



**A study of depth estimation and  
depth based applications**  
(奥行きデータの推定とその応用に関する研究)

**TRAN ANH TU**

September 2013

**A study of depth estimation and  
depth based applications**  
(奥行きデータの推定とその応用に関する研究)

by

**Tran Anh Tu**

A Thesis Submitted to  
Graduate School of Engineering of  
Hiroshima University  
in Partial Fulfillment of the  
Requirements for the Degree of

**Doctor of Engineering**

Department of Information Engineering,  
Graduate School of Engineering



September 2013

Department of Information Engineering  
Graduate School of Engineering  
HIROSHIMA UNIVERISTY



THIS IS TO CERTIFY THAT THE DISSERTATION SUBMITTED BY

**Tran Anh Tu**

ENTILED

**A study of depth estimation and depth based applications**  
(奥行きデータの推定とその応用に関する研究)

HAS BEEN ACCEPTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF

**Doctor of Engineering in Information Engineering**

**BOARD OF DISSERTATION COMMITTEE**

**Dr. Koichi Harada** (Supervisor)  
Professor  
Dept. of Information Engineering  
Graduate School of Engineering  
Hiroshima University, Japan

**Dr. Kazufumi Kaneda** (Member)  
Professor  
Dept. of Information Engineering  
Graduate School of Engineering  
Hiroshima University, Japan

**Dr. Takio Kurita** (Member)  
Professor  
Dept. of Information Engineering  
Graduate School of Engineering  
Hiroshima University, Japan

**Dr. Tadashi Shima** (Member)  
Associate Professor  
Dept. of Information Engineering  
Graduate School of Engineering  
Hiroshima University, Japan

September 2013

*To my family*

## ACKNOWLEDGMENT

First of all, I would like to thank all of people who have given their time, assistance and patience so generously. The thesis would be impossible to be completed if without their great supports.

I would like to express my deepest gratitude to my supervisor **Professor Koichi Harada**. It has been my great honor to work under his supervision in my PhD program. I extremely appreciate him for accepting me as his student and giving me valuable opportunity to work on this interesting topic. His proficient and valuable guidance made me think in more depth throughout this work. Our numerous technological discussions and his many constructive comments have greatly improved this work.

I would like to extend my deepest thanks to **Professor Kazufumi Kaneda, Professor Takio Kurita** and **Associate Professor Tadashi Shima** for taking their precious time to serve on the committee of my PhD thesis and oral defense. I appreciate their great effort in reviewing my manuscript and making useful comments.

I would like to thank to my friends, here at Hiroshima and elsewhere, who always gave me in-time encouragement to accomplish this thesis and throughout my studies.

Last but not least my deepest thanks go to my family- my fathers and mothers, my wife and my son for their love and their best supports.

# ABSTRACT

Three dimensional video ( $3DV$ ) and multi-view imaging technologies may be the next step in the evolution of motion picture formats, as we presently witness the appearance of  $3D$  displays, multi-camera systems with dense or sparse camera configuration, coding systems. Going with the demand of entertainment and progressive development of digital devices, developing  $3D$  processing algorithms, related applications and systems have been attracted extensive attentions in the industrial and research communities. Depth inference from stereo and multi-view images is one of the most fundamental techniques in  $3D$  digital imaging applications since it provides the perception and visualization of the real word environment in  $3DV$  as well as a useful cue for other applications.

This thesis devotes to firstly study depth estimation from multi-view images and then use this useful information for three applications including: one of the key applications in  $3DTV$ , namely free viewpoint synthesis, and other two applications object segmentation and multiple moving object tracking.

The first part (*Chapter 3*) of this thesis addresses the problem of depth estimation from multiple views. The depth information disappears after taking an  $2D$  image from a  $3D$  scene. To recover this missing information, the depth can be estimated from two or more images by finding the correspondence pairs among them. Initially, we introduce the basic geometric model that enables the triangulation of corresponding pixel points across the views. While the basic physics and geometry relating visual disparity to scene structure are well understood automatically measuring this disparity by establishing dense and accurate inter image correspondence is a challenging task. Some difficulties such as the unable setting the identical internal cameras' parameters, the change of illumination across the views, texture-less regions and occlusion can result in an unreliable identification of the point-correspondences and thus in inaccurate depth values. Next, we review the previous works on estimation of depth image using a single image/mono video, two views and multiple views. Finally, we have proposed a method that allows the use of several un-rectified images simultaneously to estimate a consistency and reliability depth image. We have introduced

three constraints, i.e. intra-line, inter-line and inter-view smoothness constraint, which enforce smooth variations of depth value in the scanline, across scanline and consistent depth value across the views. The proposed algorithm combines two stages: the first stage serves as a calculation of initial depth images and the second stage enhances the depth initial depth images in the first step by enforcing consistent depth across the views. The three smooth constraints can be efficiently integrated into one dimensional optimization dynamic program algorithm. Experiments have shown that the proposed method yields reasonably depth image quality for various multi-view data sets.

After investigating and presenting the depth estimation algorithm, the next part (*Chapter 4*) of this thesis focuses on the depth based image rendering for *3D* video and *3DTV* systems. In *3DV/3DTV*, the viewer can ideally navigate through the *3D* domain and selects his own viewpoint. The chosen viewpoint may not only be selected from available multi-view camera views, but also any viewpoint between these cameras. Obviously, this feature requires a smart synthesis algorithm that allows free-viewpoint view rendering. In chapter 4, we have reviewed the recent advancements in viewpoint synthesis for *3DTV* and then proposed a novel method and showed its performance. Our contribution is a novel synthesis method that enables to render a free-viewpoint from multiple existing cameras. The proposed method solves the main problems of depth based synthesis by applying forward depth map following with inverse warping texture, performing pixel classification to generate an initial new view from stable pixels and using Graph cut to select the best candidate for unstable pixels. By defining the types of pixels and using Graph cuts, the color is consistent and the pixels wrapped incorrectly because of inaccuracy depth maps are removed. The remained disoccluded pixels are inpainted by using depth and texture neighboring pixel value. Considering depth information for inpainting, blurring between foreground and background textures are reduced. Experimental results show that the proposed method has strength in artifact reduction. Objective evaluation has shown that our method get a significant gain in Peak Signal Noise Ratio (*PSNR*) and Structure Similarity Index (*SSIM*) comparing to some other existing methods. Another advantage of our method is that we can use a set of un-rectified images in multi-view system to create a new view with higher quality

As the estimated depth information available, our concern is to apply the usefully estimated  $3D$  information for the object segmentation method (*Chapter 5*). Even though image segmentation has been addressed in extensive literatures, the results are not satisfactory in many situations. A major difficulty lies in the fact that semantic objects are not homogeneous with respect to the low-level features in single image, such as color or texture properties. Fortunately, depth information recovered from multi-view serves as an important cue for segmentation. We have proposed a method using both depth and color cues, which requires no interactive operation, to segment human object from multi-view video. Our method consist of two stages: for initial frame of the video sequence, the interested object is automatically extracted based on saliency model and iterated Graph cut. After having segmented object in first frame, from the second frame we have proposed a method combining Bayesian estimation and minimizing energy function using Graph cut to segment object. We use Gaussian Mixture Model (*GMM*) in RGB space for the color cue and histogram model for depth cue. Based on these probabilistic models, the probability of each pixel to be in foreground is computed based on Bayesian estimation and the results are used to create the tri-map including foreground (F), background (B) and uncertain region (U). Graph cut is then performed on the uncertain region. In the energy function for Graph cut optimization, the color, depth and spatial-temporal coherence are integrated in data term and the penalty cost of the neighboring pixels with different labels is encoded in smoothness term. Experiment results on test sequences are encouraging and showed that our method is more effective than the case using only color cue.

The final work in this thesis is to using the estimated depth information for object tracking (*Chapter 6*). Detection and tracking of objects is very importance research area of computer vision and has a wide range of application. Many researchers have investigated object tracking and different approaches have been presented. Some of approaches can achieve good results in some cases, such as when the target object has distinct color distribution from the background. However, multi objects tracking is still a difficult task due to various aspects, including inaccurate motion vector estimation, variation of the non-rigid object appearance and confusions in multiple targets' identities when their projections in the image are close. Moreover, object regions with various tracking issues such as



## Abstract

---

appearance and disappearance, splitting and merging, without and with occlusion should be dealt with in the tracking algorithm. We have proposed a novel tracking method aiming at detecting objects and maintaining their label/identification over the time. The main key factors of this method are to use depth information and different strategies to track objects under various occlusion scenarios. The foreground objects are detected and refined by background subtraction and shadow cancellation. The occlusion detection is based on information of foreground blobs in successive frames. The occlusion regions are projected to the projection plane  $XZ$  (ground plane) to analysis occlusion situations. According to the occlusion analysis results, different objects correspondence strategies are introduced to track object under various occlusion scenarios including tracking occluded objects in similar depth layer and in different depth layers. The experimental results show that our proposed method can track the moving objects under the most typical and challenging occlusion scenarios.

## LIST OF FIGURES

Figure 1-1. Organization of this Thesis.....	4
Figure 2-1. The ideal pinhole camera model.....	9
Figure 2-2. Image coordinate system (right side); Non ideal image sensor with non-square and skewed pixels (left side). .....	10
Figure 2-3. Radial distortion: rays farther from the center of a simple lens are bent to much compared to the rays that pass closer to the center resulting the sides of a square appear to bow out of the image plane .....	12
Figure 2-4. The relationship between the camera and world coordinate system. ....	14
Figure 2-5. A planar chessboard pattern with 3D world coordinate system. ....	16
Figure 2-6. Two perspective views of the same 3D scene. ....	20
Figure 2-7. The epipolar and epipolar constraint. ....	21
Figure 2-8. Two views of scene with 4 epipolar lines.....	22
Figure 2-9. The essential matrix E contains all of the information about translation T and rotation R, which describes the location of the second camera relative to the first in world coordinates. ....	24
Figure 2-10. Stereo image rectification. The epipolar lines become collinear and parallel to the image scan-line. ....	26
Figure 2-11. An example of graph construction in [15] (Edge cost are reflected by the thickness). ....	30
Figure 2-12. An example image labeling. ....	31
Figure 3-1. An example of 3D depth camera products. ....	35
Figure 3-2. The restricted case of depth estimation. ....	36
Figure 3-3. An example of Sum of Absolute Different (SAD). ....	37
Figure 3-4. An example of depth estimated from single still image [31]. ....	40
Figure 3-5. Multi-view system in Virtualized Reality Project developed by CMU [48]. ....	44
Figure 3-6. A self-reconfigurable camera array system with 48 cameras [50]. ....	45
Figure 3-7. “100-camera system” at Nagoya University with different camera arrangement: 1-D line, 1-D arc, and 2-D array [51]. ....	46
Figure 3-8. A configuration of MSR with 8 cameras [52]. ....	46
Figure 3-9. An example of our Chessboard calibration Technique.....	49
Figure 3-10. The spatial configuration of the two cameras and the calibration planes.....	50
Figure 3-11. Stereo rectification: (a) original pair; (b) rectified pair; note that the barrel distortion (in (a)) has been corrected and the scan lines are aligned in the rectified images. ....	51
Figure 3-12. Estimated disparity maps from stereo images captured by Minoru 3D webcam. ....	52
Figure 3-13. Estimated disparity maps using stereo images from Middlebury Stereo Datasets [63]. ....	53
Figure 3-14. Correlation table with all matching cost for all depth value a long scanline.....	56

## List of Figures

---

Figure 3-15. Given a pixel $p(x,y)$ in the reference view, the algorithm test the vector of depth candidate $Z_c$ . For example, candidate $Z_i$ yields the consistent color across views, it should be selected as the final depth value.....	58
Figure 3-16. Original “Break-dancers” image (on the left) and the corresponding segmented image using mean shift segmentation algorithm (on the right). .....	61
Figure 3-17. The idea to objectively evaluate the quality of depth map via view synthesis. ....	64
Figure 3-18. An estimated depth image for “Break-dancers”.....	66
Figure 3-19. An estimated depth image for “Ballet”.....	66
Figure 3-20. Comparing the proposed method with others.....	68
Figure 4-1. Example of advanced 3DTV concept based on Multiple View plus Depth (MVD) (9 outputs views out of 3 input views plus depth); Pos: viewpoint, R: right eye, L: left eye, V: view/image, D: Depth, DIBR: Depth image based rendering.....	70
Figure 4-2. An example of 3DTV system with multiple view plus depth format [56]. .....	71
Figure 4-3. The two projection point $p_1$ and $p_2$ of a point $P_w$ .....	72
Figure 4-4. Proposed new view synthesis algorithm.....	75
Figure 4-5. The projected depth maps from two reference cameras (from the left side and from the right side). .....	76
Figure 4-6. Median filter depth maps.....	77
Figure 4-7. Image synthesis process using inverse warping. ....	77
Figure 4-8. An example of interpolating color of the synthetic pixel from the four neighboring pixels in the reference image. ....	78
Figure 4-9. Obtained color images by inverse warping. ....	78
Figure 4-10. Weighted interpolation based on angular distances. ....	81
Figure 4-11. Initial synthesized view with 3 types of pixels: the white color pixels are unstable pixels, the red color pixels are disoccluded pixels and the remaining pixels are stable pixels. ....	81
Figure 4-12. Refinement of initial synthesized image (image in Figure 4-11) by using Graph cut (the red color pixels are disoccluded pixels).....	84
Figure 4-13. A configuration of “Break-dancer” and “Ballet” sequences with 8 cameras [52].....	87
Figure 4-14. Experimental scenario: view 3 and 5 are used to synthesize view 4. ....	88
Figure 4-15. Example of the synthetic view. ....	88
Figure 4-16. PSNR and SSIM comparison: (a) PSNR for “Break-dancer”, (b) SSIM for “Break-dancer”, (c) PSNR for “Ballet”, (d) SSIM for “Ballet”.....	89
Figure 4-17. Varying the distance between the two reference cameras. ....	90
Figure 4-18. PSNR versus distance between cameras for: (a) “Break-dancer” sequence, (b) “Ballet” sequence. ....	91
Figure 4-19. Number of unstable pixels (%) comparing to image size.....	91
Figure 5-1. An example of region-based and object-based image segmentation.....	94
Figure 5-2. The illustration of proposal algorithm. ....	99
Figure 5-3 Stereo depth image. ....	100
Figure 5-4. Object segment based on SM and GrabCut.....	102

## List of Figures

---

Figure 5-5. Tri-map (black color: background region; white color: foreground region; grey color: uncertainly region). .....	105
Figure 5-6. Quantitative comparison.....	107
Figure 5-7. Segmentation works well with non-stationary background.....	107
Figure 5-8. Segmentation on stereo video captured by Minoru 3D webcam .....	108
Figure 6-1. The flowchart of proposed tracking method.....	110
Figure 6-2. Color image and depth image.....	111
Figure 6-3. Foreground segmentation. ....	113
Figure 6-4. Shadow regions detection.....	113
Figure 6-5. The image plane XY and ground plane XZ.....	116
Figure 6-6. Projected foreground blobs in XZ plane.....	117
Figure 6-7. An example of image segmentation by means of depth ranges.....	118
Figure 6-8. An example of color histogram. ....	119
Figure 6-9. Tracking occluded objects in different depth layers.....	119
Figure 6-10. Bhattacharyya distance between two histograms. ....	120
Figure 6-11. Tracking occluded objects in different depth layers.....	121
Figure 6-12. An example of projection image. ....	122
Figure 6-13. An example of tracking occluded objects in one depth layer. ....	123
Figure 6-14. Result of tracking non-occluded objects. ....	124
Figure 6-15. Tracking occluded objects in different depth layers.....	124
Figure 6-16. Tracking occluded objects in one depth layer. ....	125
Figure 6-17. Tracking fully occluded objects. ....	125

## LIST OF TABLES

Table 2-1. The two different geometries, the transformations allowed in each, and the measures that remains invariant under those transformations. ....	8
Table 3-1. Minoru 3D webcam specification. ....	48
Table 3-2. Calibration parameters of Minoru 3D webcam. ....	50
Table 3-3. The quality of synthesized image resulting of two different depth estimation algorithms. ....	65
Table 4-1. The measured synthetic image qualities are compared with other methods. ....	90
Table 4-2. Calculation time for one frame with varying numbers of unstable pixels. ....	92
Table 7-1. Calibration parameters of the multi-view sequence “Break-dancers”. ....	133
Table 7-2. Calibration parameters of the multi-view sequence “Ballet”. ....	134

# Contents

<b>Acknowledgment</b> .....	<b>i</b>
<b>Abstract</b> .....	<b>ii</b>
<b>List of Figures</b> .....	<b>vi</b>
<b>List of Tables</b> .....	<b>ix</b>
<b>1 Introduction</b> .....	<b>1</b>
1.1 Motivation and Objective of Thesis .....	1
1.2 Structure of Thesis.....	4
<b>2 Background</b> .....	<b>7</b>
2.1 Camera Models and Calibration .....	7
2.1.1 Projective Geometry.....	7
2.1.2 Pinhole Camera Model.....	8
2.1.3 Projection of 2D Point to 3D Points.....	15
2.1.4 Camera Calibration .....	16
2.2 Two-view Geometry .....	20
2.2.1 Epipolar Geometry .....	21
2.2.2 Stereo Calibration.....	24
2.2.3 Stereo Rectification.....	26
2.3 Background of Graph Cuts Theory .....	28
2.4 Summary.....	32
<b>3 Depth Estimation</b> .....	<b>34</b>
3.1 Introduction .....	34
3.2 The Formulation Depth From Stereo.....	35
3.3 Previous Works on Depth Estimation.....	38
3.3.1 Single Image/Video Depth Estimation.....	38
3.3.2 Stereo Depth Estimation.....	41
3.3.3 Multi- view Depth Estimation.....	43
3.4 Experimental Example: Stereo Calibration, Rectification and Stereo Correspondence..	48
3.5 Proposed Depth Estimation Algorithm.....	54
3.5.1 Introduction .....	54
3.5.2 Proposal Depth Estimation from Multiple Images.....	55
3.5.3 Experimental Results .....	63
3.6 Summary and Conclusions .....	66

<b>4</b>	<b>Depth Image Based Synthesis .....</b>	<b>69</b>
4.1	Introduction .....	69
4.2	Current Research on Depth Image Based Synthesis.....	72
4.2.1	Basic Principle: 3D Image Warping.....	72
4.2.2	Previous Works .....	73
4.3	Proposed Algorithm and Its Performance.....	74
4.3.1	Algorithm Overview .....	74
4.3.2	Proposed Depth Image Based Synthesis using Graph Cuts .....	75
4.3.3	Experimental Results .....	85
4.4	Summary and Conclusions .....	92
<b>5</b>	<b>Depth Assisted Object Segmentation .....</b>	<b>94</b>
5.1	Introduction .....	94
5.2	Current Research on Depth Based Object Segmentation .....	96
5.3	Proposed Segmentation Algorithm Using Depth Information .....	98
5.3.1	Algorithm Overview .....	98
5.3.2	Segmentation Algorithm .....	100
5.3.3	Experimental Results .....	106
5.4	Conclusions .....	108
<b>6</b>	<b>Depth Aided Object Tracking.....</b>	<b>109</b>
6.1	Introduction .....	109
6.2	Proposed Tracking Method .....	111
6.2.1	Method Overview.....	111
6.2.2	Depth-Aided Tracking Multiple Objects under Occlusion .....	111
6.2.3	Experimental Results .....	123
6.3	Conclusions .....	126
<b>7</b>	<b>Conclusions and Outlook .....</b>	<b>127</b>
7.1	Summary and Conclusions .....	127
7.2	Outlook .....	130
<b>A</b>	<b>Appendices.....</b>	<b>133</b>
A.1	Calibration Parameters of the Multi-view Sequences “Break-Dancers” and “Ballet” ..	133
<b>B</b>	<b>Abbreviations .....</b>	<b>135</b>
<b>C</b>	<b>List of Publications by Author.....</b>	<b>136</b>
	<b>References .....</b>	<b>137</b>





# 1

## INTRODUCTION

### 1.1 MOTIVATION AND OBJECTIVE OF THESIS

The conventional two-dimensional ( $2D$ ) images captured by a traditional single camera lose three-dimensional ( $3D$ ) information especially the depth information of the  $3D$  scene, which is a useful and important cue for perception and visualization of the real world environment. In the recent years, with the fast improvement of the capability of personal computers and digital equipment, more and more multi-camera systems with dense or sparse, wide-baseline or narrow-baseline camera configuration become available, which significantly broaden the multi-view applications and enhance the user experience. Going with the demand of entertainment and progressive development of digital devices, developing  $3D$  processing algorithms, related applications and systems have been attracted extensive attentions in the industrial and research communities.

Recently, multi-view imaging technologies are becoming important with the ongoing convergence of extensive visualizations, and greatly influence our life in the area of surveillance for environmental security, entertainment, and virtual view synthesis for three dimensional television ( $3DTV$ ) and free viewpoint television ( $FTV$ ) system. Additionally, content-based applications enable analysis and interpretation of data by accessing and manipulating semantic objects, offering user flexibility for data exploitation in the object level. This popularity and flexibility are the motivation of our efforts to learn some key technologies and applications in a multi-view system such as multi-view acquisition, depth estimation, depth based view synthesis for  $3DTV$ , depth assisted object segmentation and tracking. We focus on learning to estimate depth from multi-view images and to use this useful information for some important applications that are view synthesis for  $3DTV$ , object segmentation and tracking.

In this thesis, firstly we study about estimate depth from multi view images. Depth inference from stereo and multi-view images is one of the most fundamental techniques in  $3D$  digital imaging

## 1.1. Motivation and Objective of Thesis

---

applications. The depth information disappears after taking an  $2D$  image from a  $3D$  scene. To recover this missing information, the depth can be estimated from two or more images by finding the correspondence pairs among them. From the earliest inquiries into visual perception, it was known that we perceive depth based on the difference in appearance between left eye and right eye. When viewing a scene with both eyes, the observers can use information derived from the different projection of objects onto each retina to judge depth. Thus, stereo or disparity measured by using two images of the same scene from different angles is the binocular cue to determine the depth information, which is negatively correlated to the distance between camera and the object. While the basic physics and geometry relating visual disparity to scene structure are well understood, automatically measuring this disparity by establishing dense and accurate inter-image correspondence is a challenging task. Depth estimation from a set of multiple views or images has been widely studied in the computer-vision research community. However, depth estimation still remains an open research topic since this is an ill-posed problem in many situations. For example, the following aspects illustrate some difficulties. First, due to the inability to set the identical internal cameras' parameters and/or the change of illumination across the views, multi-view images may not conform to the photo-consistency rule. The projections of the same  $3D$  point into different views can have different intensity/color values. This results in an unreliable identification of the point-correspondences and thus in inaccurate depth values. Second, object surfaces appear differently depending on the viewpoint. The third, it may occur the appearance of texture-less regions and repeated patterns in the scene. This will increase the ambiguity while finding reliable corresponding points, thereby resulting in inaccurate depth values. The fourth, in some cases, particular background regions may be visible from a given camera viewpoint but may not be visible from a different camera viewpoint, so that, it is not possible to identify point-correspondences across the views. In this thesis, first we review the existing methods and then propose the algorithm which uses simultaneously multiple images/views to estimate depth.

An important application of depth maps is the depth-based image rendering for  $3D$  video and  $3DTV$  systems. In the near future, most video content will be available in  $3D$ .  $3D$  does not only

mean stereo video, but also multi-view video, where a user can navigate around and change its viewpoint in a similar way as it is already known from today's video games. Multi-view video is recorded using arrays of cameras, which are capturing the same scene from different viewpoints. This technique especially enables applications such as *3DTV* and *FTV*. Comparing with the *2DTV*, the most impressive progress of *3DTV* and *FTV* is to offer the users a *3D* depth feeling of the observed scene and the flexibility of interactively selecting the freedom viewpoint of the real environment. The chosen free viewpoint may not only be selected from available multi-view cameras, but also any viewpoint between these cameras. It requires a smart rendering algorithm that allows free viewpoint view synthesis. In this thesis, we concern such multi view synthesis algorithm and aims at proposing best synthesis algorithm when using multi-view texture and depth images of the scene.

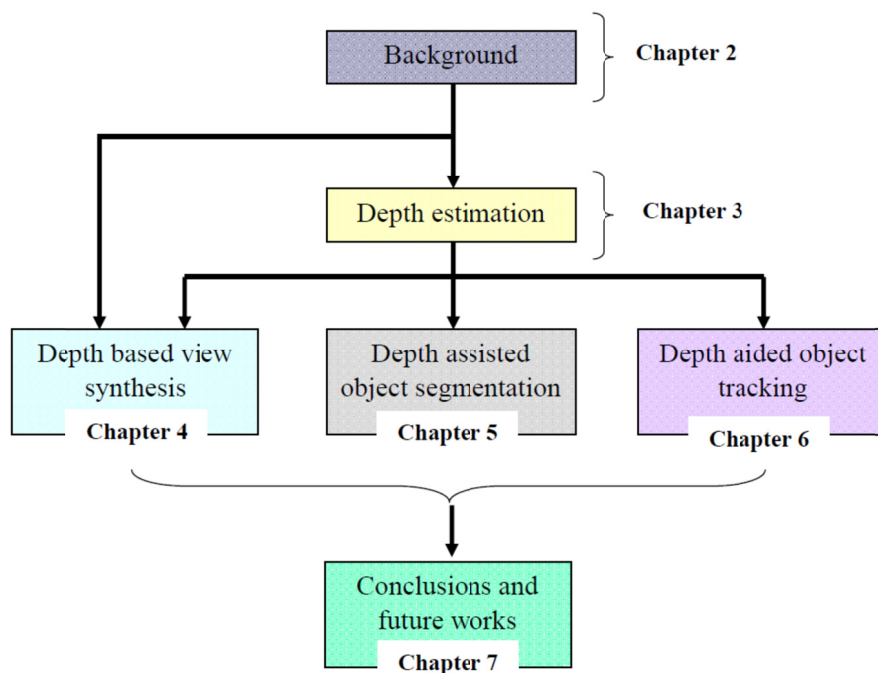
As the estimated depth information available, our next concern is to apply the usefully estimated *3D* information for the object segmentation method. Even though image segmentation has been addressed in extensive literature, the results are not satisfactory. A major difficulty lies in the fact that semantic objects are not homogeneous with respect to the low-level features in single image, such as color or texture properties. Fortunately, depth information recovered from multi-view serves as an important cue for segmentation. However, due to ill-posed nature of depth estimation, errors may occur in the depth map. To obtain more robust segmentation results for object-level manipulation, we consider to integration of depth, color, and other image cues in our algorithms.

The final work in this thesis is to using the estimated depth information for object tracking. Detection and tracking of objects is very importance research area of computer vision and has a wide range of application. Many researchers have investigated object tracking and different approaches have been presented. Some of approaches can achieve good results in some cases, such as when the target object has distinct color distribution from the background. However, multi objects tracking is still a difficult task due to various aspects, including inaccurate motion vector estimation, variation of the non-rigid object appearance and confusions in multiple targets' identities when their projections in the image are close. Moreover, object regions with various tracking issues such as appearance and

disappearance, splitting and merging, without and with occlusion should be dealt with in the tracking algorithm. Depth information will be used in our tracking method. This information is employed to analysis occlusion and help to track objects under various occlusion scenarios.

### 1.2 STRUCTURE OF THESIS

This thesis addresses the following topics: depth estimation from multi-view images, depth based view synthesis, depth assisted object segmentation and depth aided moving object tracking. **Figure 1-1** shows the organization and relationship between the chapters.



**Figure 1-1. Organization of this Thesis.**

#### **Chapter 2: Background**

- Providing the summary of geometry of multiple views; the relationship of 3D points and 2D point in image plane;
- Describing a method for computing the internal and external parameters of the cameras, which is known as camera calibration;

- Describing a stereo rectification process which is correcting the individual images so that they appear as if they had been taken by two cameras with row-aligned image planes. This help to reduce the search of point correspondences when depth is estimated;
- Introducing a very short summary of Graph Cuts, which is use to minimize our energy functions in the in later chapters.

### **Chapter 3: Depth Estimation**

- Providing the comprehensive review the previous works on estimation of depth image using a single image/mono video, two views and multiple views;
- Presenting the proposed depth estimation method and experimental results. The proposed method is utilizing simultaneously all views and using one dimensional optimization strategy;

### **Chapter 4: Depth Image Based Synthesis**

- Briefly introducing the advances in three-dimensional video/television (*3DV / 3DTV*);
- Providing the review the previous works on estimation of depth based image synthesis;
- Presenting a novel view synthesis algorithm and its performance assuming that the depth maps, textures of multi-view cameras and their parameters are available;

### **Chapter 5: Depth Assisted Object Segmentation**

- Briefly introducing the object segmentation topic;
- Providing the current research on depth based object segmentation;
- Presenting the proposed object segmentation algorithm, in which depth is fused with color and spatial-temporal coherence in an energy function. This function is optimizing by Graph Cuts.

**Chapter 6: Depth Aided Object Tracking**

- Providing the current research on depth based object tracking;
- Presenting the proposed tracking method, in which depth information and different strategies are used to track objects under various occlusion scenarios.

**Chapter 7: Conclusion and Outlook**

- Providing a summary, some conclusions as well as an outlook on future work.

## 2

# BACKGROUND

### 2.1 CAMERA MODELS AND CALIBRATION

#### 2.1.1 *Projective Geometry*

Projective geometry serves as a mathematical model for how images of the  $3D$  world are formed. It is used to model the image formation process, generate synthetic images, or reconstruct  $3D$  objects from multiple images. To model lines, planes or points in a  $3D$  space, usually the Euclidean geometry is employed. In Euclidean space, a point defined in three dimensions is represented by a 3-element vector  $(X, Y, Z)^T$ ; the sides of objects have lengths, intersecting lines determine angles between them, and two lines are said to be parallel if they lie in the same plane and never meet. Moreover, these properties do not change when the Euclidean transformations (translation and rotation) are applied. However, when we consider the imaging process of a camera, it becomes clear that Euclidean geometry is insufficient: lengths and angles are no longer preserved, and parallel lines may intersect.

Projective geometry establishes an attractive framework to circumvent the above disadvantages of the Euclidean geometry. In the projective space, the same point is described using a 4-element vector  $(X_1, X_2, X_3, X_4)^T$  such that

$$X = X_1/X_4, \quad Y = X_2/X_4, \quad Z = X_3/X_4, \quad (2-1)$$

where,  $X_4 \neq 0$ . Generally, the coordinates  $(X, Y, Z)^T$  and  $(X_1, X_2, X_3, X_4)^T$  are called inhomogeneous coordinates and homogeneous coordinates [1], respectively.

As a generalization, the mapping from a point in the  $n$ -dimensional Euclidean space to a  $(n+1)$  dimensional projective space can be written as

## 2.1. Camera Models and Calibration

---

$$\underbrace{(X_1, X_2, \dots, X_n)^T}_{\text{Euclidean space}} \rightarrow \underbrace{(\lambda X_1, \lambda X_2, \dots, \lambda X_n, \lambda)^T}_{\text{Projective space}}, \quad (2-2)$$

where  $\lambda \neq 0$  corresponds to a free scaling parameter. This free scaling parameter  $\lambda$  is often called the homogeneous scaling factor.

Projective geometry models well the imaging process of a camera because it allows a much larger class of transformations than just translations and rotations, a class which includes perspective projections. **Table 2-1** shows the relationships between two different geometries.

**Table 2-1. The two different geometries, the transformations allowed in each, and the measures that remains invariant under those transformations.**

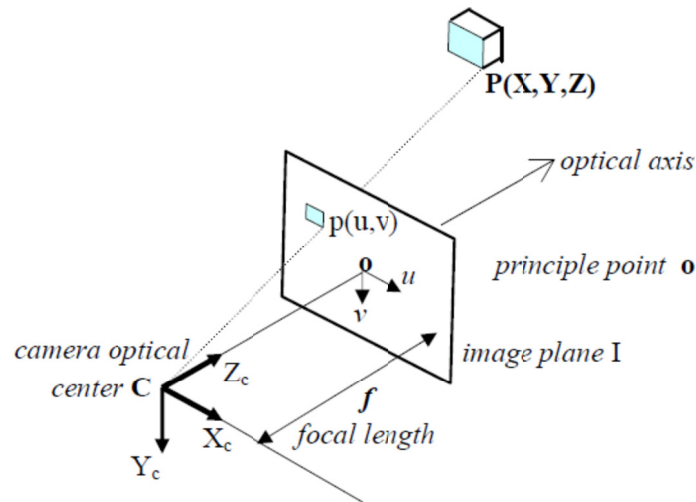
	Euclidean	Projective
Transformations:		
rotation	x	x
translation	x	x
reflection	x	x
uniform scaling		x
non-uniform scaling		x
perspective projection		x
Invariants:		
length	x	
angle	x	
ration of lengths	x	
parallelism	x	
cross ratio	x	x

### 2.1.2 Pinhole Camera Model

Pinhole camera is a simple camera without a lens and with a single small aperture, effectively a light-proof box with a small hole in one side. Light from a scene passes through this single point and



projects an inverted image on the opposite side of the box. The simplest form of real camera comprises a pinhole and an imaging plane (or screen). Because the pinhole lies between the imaging plane and the observed 3D world scene, any ray of light that is emitted or reflected from a surface patch in the scene is constrained to travel through the pinhole before reaching the imaging plane. Therefore, there is a correspondence between each 2D area on imaging plane and the area in 3D world as observed through the pinhole from the imaging screen. The ideal pinhole camera model describes the relationship between a 3D point  $P = (X, Y, Z)^T$  and its corresponding 2D projection point  $p = (u, v)^T$  onto the image plane. **Figure 2-1** illustrates the ideal model of pinhole camera. When using a pinhole camera model, this geometric mapping from 3D to 2D is called a *perspective projection*.



**Figure 2-1. The ideal pinhole camera model.**

We denote the center of the perspective projection (the point in which all the rays intersect) as the *optical center* or *camera center* and the line perpendicular to the image plane passing through the optical center as the *optical axis*. Additionally, the intersection point of the image plane with the optical center as the *principal point*  $o = (o_x, o_y)^T$ . The distance from the image plane to optical center point is known as the *focal length*  $f$ . The mathematical model of pinhole camera that describes a perspective projection of 3D points onto the image plane can be described as follows.

## 2.1. Camera Models and Calibration

Let us simplify by consider camera with the optical axis being collinear to the  $Z_{cam}$  axis and the optical center being located at the origin of a  $3D$   $3D$  coordinate system, as shown in **Figure 2-1**. The projection of a  $3D$  world point  $P = (X, Y, Z)^T$  onto the image plane at pixel position  $p = (u, v)^T$  can be written as

$$u = \frac{fX}{Z}, \quad v = \frac{fY}{Z}. \quad (2-3)$$

Avoiding such a non-linear division operation, the Equation (2-3) can be reformulated using the projective geometry framework, as

$$(\lambda u, \lambda v, \lambda)^T = (fX, fY, Z)^T. \quad (2-4)$$

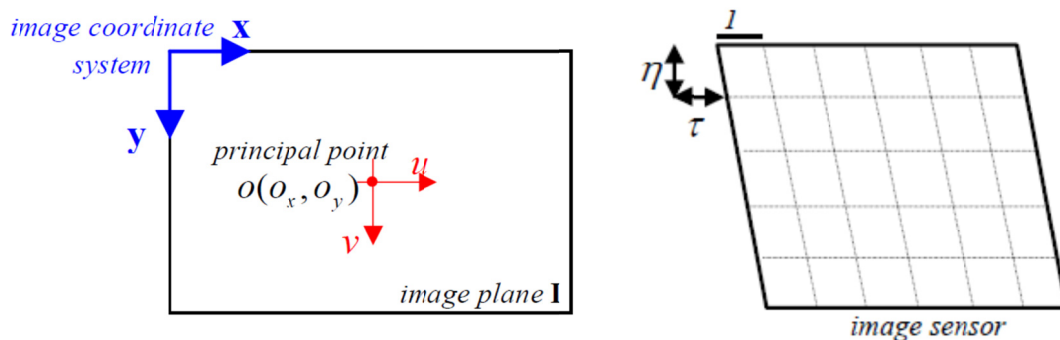
This relation can be the expressed in matrix notation by

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (2-5)$$

where  $\lambda = Z$  is the homogenous scaling factor.

Equation (2-5) constitutes a foundation of the pinhole camera. The pinhole camera model can be defined by two sets of parameters: intrinsic parameters and extrinsic parameter.

### 2.1.2.1 Intrinsic Camera Parameters



**Figure 2-2. Image coordinate system (right side); Non ideal image sensor with non-square and skewed pixels (left side).**

To formulate above relation described in Equation (2-5), we have assumed that the origin of the pixel coordinate system corresponds to the principal point  $o = (o_x, o_y)^T$ , located at the center of the image. However, most of the current imaging systems define the origin of the pixel coordinate system at the top-left pixel of the image. A conversion of coordinate systems is thus necessary. Using homogeneous coordinates, the principal point position can be readily integrated into the projection matrix. Now the perspective projection equation becomes

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & o_x & 0 \\ 0 & f & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \quad (2-6)$$

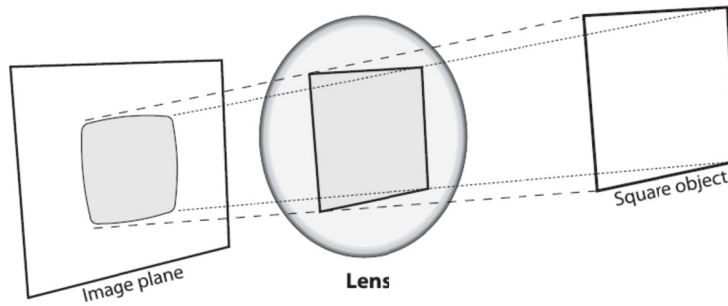
We have assumed that the pixels of image sensor are square, i.e. aspect ratio is 1:1 and pixels are not skewed to derive the relation (2-6). However, both assumptions may not always be valid. In practice, the pixel aspect ratio is often provided by the image-sensor manufacturer. The individual pixel on typical low cost camera is rectangular rather than square. For example, an NTSC TV system defines non-square pixels with an aspect ratio of 10: 11. Moreover, pixels can potentially be skewed, especially in the case that the image is acquired by a frame grabber due to an inaccurate synchronization of the pixel-sampling process. Two parameter  $\eta$  and  $\tau$  are used to model the pixel aspect ratio and skew of the pixel respectively as shown in **Figure 2-2**. The Equation (2-7) can be update as

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & \tau & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = [K \mid 0_3]P, \quad (2-7)$$

with  $P = (X, Y, Z, 1)^T$  being a 3D point defined with homogenous coordinates and  $f_y = \eta f_x$ . The intrinsic parameters are denoted as  $K$ , and  $0_3$  is denoted by  $0_3 = [0, 0, 0]^T$ . With nowadays digital cameras, it can be assumed that pixels are non-skewed ( $\tau = 0$ ).

2.1.2.2 *Lens Distortion*

In theory, it is possible to define a lens that will introduce no distortions. However, in practice, no lens is perfect. Distortions encountered in real optical systems arise mostly from the nonlinearity of physical elements, as well as from the dependence of the optical parameters on the wavelength of the incident light. Here we briefly describe the two main lens distortions and a standard technique to model them.



**Figure 2-3. Radial distortion: rays farther from the center of a simple lens are bent to much compared to the rays that pass closer to the center resulting the sides of a square appear to bow out of the image plane .**

The largest distortion is radial distortions. In practice, radial lens distortion causes straight lines to be mapped as curved lines, this is also known as barrel distortion. The lens of real cameras often noticeably distorts the location of pixels near the edges image as shown in **Figure 2-3**. With some lens, particularly in cheap webcam, rays father from the center of the lens are bent more than those closer in. For the radial distortions, the distortion is 0 at the center of image and increase as we move toward the periphery. This distortion is small and can be characterized by the few term of Taylor series expansion. For the cheap web cameras, we usually use first three such terms, which is conventionally called  $k_1, k_2$  and  $k_3$ . In general, the radical location of a point on the image will be rescaled according to equation [2]:

$$\begin{aligned} x_{corrected} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corrected} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \end{aligned} \tag{2-8}$$

where,  $(x, y)$  is original location of distorted point on the image and  $(x_{corrected}, y_{corrected})$  is the new location as the result of correction.

The second largest common distortion is tangential distortion. This distortion is due to manufacturing defects resulting from the lens not being exactly parallel to the imaging plane. Tangential distortion is minimally characterized by two parameters  $p_1$  and  $p_2$  [2], such that:

$$\begin{aligned}x_{corrected} &= x + [2p_1y + p_2(r^2 + 2x^2)] \\y_{corrected} &= y + [p_1(r^2 + 2y^2) + 2p_2x].\end{aligned}\tag{2-9}$$

There are many kinds of distortions that occur in imaging systems, but they are lesser effect than radial and tangential distortions. Hence almost research will not deal with them.

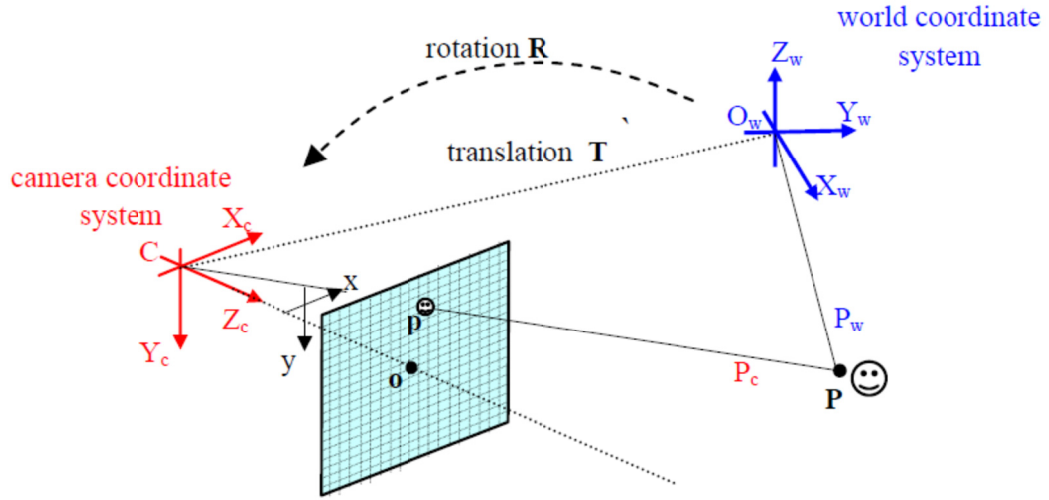
We have five distortion coefficients that model the influence of the radial and tangential distortions of the optical system. These coefficients are the new intrinsic parameters of the camera model. In OpenCV[2], they are typically bundled into one distortion vector 5 by 1, which is containing  $k_1, k_2, p_1, p_2$ , and  $k_3$ .

### 2.1.2.3 Extrinsic Parameter

So far, we assume that 3D points are expressed in the camera coordinate system. In practice, they can be expressed in any 3D coordinate system. In order to understand how points in the real world are related mathematically to the points in the images plane, two coordinate systems (see the **Figure 2-4**) are of particular interest:

1. The world coordinate system at  $O_w$  (denoted here with a subscript ‘ $W$ ’ for ‘world’), which is independent of placement and parameter of the camera.
2. The camera coordinate system at the optical center  $C$  (denoted by ‘ $C$ ’ for ‘camera’).

The two coordinate system are related by translation expressed by vector  $T$  and rotation represented by matrix  $R$ . The translation vector  $T$  describes a change in position of the coordinate center  $C$  and  $O_w$ ,  $T = C - O_w$ . The rotation, in turn, changes the corresponding axes of each system. This change is described by the orthogonal matrix  $R$  of dimension  $3 \times 3$  ( $RR^T = 1$ ) [3].



**Figure 2-4. The relationship between the camera and world coordinate system.**

For a given point  $P$ , its coordinate related to camera and external coordinate related to world coordinate are connected by following formula:

$$P_c = RP_w + T = R(P_w - C)$$

$$\text{or } \begin{bmatrix} P_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} P_w \\ 1 \end{bmatrix} = \begin{bmatrix} R & -RC \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} P_w \\ 1 \end{bmatrix}, \quad (2-10)$$

where  $P_c$  expresses placement of a point  $P$  in the camera coordinate system,  $P_w$  is its placement in the external coordinate system,  $R$  stands for the rotation matrix and  $T$  is the translation matrix between origins of those two coordinate systems,  $C$  is position of camera optical center. The matrices  $R$  and  $T$  can be specified as follows:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}_{3 \times 3}, \quad T = -RC = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}_{3 \times 1}. \quad (2-11)$$

Substituting (2-10) into (2-7) we obtain the linear equation of the pinhole camera:

$$\lambda p = [K | 0_3] \begin{bmatrix} R & -RC \\ 0_3^T & 1 \end{bmatrix} P = MP, \quad (2-12)$$

where  $p = (x, y)^T$  is an image point of  $P = (X, Y, Z, 1)^T$  under transformation  $M$  performed by pinhole camera. The matrix  $M$  is called a *projection matrix*. Equation (2-12) defines a transformation of the projecting a 3D point onto the image plane.

### 2.1.3 Projection of 2D Point to 3D Points

In the previous section, the process of projecting a 3D point onto the 2D image plane was described. This part will present how a 2D point can be back-projected to the 3D space and derive the corresponding coordinates. Considering a 2D point  $p$  in an image, there exists a collection of 3D points that are mapped and projected onto the same point  $p$ . This collection of 3D points constitutes a ray connecting the camera center  $C = (C_x, C_y, C_z)$  and  $p = (x, y, 1)^T$ .

The Equation (2-12) can rewrite as

$$\lambda p = KR \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - KRC. \quad (2-13)$$

From Equation (2-13) the ray  $P(\lambda)$  associated to a pixel  $p = (x, y, 1)^T$  can be defined as

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \underbrace{C + \lambda R^{-1}K^{-1}p}_{\text{ray } P(\lambda)} = \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} + \lambda \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}, \quad (2-14)$$

where  $\lambda$  is the positive scaling factor defining the position of the 3D point on the ray.

If  $Z$  is known, from Equation (2-14) it is possible to obtain the coordinates  $X$  and  $Y$  by calculating  $\lambda$  using the following relation

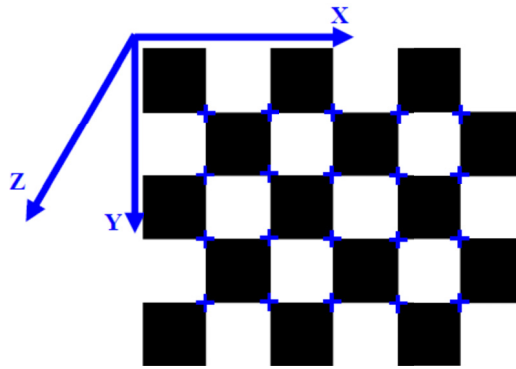
$$\lambda = \frac{Z - C_z}{r_3}, \quad (2-15)$$

where  $r_3$  is computed by Equation (2-14).

The back-projection of 2D point to 3D points is important for depth estimation and image rendering, which will be used later chapters in this Thesis.

### 2.1.4 Camera Calibration

Camera calibration is a process of finding the intrinsic and extrinsic parameter of a camera or subset of these. This topic has attracted great attention among researchers resulting an ample literature, for instance [4-10]. An evaluation of the three common calibration method of Tsai [5], Heikkila [6] and Zhang [10] can be found in the paper [11]. In this section, we do not present a new calibration algorithm but instead shortly describe the method proposed by Zhang [10].



**Figure 2-5. A planar chessboard pattern with 3D world coordinate system.**

The estimation of camera parameters is based on a calibration rig with known geometry. In principle, any appropriately characterized object could be used as a calibration object, yet the practical choice is a regular pattern, such as a planar chessboard pattern (see **Figure 2-5**).

The first stage of the camera calibration procedure is to establish correspondences between 2D points in the image and 3D points on the chessboard, i.e., so-called point-correspondences. Because each 3D feature point belongs to the board plane  $Z = 0$ , the projection of each 2D point onto the image plane can be written as

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = s[K \mid 0_3] \begin{bmatrix} R & -RC \\ 0_3^T & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z = 0 \\ 1 \end{pmatrix}, \quad (2-16)$$

which can be rewrite to



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = sK[r_1 \ r_2 \ r_3 \ t] \begin{pmatrix} X \\ Y \\ Z=0 \\ 1 \end{pmatrix} = sK[r_1 \ r_2 \ t] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \quad (2-17)$$

where,  $r_i$  corresponds to the  $i^{th}$  column of the rotation matrix  $R$  and  $t = -RC$ .

The *homography matrix*  $H$  that map a planer object's point on the imager is then described by  $H = sK[r_1 \ r_2 \ t]$ . Note that  $H$  is now a 3-by-3 matrix. The calculation of the camera parameters requires the estimation of the homography matrix  $H$ . The homography matrix  $H$  related to the positions of the points on the source image plane (model plane) to the points on the destination image plane (imager plane). Given an image of model plane, an homography can be estimated (more detail in [10]).

Assuming that the homography  $H$  is calculated, we will write  $H$  out as column vector,  $H = [h_1 \ h_2 \ h_3]$ , where each  $H$  is a 3-by-1 vector. From Equation (2-17), we get:

$$H = [h_1 \ h_2 \ h_3] = sK[r_1 \ r_2 \ t] \quad (2-18)$$

Reading off these equations, we have:

$$\begin{aligned} h_1 &= sKr_1 \quad \text{or} \quad r_1 = \lambda K^{-1}h_1 \\ h_2 &= sKr_2 \quad \text{or} \quad r_2 = \lambda K^{-1}h_2 \\ h_3 &= sKr_3 \quad \text{or} \quad r_3 = \lambda K^{-1}h_3, \end{aligned} \quad (2-19)$$

where,  $\lambda = 1/s$ .

The rotation vectors are orthogonal to each other (that means the rotation vector's dot product is 0 and the vector's magnitudes are equal), so:

$$\begin{aligned} r_1^T r_2 &= 0 \\ \|r_1\| &= \|r_2\| \quad \text{or} \quad r_1^T r_1 = r_2^T r_2. \end{aligned} \quad (2-20)$$

For any vector  $a$  and  $b$  we have  $(ab)^T = b^T a^T$ , from equations (2-19) and (2-20) we have:

$$\begin{aligned} h_1^T (K^{-1})^T K^{-1} h_2 &= 0 \\ h_1^T (K^{-1})^T K^{-1} h_1 &= h_2^T (K^{-1})^T K^{-1} h_2. \end{aligned} \quad (2-21)$$

## 2.1. Camera Models and Calibration

---

Setting  $B = (K^{-1})^T K^{-1}$ , we have

$$B = (K^{-1})^T K^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-o_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-o_y}{f_y^2} \\ \frac{-o_x}{f_x^2} & \frac{-o_y}{f_y^2} & (\frac{o_x}{f_x^2} + \frac{o_y}{f_y^2} + 1) \end{bmatrix}. \quad (2-22)$$

Because  $B$  is symmetric, it can be written as a reduced vector  $b$  with only 6 terms, thus

$b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$ . The two constrains in Equation (2-21) have the general form

$h_i^T B h_j$  as:

$$h_i^T B h_j = v_{ij}^T b = \begin{bmatrix} h_{i1} h_{j1} \\ h_{i1} h_{j2} + h_{i2} h_{j1} \\ h_{i2} h_{j2} \\ h_{i3} h_{j1} + h_{i1} h_{j3} \\ h_{i3} h_{j2} + h_{i2} h_{j3} \\ h_{i3} h_{j3} \end{bmatrix}^T \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{21} \\ B_{33} \end{bmatrix}. \quad (2-23)$$

Using this definition for  $v_{ij}^T$  the two constrains in (2-21) can be written as:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0. \quad (2-24)$$

Note that the vector  $b$  which summarizes the intrinsic parameters is a 6 element vector so that 6 equations are necessary to recover all camera parameters. Therefore, since each homography provides 2 linear equations, at least 3 homographies or captured images are sufficient. Assuming that we collect  $N$  images of chessboards together, the linear system composed of  $N$  instances of Equation (2-24) can be written as

$$Vb = 0, \quad (2-25)$$

where  $N$  is  $2N$ -by-6 matrix. If  $N \geq 3$ , this linear can be solved by employing the standard technique of Singular Value Decomposition ( $SVD$ ).

Using the computed solution vector  $\mathbf{b}$ , the intrinsic parameters can be derived as follows:

- The camera intrinsic parameters:

$$\begin{aligned}
 f_x &= \sqrt{\lambda/B_{11}} \\
 f_y &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\
 c_x &= -B_{13}f_x^2/\lambda \\
 c_y &= (B_{12}B_{13} - B_{11}B_{22})/(B_{11}B_{22} - B_{12}^2) \\
 &\text{with } \lambda = B_{33} - (B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{22}))/B_{11}.
 \end{aligned} \tag{2-26}$$

- The camera extrinsic parameters:

The rotation matrix and translation vector can be recovered from the Equation (2-19):

$$\begin{aligned}
 r_1 &= \lambda K^{-1}h_1 \\
 r_2 &= \lambda K^{-1}h_2 \\
 r_3 &= r_1 \times r_2 \\
 t &= \lambda K^{-1}h_3,
 \end{aligned} \tag{2-27}$$

where  $\times$  denote a cross product. The scaling parameters are determined by  $\lambda = 1/\|K^{-1}h_1\|$ .

To refined the obtained camera parameters, it may perform an nonlinear minimization of a projection function, which is solved with the Levenberg-Marquardt Algorithm [12]. This function is projecting the  $3D$  points onto the image plane and accumulating the differences between corresponding points as following:

$$\sum_j^N \sum_i^n \left\| p_{ij} - \left( [K \mid 0_3] \begin{bmatrix} R_j & -R_j C_j \\ 0 & 1 \end{bmatrix} P_{ij} \right) \right\|^2, \tag{2-28}$$

where  $j$  is the image index and  $i$  denotes to the point correspondence index.

The recommended calibration procedure can be summarized as follows.

1. Print a chessboard pattern and attach it to a planar surface;
2. Take  $N$  (at least 3) images of the model plane under different orientations by moving either the plane or the camera;
3. Detect the feature points in the images and calculate the  $N$  homography transforms;

## 2.2. Two-view Geometry

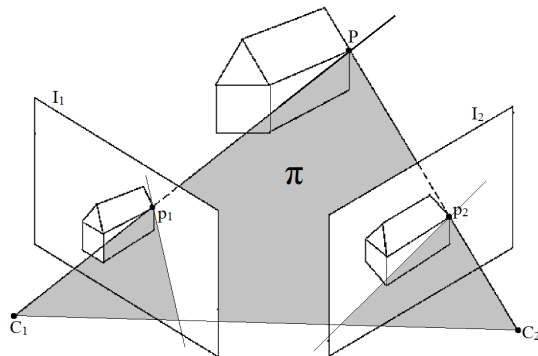
---

4. using the  $N$  homography transforms, calculate the intrinsic and extrinsic parameters;
5. Refine the calculated camera parameters.

### 2.2 TWO-VIEW GEOMETRY

In the previous section, we have introduced the geometry of a single camera. We now describe the case of two camera geometry. The two-view geometry is the intrinsic geometry of two different perspective views of the same 3D scene (see **Figure 2-6**). The two perspective views may be acquired simultaneously, for example in a stereo rig, or sequentially, for example by a moving camera. From the geometric viewpoint, the two situations are equivalent, but notice that the scene might change between successive snapshots.

Most 3D scene points must be visible in both views simultaneously. This is not true in the case of occlusions, i.e., points visible in only one camera. Any unoccluded 3D scene point  $P = (X, Y, Z, 1)^T$  is projected to the left and right view as  $p_1 = (x_1, y_1, 1)^T$  and  $p_2 = (x_2, y_2, 1)^T$ , respectively (see the **Figure 2-6**). Image points  $p_1$  and  $p_2$  are called corresponding points as they represent projections of the same 3D scene point  $P$ .



**Figure 2-6. Two perspective views of the same 3D scene.**

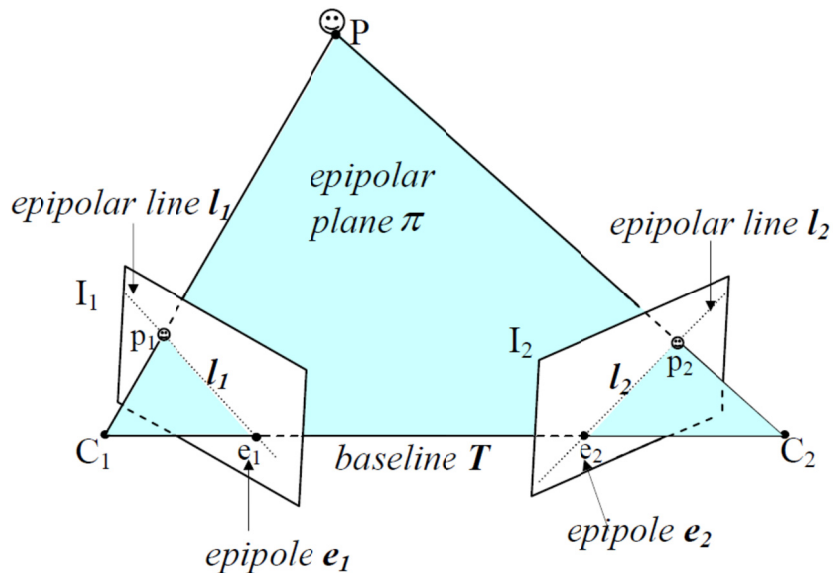
Algebraically, each perspective view has an associated 3-by-4 camera projection matrix  $M$  which represents the mapping between the 3D world and an 2D image. We will refer to the camera projection matrix of the left view as  $M_1$  and of the right view as  $M_2$ . Based on Equation (2-12), we have:

$$\begin{aligned}\lambda_1 p_1 &= M_1 P, \\ \lambda_2 p_2 &= M_2 P.\end{aligned}\tag{2-29}$$

Geometrically, the position of the image point  $p_1$  in the left image plane  $I_1$  can be found by drawing the optical ray through the left camera projection center  $C_1$  and the scene point  $P$ . The ray intersects the left image plane  $I_1$  at  $p_1$ . Similarly, the optical ray connecting  $C_2$  and  $P$  intersects the right image plane  $I_2$  at  $p_2$ .

The knowledge of image correspondences enables scene reconstruction from images. The position of  $P$  is calculated by triangulation of the two corresponding points, using the geometry of the two cameras. The geometry of the two cameras relates to the respective position and orientation and internal geometry of each individual camera. The underlying geometry that describes the relationship between both cameras is known as the *epipolar geometry*. This is discussed in *Section 2.2.1*.

### 2.2.1 Epipolar Geometry



**Figure 2-7. The epipolar and epipolar constraint.**

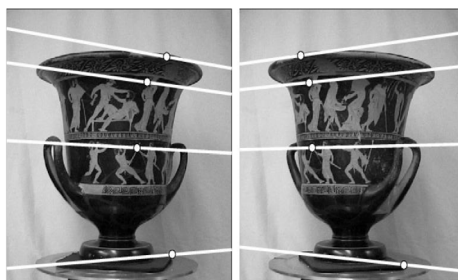
The epipolar geometry describes the geometric relationship between two perspective views of the same 3D scene. The key finding is that corresponding image points must lie on particular image lines, which can be computed without the information on the camera calibration. This implies that,

## 2.2. Two-view Geometry

---

given a point in one image, one can search for the corresponding point in the other image along a line and not in a  $2D$  region, a significant reduction in complexity.

**Figure 2-7** illustrates the rules of the *epipolar geometry*. Any  $3D$  point  $P$  and the camera projection center  $C_1$  and  $C_2$  define a plane that is called the *epipolar plane*. The projections of the point  $P$ , image points  $p_1$  and  $p_2$ , also lie in the epipolar plane since they lie on the rays connecting the corresponding camera projection center and point  $P$ . The corresponding *epipolar lines*,  $l_1$  and  $l_2$ , are the intersections of the epipolar plane with the image planes. The line connecting the camera projection centers  $C_1$  and  $C_2$  is called the *baseline*. The baseline intersects each image plane in a point called *epipole*. By construction, the left epipole  $e_1$  is the image of the right camera projection center  $C_2$  in the left image plane. Similarly, the right epipole  $e_2$  is the image of the left camera projection center  $C_1$  in the right image plane. All epipolar lines in the left image go through  $e_1$  and all epipolar lines in the right image go through  $e_2$ . Therefore, the search of point-correspondences can be limited to a search along the epipolar line instead of an exhaustive search in the image. An example of two views with the computed epipolar lines superimposed onto the images is given in **Figure 2-8**.



**Figure 2-8. Two views of scene with 4 epipolar lines.**

We'll now summarize some facts about stereo camera epipolar geometry:

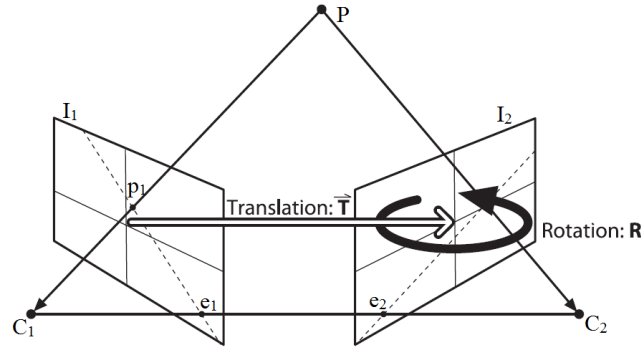
- Every  $3D$  point in view of the cameras is contained in an epipolar plane that intersects each image in an epipolar line;

- Given a feature in one image, its matching view in the other image must lie along the corresponding epipolar line. This is known as the epipolar constraint;
- The baseline is the line going through the two cameras centers;
- The epipole is the image-point determined by the intersection of the image plane with the baseline;
- The epipolar constraint means that the possible two-dimensional search for matching features across two imagers becomes a one-dimensional search along the epipolar lines once we know the epipolar geometry of the stereo rig. This is not only a vast computational savings, it also allows us to reject a lot of points that could otherwise lead to spurious correspondences;
- Order is preserved. If points  $A$  and  $B$  are visible in both images and occur horizontally in that order in one imager, then they occur horizontally in that order in the other imager.

As we pointed out in previous part, each camera is described by a set of extrinsic parameter. They determine placement of a camera in respect to external coordinate system. With each of the cameras of the stereo systems we associate a separate coordinate system with its center coinciding with the central point of the camera. The  $Z$  axis of such coordinate system is collinear with the optical axis of the camera. In both coordinate systems,  $P_1 = (X_1, Y_1, Z_1)$  and  $P_2 = (X_2, Y_2, Z_2)$  represents the same 3D point  $P$ . On the other hand, on the respective image planes,  $p_1 = (x_1, y_1, z_1)$  and  $p_2 = (x_2, y_2, z_2)$  determine two different images of the 3D point  $P$ . We note that  $z_1 = f_1$  and  $z_2 = f_2$ .

The information about translation  $T$  and rotation  $R$ , which describes the changing of the second camera (right camera) to the first (left camera) in world coordinates, is contained in *essential matrix*  $E$  (see **Figure 2-9**). The essential matrix  $E$  obeys the following constraint:

$$(P_2)^T E P_1 = 0 \quad \text{or} \quad (p_2)^T E p_1 = 0. \quad (2-30)$$



**Figure 2-9. The essential matrix  $E$  contains all of the information about translation  $T$  and rotation  $R$ , which describes the location of the second camera relative to the first in world coordinates.**

The matrix  $E$  contains all of the information about the geometry of the two cameras related to each other but no information about the cameras themselves. In practice, we are usually interested in pixel coordinates. In order to find a relationship between a pixel in one image and the corresponding epipolar line in the other image, we will have to introduce intrinsic information about the two camera. For the pixel coordinate  $p$  we substitute  $q$  and camera intrinsic matrix that relates them, that means  $q = Kp$  or equivalent  $p = K^{-1}q$ . Hence, the Equation (2-30) for  $E$  becomes:

$$q_2^T (K_2^{-1})^T E K_1^{-1} q_1 = 0. \quad (2-31)$$

Finally we obtain

$$q_2^T F q_1 = 0, \quad (2-32)$$

where the matrix

$$F = (K_2^{-1})^T E K_1^{-1}, \quad (2-33)$$

is called the *fundamental matrix*. It describes the epipolar geometry in term of pixel coordinates.

By providing a number of known correspondences, we can compute the fundamental matrix  $F$ , and then we can compute the epipolar lines.

### 2.2.2 Stereo Calibration

The problem of stereo calibration consists of determination of the parameters of the two cameras and the geometrical relationship between the two cameras in space.



The parameters of the two cameras can be computed based on the single-camera calibration method in the previous *section 2.1.4*. Now we concern how to find the rotation matrix  $R$  and translation vector  $T$  between the two cameras, as depicted in **Figure 2-9**. The rotation matrix  $R$  describes a relative rotation between coordinate system of the two cameras and the vector  $T$  describes a translation of the two camera centers.

Let us assume that the extrinsic parameters are already known for the two camera of stereo system (this can be done by single-camera calibration method in the previous *section 2.1.4*). There are given by four matrices:  $R_1$  and  $T_1$  for the left camera (left camera center  $C_1$ ),  $R_2$  and  $T_2$  for the right camera (right camera center  $C_2$ ). Using the relation in Equation (2-10), which connects coordinate of a certain 3D point  $P_w$  from world coordinate system with the camera coordinate system, we obtain

$$\begin{aligned} P_1 &= R_1(P_w - C_1) \\ P_2 &= R_2(P_w - C_2), \end{aligned} \quad (2-34)$$

where,  $P_1$  and  $P_2$  are the location of the 3D point  $P_w$  from the coordinate system of the left and right cameras respectively.

After factoring out  $P_w$  from the Equation (2-34), we get

$$P_2 = R_2 R_1^T P_1 + R_2(C_1 - C_2). \quad (2-35)$$

On the other hand, it is evident from **Figure 2-9** that the two matrices  $P_1$  and  $P_2$  are related by

$$P_2 = R P_1 + T. \quad (2-36)$$

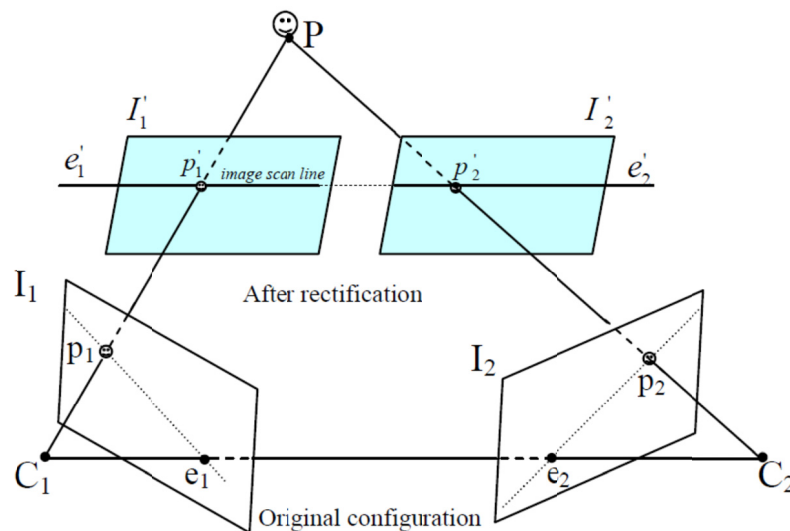
Comparing two Equation (2-35) and (2-36), the following relations can obtains:

$$\begin{aligned} R &= R_2 R_1^T, \\ T &= R_2(C_1 - C_2) = T_2 - R T_1, \end{aligned} \quad (2-37)$$

where,  $R$  and  $T$  are the sought calibration matrices of the stereo system,  $T_i = -R_i C_i$  with  $i \in \{1,2\}$ .

2.2.3 Stereo Rectification

As discussed in previous parts, the three-dimensional structure can be extracted by determining the correspondences in the two views and that the point-correspondences can be searched along the epipolar line only. To simplify the search, people try to capture images such that all epipolar lines are parallel and horizontal. In the perfect case, the search of point correspondences can be performed along the horizontal raster lines of both images. Unfortunately, it is difficult to accurately align and orient the two cameras such that epipolar lines are parallel and horizontal. Instead of physically stereo setups, an alternative approach will be to mathematically align the two cameras into one viewing plane so that pixel rows between the cameras are exactly aligned with each other, as illustrated in **Figure 2-10**. This process is called *image rectification*.



**Figure 2-10. Stereo image rectification. The epipolar lines become collinear and parallel to the image scan-line.**

Some algorithms have been introduced to do image rectification. For example, in [13], authors have introduced an algorithm which performs rectification given a *weakly calibrated* stereo rig, i.e., a rig for which only point correspondence between image are given (or equivalently for which the fundamental matrix could be computed). In case the stereo rig is *calibrated*, i.e., the cameras' parameters, mutual position and orientation are known, Bouguet's algorithm [14] is one of the most well-known method. Given the rotation matrix  $I$  and translation vector ( $T$ ) between the stereo images, Bouguet's algorithm for stereo rectification simply attempts to minimize the amount of

change re-projection produces for each of the two images while maximizing the viewing area. This method can be described as following.

To minimize image re-projection distortion, the rotation matrix  $R$  that rotates the right camera image plane into the left's plane is split in half between the two cameras. We get the two result rotation matrix  $r_1$  and  $r_2$  for the left and right camera, respectively. Each camera rotates half a rotation, so their principal rays each end up parallel to the vector sum of where their original principal rays have been pointing. However, this rotation just puts the cameras into coplanar alignment but not into row alignment.

Now we build the rotation matrix that will take the left camera's epipole to infinity and align the epipolar lines horizontally. This rotation is described by a matrix  $R_{rect}$ , which consists of three mutually orthogonal unit vectors:  $e_1$ ,  $e_2$  and  $e_3$ . Taking the principal point  $o = (o_x, o_y)$  as the left image origin, the vector  $e_1$  is directly along the translation vector between two cameras and is given as:

$$e_1 = \frac{T}{\|T\|}. \quad (2-38)$$

The vector  $e_2$  is orthogonal to the  $e_1$ . Since,

$$[-T_y, T_x, 0] \cdot [T_x, T_y, T_z]^T = 0, \quad (2-39)$$

the  $e_2$  take the form

$$e_2 = \frac{[-T_y, T_x, 0]^T}{\sqrt{T_x^2 + T_y^2}}. \quad (2-40)$$

The third vector  $e_3$  has to be simultaneously orthogonal to the vector  $e_1$  and  $e_2$ . Therefore, it can be found by using the cross product:

## 2.3. Background of Graph Cuts Theory

---

$$e_3 = e_1 \times e_2. \quad (2-41)$$

The rotation matrix is now:

$$R_{rect} = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix}_{3 \times 3}. \quad (2-42)$$

This matrix rotates the left camera so that the epipolar lines become horizontal and the epipoles are at infinity. The row alignment of the two cameras is then obtained by:

$$\begin{aligned} R_1 &= R_{rect} \cdot r_1 \\ R_2 &= R_{rect} \cdot r_2. \end{aligned} \quad (2-43)$$

The matrices  $R_1$  and  $R_2$  are the rectification transform (rotation matrix) for the left and right cameras, respectively, which we want to find.

The resulting rectified image is employed in a lot of stereo camera setups and stereo algorithms as shown in the next chapter.

### 2.3 BACKGROUND OF GRAPH CUTS THEORY

In this thesis, we use graph cut to minimize the energy function in depth estimation (in *Chapter 3*), depth based view synthesis application (in *Chapter 4*) and depth assisted object segmentation (in *Chapter 5*). Thus, in the next parts, let's us introduce briefly graph cuts theory.

Energy minimization is a natural framework for many vision applications. It has several advantages. It allows a clean specification of the problem to be solved, as distinct from the algorithm used to solve it. In addition, energy minimization naturally allows the use of soft constraints, such as spatial coherence. Finally, energy minimization avoids being trapped by early hard decisions.

Solving a problem via an energy minimization consists of two major steps. First, an objective function is formulated. It maps all possible solutions to real numbers, and it shows how good (or bad) a candidate solution is. An objective function is usually a sum of terms corresponding to different constraints of the problem, either soft or hard. The second step of the approach is to minimize the

energy function. This is often a very hard task. Energy functions that arise in vision usually have thousands of dimensions and many local minima.

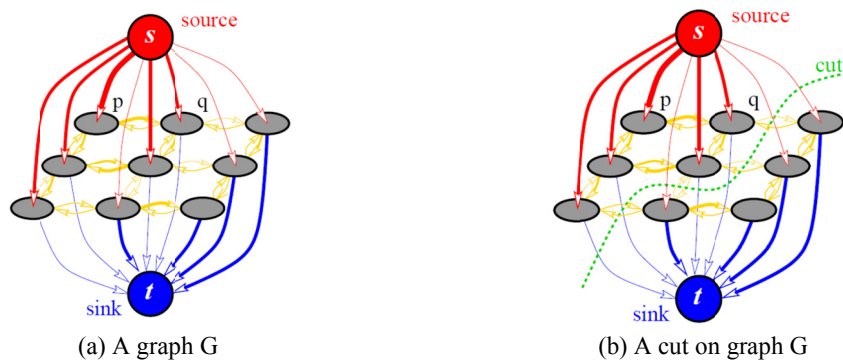
Recently, new fast energy minimization techniques based on graph cuts have emerged. These techniques can be applied to a restricted class of energy functions of discrete variables. An advantage of these methods is that in certain cases they can produce a global minimum of the energy or in other cases a local minimum with some strong properties.

Graph cut theory was first applied in the field of computer vision at the end of 1980s' by Greig et. Al.[15]. In [15], they showed that a certain important energy function in vision can be efficiently solved by powerful min-cut/max flow algorithm. Image restoration was taken as an example by obtaining the maximum a posterior (MAP) estimation of a binary image using the graph cut technique. Unfortunately, the graph cut technique in [15] remained unnoticed for almost 10 years mainly because binary image restoration looked very limited as an application. In the late 90's new computer vision techniques appeared that figured how to use min-cut/max-flow algorithms on graphs for more interesting non-binary problems. The results in [16] showed that iteratively running min-cut/max-flow algorithms on appropriate graphs can be used to find provably good approximate solutions for even more general multi-label case when interaction penalties are metrics. A growing number of publications in vision use graph based energy minimization techniques for applications like image segmentation [17], stereo reconstruction [18], object detection and tracking [19], augmented reality and others.

Graph cut based methods construct a graph topology to minimize the specified energy function activated by the max-flow/min-cut algorithm [20], so that the min-cut on the graph is of minimal energy among all the cuts separating the terminals. Theoretically,  $G = \langle V, E \rangle$  is a graph which consists of a set of nodes  $V$  and a set of edges  $E$  that connect them. The node set  $V = \{s, t\} \cup P$  contains two special terminal nodes, which are called the source  $s$ , and the sink  $t$ , and a set of non-terminal nodes  $P$ . **Figure 2-11(a)** shows a simple example of a graph with the terminals  $s$  and  $t$ . Generally, non-terminal nodes represent pixels (or voxels), and source and sink correspond to the

### 2.3. Background of Graph Cuts Theory

labels assigned to the non-terminal nodes. Each graph edge is assigned some nonnegative weight or cost  $w(p, q)$ . A cost of a directed edge  $(p, q)$  may differ from the cost of the reverse edge  $(q, p)$ . An edge is called a  $t$ -link if it connects a non-terminal node in  $P$  with a terminal. An edge is called a  $n$ -link if it connects two non-terminal nodes. A set of all (directed)  $n$ -links will be denoted by  $N$ . The set of all graph edges  $E$  consists of  $n$ -links in  $N$  and  $t$ -links  $\{(s, p), (p, t)\}$  for non-terminal nodes  $p \in P$ . In **Figure 2-11**,  $t$ -links are shown in red and blue, while  $n$ -links are shown in yellow.



**Figure 2-11. An example of graph construction in [15] (Edge cost are reflected by the thickness).**

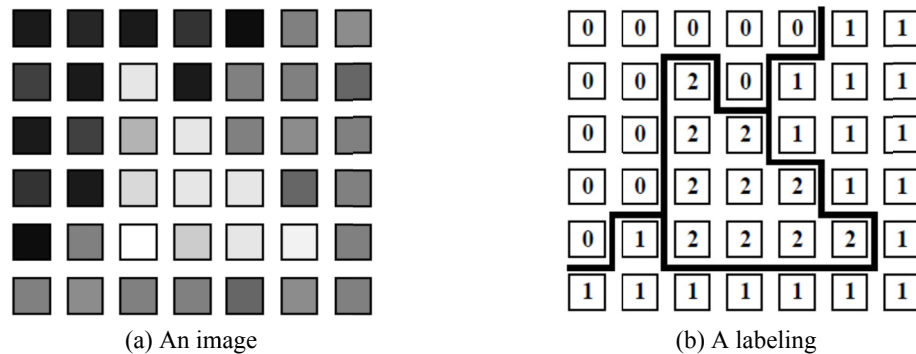
An  $s/t$  cut  $C$  (sometimes it is called a cut) is a partitioning of the nodes in the graph into two disjoint subsets  $S$  and  $T$  such that the source  $s$  is in  $S$  and the sink  $t$  is in  $T$ . **Figure 2-11(b)** shows one example of a cut. The cost of a cut  $C = \{S, T\}$  is the sum of costs/weights of “boundary” edges  $(p, q)$  such that  $p \in S$  and  $q \in T$ . If  $(p, q)$  is a boundary edge, then sometimes it is said that cut  $C$  severs edge  $(p, q)$ . The minimum cut problem is to find a cut that has the minimum cost among all cuts. One of the fundamental results in combinatorial optimization is that the minimum  $s/t$  cut problem can be solved by finding a maximum flow from the source  $s$  to the sink  $t$ . For informal example, maximum flow is the maximum “amount of water” that can be sent from the source to the sink by interpreting graph edges as directed “pipes” with capacities equal to edge weights. The theorem of Ford and Fulkerson [20] shows that a maximum flow from  $s$  to  $t$  saturates a set of edges in the graph dividing the nodes into two disjoint parts  $\{S, T\}$  corresponding to a

minimum cut. Thus, min-cut and max-flow problems are equivalent. In fact, the maximum flow value is equal to the cost of the minimum cut.

The min-cut/max-flow algorithm aims to minimize the energy function over the image labeling. How to define the energy function for efficiently representing the properties of the image becomes the key problem in the graph cut theory as well as its applications. The energies addressed by Greig et. Al. and by most later graph based methods can be represented as a posterior energy in the well-known maximum a maximum a posteriori estimation of a Markov Random Field (MAP-MRF) framework:

$$E(f) = \sum_{p \in P} E_p(f_p) + \lambda \sum_{(p,q) \in N} V_{p,q}(f_p, f_q) \quad (2-44)$$

where,  $f = \{f_p | p \in P\}$  is the labeling of image,  $P$  is the set of pixels, and  $N$  is the pixel's neighborhood system. An example of image labeling is shown in **Figure 2-12**. The first term,  $D_p(f_p)$ , is called the data term and corresponds to the  $t-link$  in the graph. It measures the likelihood of a certain pixel  $p$  assigned to the label  $f_p$  and how well the label  $f_p$  fits the pixel  $p$  given the observed data. The second term,  $V_{p,q}(f_p, f_q)$  (called smoothness term) is represented by the  $n-link$  in the graph. It evaluates the penalty of discontinuities between  $p$  and  $q$  which are assigned with  $f_p$  and  $f_q$  respectively.  $\lambda$  is a parameter to weigh the importance of these two terms.



**Figure 2-12. An example image labeling.**

In **Figure 2-12(a)**, an image is a set of pixels  $P$  with observed intensity  $I_p$  for each  $p \in P$ . A labeling  $f$  shown in **Figure 2-12(b)** assigns some label  $f_p \in \{0,1,2\}$  to each pixel  $p \in P$ . Such

## 2.4. Summary

---

label can represent depth (in stereo), object index (in segmentation), original intensity (in image restoration) or other pixel properties. Normally, the set of possible labels at each pixel is finite. The thick lines in **Figure 2-12(b)** show labeling discontinuities between neighboring pixels.

Graph cut is not limited to solving the regular binary-label segmentation problem, but is also applicable to multi-label energy minimization. In [16], a multi-way cut based on expansion move ( $\alpha$  expansion or  $\alpha - \beta$  swap) is proposed to handle the multi-value labeling by repeatedly minimizing the energy function containing three or fewer binary variables. A fast algorithm for graph-cut optimization including expansion move with source code is implemented and public in [21], which drives the extensive application of graph cut technique in various optimization problems.

## 2.4 SUMMARY

In this chapter, the projective geometry has been introduced firstly which uses homogenous coordinates to describe the position of  $2D$  and  $3D$  points. Employing the homogenous coordinates, we show that the projection of  $3D$  points onto the  $2D$  image plane can be defined by a linear projection matrix. This projection matrix can be decomposed into intrinsic and extrinsic camera parameters. The intrinsic matrix  $K$  comprises the internal parameters of camera such as focal length principal point and skew parameter. The extrinsic matrix  $[R | T]$  indicates the external position and the orientation of the camera in the  $3D$  world coordinate. We also present how a  $2D$  point can be back projected to the  $3D$  space, which is important for depth estimation and view synthesis in later chapter in this thesis. Then, the calibration technique using a planar chessboard pattern that enable the estimation of these camera parameters has been presented. After introducing the geometry of a single camera, we have introduced the case of the two view geometry which describes the geometry relationship between two images. The key relationship is that corresponding image points must lie on particular image lines, i.e. epipolar lines. To simplify the search of point correspondences in the two views, we have described the stereo calibration and stereo rectification which are particularly useful to estimate the  $3D$  structure of a scene addressing in the next chapter. The stereo calibration consists of determination of the parameters of the two cameras and the geometrical relationship between the



two cameras in space. In contrast, stereo rectification is the process of correcting the individual images so that they appear as if they had been taken by two cameras with row-aligned image planes. Finally, the graph cut is briefly introduced which is used to minimize our energy functions in the in later chapters.

## 3

# DEPTH ESTIMATION

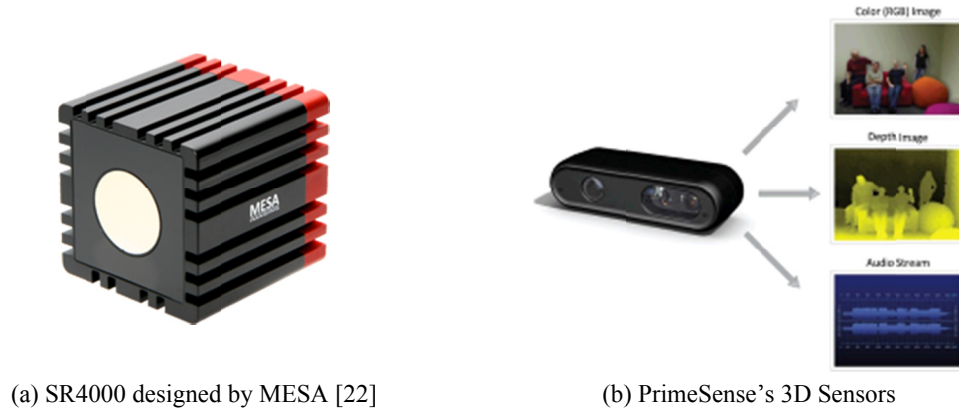
### 3.1 INTRODUCTION

From the earliest inquiries into visual perception, it was known that we perceive depth based on the difference in appearance between left eye and right eye. Under simple imaging configuration (both eyes or cameras looking straight ahead), the amount of horizontal motion or disparity is inversely proportional to distance from the observer (as we will see in later section). While the basic physics and geometry relating visual disparity to scene structure are well understood (*section 2.2.1*), automatically measuring this disparity by establishing dense and accurate inter image correspondence is a challenging task.

An *2D* image captured by a traditional single camera loses depth information of the *3D* scene, which is a useful and important cue for perception and visualization of the real world environment. Since depth information provides the users with *3D* feeling of the scene, depth acquisition and reconstruction have been attracted extensive attentions in the industrial and research communities. In general, depth of the scene can be obtained by two approaches: *active methods* by measuring depth directly from digital device such depth camera and time flight (*ToF*) camera; *passive methods* by estimating depth from the captured images in a computational way.

Recently, with the demand of entertainment and progressive development of digital device, camera capable of generating *3D* models has been emerged. *3D* depth camera capturing video with the depth information in real-time is being widely used in *3D* broadcasting and virtual reality system to provide the user with interactive and realistic experience of *3D* world. Many kinds of *3D* camera products currently are available on the market, especially a new class of active depth sensing system based on the time-of-flight (*ToF*) principle such as SR4000 designed by MESA [22]. *ToF* cameras are active sensors that determine the per-pixel depth value by measuring the time taken by infrared light to travel to the object and back to camera. PrimeSense's *3D* Sensors [23] is a digital

device manufactured by a fables semiconductor company. It works by coding the scene with near-IR light, which is invisible to the human eye. The solution then used a standard off the shelf *CMOS* image sensor to read the coded light back from the scene. This is the process that enables depth acquisition and makes PrimeSense’s solutions accurate. **Figure 3-1** shows an example of *3D* depth camera products.



**Figure 3-1. An example of 3D depth camera products.**

Even though the depth camera provide a convenient and straightforward way for acquisition of depth dimension, the cost of system increases by introducing depth camera with additional price. However, their accuracy use to be much higher and some of them are used to obtain the ground truth.

In this chapter, we mainly focus on passive methods, which will reconstruct depth from multiple images/views of the same scene using computational algorithms. A comprehensive review and our depth estimation algorithm will be discussed in the following sections.

### 3.2 THE FORMULATION DEPTH FROM STEREO

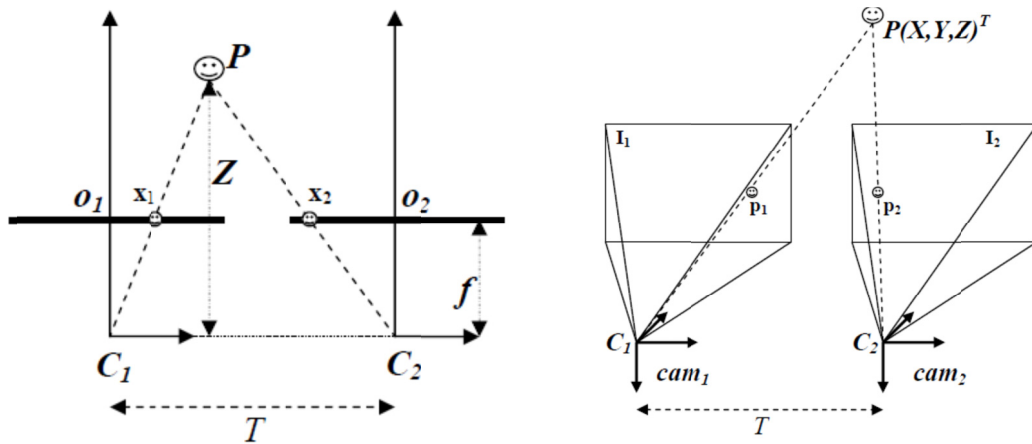
Assume that we have a perfectly undistorted, aligned and measured stereo rig as shown in **Figure 3-2**: two cameras whose image plane are exactly coplanar with each other, with exactly parallel optical axes that are a known distance part and with equal focal length  $f_1 = f_2$ . Also, assume that the principle point  $O_{1x}$  and  $O_{2x}$  have been calibrated to have same pixel coordinate in their respective left and right image. Let’s further assume that images are row aligned and that every pixel row of one camera aligns exactly with the corresponding row in the other camera (remembering that the process of rectification (in *section 2.2.3*) is how we get things done mathematically when these assumptions

### 3.2. The Formulation Depth From Stereo

are not physically true). We will also assume that we can find a point  $P$  in the physical world in the left and the right image at  $p_1$  and  $p_2$ , which will have the respective horizontal coordinate  $x_1$  and  $x_2$ . In this simplified case as shown in **Figure 3-2(a)**, we can derive the depth  $Z$  by using similar triangles as:

$$\frac{T - (x_1 - x_2)}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{fT}{(x_1 - x_2)} = \frac{fT}{d}. \quad (3-1)$$

The Equation (3-1) shows that the depth is inversely proportional to the disparity between these views, where the disparity is defined simply by  $d = x_1 - x_2$ .



(a) with a perfectly undistorted, aligned stereo rig      (b) two aligned camera capturing rectified images

**Figure 3-2. The restricted case of depth estimation.**

In the previous chapter, we have introduced the fundamentals of multi-view geometry that model the projection of a 3D point onto the 2D image plane. Now, we employ the geometry of multiple views to solve the inverse problem of estimating the 3D position (depth) of a point using multiple 2D images. The previous assumptions can be considered as the restricted case of two rectified views (see **Figure 3-2(b)**). Without loss of generality, the world coordinate system is selected such that it coincides with the coordinate system of camera 1. In such a case, it can be deduced that  $C_1 = 0_3$  and  $R_1 = I_{3 \times 3}$ . Since images are rectified, both rotation matrices are equal:  $R_1 = R_2 = I_{3 \times 3}$ . Camera 2 is located on the  $X$  axis so  $C_2 = (T, 0, 0)$ . Finally, both cameras are identical, so that the internal camera parameter matrices are equal, leading to

$$K = K_1 = K_2 = \begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 0 \end{pmatrix}. \quad (3-2)$$

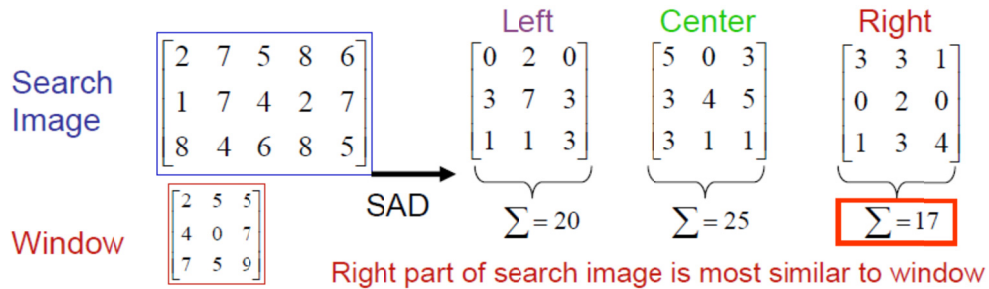
According to the Equation (2-13), we have

$$\lambda_1 \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \text{and} \quad \lambda_2 \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - K \begin{pmatrix} T \\ 0 \\ 0 \end{pmatrix}. \quad (3-3)$$

By combining both previous relations, it can be derived that

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 - \frac{fT}{Z} \\ y_1 \end{pmatrix}. \quad (3-4)$$

Again the Equation (3-4) provides the relationship between two corresponding pixels and depth  $Z$  of the  $3D$  point for the case of two rectified views. It is the identical the relationship we obtained in Equation (3-1).



**Figure 3-3. An example of Sum of Absolute Different (SAD).**

According to the above described restricted case, a simple depth estimation algorithm can be shown as follows. The left and right rectified images are denoted by  $I_1$  and  $I_2$ , respectively. To estimate depth, it is necessary to find the correspondence point  $(p_1, p_2)$  for each pixel. Selecting the pixel  $p_1$  as a reference, we can find the pixel  $p_2$  that corresponds to pixel  $p_1$  along the epipolar line (as shown in *Chapter 2*). Since the images are rectified, we can search the point  $p_2$  along the same row (same  $y$  coordinate) as the point  $p_1$  in the left image. The similarity between pixels  $p_1$  and  $p_2$  is

### 3.3. Previous Works on Depth Estimation

---

measured using the matching window block ( $W$ ) surrounding pixels. Block matching works by using the Sum of Absolute Different (SAD) (see an example in **Figure 3-3**). To limiting the length of a search, the minimum disparity  $d_{\min}$  (the search should start) and the maximum disparity  $d_{\max}$  are defined (usually  $d_{\min} = 0$ ). The disparity  $d$  of a pixel  $p_1 = (x, y)$  in the left view  $I_1$  can be find by

$$d(x, y) = \arg \min_{d_{\min} \leq \tilde{d} \leq d_{\max}} \sum_{(i, j) \in W} |I_1(x+i, y+j) - I_2(x+i-\tilde{d}, y+j)|. \quad (3-5)$$

Repeating the process shown in Equation (3-5) for each pixel in  $I_1$ , a dense disparity maps is obtained. From the obtained disparity map, we can get the depth image by using the relationship in Equation (3-1). This is a very simple estimation depth algorithm, however the estimated disparity values is inaccurately. For example, when capturing two images with two different cameras, the contrast settings and illumination may differ. This results in different intensity levels across the views yielding unreliable matches.

The estimation of depth using correspondence point is challenging problems in many situations such as in texture-less region, in occluded regions or change of illumination across views. In the next section, we are review the state of art work on depth estimation from literature.

## 3.3 PREVIOUS WORKS ON DEPTH ESTIMATION

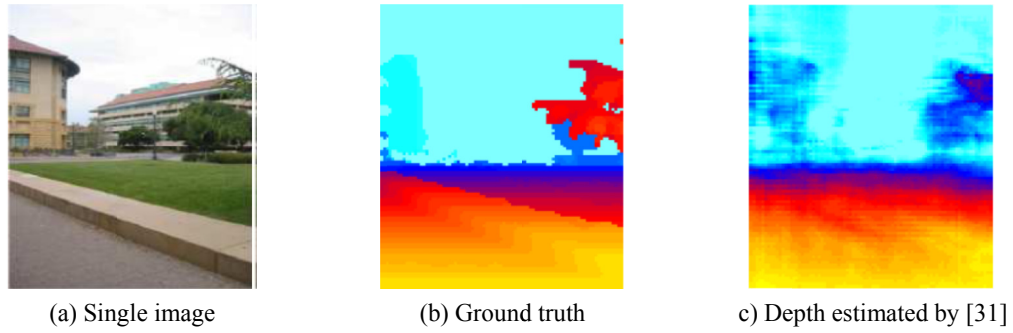
### 3.3.1 *Single Image/Video Depth Estimation*

With the recent explosive for  $3D$  media contents, converting existing single still images or monocular videos to  $3D$  contents is a problem of considerable practical interest. In the following part, we will briefly review the research works of this topic including depth estimation from a single still image and depth estimation algorithms used in  $2D$  to  $3D$  video conversion.

Depth estimation from a single still image is a difficult task, since depth typically remains ambiguous given only local image features. Some works try to solve this problem by using single using monocular cues such as texture variations, texture gradients, occlusion, haze, defocus [24, 25]. For example, many objects' texture will look different at different distances from the viewer. Texture

gradients, which capture the distribution of the direction of edges, also help to indicate depth. Haze is another depth cue, and is caused by atmospheric light scattering. There are some algorithms that can perform depth reconstruction from single images in very specific settings. For example, in [26] authors performed surface reconstruction from single images for known, fixed, objects such as hands and faces. Methods such as shape from shading [27] and shape from texture [28] generally assume uniform color and/or texture, and hence would perform poorly on the complex, unconstrained, highly textured images that we consider. In [29], authors reconstructed high quality *3D* models from several images, but they required that the images also contain “assistant” objects of known shapes next to the target object. In some recent works [30, 31], they have presented the algorithms from monocular image features. For example, in [31], they use a hierarchical, multi-scale Markov Random Field (MRF) that incorporates multi-scale local and global image features, and models the depths and the relation between depths at different points in the image. They divide the image into small rectangular patches, and estimate a single depth value for each patch. They use two types of features: absolute depth features – used to estimate the absolute depth at a particular patch – and relative features, which we use to estimate relative depths (magnitude of the difference in depth between two patches). These features try to capture two processes in the human visual system: local feature processing (absolute features), such as that the sky is far away; and continuity features (relative features), a process by which humans understand whether two adjacent patches are physically connected in *3D* and thus have similar depths. They chose features that capture three types of local cues: texture variations, texture gradients, and color.

**Figure 3-4** shows an example of depth estimated from single still image by algorithm [31]. Although, there are many studies trying to study depth from a single image, it still remains a challenging problem.



**Figure 3-4. An example of depth estimated from single still image [31].**

The  $3D$  stereo contents are not rich enough but there are still large numbers of  $2D$  videos exist in different compressed formats. If they are converted to  $3D$  videos, this will offer a more realistic sense of the scene to the viewer. That is motivation for research in converting  $2D$  to  $3D$  video. The main purpose of the  $2D$ -to- $3D$  video conversion is to generate the second view video based on the content of the  $2D$  video, which involve two processes: (1) Depth Estimation and (2) Depth Image Based Rendering (DIBR) (see detail in *Chapter 4*). There are three commonly used depth estimation methods from  $2D$  to  $3D$  conversion applications. “Depth from blur” [32] using the focuses information of the capture camera to generate the depth map. The depth information is estimated based on the amount of blur of the object. “Vanishing Point based Depth Estimation” [33] based on the vanishing point that is the farthest point of the whole image to get the depth map. “Depth from Motion Parallax” [34] is based on the fact that objects with different motions usually have different depths. For example, near objects move faster than far objects and so relative motion can be used to estimate the depth map. This method is widely used for the depth estimation in  $2D$ -to- $3D$  video conversion. The motion information can be obtained by block matching algorithm between two consecutive frames. The relative depth information is calculated by

$$D(i, j) = \lambda \sqrt{MV(i, j)_x^2 + MV(i, j)_y^2}, \quad (3-6)$$

where,  $MV(i, j)_x$ ,  $MV(i, j)_y$  are the motion vectors corresponding to the  $X$  and  $Y$  axis direction.

$D(i, j)$  is the depth information at pixel  $p(i, j)$  and  $\lambda$  is a scale factor.



### 3.3.2 Stereo Depth Estimation

In computer vision, the topic of stereo matching has been one of the most widely studied and fundamental problems and continues to be the most active research areas. Most stereo matching algorithms today focus on dense correspondence, since this is required for applications such as depth image based synthesis (detail in *Chapter 4*), modeling and some other depth based applications (object segmentation in *Chapter 5*, multiple objects tracking in *Chapter 6*). Now, we review the dense correspondence algorithms based on the taxonomy and categorization scheme proposed by Scharstein and Szeliski [35]. In [35], the authors have introduced a set of algorithmic blocks from which a large set of algorithms can be constructed. It is based on the observation that most of stereo algorithms perform some subset of following components:

1. matching cost computation or cost function;
2. support of cost aggregation;
3. disparity calculation and optimization;
4. Post processing for refining disparity map.

For example, the introduced simple algorithm in section 3.2 can be broken down into step 1, 2, 3 as:

1. the matching cost is the absolute different of intensity values at a given disparity;
2. the support of the matching function is done by summing of matching cost over square window with constant disparity;
3. Disparity is calculated by selecting the minimal aggregated value at each pixel.

#### 3.3.2.1 Matching Cost

The first component of any dense stereo matching algorithms is matching cost or cost function that measures correlation or similarity between pixels in order to determine how likely they are to be in correspondence.

### 3.3. Previous Works on Depth Estimation

---

The most common pixel based matching costs include, among others, absolute intensity different (Sum of Absolute Different –SAD, example in **Figure 3-3**), squared intensity different (Sum of Squared Different – SSD), Cross Correlation (CC), Normalized Cross Correlation (NCC).

#### 3.3.2.2 *Support of the Cost Aggregation*

The second component of dense stereo matching algorithms is the support of the matching cost function. These include single-pixel windows [36], square windows, adaptive windows [37], shiftable windows [38-40]. Typically, to obtain a reliable matching metric, a large region support should be used. However, whereas using a large window provides a reliable matching support at an object surface with smoothly varying depth, an unreliable support is obtained at object boundaries. To solve this problem, a segment-based approach [41, 42] decomposes the image into a sufficiently large number of object segments (using for example a color segmentation technique) and assumes that a single depth value is computed for each segment. Therefore, as the segmentation would follow the shape of objects, an advantage of such a segment-based approach is that the depth is more accurately estimated at the boundaries of objects.

#### 3.3.2.3 *Disparity Calculation and Optimization*

The most important component is the disparity calculation and optimization strategy. Accordingly the stereo matching algorithms can be categorized into local and global methods.

Local approaches calculate independently the disparity of each pixel using the single matching cost of the selected pixel, and implicitly make smoothness assumptions by aggregating support. The local methods emphasize the matching cost definition and cost aggregation steps. The final disparities are computed by simply choosing the disparity at each pixel associated with the minimum cost value. Thus, these methods perform a local “winner-take-all” (WTA) optimization at each pixel. Local optimizations typically yield accurate disparity estimates in textured regions. However, large texture-less regions tend to produce fuzzy disparity estimates.

In contrast, global approaches explicitly enforce the smoothness assumptions of disparity field and determine all the disparities simultaneously by applying global cost minimization using various

optimization techniques. Global methods perform almost all of their work during the disparity computation phase and often skip the aggregation step. For most of global methods, firstly the global energy needs to be defined. The objective is to find a disparity function  $d$  that minimizes a global energy. The general global energy is defined as

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d). \quad (3-7)$$

The data term,  $E_{data}(d)$ , measures how well the disparity function  $d$  agrees with the input image pair. The smoothness term,  $E_{smooth}(d)$  encodes the smoothness assumptions made by the algorithm. To make the optimization computationally tractable, the smoothness term is often restricted to only measuring the differences between neighboring pixels' disparities.

Once the global energy has been defined, a variety of algorithms can be used to find a minimum such as simulated annealing [43], dynamic programming [44], belief propagation [45] or graph-cut [18, 46]. The objective is to find a disparity function that minimizes a global energy, where various constraints are applied to reduce the uncertainties of disparity map.

Since new stereo matching algorithms continue to be introduced, however according to recent advances [47], region-based stereo methods are more favored due to their better disparity smoothness regularization. For optimization strategies, global approaches in general produce more accurate disparity map, while local approaches have slightly poor results, and tend to produce outliers in the homogenous area, blur disparity at discontinuous boundary and match failure in occlusion, but is superior with respect to computational complexity.

### 3.3.3 Multi-view Depth Estimation

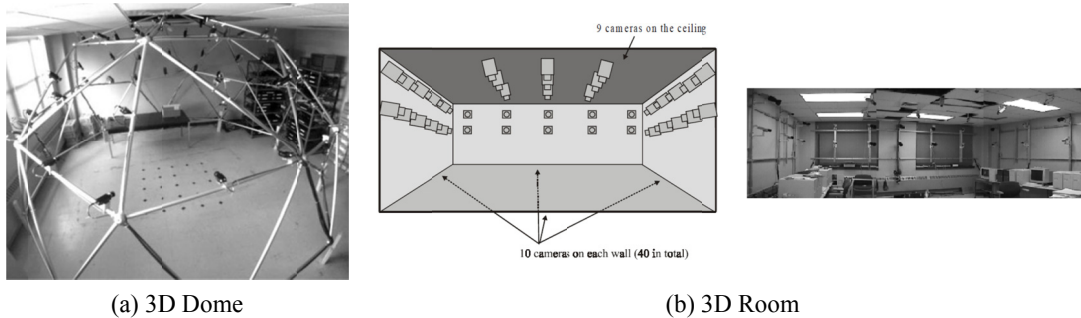
#### 3.3.3.1 Multi-camera System

In the recent years, with the fast improvement of the capability of personal computers and digital equipment, more and more multi-camera systems with dense or sparse, wide-baseline or narrow-baseline camera configuration become available, which significantly broaden the multi-view

### 3.3. Previous Works on Depth Estimation

---

applications and enhance the user experience. In this part, we introduce some of among multi-camera, which have been developed by researchers so far.

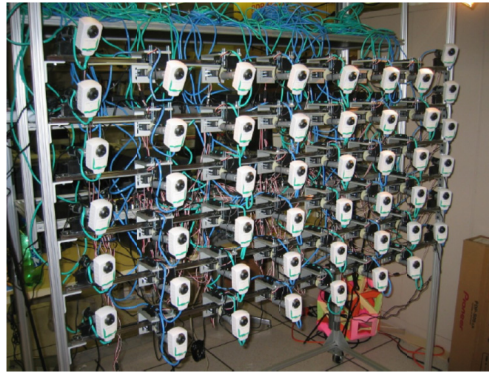


**Figure 3-5. Multi-view system in Virtualized Reality Project developed by CMU [48].**

The Virtualized Reality Project, started and developed by the Carnegie Mellon University (CMU) Robotics Institution in 1994, is considered to be the pioneering work in this field. Their virtualization setup, called the *3D-Dome* [48], is probably the first integration of synchronized cameras at multiple viewpoints. The *3D-Dome* (as shown in **Figure 3-5(a)**), consists of 51 cameras mounted on a 5 meter diameter geodesic dome. The cameras looked at the center of the dome and had a volume of intersection close to 3m x 3m x 2m. It uses consumer *VCRs* to record the synchronized video from each monochrome, analog charge-coupled device (*CCD*) camera. The system used lenses with 3.6 mm focal length for a field of view close to 90 degrees. The resolution of each camera is  $512 \times 512$  and the capture rate is 30 frames per second (*fps*). In 1998, they have constructed the second generation, called the *3D-Room* [49]. The *3D-Room* is 6.1(L)\*6.1(W)\*2.7(H) meters. As shown in **Figure 3-5(b)**, 49 cameras are distributed inside the room: 10 cameras are mounted on each of the four walls, and 9 cameras on the ceiling. It makes use of 49 synchronized color S-Video cameras to capture the 640x480 video at 30 *fps*. The computing system of the *3D-Room* is composed of a control *PC* and 17 digital *PCs*. By now, this project has been developed to the third generation, *3D-Cage*. It has 48 cameras controlled by 25 *PCs* mounted on the gird of all walls. It can continuously capture full color images with 640x480 resolution at 30 *fps* for over 2 hours.

In [50], they presented a self-reconfigurable camera array system that can interactively capture and render 3D virtual scene. This camera system, as shown **Figure 3-6**, is composed of 48 (8x6) Axis

205 network cameras located on 6 linear guides. It can capture  $640 \times 480$  videos at up to 30 fps. The cameras have built-in HTTP servers to respond to HTTP request and send out motion JPEG sequences. The most distinguishing characteristic of the system is its reconfiguration because the cameras are mounted on a mobile platform.



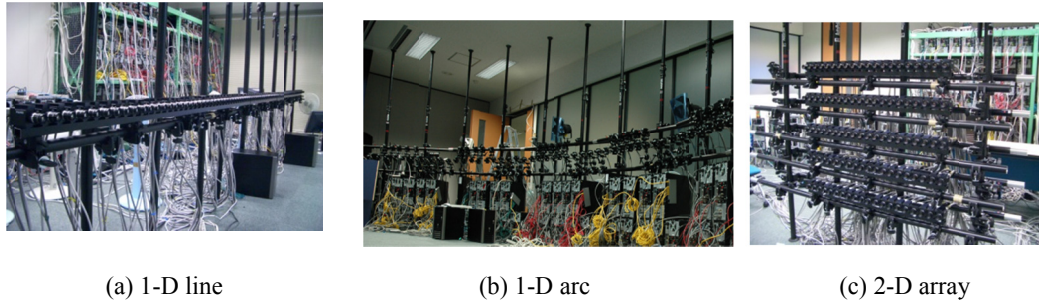
**Figure 3-6. A self-reconfigurable camera array system with 48 cameras [50].**

3DTV and Free viewpoint Television (*FTV*) are the new forms of media communication representing very important multi-view applications. An example of the multi-view image capturing system for *FTV* application developed at Nagoya University [51] is shown in **Figure 3-7**. The system consists of one host server *PC* and 100 client *PCs* (called ‘nodes’) that are equipped with JAI PULNiX TM-1400CL cameras. The interface between camera and *PC* is Camera-Link. The system has one sync-generator that generates a synchronization signal and the sync signal is distributed to all the nodes. The system is able to capture 100 synchronized high-resolution video signals at 30fps. They captured video sequences with different camera arrangements. The first configuration is a 1-D line arrangement (see the **Figure 3-7(a)**), in which 100 cameras are aligned in a line with the camera interval 5cm. Hence, the viewing zone is 5 meters in length. The orientation of camera is set so that the optical axis of each camera is converged to one reference point near object. The second system configuration is a half-round camera arrangement (see the **Figure 3-7(b)**), in which 100 cameras are set in half-circle shape with radius 450cm. In this case, the camera interval is 15.7cm. Camera orientation is set to ‘converged’ in a same way as the 1-D line arrangement. The third system configuration is 2-

### 3.3. Previous Works on Depth Estimation

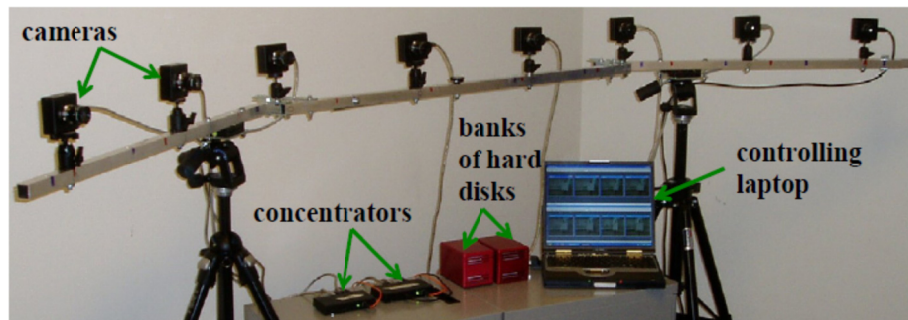
---

D array camera arrangement (see the **Figure 3-7(c)**), in which 100 cameras are aligned in 20(H) x 5(V) in camera interval 5cm and 20cm, respectively. The optical axes were set to parallel in this case.



**Figure 3-7. “100-camera system” at Nagoya University with different camera arrangement: 1-D line, 1-D arc, and 2-D array [51].**

Apart from these large camera arrays with tens of cameras, some multi-view capturing systems adopt sparse configuration with small number of cameras to capture the images for certain applications. The multi-camera system [52] developed by Interactive Visual Media Group in Microsoft Research uses an 8-camera array to capture the videos off-line. Eight cameras are placed along a 1D arc spanning about  $30^{\circ}$  from one end to the other as shown in **Figure 3-8**. The images are captured with high resolution 1024×768 at 15 fps by high-quality PtGrey color cameras with 8mm lenses. To handle real-time storage of all the input videos, they commissioned PtGrey to build us two concentrator units. Each concentrator synchronizes four cameras and pipes the four uncompressed video streams into a bank of hard disks through a fiber optic cable. The two concentrators are synchronized via a FireWire cable.



**Figure 3-8. A configuration of MSR with 8 cameras [52].**

### 3.3.3.2 *Literature Review for Multi-view Depth Estimation*

As introducing in the above *sub-section (3.3.3.1)*, with the growing capability of capturing devices, multi-view capture system with dense or sparse camera array can be built with ease, which motivates the development of multi-view techniques and its related applications.

Multi-view depth map estimation has received more attention so far and resulted in a number of methods with different accuracy and complexity. In [46], a graph model of multi-view stereo images was proposed with the visibility constraint. The graph model used all the pixels of the multi-view images as nodes in the graph. Because the graph model exploited all possible interactions among all the pixels of the multi-view images, it made huge improvement. However, the graph was getting bigger as the number of multi-view images increased. Because of the property, computational times of the graph increased as proportional to the number of multi-view images. In [53], depth map for each view is estimated by conventional algorithm and then the total error minimization process using the graph cut algorithm is used to obtain simultaneously the depth images at three viewpoints. This method gives good results but needs to use conventional algorithm to estimate depth map for each view and long computation time because of graph cut algorithm. Some people use belief propagation to estimate depth maps. In [54], depth maps are obtained by segmentation and then belief propagation is used for refining step. Belief propagation and camera optical flow are used in [55], where depth maps are de-noised. Although depth estimation based on graph model and belief propagation yields an accurate depth map, they are computationally complex and difficulty in real time implementation. In [52] multiple images are also utilizing for the estimation of depth map of each view. Multi stage segmentation method is used to estimate the depth maps. Each independently segmented color texture image was followed by computing and iteratively refining the disparity space distribution for each segment and, finally, image matting is used by computing the alpha values for pixels along the disparity discontinuities. In [56], pairs of images among multiple views are used to estimate depth map by stereo algorithms and then the depth maps are regularized in a filtering step and merged for the final depth map. Because depth image are estimated pairwise, the consistency of depth estimates across the views is not enforced. Another depth estimation method proposed in [57]

### 3.4. Experimental Example: Stereo Calibration, Rectification and Stereo Correspondence


is based on stereo matching. The bi-directional disparity search method and regularization to a spatial neighborhood of pixels are applied. And the energy function is minimized by graph cuts algorithm. The resulting computation is less complexity. However, the generated depth maps can be inconsistent across viewpoints since the inter-view is not fully exploited. Another point is that all above papers are working with rectified images.

#### 3.4 EXPERIMENTAL EXAMPLE: STEREO CALIBRATION, RECTIFICATION AND STEREO CORRESPONDENCE

In the previous sections, we have introduced the mathematic relationship behind the camera calibration and stereo rectification process. In this section, we will show our experimental results of these processes.

We use the low-cost Minoru 3D webcam [58] for our experiment. The specifications of this webcam are provided in **Table 3-1**. Our implementation use C++ and the OpenCV's Libraries [59].

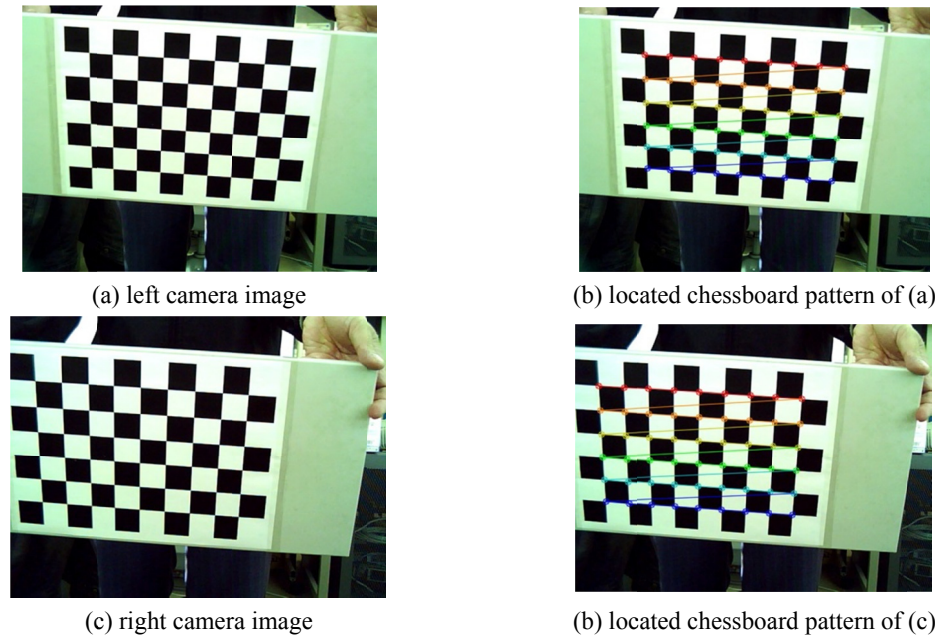
**Table 3-1. Minoru 3D webcam specification.**

	Interface:	USB 2.0
	Max Resolution:	800x600
	Maximum frame rate:	30 fps
	Horizontal FOV	42 <sup>0</sup>
	Manual focus	from 10cm to infinity
	VGA CMOS Sensor	
	Indoor use only since the aperture is always open	

- ***Stereo calibration***

To calibrate a camera, OpenCV proposed to use a chessboard pattern to generate the set of 3D scene points. This pattern creates points at the corner of each square and since this pattern is flat we can assume that the chessboard is located at  $Z = 0$  with the  $X$  and  $Y$  axes well aligned with the grid. The performing calibration includes two steps: chessboard image capture and calibration calculation





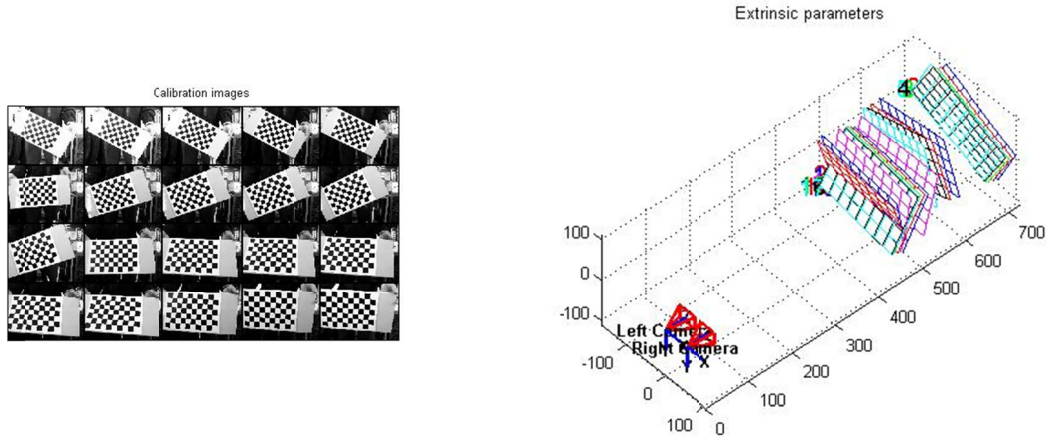
**Figure 3-9. An example of our Chessboard calibration Technique.**

Printed of the chessboard pattern onto a flat surface, our implementation includes follows steps:

- Providing details about chessboard pattern: number of inner vertical and horizontal squares and the size of a square.
- Holding the chessboard pattern in front of the webcam so that the chessboard is visible in both views (see the **Figure 3-9(a)** and **Figure 3-9(c)**).
- Detecting the corners of showed chessboard pattern (see the **Figure 3-9(b)** and **Figure 3-9(d)**). It is nice that OpenCV has functions to do this work automatically with sub-pixel accuracy (*findChessboardCorners()* and *cornerSubPix()*). When a set of chessboard corners has been successfully detected, these points are added to our vector of image and scene points.
- Once a sufficient number of chessboard images have been processed, we can initiate the computation the calibrate parameters by using OpenCV's function *stereoCalibrate()*. In [2], authors suggest that using enough number of chessboard image so that  $2NK \geq 6K + 4$  holds, where  $N$  = number of corners and  $K$  = number of images, should be sufficient to get an accurate calibration as long as the chessboards are taken from different viewpoints at different depths. In practice, 10 to 20 chessboard images are sufficient. In our implementation, we took 20 images

### 3.4. Experimental Example: Stereo Calibration, Rectification and Stereo Correspondence

and the spatial configuration of the two cameras and the calibration planes are being displayed in a form of a 3D plot in **Figure 3-10**.



(a) Calibration images

(b) Captured calibration planes in a form of a 3D plot

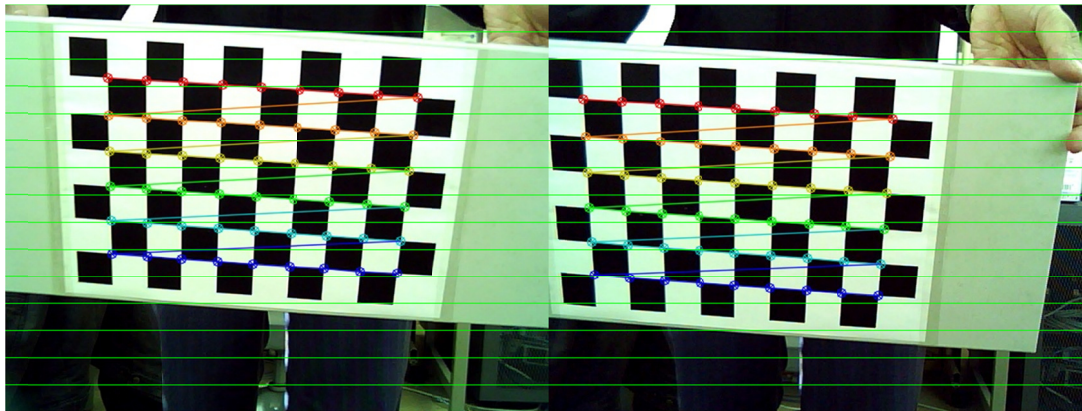
**Figure 3-10. The spatial configuration of the two cameras and the calibration planes.**

The outputs of our calibration process are the intrinsic parameters, distortion coefficients of the left and right cameras, respectively. We also find out the rotation matrix and translation vectors relating the right camera to the left camera. These outputs are shown in **Table 3-2**.

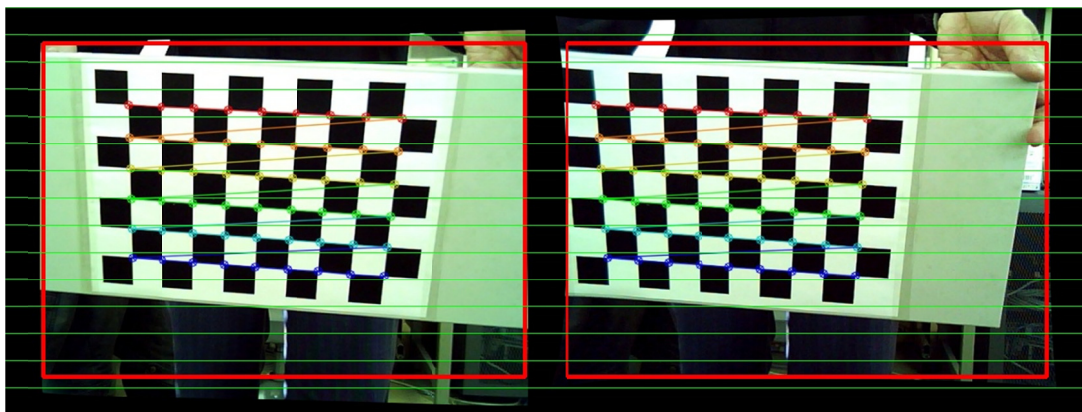
**Table 3-2. Calibration parameters of Minoru 3D webcam.**

	Left camera	Right camera
Intrinsic parameters	$K_{left} = \begin{bmatrix} 853.493 & 0.0 & 336.82 \\ 0.0 & 853.493 & 212.342 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$K_{right} = \begin{bmatrix} 853.493 & 0.0 & 346.534 \\ 0.0 & 853.493 & 219.953 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$
Distortion	$D_{left} = [-0.0786 \quad -0.6181 \quad 0.0 \quad 0.0 \quad 0.0]$	$D_{right} = [-0.1567 \quad 1.2356 \quad 0.0 \quad 0.0 \quad 0.0]$
Extrinsic parameters (position of right camera with respect to left camera)	$R_{left} = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$R_{right} = \begin{bmatrix} 0.9999 & -0.0028 & 0.013 \\ 0.0025 & 0.9997 & 0.0219 \\ -0.013 & -0.0219 & 0.9997 \end{bmatrix}$
Translation vector	$T_{left} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$	$T_{right} = \begin{bmatrix} -5.3977 \\ -0.0432 \\ -0.3967 \end{bmatrix}$

- *Stereo Rectification*



(a) Original left and right images pair



(b) Rectified left and right image pair

**Figure 3-11. Stereo rectification: (a) original pair; (b) rectified pair; note that the barrel distortion (in (a)) has been corrected and the scan lines are aligned in the rectified images.**

As discussed in previously, it is easiest to compute the stereo disparity when the two image planes align exactly. However, a perfectly aligned configuration is rare with a real stereo system, since the two cameras almost never have exactly coplanar and row-aligned image planes. Instead of physically stereo setups, an image rectification approach will be to mathematically align the two cameras into one viewing plane so that pixel rows between the cameras are exactly aligned with each other. We have introduced the Bouguet's rectification algorithm in *section 2.2.3* and it is luckily that this algorithm was implemented in OpenCV.

Applying the Bouguet rectification method, we obtain the results of stereo undistortion and rectification of a stereo pair of images as shown in **Figure 3-11**.

### 3.4. Experimental Example: Stereo Calibration, Rectification and Stereo Correspondence

- **Generation of Disparity Image**

Once the calibration and rectification of the camera system is done, the generation of disparity image can be performed. Here we show the implemental results of three correspondence: Block Matching (BM) based on the research in [60], Semi Global Block Matching (SGBM) based on research in [61] and Graph cuts proposed by [62]. According to the research community evaluation on Middlebury stereo images test, Graph cuts is most accuracy of correspondence algorithms and performs slowest; Block Matching is the worst accuracy but it takes a shortest time. SGBM method yields average accuracy and performs rather fast.

**Figure 3-12** shows the estimated disparity from stereo images captured by Minoru 3D webcam.



(a) Color image (left camera)



(b) Disparity map estimated by BM method



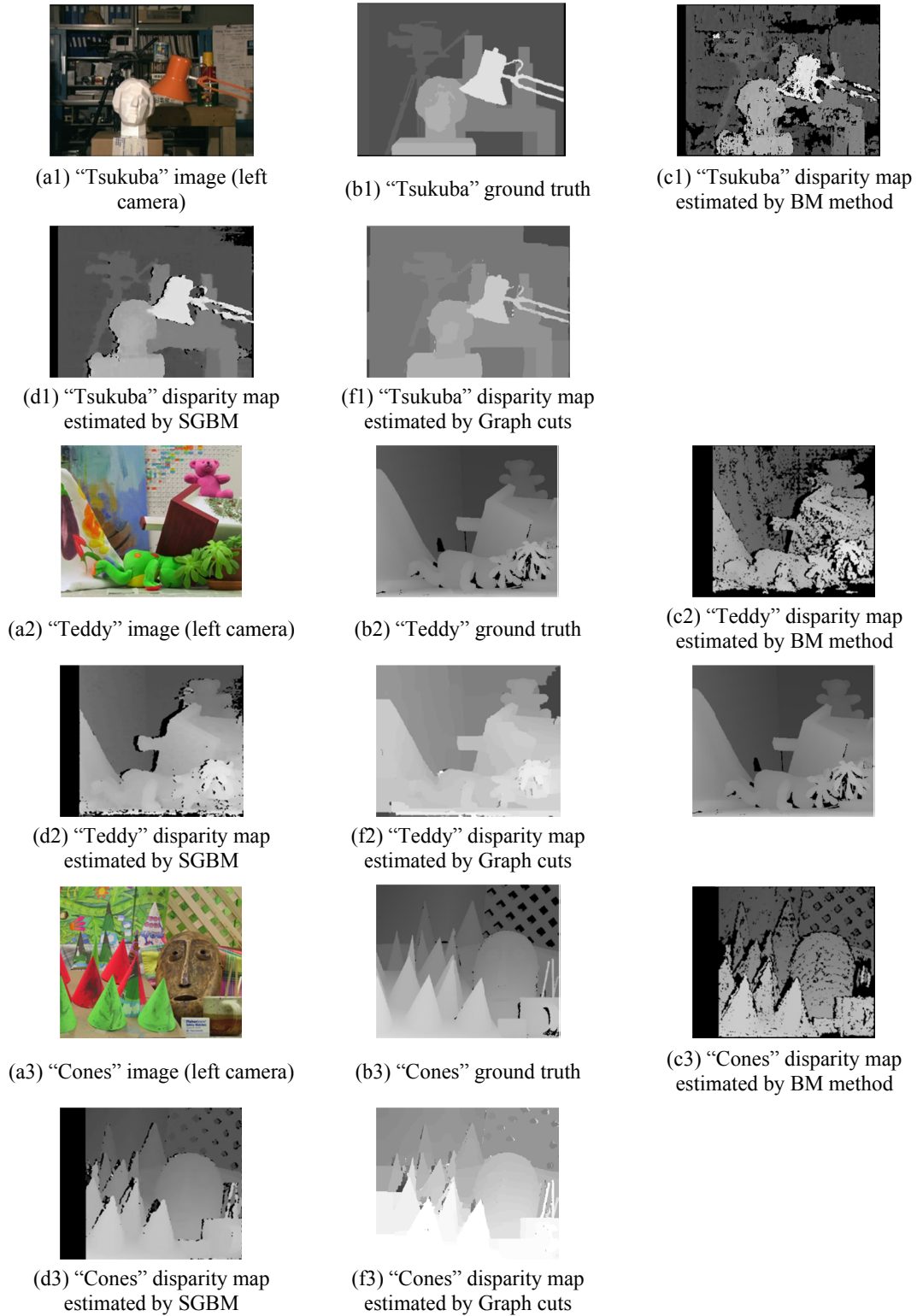
(c) Disparity map estimated by SGBM



(d) Disparity map estimated by Graph cuts [62]

**Figure 3-12. Estimated disparity maps from stereo images captured by Minoru 3D webcam.**

In our implementation, we also estimate the disparity maps using some stereo images given by Middlebury Stereo Datasets [63]. **Figure 3-13** shows our implementation results.



**Figure 3-13. Estimated disparity maps using stereo images from Middlebury Stereo Datasets [63].**

#### 3.5 PROPOSED DEPTH ESTIMATION ALGORITHM

This section presents a multi-view depth estimation algorithm that utilizes all views simultaneously, enforces a smooth variation of depth pixels within the image and consistent depth images across the views. The proposal algorithm is based on three constraints, i.e. *intra-line*, *inter-line* and *inter-view* smoothness costs which make the smooth variation of depth values in scanline, between scanline and consistent depth among views. These smooth costs are integrated into the one dimension optimization dynamic programming algorithm which helps in finding a global optimum. Experimental results on several multi-view data sets are encouraging.

##### 3.5.1 Introduction

This proposal method deals with the problem of estimation depth of objects in the scene from a set of multiple views or images. Depth information recovered from multi-view image/video serves as an important cue for many applications such as virtual view synthesis, multi-view video coding, multi-view object segmentation and others. However, estimating an accurate depth map is a complex process, which makes real-time implementation challenging.

In *section 3.3*, we already introduced the previous works on depth estimation. Most of the introduced methods estimate depth using rectified views. As we known in the rectification image section, the rectification process can lead to excessive and unwanted image distortion. Moreover, multiple pre-processing (image rectification) and post processing (image de-rectification) procedures may be necessary. In multi-view system, multiple images are available so that an efficient depth estimation algorithm should employ all views available. Many algorithms estimate depth map from multiple views are reviewed in *section 3.3.3*. In the following parts, we will propose the algorithm that utilizes multiple un-rectified images simultaneously to estimate depth. Three smooth costs, i.e. *intra-line*, *inter-line* and *inter-view* smooth costs are used to enforce a smooth variation of depth pixels within the image and among the images. These smooth costs can be integrated into the one dimension optimization dynamic programming algorithm which helps in finding a global optimum with polynomial complexity.

### 3.5.2 Proposal Depth Estimation from Multiple Images

It should be noted that there exists a relation between the depth of an object and its corresponding in the image. For instance, the near and far objects are seen as large and small in the image, respectively. Thus, a near object can be observed with higher resolution than a far object so that the depth resolution of a near object should be higher than that of a far object. To dealing with depth sampling, we use the plenoptic sampling method presented in [64]. In [64], they employ a non-linear quantization of depth between a minimum and maximum depth  $Z_{\min}$  and  $Z_{\max}$ . Since the depth image is usually the gray image with intensity with the range between 0 and 255, i.e.  $0 \leq d \leq 255$ , the relation among depth  $Z$ , nearest  $Z_{\min}$ , and farthest  $Z_{\max}$  can be expressed as:

$$Z = \frac{1.0}{\frac{d}{255.0} \cdot \left( \frac{1.0}{Z_{\min}} - \frac{1.0}{Z_{\max}} \right) + \frac{1.0}{Z_{\max}}}. \quad (3-8)$$

In such a depth image, pixel value  $d_{\min} = 0$  represents the farthest 3D point with depth being  $Z_{\max}$ , and the pixel value  $d_{\max} = 255$  represents the nearest 3D point with depth being  $Z_{\min}$ .

We propose an algorithm which is based on one-dimensional optimization to estimate depth images. A vector of depth pixel value  $D = \{d_{\min}, d_1, \dots, d_{\max}\}$  (or equivalent the vector of  $Z = \{Z_{\max}, \dots, Z_1, Z_{\min}\}$ ) along a scanline is computed to minimize the energy function  $E(D)$ , defined by

$$E(D) = E_{data}(D) + E_{smooth}(D), \quad (3-9)$$

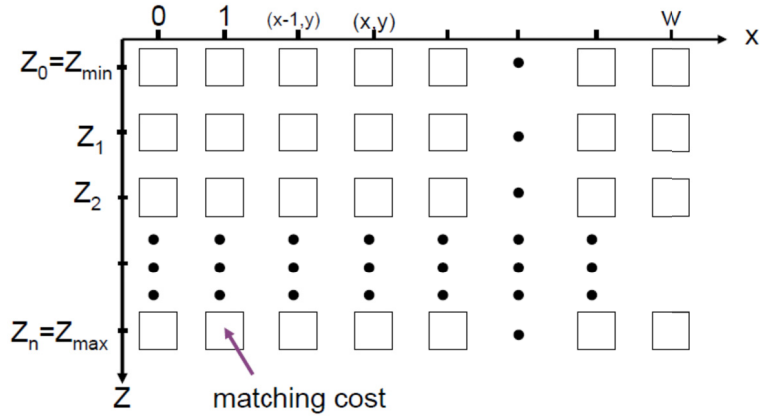
where,  $E_{data}(D)$  and  $E_{smooth}(D)$  are the matching cost and smoothness cost, respectively. These costs are defined as follows.

#### 3.5.2.1 Matching Cost $E_{data}(D)$

Our approach employs multiple views and estimates the depth using a correlation table  $C(x, y, Z_c)$  similar to the Disparity Space Image (DSI) in [65]. The  $C(x, y, Z_c)$  structure is a 2D table of size

### 3.5. Proposed Depth Estimation Algorithm

$Z_{\max} \times w$  that contains the matching cost of all pixels along a scanline at all possible candidate depth values  $Z_c \in [Z_{\min}, Z_{\max}]$ . Here,  $Z_{\max}$  and  $w$  correspond to the number of possible depth values and the column of image (image width), respectively. The objective of the algorithm is then to calculate the depth sequence that minimizes the total cost associated with the path through the correlation table. This minimum-cost path can be calculated using a dynamic programming algorithm.



**Figure 3-14. Correlation table with all matching cost for all depth value a long scanline.**

In our algorithm, an entry in the correlation table  $C(x, y, Z_c)$  is calculated using the sum of the correlation measures over all image views  $I$ , specified by

$$C(x, y, Z_c) = \sum_{k \neq k_r} \sum_{(i,j) \in W} |I_{k_r}(x-i, y-j) - I_k(x_k-i, y_k-j)|. \quad (3-10)$$

Given selected pixel from the *reference view*  $p_{k_r} = (x_{k_r}, y_{k_r}, 1)^T$  and candidate depth value  $Z_c$ , the candidate pixel positions  $p_k = (x_k, y_k, 1)^T$  in neighboring views are calculated as follows.

In the first step, for each pixel  $p_{k_r} = (x_{k_r}, y_{k_r}, 1)^T$  in the reference image and a candidate depth value  $Z_c \in \{Z_0, Z_1, \dots, Z_n\}$ , we need to find the 3D world point  $P = (X, Y, Z_c)^T$ . The 3D world point  $P$  can be computed using the back-projection relation which is based on internal and external camera parameters that indicate the location and orientation of cameras according to Equation (2-14) and can be written as:



$$(X, Y, Z_c)^T = \lambda_{k_r} R_{k_r}^{-1} K_{k_r}^{-1} p_{k_r} - R_{k_r}^{-1} T_{k_r}, \quad (3-11)$$

where  $K_{k_r}$  is an  $3 \times 3$  intrinsic matrix describing internal parameters of the reference camera  $k_r$ .  $R_{k_r}$  is a  $3 \times 3$  rotation matrix indicating the external orientation of the reference camera and  $T_{k_r}$  is a  $3 \times 1$  translation vector describing the external position the reference camera. To compute the intrinsic parameter  $K$  and extrinsic parameters, i.e.  $R$  and  $T$  of a camera, people usually use the calibration technique as discussing in *section 2.1.4* and *section 2.2.2*.  $\lambda_{k_r}$  is a positive scaling factor defining the position of the 3D point.

If we know the scaling factor  $\lambda_{k_r}$ , we can obtain the value of  $X$  and  $Y$  of the 3D point. Since we know  $Z_c$  thus  $\lambda_{k_r}$  is calculated by:

$$\lambda_{k_r} = \frac{Z_c + C_3}{z_3}, \quad (3-12)$$

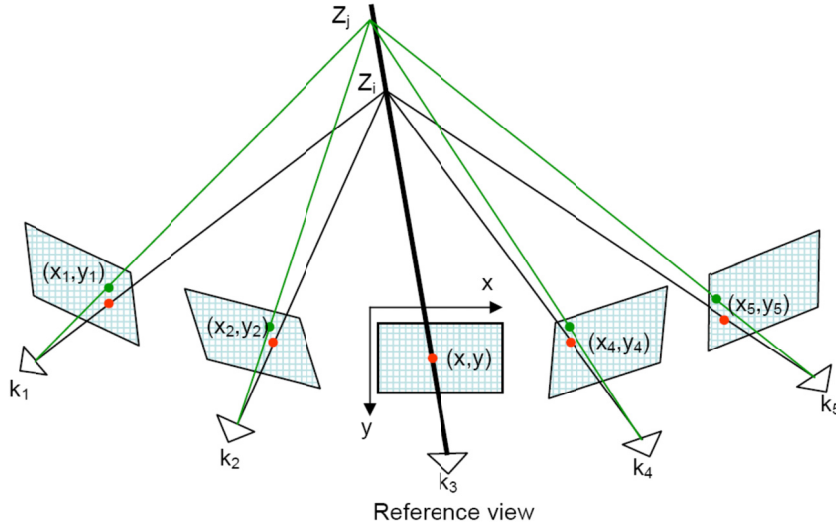
where  $(z_1, z_2, z_3) = R_{k_r}^{-1} K_{k_r}^{-1} p_{k_r}, (C_1, C_2, C_3) = -R_{k_r}^{-1} T_{k_r}$ .

In the second step, the candidate pixel positions in neighboring views are calculated by projecting the calculated 3D point  $P = (X, Y, Z_c)^T$  onto each neighboring view  $k$  according to the Equation (2-13), providing a corresponding pixel corresponding pixel  $(x_k, y_k, 1)^T$  for each view for which:

$$\lambda_k (x_k, y_k, 1)^T = K_k R_k (X, Y, Z_c)^T + K_k T_k, \quad (3-13)$$

where,  $K_k, R_k$  and  $T_k$  are representing the intrinsic matrix, rotation matrix and translation vector of the neighboring camera  $k$ , respectively. These two steps are repeated for depth candidates  $Z_c$ , so that a corresponding similarity can be measured.

By exploiting camera parameters, multiple views can be employed simultaneously so a reliable similarity value can be obtained. The similarity measure among the selected pixel and the pixels in neighboring views is measured using the SAD and the matching cost is stored in a correlation table  $C(x, y, Z_c)$ . This idea is illustrated by **Figure 3-15**.



**Figure 3-15.** Given a pixel  $p(x,y)$  in the reference view, the algorithm test the vector of depth candidate  $Z_c$ . For example, candidate  $Z_i$  yields the consistent color across views, it should be selected as the final depth value.

The term  $E_{data}(D)$  in Equation (3-9) represents the matching cost of all pixels along the scanline defined as:

$$E_{data}(D) = \sum_{x=0}^w C(x, y, Z_c), \quad (3-14)$$

where  $C(x, y, Z_c)$  represents the cost of matching function computed by Equation (3-10).

### 3.5.2.2 Smoothness Cost $E_{smooth}(D)$

To enforce a smooth variation of depth pixels within the image and among the images, we introduce three constraints:

- 1) Smooth variation of depth values intra scanline between two neighboring pixels in the reference image (namely intra-line penalty cost);
- 2) Smooth variation of depth values between two scanlines (inter-line penalty cost);
- 3) And consistent depth values across views (inter-view penalty cost).

A depth image is estimated by minimizing two objective functions which are combined above constraints with dynamic programming algorithm in two following steps.

**Step 1: Initial Depth Map for Each View**

In this step, we wish to enforce the constraints that consecutive image scanlines should have smooth depth variations along and across the lines. We denote the objective function for one view at the first step as:

$$E_1(D) = E_{data}(D) + E_{smooth1} \quad (3-15)$$

Depth image for each view is estimated with the intra-line and inter-line constraints, so we define the smoothness cost for this step as:

$$E_{smooth1} = E_{intra-line}(D) + E_{inter-line}(D) \quad (3-16)$$

The term  $E_{intra-line}(D)$  in Equation (3-16) is designed to make it more likely that a pair of adjacent pixels  $p(x-1, y)$  and  $q(x, y)$  along a scanline in the reference image with similar intensity would end up with similar disparity. For example, if the neighbor pixel  $p$  and  $q$  have the similar intensity, the intra-line penalty cost is low if they have similar depth value and the cost is high if they have different depth value.

Practically, the cost function integrating the intra-line penalty cost can be written as:

$$E_{intra-line}(D) = \begin{cases} E_{intra\_1}(D) & \text{if } |d_{k_r}(x,y) - d_{k_r}(x-1,y)| \leq T_d \\ E_{intra\_2}(D) & \text{otherwise} \end{cases} \quad (3-17)$$

The two functions  $E_{intra\_1}(D)$  and  $E_{intra\_2}(D)$  are defined as following

$$E_{intra\_1}(D) = \begin{cases} \sum_x \lambda_1 |d_{k_r}(x,y) - d_{k_r}(x-1,y)| & \text{if } |I_{k_r}(x,y) - I_{k_r}(x-1,y)| < T_c \\ \sum_x \alpha_1 \lambda_1 |d_{k_r}(x,y) - d_{k_r}(x-1,y)| & \text{otherwise} \end{cases} \quad (3-18)$$

$$E_{intra\_2}(D) = \begin{cases} \sum_x \lambda_1 T_d & \text{if } |I_{k_r}(x,y) - I_{k_r}(x-1,y)| < T_c \\ \sum_x \alpha_1 \lambda_1 T_d & \text{otherwise} \end{cases}$$

Here,  $D$  is the vector of depth pixel values  $d_{k_r}(x, y)$  along a scanline,  $d_{k_r}(x, y)$  the estimated depth at position  $(x, y)$  in the reference image with index  $k_r$ ,  $k_r$  varies between  $1 \leq k_r \leq N$ , for the  $N$

### 3.5. Proposed Depth Estimation Algorithm

---

views.  $\lambda_1$ , and  $\alpha_1 > 1$  (typical scalar factor) are control parameters used to penalty variations along the scanline.  $T_d$  is a different depth threshold and  $T_c$  is threshold of intensity variation (in our experiment, we set  $\lambda_1 = 1$ ,  $\alpha_1 = 3$ ,  $T_d = 3$  and  $T_c = 10$ ).

Since one-dimension dynamic programming optimization is performed independently for each scanline, which has led to horizontal line based streaking artifacts in the depth image. Voiding this artifact, we introduce an inter-line penalty cost  $E_{\text{inter-line}}(D)$  to enforce smooth variations of depth values across scanlines. In practice, the inter-line penalty cost can be defined as:

$$E_{\text{inter-line}}(D) = \begin{cases} \sum_x \lambda_2 |d_{k_r}(x, y) - d_{k_r}(x, y-1)| & \text{if } |d_{k_r}(x, y) - d_{k_r}(x, y-1)| < T_d \\ \sum_x \alpha_2 \lambda_2 T_d & \text{otherwise} \end{cases}, \quad (3-19)$$

where, similar to intra-line cost  $\lambda_2$ , and  $\alpha_2$  (typical scalar factor) are control parameters used to penalty variations between two the scanlines.  $T_d$  is a different depth threshold (in our experiment, we set  $\lambda_2 = 1$ ,  $\alpha_2 = 1$  and  $T_d = 3$ ).

#### **Step 2: Depth Image Refinement**

The aim of this second step is to refine the initialization depth images from the first step such that the final depth images are consistent across the views. In this step, we wish to enforcer the constraint that objects should have consistent world depth values in all depth images. We encode the inter-view constraint using the following objective function:

$$E_2(D) = E_1(D) + E_{\text{inter-view}}(D). \quad (3-20)$$

The inter-view penalty costs can be written as

$$E_{\text{inter_views}}(D) = \sum_x \sum_{k \neq k_v}^N \lambda_v |d_{k_r}(x, y) - d_k(x_k, y_k)|, \quad (3-21)$$

where,  $\lambda_v$  is control parameters used to penalty variations among views. The depth  $d_{k_r}(x, y)$  and  $d_k(x_k, y_k)$  correspond to the depth of pixel  $(x, y)$  in the reference view with index  $k_r$  and the depth

of pixel  $(x_k, y_k)$  neighboring views  $k$ , respectively. The value of the depth  $d_k(x_k, y_k)$  is determined by projecting pixel  $(x, y)$  in the reference view  $k_r$  onto the neighboring views  $k$  which provides the pixel position  $(x_k, y_k)$  and the corresponding depth  $d_k(x_k, y_k)$  was computed in the first step.

### 3.5.2.3 Noise Reduction Based on Plane Fitting

The goal of this optional processing is to improve the quality of depth images for each view by applying a noise reduction algorithm. The most challenge is filter noisy pixels whereas preserving the accuracy of depth pixels along object borders. To solving this problem, we propose noise-reduction algorithm proceeds based on plane fitting as following steps:

**Step 1: the color texture image corresponding to the considered depth image is segmented.**

The aim of this step is to segment the color image into regions correspond to a piece of the object surfaces. Because the object surfaces are likely smooth regions in the depth image so the corresponding depth image segments can be approximated by plane. Any algorithm that decompresses an image into homogeneous color region will work for our purpose such as watershed segmentation algorithm [66], mean shift segmentation [67]. For our implementation, we used mean shift segmentation algorithm [67], an example is shown in **Figure 3-16**.



**Figure 3-16. Original “Break-dancers” image (on the left) and the corresponding segmented image using mean shift segmentation algorithm (on the right).**

**Step 2: plane fitting within each segmented region by means of 3D planar surfaces according to:**

$$ax + by + c = Z_p, \quad (3-22)$$

### 3.5. Proposed Depth Estimation Algorithm

---

where  $(x, y)$  is an image point  $p$  and corresponding depth value  $Z_p$ .  $a, b$  and  $c$  are the linear function parameters of each segmentation.

$$A[a, b, c]^T = B, \quad (3-23)$$

where each row of  $A$  is  $[x, y, 1]$  vector for a pixel and each row of  $B$  is its corresponding  $Z$ .

To estimate unknown parameters  $(a, b, c)$ , we using the random sample consensus (RANSAC) algorithm [68]. The RANSAC calculate the plane parameters as follows:

1. In each segmented region, first we randomly select three pixels denoted by  $(x_i, y_i)$  and their corresponding depth value  $Z_i$ , with  $i = 1, 2, 3$ . A test plane from the three selected points is created by:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix}. \quad (3-24)$$

2. The candidate parameters  $(a, b, c)$  are computed by solving the linear equations system.
3. Test the remaining points in the segmented region against this fitted plane and counts the number of pixels (inliers) that support for this model with in a given error threshold. The equation used to determining if a given point is a inliers is

$$D_i(x_i, y_i, Z_i) = \frac{|ax_i + by_i + c - Z_i|}{\sqrt{a^2 + b^2 + 1}}, \quad (3-25)$$

$$Inlier(x_i, y_i, Z_i) = \begin{cases} 1 & D_i(x_i, y_i, Z_i) \leq T_D \\ 0 & D_i(x_i, y_i, Z_i) > T_D \end{cases}$$

where  $D_i(x_i, y_i, Z_i)$  is the distance from point  $(x_i, y_i, Z_i)$  to the fitted plane and  $T_D$  is a distance threshold.

4. Repeat steps 2 and 3 until stopping criteria is reached. Usually iteration until the maximum number of inliers is greater than a set threshold, or reaches the maximum iteration number.

The parameters with biggest support are selected as the planar parameters of segmented region.

This technique is performed for all image segments and we obtain the refined depth map.

### 3.5.3 Experimental Results

In this section, we evaluate the accuracy of our depth estimation algorithm describing in previous section. The proposed depth estimation algorithm has been tested on some multi-view data sets such as “Break-dancers” and “Ballet” [69], which is utilized in multi-view coding, rendering and depth estimation research. In the following parts, we first provide an objective evaluation and, second, subjective evaluation of depth image quality.

For objective evaluation, basically we have two approaches. The first approach is a direct method which is based on the comparison with ground-truth depth image. However, the ground truth multi-view depth images are not usually available. The second approach is an indirect method which is based on rendering quality evaluation. This technique is based on the notion that the quality of the rendered image depends on the accuracy of the estimated depth image. Generally, the more accurate a depth image is, the higher the rendering quality becomes. In this experiment, the synthesis of virtual image is carried out using the simply 3D warping synthesis algorithm, which is presented in the next *Chapter 4*. The synthesis image ( $I_s$ ) is rendered at the same position and orientation of a selected view ( $I_c$ ), i.e. virtual camera has the same external and internal parameters as selected camera. In this method, the quality of calculated depth image is measured objectively by calculating the Peak Signal Noise Ratio (*PSNR*) of the synthesis image  $I_s$  and the captured image  $I_c$ . This idea is shown in **Figure 3-17**.

Before computing *PSNR*, the image is converted from RGB color space to YUV color space, and Y channel is used for calculation. Y channel is defined by

$$Y(i,j) = 0.299R(i,j) + 0.587G(i,j) + 0.114B(i,j). \quad (3-26)$$

The *PSNR* can be calculated by

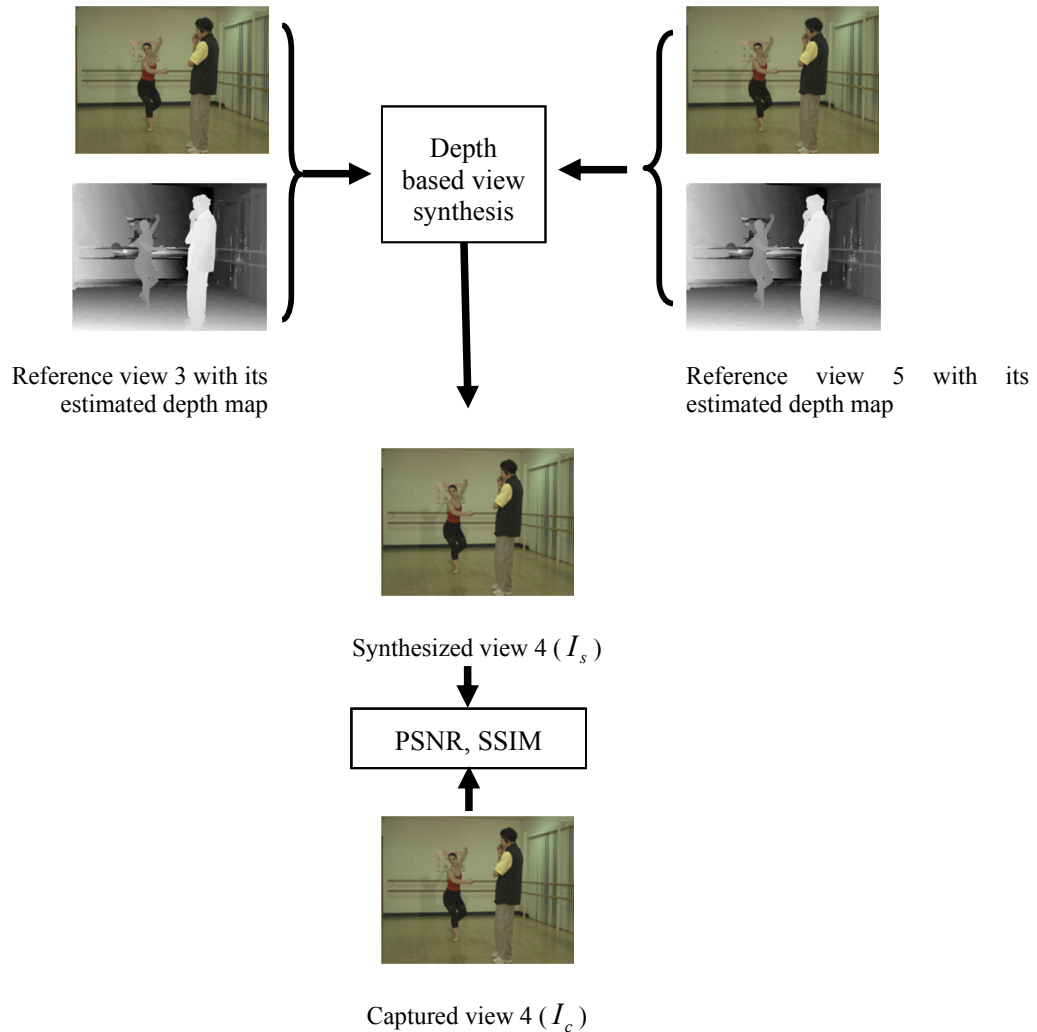
### 3.5. Proposed Depth Estimation Algorithm

$$PSNR = 10 \log_{10} \left( \frac{255^2}{MSE} \right), \quad (3-27)$$

and Mean Square Error ( $MSE$ ) is computed by

$$MSE = \frac{1}{w \cdot h} \sum_{i=1}^w \sum_{j=1}^h \|Y_{captured}(i, j) - Y_{synthesis}(i, j)\|^2, \quad (3-28)$$

where  $w$  and  $h$  are the width and the height of the images, respectively.



**Figure 3-17. The idea to objectively evaluate the quality of depth map via view synthesis.**

The  $PSNR$  values which measure the quality of synthesized image resulting of two different depth images are summarized in **Table 3-3**. Here, we evaluation in two scenarios: two views and multi-views depth estimation based on one dimension dynamic programming optimization. It can be seen



that our proposed algorithm that uses multi-views to compute depth images consistently and yields better results comparing with two-view depth estimation, for example improvement of 3.3 dB for the “Break-dancer” and 2.7 dB for “Ballet”, respectively.

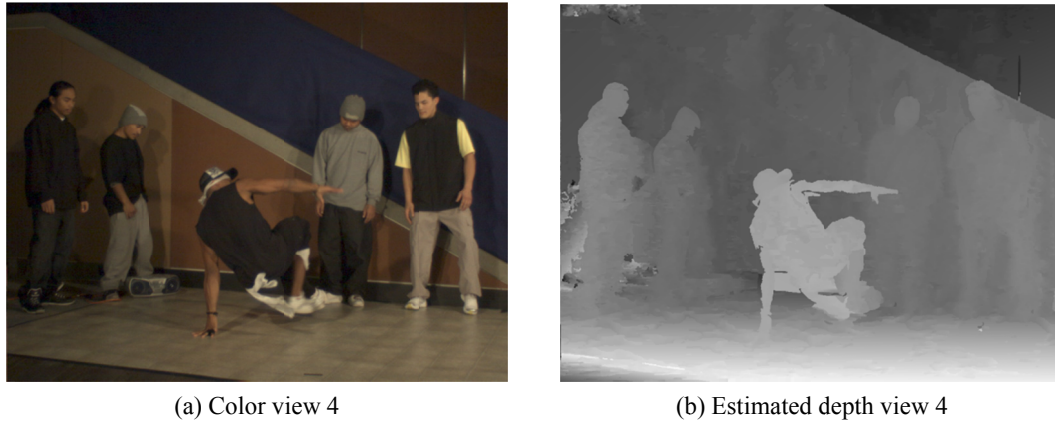
**Table 3-3. The quality of synthesized image resulting of two different depth estimation algorithms**

	Two views	Multi-views (8 views)
“Break-dancers”	28.6 dB	31.9 dB
“Ballet”	26.1 dB	28.7 dB

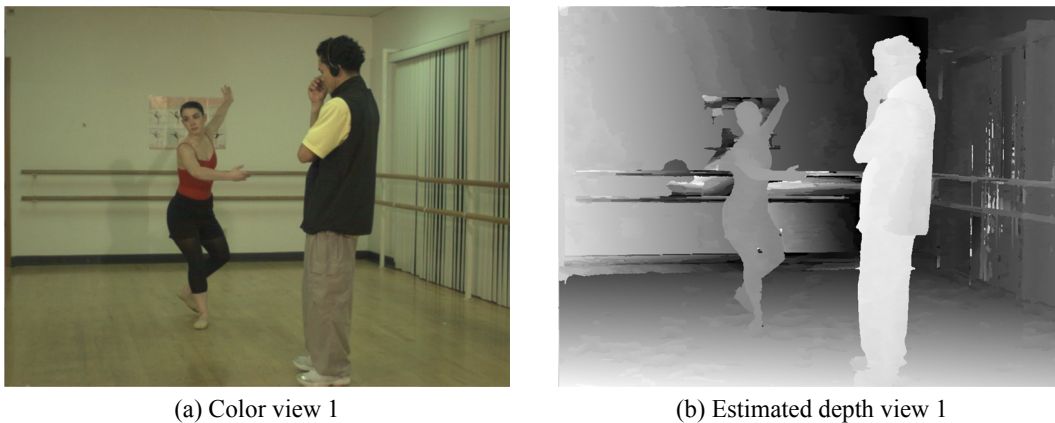
It should be noted that the higher *PSNR* values can be obtained via other better image rendering algorithms, whereas in this work, our simple algorithm is utilized during the rendering stage. It also note that the *PSNR* might not match the subjective visual quality, i.e. human subjective perception, however this is simple image quality metric that has been widely accepted in research community.

Next we show the subjective quality of the estimated depth using the proposed algorithm. The estimated depth images for the reference view of different data sets are presented in **Figure 3-18** and **Figure 3-19**, where the whiter intensities indicate the regions closer to the camera. It can be seen that the estimated depth map have consistent depth value across the lines so that the streaking artifacts are reduced. Second, the result shows that the depth discontinuity between the foreground objects and background wall is accurately estimated. It can be noted that the depth of colorless and texture-less can be accurately estimated. For instance, the texture-less walls in **Figure 3-19** are correctly estimated.

Up till now, as our knowledge there is a few contribution exist which employ the complex “Break-dancer” and “Ballet” multi-view sequences. We have found two papers that use either both sequences [52] or “Break-sequence” only [70]. In [52], authors proposed method which perform an over segmentation of the color image and compute the depth of each segment using a belief propagation algorithm. In [70], the proposal is based on modified plane sweeping algorithm. In **Figure 3-20** shows the estimated depth images by the proposed method comparing with the results in these two papers.



**Figure 3-18. An estimated depth image for “Break-dancers”.**



**Figure 3-19. An estimated depth image for “Ballet”.**

### 3.6 SUMMARY AND CONCLUSIONS

In the previous chapter, we have introduced the multi-view geometry that model the projection of a 3D point onto the 2D image plane. Based on this framework, in this chapter we describe algorithms for estimating the depth of pixels.

In the first part of this chapter, we focus on a comprehensive review the previous works on estimation of depth image using a single image/mono video, two views and multiple views. With the recent explosive for 3D media contents, converting existing single still images or monocular videos to 3D contents is a problem of considerable practical interest. Depth estimation from a single image is a difficult task, since depth typically remains ambiguous given only local image features. Most of algorithms try to explore the monocular cues, image features and human visual features to estimate

depth. Until now, there are three commonly used depth estimation methods from 2D to 3D video conversion applications. “Depth from Motion Parallax” method, which is based on the fact that objects with different motions usually have different depths, is most widely used for the depth estimation in 2D-to-3D video conversion. Depth estimation from two views is one of the most widely studied and fundamental problem and continues to be the active research areas. We have reviewed the two-view depth estimation algorithms based on the taxonomy and categorization scheme proposed by [35], which comprises a set of algorithmic blocks from which a large set of algorithms can be constructed. Next, we have introduced the multi-view system which significantly broadens the multi-view applications and enhances the user experience. Then we also review the previous works on multi-view depth estimation.

In the second part of this chapter, a novel algorithm is proposed for the estimation of depth from multi-view images utilizing calibration parameters to provide consistency and reliability. We have introduced three constraints, i.e. intra-line, inter-line and inter-view smoothness constraint, which enforce smooth variations of depth value in the scanline, across scanline and consistent depth value across the views. The proposed algorithm combines two steps: the first step serves as a calculation of initial depth images and the second step enhances the depth initial depth images in the first step by enforcing consistent depth across the views. The three smooth constraints can be efficiently integrated into one-dimensional optimization dynamic programming algorithm. Experiments have shown that the proposed smooth constraints yield reasonably good depth image quality for various multi-view data sets.

One important application of depth image is a depth-based view synthesis for free-viewpoint video and 3D-TV systems, which are detailed in the next chapter.



(a) Color image



(b) Depth image computed by the proposed method.



(c) Depth image computed by [52].



(d) Depth image computed by [70].

**Figure 3-20. Comparing the proposed method with others.**

## DEPTH IMAGE BASED SYNTHESIS

### 4.1 INTRODUCTION

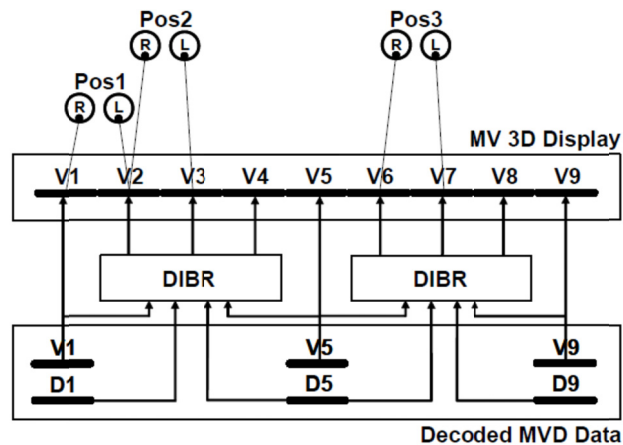
There is an increasing interest in three-dimensional video/television (3DV/3DTV) systems since the technology advances rapidly in 3D scene capturing, processing, transmission and displaying. With the growing capability of capturing devices, multi-view capture system with dense or sparse camera array can be built with ease, Free-viewpoint television (FTV) [71] system has attracted increasing attention. In FTV system, user can freely select the viewpoint of any dynamic real world scene. The chosen free-viewpoint cannot only be selected from available multi-view camera views, but also any viewpoint between these cameras.

Creating 3D depth impression requires that a viewer looking at 3D display see a different view with each eye. There are roughly two categories of such 3D display: stereoscopic displays with glasses and autostereoscopic displays without wearing glasses. For stereoscopic displays that require the viewer to wear glasses, two views are emitted at the display while the accompany glasses allow only one view to go through each eyeglasses. There are some typical glassed including anaglyph glasses, polarized glasses, and shutter glasses. The necessity to wear glasses is considered as a main obstacle for success of 3DV in home user environment. The invention of autostereoscopic display gives us an opportunity to overcome this problem. Several images are emitted at the same time but the technology ensures that users only see a stereo pair from a specific viewpoint. For example, the high resolution LCD screens with slanted lenticular lens technology as commercially available from Phillips [72] are capable of displaying 9 and more simultaneous views, of which only a stereo pair is visible from a specific viewpoint. With this, multiuser 3D sensation without glasses is enable, for example in a living room. This principle is shown in **Figure 4-1**. For example, at position 1, a user sees only view 1 and view 2 with right eye and left eye, respectively. At another position 3, a user sees only view 6 and view 7, hence multi-user 3D viewing is supported. All views are properly arranged such that views 1 and 2, then view 2 and 3 and so on are stereo pairs with proper human eye

## 4.1. Introduction

distance baseline, thus a user moving in front of such 3D display system will perceive a 3D impression with head motion parallax.

To provide input for 3D displays, one approach is to store/transmit video of multiple views. Comparing to normal 2D video, this is an extreme increase of store/transmission cost. To compress such contents, the potential solutions include simulcast of each view, multi-view video coding (MVC) [73] as an amendment to H.264/AVC. However, when the number of views is pretty large, the cost in bit rate may be too expensive. It has been shown in [74] that the total bit rate of MVC increase linearly with the number of views  $N$ . Therefore, future display with more views would require even total higher bit rate. Further, even with a very large number of video views, it is still not capable of covering any arbitrary viewpoint.

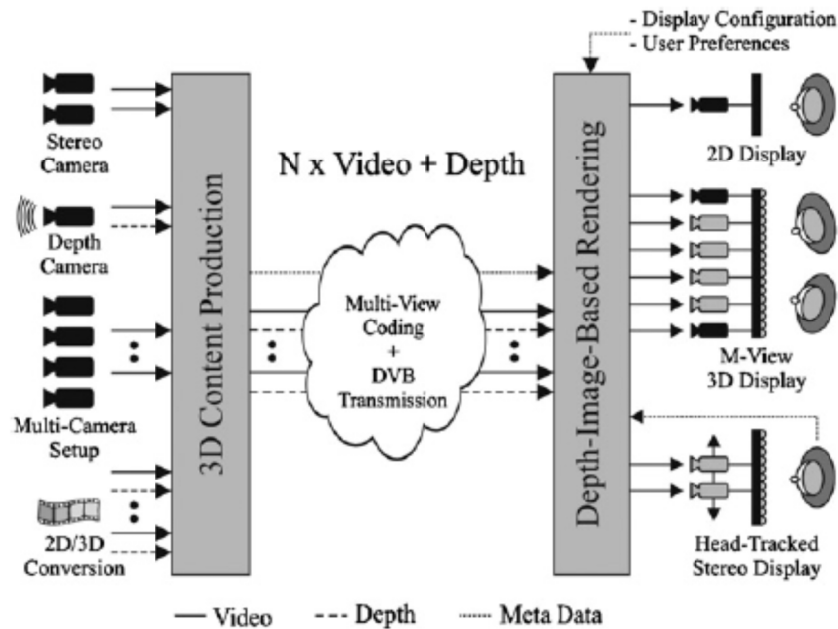


**Figure 4-1. Example of advanced 3DTV concept based on Multiple View plus Depth (MVD) (9 outputs views out of 3 input views plus depth); Pos: viewpoint, R: right eye, L: left eye, V: view/image, D: Depth, DIBR: Depth image based rendering.**

To overcome these disadvantages, an alternative 3D video format that is the multiview video plus depth (MVD) is being explored [75]. A corresponding standard known as MPEG-C Part 3 has been released by MPEG. Instead of using a large number of views, video and depth from a sparse camera arrangement are utilized while intermediate views are synthesized. When using the MVD format only a subset  $M$  of the  $N$  display views is transmitted. **Figure 4-1** illustrates a typical of MVD example, in which three views plus their corresponding depth are transmitted to decoder at a receiver. From the decoded video plus depth, additional views can be synthesized using the depth based image rendering technique [76], which involves the 3D-projection or 2D-warping from a viewpoint into another view.

The rendering may be a built-in function of displays or be implemented outside a display such as in a set-up box.

**Figure 4-2** shows an example of 3DTV system with MVD format [56]. It consist of the main components such as 3D video capture and content production, 3D content encoder, transmission/storage, decoder, depth image based synthesis and display. The success of the 3DTV system in **Figure 4-2** depends a lot on the quality of view synthesis at receiver. However, high quality with DIBR is a challenging task especially when the depth map is noisy and no extra scene information such as 3D surface properties is known. To render a high quality image at arbitrary view point, one has to manage three main challenges as pointed out in [52]. First, empty pixels and holes due to sampling of the reference image have to be closed. Secondly, pixels at borders of high discontinuities cause contour artifacts. The third challenge involves inpainting disocclusions that remain after blending the projected images (these are invisible from any of the surrounding cameras).



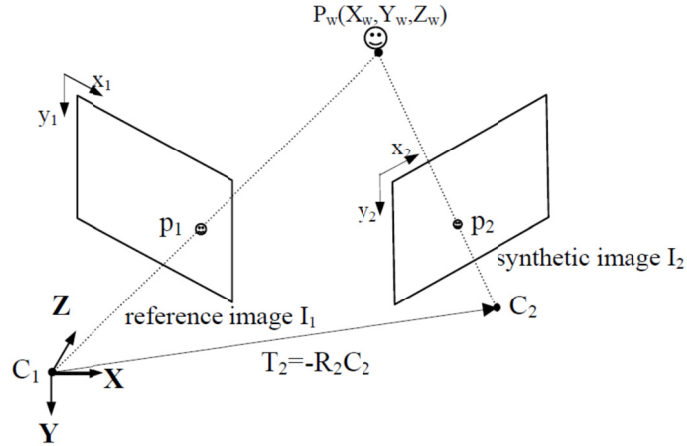
**Figure 4-2.** An example of 3DTV system with multiple view plus depth format [56].

In this chapter, we will discuss the view synthesis techniques for 3D video. The next section, we will review image rendering techniques. Afterward, we present a novel view synthesis algorithm assuming that the depth maps, textures of multi-view cameras and their parameters are available.

## 4.2 CURRENT RESEARCH ON DEPTH IMAGE BASED SYNTHESIS

### 4.2.1 Basic Principle: 3D Image Warping

The basic idea of most depth image based synthesis method is to perform 3D warping to the virtual viewpoint using texture and depth information of the reference camera [77]. Let us specify this in some more detail.



**Figure 4-3. The two projection point  $p_1$  and  $p_2$  of a point  $P_w$**

Let  $P_w = [X_w, Y_w, Z_w, 1]^T$  be the world point, which are captured by both cameras;  $p_1 = [x_1, y_1, 1]^T$  and  $p_2 = [x_2, y_2, 1]^T$  be its projection onto reference and synthetic image planes, respectively. This is illustrated in **Figure 4-3**. According to the Equation (2-13), the two pixel positions  $p_1$  and  $p_2$  are related with the point  $P_w$  by the cameras' parameters as following relationship:

$$\lambda_1 p_1 = K_1 R_1 \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - K_1 R_1 C_1, \quad (4-1)$$

$$\lambda_2 p_2 = K_2 R_2 \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} - K_2 R_2 C_2, \quad (4-2)$$

where,  $K_i$  is a  $3 \times 3$  upper triangular matrix representing the intrinsic parameter matrix of the camera  $i$ , with  $i \in \{1, 2\}$ . The  $3 \times 3$  orthogonal matrix  $R_i$  and  $C_i$  represent the orientation and the position of the camera  $i$ , respectively.



Rearranging Equation (4-1) we can derive 3D coordinate of the scene point  $P_w$  :

$$(X_w, Y_w, Z_w)^T = (K_1 R_1)^{-1} \cdot (\lambda_1 p_1 + K_1 R_1 C_1). \quad (4-3)$$

Substituting Equation (4-3) into Equation (4-2), we obtain the synthetic pixel position  $p_2$  :

$$\lambda_2 p_2 = K_2 R_2 (K_1 R_1)^{-1} \cdot (\lambda_1 p_1 + K_1 R_1 C_1) - K_2 R_2 C_2. \quad (4-4)$$

Assuming that the world coordinate system is the same as the reference camera coordinate system and looks at along  $Z$ -direction, i.e.,  $C_1 = (0,0,0)$ ,  $R_1 = I_{3 \times 3}$  and  $\lambda_1 = Z_w$ , Equation (4-4) can rewrite as following:

$$\lambda_2 p_2 = K_2 R_2 K_1^{-1} \cdot Z_w p_1 - K_2 R_2 C_2, \quad (4-5)$$

where,  $Z_w$  is defined by the pixel value at coordinate point  $p_1$  in the reference image.

The relationship (4-5) constitutes the 3D image warping equation that enables the synthesis of the virtual view from a reference texture and a corresponding depth map. Given a pixel point  $p_1 = [x_1, y_1, 1]^T$  from the reference image and its corresponding depth value  $Z_w$ , we can calculate a pixel point  $p_2$  on the synthesis image. This equation specifies the computation for one pixel only so that it has to be performed for the entire image.

#### 4.2.2 Previous Works

In this paragraph, we describe some recent research on free-viewpoint DIBR algorithm. In [52], author has developed a free-viewpoint rendering algorithm which is based on layered representation. For texture mapping, 3D meshes are created and the rendering is implemented on a GPU. Although the results look good, the method is complex and requires a considerable amount of pre- and post-processing operations. This work is extended in [78] where the depth map is decomposed into three layers and these layers are warped separately. The warp results are obtained for each layer and merged. To deal with artifacts, they have introduced three post-processing algorithms. In [79], a new viewpoint is rendered by some steps. First, the depth maps of the reference cameras are warped to the new viewpoint. Then the empty pixels are filled with a median filter. Afterwards, the depth maps are

### 4.3. Proposed Algorithm and Its Performance

---

processed with a bilateral filter. Then, the textures are retrieved by performing an inverse warping from the projected depth maps back to the reference cameras. Ghost contours are removed by dilating the disocclusions. Finally, the texture images are blended and the remaining disocclusions are inpainted using the method proposed by Telea [80]. Although, the results look good, this method is remaining some issues such as not removing all holes by median filter, assigning a none-zero value for some pixels in disocclusion regions. This work is improving in [81] by introducing three enhancing techniques. First, re-sampling artifacts are filled in by a combination of median filtering and inverse warping. Second, contour artifacts are processed while omitting warping of edges at high discontinuities. Third, disocclusion regions are inpainted with depth information. The quality of this method is higher than the work in [79], but still having disadvantages. For example, they have to define the label of pixel at high discontinuities. The color consistency during blending is not verified to avoid jagged edges at straight line after blending. The work in [82] combines depth based hole filling and inpainting to restore the disoccluded pixels more accurately compared to inpainting method without using depth information. This method produces a notable blur and can be computationally inefficient when disoccluded region is larger in new view. .

## 4.3 PROPOSED ALGORITHM AND ITS PERFORMANCE

### 4.3.1 *Algorithm Overview*

In this section, we introduce a new free-viewpoint rendering algorithm from multiple color and depth images. First, the depth maps for the virtual views are created by warping the depth maps of reference cameras. We process the wrapped depth maps with median filter. Depth maps consist of smooth regions with sharp edges, so filtering with a median will not degrade the quality. Then, the textures are retrieved by performing an inverse warping from the warped depth maps to the reference cameras. This allows a simple and accurate re-sampling of synthetic pixel. After that, all warped depth and warped texture images are used to classify pixel as stable, unstable and disoccluded regions. An initial virtual view is created based on weighted interpolation of stable pixels. To refine the synthetic view, best candidates for unstable pixels are optimally selected by Graph cuts. By defining the types of pixels and using Graph cuts, the color is consistent and the incorrectly wrapped

pixels because of inaccuracy depth maps are removed in the refined view. The remaining disoccluded pixels are inpainted by using depth and texture neighboring pixel values. Considering depth information for inpainting, blurring between foreground and background textures is reduced.

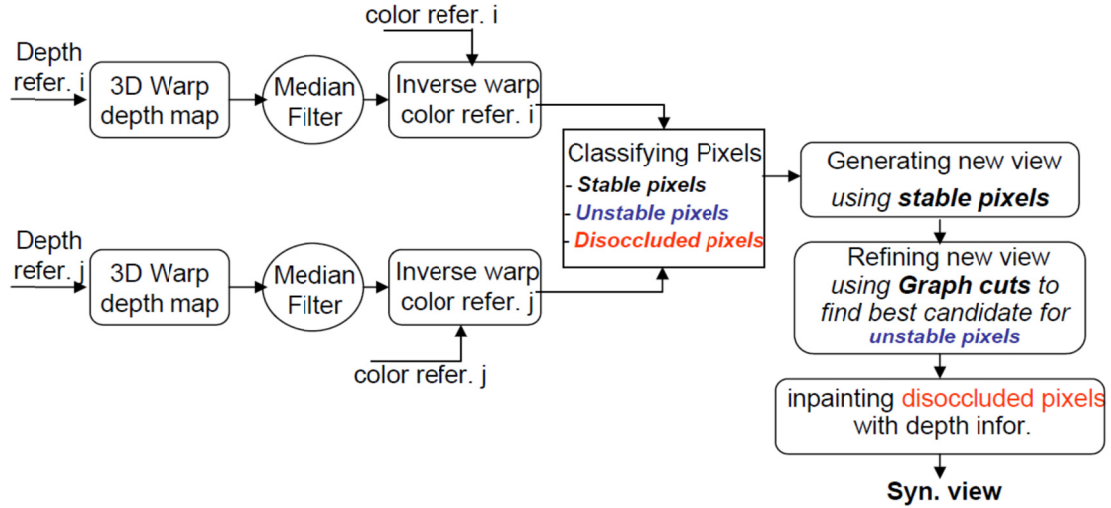


Figure 4-4. Proposed new view synthesis algorithm.

### 4.3.2 Proposed Depth Image Based Synthesis using Graph Cuts

Our proposal is shown in **Figure 4-4** and it consists of six steps. These steps are explained below.

#### 4.3.2.1 3D Warping the Depth Maps

3D warping projects an image to another image plane. As described in Equation (4-5), given a pixel point  $p_1 = [x_1, y_1, 1]^T$  from the reference image and its corresponding depth value  $Z_w$ , we can calculate a pixel point  $p_2$  on the synthesis image.

In this step, the problem that several points can be projected to the same point in virtual image is solved by using simple *z-buffering* technique. Another issue of this process is that a pixel  $p_1$  of reference view is not usually projected on to a point  $p_2$  at integer pixel position. To obtain an integer pixel position, we map the sub-pixel  $p_2$  to the nearest integer pixel  $\hat{p}_2$  as follows equation:

$$\hat{p}_2 = (\hat{x}_2, \hat{y}_2, 1) = (\lfloor x_2 + 0.5 \rfloor, \lfloor y_2 + 0.5 \rfloor, 1). \quad (4-6)$$

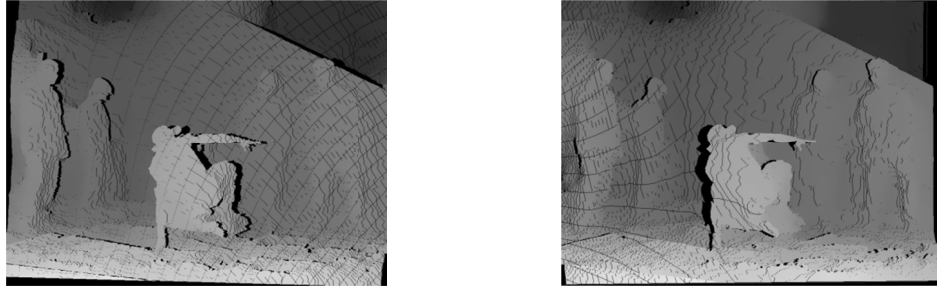
### 4.3. Proposed Algorithm and Its Performance

---

In our method, only depth maps of reference cameras are projected to virtual image plane. The warping is specified by:

$$[Z_{syn}, \hat{p}_2] = 3D\_Warp(Z_{ref}, p_1), \quad (4-7)$$

where,  $Z_{ref}$  is depth map of a reference camera,  $3D\_Warp$  is warping operation as above describing (see section 4.2.1). The projected depth maps from two reference cameras for an arbitrary scene are shown in **Figure 4-5**.



**Figure 4-5. The projected depth maps from two reference cameras (from the left side and from the right side).**

#### 4.3.2.2 Median Filter the Warped Depth Map

In this step, we consider the blank points that appeared in projected depth map. The reasons for the appearance of these blank points are round off errors of the image coordinate by Equation (4-6) and depth discontinuities. It can cause one pixel wide blank region to appear. This blank region can be filled by median filter with a window of  $3 \times 3$  pixels. Depth maps consist of smooth regions with sharp edges, so filtering with a median will not degrade the quality.

This step can describe as:

$$Z_{syn\_filtered} = Median(Z_{syn}), \quad (4-8)$$

where,  $Median$  is a median filter with a window  $3 \times 3$  pixels,  $Z_{syn\_filtered}$  is output of median filter.

The image in **Figure 4-5** can be processed by using median filter to obtained images in **Figure 4-6**.

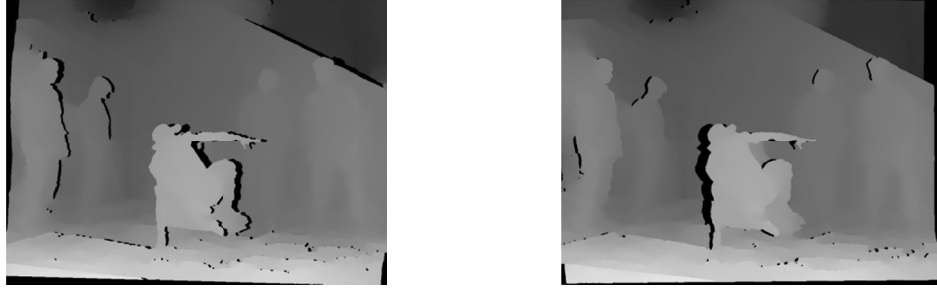


Figure 4-6. Median filter depth maps.

#### 4.3.2.3 Retrieve Texture Image by Inverse Warping

In this step, the textures are retrieved by performing inverse warping from filtered projected depth maps back to the reference cameras. **Figure 4-7** illustrates the image rendering process using inverse warping.

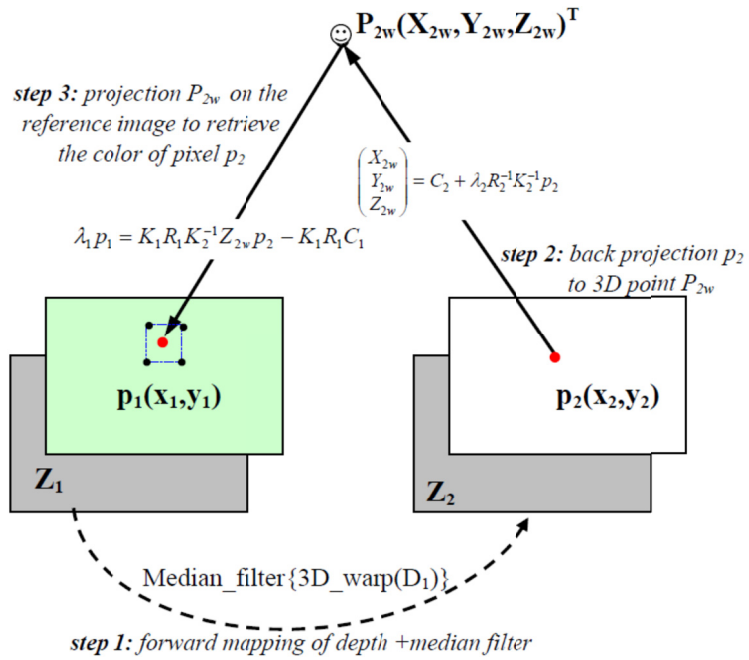
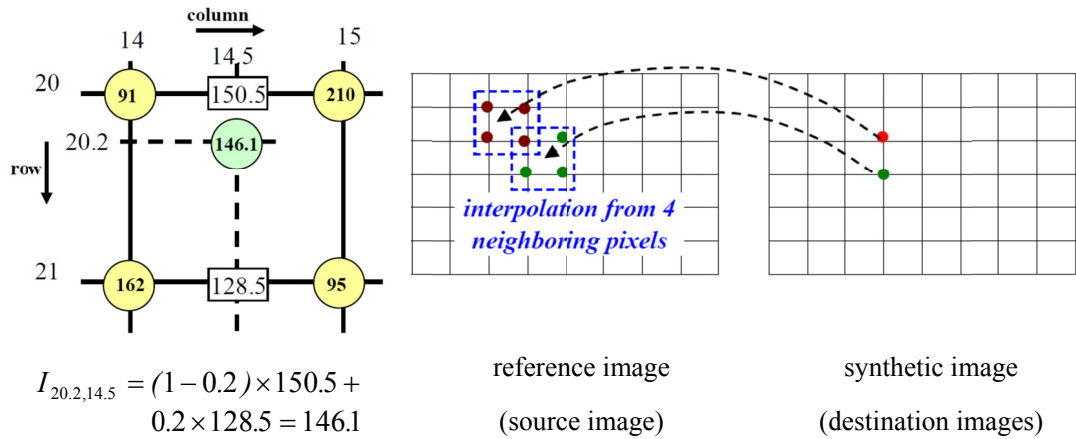


Figure 4-7. Image synthesis process using inverse warping.

For each pixel  $p_2$  of the filtered projected depth image, a 3D world point  $P_{2w} = (X_{2w}, Y_{2w}, Z_{2w})$  is calculated based on the Equation (4-3).  $Z_{2w}$  is defined by the depth value at coordinate  $p_2$  in the filtered projected depth image. Then, the calculated 3D point  $P_{2w}$  is projected onto respective reference textures image by employing the Equation (4-5). Finally, the color of the

### 4.3. Proposed Algorithm and Its Performance

synthetic destination pixel  $p_2$  is interpolated from the surrounding pixel  $p_1$  in the reference color image. An example of interpolating color of the destination pixel from the four neighboring pixels in the source image is shown in **Figure 4-8**.



**Figure 4-8. An example of interpolating color of the synthetic pixel from the four neighboring pixels in the reference image.**

This step can be specified by:

$$[I_{syn}, p_2] = 3D\_Warp^{-1}(Z_{syn\_filtered}, p_2). \quad (4-9)$$

The advantage of an inverse warping operation is that all pixels of the destination image are correctly defined and the color disoccluded pixels can be inferred by back projected 3D point  $P_{2w}$  onto multiple source image planes, covering all regions of video scene. **Figure 4-9** shows the retrieved color images by inverse warping using depth maps in **Figure 4-6**.



**Figure 4-9. Obtained color images by inverse warping.**

#### 4.3.2.4 Pixel Classification and Initial New View Creation

Formally, suppose that we have a set of  $N$  texture images  $I = \{I_1, I_2, \dots, I_N\}$  and  $N$  depth images  $Z = \{Z_1, Z_2, \dots, Z_N\}$ . Let  $I_m(p)$  and  $Z_m(p)$  be the color and depth value at position of  $m$ -th image.

In this step, we describe the type of pixels in the synthetic view. We go through each pixel  $p \in P$  of all  $N$  input images and classify as stable, unstable and disoccluded pixels. To detect the types of pixel, we set the thresholds (depth threshold  $t_z$  and color threshold  $t_c$ ) and examine the color and depth values for pixel  $p \in P$ . For each color channel, the color threshold  $t_c$  is set to be 15 in our case. Depth threshold is the brightness in the depth map. In our experiments,  $t_z$  is set to 5 for the 8 bits depth quantization.

A pixel is classified as:

+ if the depth value of a pixel  $p \in P$  at all  $N$  input depth images is less than depth threshold  $t_z$ , we classify the pixel  $p$  as the disoccluded pixel. The color and depth values of the pixel  $p$  at synthetic view are set temporally to zero.

$$I_{new}(p) = 0, Z_{new}(p) = 0, \quad \text{if } Z_k(p) \leq t_z, \forall k = 1, 2, \dots, N. \quad (4-10)$$

+ if the depth value of a pixel  $p \in P$  at only one input image is higher than the depth threshold  $t_z$  and at all remaining  $(N - 1)$  images is less than  $t_z$ , we classify the pixel  $p$  as the stable pixel. This is case the pixel  $p$  is visible in only one view. The values of the pixel  $p$  at synthetic view are just copied from the values of the pixel  $p$  in the visible view.

$$I_{new}(p) = I_k(p), Z_{new}(p) = Z_k(p), \quad \text{if } Z_k(p) > t_z, Z_m(p) \leq t_z, \forall m = 1, 2, \dots, N, m \neq k. \quad (4-11)$$

+ If the depth value of a pixel  $p \in P$  is higher than the depth threshold  $t_z$  in more than one view, we examine both the color and depth values of the pixel  $p$  to detect the types of pixel.

### 4.3. Proposed Algorithm and Its Performance

---

First step, for each view  $k, k = 1, 2, \dots, N$ , we examine pixel  $p$ . If the depth value of the pixel  $p$  is higher than the depth threshold  $t_Z$ , then we check other views  $j, j = 1, 2, \dots, N, j \neq k$ . If the view  $j$  has both a depth value of the pixel  $p$  higher than the depth threshold  $t_Z$  and has color similarity at  $p$  of view  $j$  and  $k$ ,  $I_j(p)$  and  $I_k(p)$  are called consistent color (the color similarity at pixel  $p$  of two input images  $j$  and  $k$  is defined based on the absolute color differences between  $I_j(p)$  and  $I_k(p)$  of  $R, G$  and  $B$  channels,  $|I_j(p) - I_k(p)| < t_C$ ). We count the total number of view  $j, j = 1, 2, \dots, N$  having the consistent color with view  $k$  ( $k = 1, 2, \dots, N, j \neq k$ ) at pixel  $p$ . Assuming that for each view  $k = 1, 2, \dots, N$ , this total number is  $S_k$ .

Second step, we find the biggest number of  $S_k$ , assuming that the biggest number is  $M$ .

If  $M \geq \lfloor N/2 + 0.5 \rfloor$ , we classify the pixel  $p$  as the stable pixel. Otherwise, the pixel  $p$  is classified as the unstable pixel. The value of unstable pixel can set to be -1 so that they can be easily identified.

The color and depth values of stable pixel  $p$  at synthetic view are rendered by blending  $M$  pixels as following weighted interpolation:

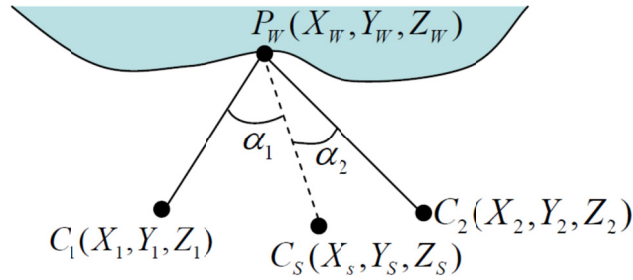
$$\begin{aligned} I_{new}(p) &= \left( \sum_{i=1}^M w_i * I_i(p) \right) / \sum_{i=1}^M w_i, \\ Z_{new}(p) &= \left( \sum_{i=1}^M w_i * Z_i(p) \right) / \sum_{i=1}^M w_i, \end{aligned} \quad (4-12)$$

where,  $w_i$  is the weight factor assigned to view  $i$ ,  $I_i(p)$  and  $Z_i(p)$  are color value and depth value of pixel  $p$  at view  $i$ . The weight assigned to each view should reflect its proximity with the view being synthesized. The views that are closer to the synthetic view should have a bigger weight. In general, case, the weight  $w_i$  can be set based on baseline spacing. However, for more precise weighting, we use the angle distance determined by the point in 3D and camera positions as shown in **Figure 4-10**. The weight factor  $w_i$  is calculated by



$$w_i = \begin{cases} e^{-c\alpha_i} & \text{if } \alpha_i < \pi/2 \\ 0 & \text{otherwise,} \end{cases} \quad (4-13)$$

where,  $i$  is view index,  $\alpha_i$  is the angular distance of view  $I$  and  $w_i$  is weight for the view at that pixel. The constant  $c$  controls the fall off as the angular distance increases. Input views for which  $\alpha_i \geq \pi/2$  are eliminated as they view the scene the other side. In practice,  $c = 1$  or  $2$  has been found to work well.



**Figure 4-10. Weighted interpolation based on angular distances.**

The new view is specified by

$$[I_{new}, Z_{new}] = InitialView(\{I_1, I_2, \dots, I_N\}, \{Z_1, Z_2, \dots, Z_N\}), \quad (4-14)$$

where, *InitialView* is the procedure of pixel classification and initial new view creation as above described.



**Figure 4-11. Initial synthesized view with 3 types of pixels: the white color pixels are unstable pixels, the red color pixels are disoccluded pixels and the remaining pixels are stable pixels.**

#### 4.3.2.5 Find the Best Candidate for Unstable Pixel by Graph Cuts

In this step, we focus on refining initial synthetic view with unstable pixels. Unstable pixels have multiple pixel candidates and we want to predict the best candidate that minimizes the energy function described in following part.

We denote  $L$  as labeling space with  $L = \{1, 2, \dots, N\}$ , representing the image index and let  $U$  be the set of unstable pixels. Let  $f_p$  be the label of unstable pixel  $p$  and  $f_p \in L$ . A labeling  $f$  is to assign a particular label  $f_p$  to a pixel  $p \in U$ . With this definition, our problem is to find the labeling  $f^*$  to fill the unstable region, such that the labeling  $f^*$  has minimum cost.

We define our energy function based on the MRF formulation:

$$E(f) = \sum_{p \in U} D_p(f_p) + \lambda \sum_{(p,q) \in N} V_{p,q}(f_p, f_q), \quad (4-15)$$

where,  $f$  is the labeling field,  $U$  is the set of unstable pixels, and  $N$  is the pixel's neighborhood system.  $D_p(f_p)$  is called the data term, which defines the cost of assigning label  $f_p$  to pixel  $p$ .  $V_{p,q}(f_p, f_q)$  denotes the smoothness term that evaluates the cost of disagreement between  $p$  and  $q$  which is assigned with  $f_p$  and  $f_q$  respectively.  $\lambda$  is a parameter to weigh the importance of these two terms.

Data term  $D_p(f_p)$  is defined by

$$D_p(f_p) = \alpha Z_{f_p}(p) \sum_{q \in N_p} (1 - O_q) |I_{f_p}(p) - I_{new}(q)| + \beta \sum_{i=1}^N |I_{f_p}(p) - I_i(p)|, \quad (4-16)$$

where  $N_p$  is neighboring pixels of  $p$ ,  $Z_{f_p}(p)$  is the depth value of pixel  $p$  at candidate  $f_p$ ,  $I_{new}(q)$  and  $O_q$  (0 or 1) are the color value and disoccluded indicator of pixel  $q$ , respectively.  $\alpha$  and  $\beta$  are weight factors.  $I_i(p)$  is color value of pixel  $p$  at input image  $i$ .  $|I_i(p) - I_j(q)|$  represents the sum of absolute color differences between  $I_i(q)$  and  $I_j(q)$  of R, G and B channels.

The first part of data term enforces the candidate pixel selected to agree with its neighboring pixels. In addition, the neighboring pixel that is disocclusion does not influence the candidate selection process. It is also penalized less cost for the selecting a candidate pixel which has smaller depth value  $Z$  because the pixel with smallest depth value is closer to the camera and more likely defined the color of synthetic pixel  $p_2$ .

The second part of Equation (4-16) is stationary cost, which defined based on color similarity at pixel  $p$  of all the input images. If the pixel  $p$  has similar color at more input images, the stationary cost is smaller.

Smoothness term  $V_{p,q}(f_p, f_q)$ : measures the penalty of two neighboring pixel  $p$  and  $q$  with different labels and is defined as follow:

$$V_{p,q}(f_p, f_q) = \frac{\|I_{f_p}(p) - I_{f_q}(p)\| + \|I_{f_p}(q) - I_{f_q}(q)\|}{2}, \quad (4-17)$$

where,  $\|\cdot\|$  denotes the Euclidean distance in RGB color spaces. The smoothness term gives a higher cost if  $f_p$  and  $f_q$  do not match well.

By incorporating such the smoothness term, we can achieve visually smooth in the synthetic image.

We apply graph cuts optimization that is public available in [83] to minimize our energy function  $E(f)$ . More detail about energy minimization with graph cuts can be found in [16, 21].

This step is specified by

$$I_{new}(U) = I_{f^*}, Z_{new}(U) = Z_{f^*}, \quad \text{with } f^* = \arg \min_f (E(f)). \quad (4-18)$$

The refinement of image in **Figure 4-11** by using graph cut to select the best candidate for unstable pixel is shown in **Figure 4-12**.



**Figure 4-12. Refinement of initial synthesized image (image in Figure 4-11) by using Graph cut (the red color pixels are disoccluded pixels).**

#### 4.3.2.6 *Inpainting Disocclusion Pixels Based on the Depth and Color Values of Neighboring Pixels*

Until this step, only the disocclusion regions are remaining. To deal with these disoccluded pixels, many papers such as [79, 82] have developed algorithms based on the inpainting method proposed by Telea [80]. Inpainting is a process of reconstructing lost or corrupted parts of images using the values of neighborhood pixels. Although, these algorithms work sufficiently well, the resulting inpainted regions contain a notable blur because of the mixture background and foreground colors at the edge of disoccluded regions. In this paper, we develop a technique based on inpainting method with depth information. We assume that the disoccluded pixels belong only to background, and we employ depth information to select accurately background pixels at the edges of disoccluded regions so that the blur can be avoided. Our method consists of several steps as follow.

First, for reducing processing time we find the small disoccluded regions by defining a window with the size of  $3 \times 3$  centered at  $p$  and counting the unstable pixel inside this window. If the number of visible pixels  $M$  inside this window is higher than 50%, then the disoccluded pixels is inpainted by a weighted interpolation from visible pixels, which is specified by

$$\begin{aligned}
 I_{new}(p_{occ}) &= \left( \sum_{i=1}^M d_i^{-1} * I_{new}(p_i) \right) / \sum_{i=1}^M d_i^{-1}, \\
 Z_{new}(p_{occ}) &= \left( \sum_{i=1}^M d_i^{-1} * Z_{new}(p_i) \right) / \sum_{i=1}^M d_i^{-1}, \quad \forall p_{occ} \in O,
 \end{aligned} \tag{4-19}$$

where,  $M$  is number of visible pixels inside the window.  $O$  is disoccluded region, and  $d_i$  is distance from disoccluded pixel  $p_{occ}$  to visible pixel  $p_i$ .  $I_{new}(p_i)$  and  $Z_{new}(p_i)$  are color and depth values of the visible pixel  $p_i$ .

Second, for each pixel  $p_o$  in remaining disoccluded regions we search in eight directions to find the pixel  $p_u$ , which has the smallest depth value  $Z_{\min}$  at the edge of disoccluded region and the distance  $d_u$  from this point to  $p_o$ . We define a window with the size of  $(d_u + \Delta) \times (d_u + \Delta)$  centered at  $p_o$  (at first,  $\Delta = 0$ ), and we count the visible pixels which have depth value  $Z$  with  $|Z - Z_{\min}| \leq 5$ . If there are not enough 50% of visible pixels inside the window, we increase the size of window by increasing  $\Delta$ . Finally, disoccluded pixels are inpainted by a weighted interpolation from visible pixels according to (4-19).

With inpainting procedure describing above, this step can summarized by

$$[I_{final}, Z_{final}] = \text{Inpaint}(I_{new}, Z_{new}). \tag{4-20}$$

### 4.3.3 Experimental Results

We quantify the proposal method performance based on Peak Signal Noise Ratio ( $PSNR$ ) and the Structural SIMilarity (SSIM) index [84] between a reference image  $I_r$  and a synthetic image  $I_s$ . The system for measurement the quality of synthesized view is illustrated in **Figure 3-17**.

We have adopted the  $PSNR$  as the quality metric for comparison for two reasons. First, an advantages of  $PSNR$  is that errors are measured on pixel basis so that errors resulting from the projection and synthesis will also to contribute to the measured  $PSNR$ . Second, it is the most proven and commonly used method to measure quality different by the research community. As describing

### 4.3. Proposed Algorithm and Its Performance

---

in section 3.5.3, before computing  $PSNR$ , the images are converted from RGB color space to YUV color space, and Y channel is used for calculation. Y channel is defined by Equation (3-26) and  $PSNR$  is calculated by using Equation (3-27).

SSIM index is a method for measuring the similarity between two images [84]. The difference with respect to PSNR is that PSNR estimates *perceived errors*; on the other hand, SSIM considers image degradation as *perceived change in structural information*. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. The SSIM index between two image signal  $x$  and  $y$  within the window  $N \times N$  is calculated as following:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (4-21)$$

where:

- $\mu_x$  and  $\mu_y$  are the average of  $x$  and  $y$ , respectively;
- $\sigma_x^2$  and  $\sigma_y^2$  are the variance of  $x$  and  $y$ , respectively;
- $\sigma_{xy}$  is the covariance of  $x$  and  $y$ ;
- $c_1 = (k_1L)^2$  and  $c_2 = (k_2L)^2$  are two variables to stabilize the division with weak denominator;  $L$  is the dynamic range of the pixel-values (typically  $L = 2^{\#bit \text{ per pixel}} - 1$ ;  $k_1 = 0.01$  and  $k_2 = 0.03$  by default).

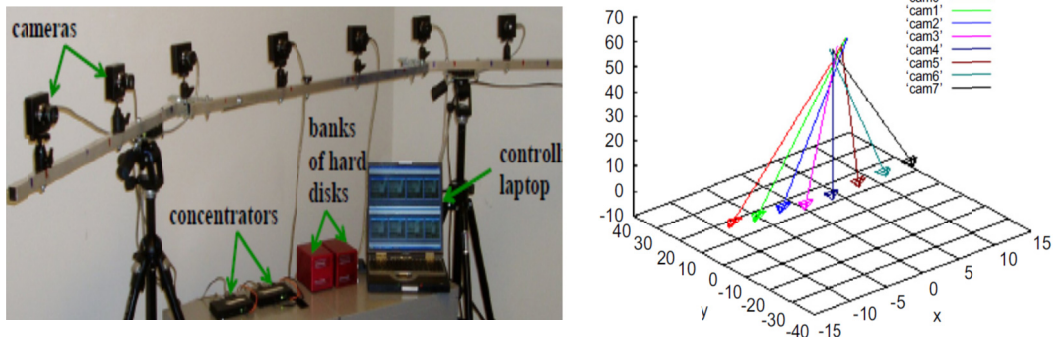
For image quality assessment, the local statistic  $\mu_x$ ,  $\sigma_x^2$  and  $\sigma_{xy}$  are typically computed within a local window  $8 \times 8$  square window, which moves pixel by pixel over the entire image. In practice, one usually requires a single overall quality measure of the entire image. We use the mean SSIM index to evaluate the overall image quality:

$$SSIM(R, S) = \frac{1}{M} \sum_{j=1}^M SSIM(R_j, S_j), \quad (4-22)$$

where,  $R$  and  $S$  are the reference and synthesized image, respectively;  $R_j$  and  $S_j$  are the image contents at the  $j$ -th local window. More detail is available online at [85].

The SSIM index value 1 is only reachable when two images are identical and the higher PSNR normally indicates that it is higher quality synthetic image.

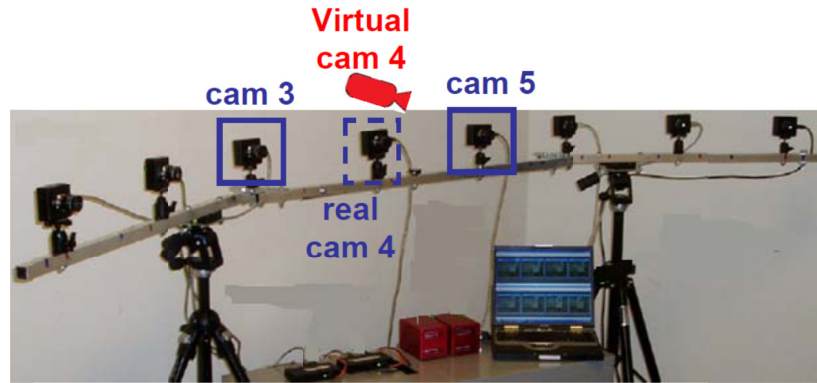
The proposed new view synthesis has been tested on “Break-dancer” and “Ballet” sequence which are generated and distribution by Interactive Visual Group at Microsoft Research [86]. These datasets include a sequence of 100 images of  $1024 \times 768$  pixels captured from 8 cameras with the calibration parameters. **Figure 4-13** shows the camera arrangement of these two sequences. Depth maps for each view are also provided. For more detail about these depth maps generation, please refer to [52].



**Figure 4-13. A configuration of “Break-dancer” and “Ballet” sequences with 8 cameras [52].**

In our experiment, the synthetic view is set to be the same as the actual camera. View 3 and 5 are used with depth maps to synthesize view 4. This experimental scenario is illustrated in **Figure 4-14** and the idea to objectively evaluate the quality of a synthesized image is shown in **Figure 3-17**.

#### 4.3. Proposed Algorithm and Its Performance



**Figure 4-14. Experimental scenario: view 3 and 5 are used to synthesize view 4.**

The example of view synthesis results is shown in **Figure 4-15**. The experimental results show that the proposed method achieved on average over 34 dB in PSNR and 0.93 index value in SSIM on the two sequence “Break-dancer “and “Ballet”.



(a) Original view image.



(b) Synthesized image (PSNR = 34.7 dB; SSIM= 0.94).



(c) Original view image.



(d) Synthesized image (PSNR = 34.6dB; SSIM= 0.95).

**Figure 4-15. Example of the synthetic view.**



## 4.3.3.1 PSNR and SSIM

Figure 4-16 shows our PSNR and SSIM comparison with those of Sohl et al. [87] over 100 frames for the “Break-dancer” and “Ballet” sequences.

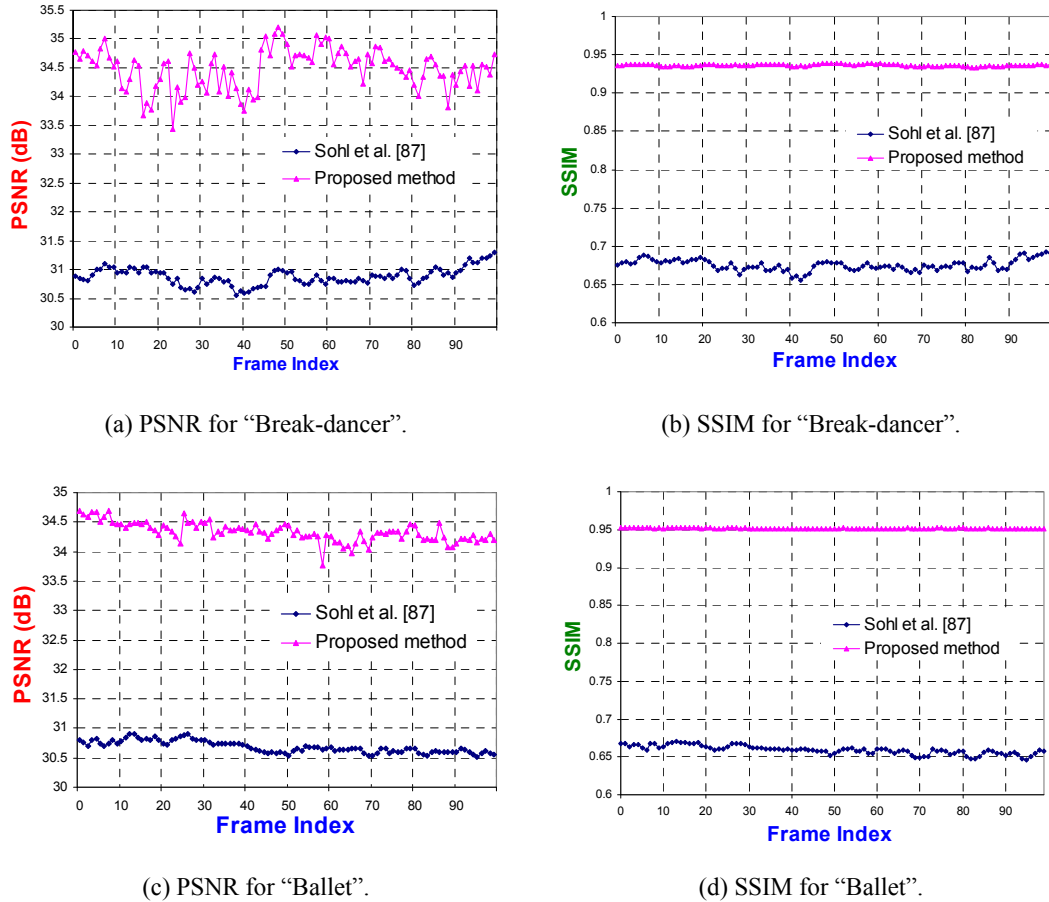


Figure 4-16. PSNR and SSIM comparison: (a) PSNR for “Break-dancer”, (b) SSIM for “Break-dancer”, (c) PSNR for “Ballet”, (d) SSIM for “Ballet”.

The measured synthetic image qualities are compared with other methods and summarized in Table 4-1. From the results, the average PSNR of proposal is superior to that of other methods such as Mori et al [79], Sohl et al. [87] with a gain of 3.0dB. The structure similarity (SSIM) of our method is higher than that of Sohl et al. method.

Moreover, in multi-view configuration, we have  $N$  cameras, which capture the scene at difference positions. For our experimental case, there are 8 cameras. Thus, instead of using only two camera views as above conventional methods, we can use more than two images to synthesize a new

### 4.3. Proposed Algorithm and Its Performance

view. Our proposal can extend to use as many views as necessary, simply by adding a processing path for supplementary view before pixel classification step (see **Figure 4-4**). Our experiment shows that using four reference views (two views on both left side and right side) to synthesis a new view, a higher PSNR (about  $0.5 \div 1dB$ ) and SSIM are obtained than the case of using two reference views.

**Table 4-1. The measured synthetic image qualities are compared with other methods.**

Method	“Break-dancer”		“Ballet”	
	PSNR(dB)	SSIM	PSNR(dB)	SSIM
Sohl et al. [87]	30.8	0.68	30.7	0.66
Mori et al. [79]	31.4	Not Reported	30.1	Not Reported
Proposed method	34.5	0.93	34.3	0.94

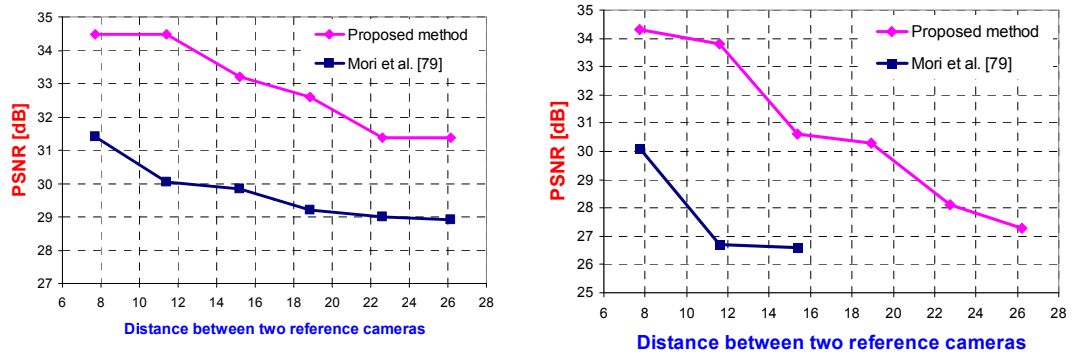
#### 4.3.3.2 PSNR versus Distance between References Cameras

Because usually the number of cameras is limited, the camera arrangement is very importance for obtaining a good quality of synthesized view. Here, we investigate the quality of synthesis with varying the distance between the two reference cameras. **Figure 4-17** shows our experimental setup.



**Figure 4-17. Varying the distance between the two reference cameras.**

**Figure 4-18** shows our quality of synthesis with varying the distance between the two reference cameras comparing the method presented by Mori et al in [79], where our measurements correspond to an average over 100 frames.



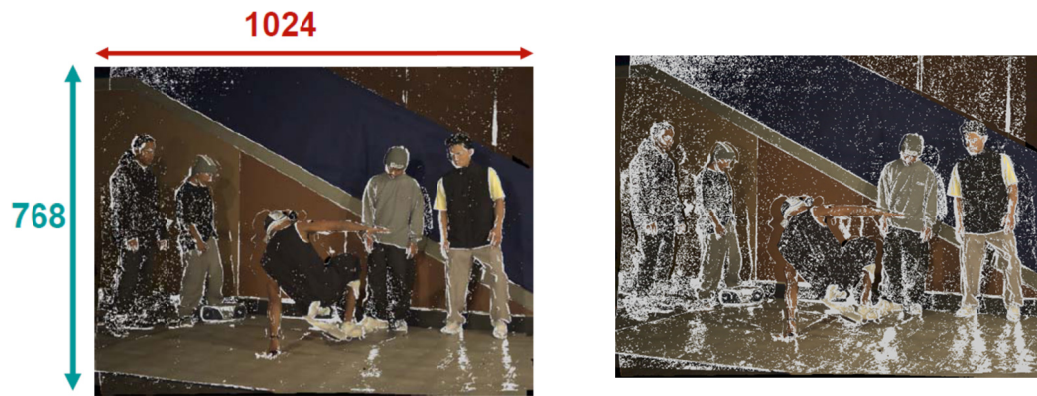
(a) "Break-dancer" sequence.

(b) "Ballet" sequence.

**Figure 4-18. PSNR versus distance between cameras for: (a) "Break-dancer" sequence, (b) "Ballet" sequence.**

#### 4.3.3.3 Timing Cost for Synthesis One View

Our experiments were performed on a PC with Intel Core i7, 3.2GHz and 8GB memory. We use C++ and OpenCV and no optimizing coding. The calculation time for one frame is the average time over 100 frames. We investigate the time for one synthesized frame with varying number of unstable pixels. **Figure 4-19** shows the number of unstable pixels in % comparing with image size.



(a) Number of unstable pixels: 6.7%

(a) Number of unstable pixels: 30.5%

**Figure 4-19. Number of unstable pixels (%) comparing to image size.**

Our result is shown in Table xxx. We can see that if the numbers of unstable pixels increase, the time for Graph Cut step is also increase. If the number of unstable pixels is 30% then the time for Graph cut is about 20% of the time for processing 1 frame.

**Table 4-2. Calculation time for one frame with varying numbers of unstable pixels.**

<b>Number of Unstable Pixels (%)</b>	<b>Time of Graph Cut (%)</b>	<b>Time of synthesis one frame (s)</b>
6.74	11.3	8.43
15.63	14.7	8.22
23.93	17.9	8.11
30.55	20.6	7.99

#### 4.4 SUMMARY AND CONCLUSIONS

In 3D-TV, the viewer can ideally navigate through the 3D domain and select his own viewpoint. The chosen viewpoint may not only be selected from available multi-view camera views, but also any viewpoint between these cameras. Obviously, this feature requires a smart synthesis algorithm that allows free-viewpoint view rendering. In this chapter, we have reviewed the recent advancements in viewpoint synthesis for 3D-TV and then proposed a novel method and showed its performance.

As discussing, an accurate manner for obtaining such a free viewpoint synthetic image is to employ a depth image based rendering method (DIBR). Such method assumes that the availability of a depth map for each camera image. The depth map encodes the distance to the viewer/camera for the content of each pixel in the camera image. More detail how to get the depth map is described in early *Chapter 3*. The basic idea of DIBR method is to perform 3D warping to the virtual viewpoint using texture image and depth map of the reference cameras. Most recent algorithms employ 3D warping from the two reference views to generate a virtual one, following the post processing procedure to enhance the quality of synthetic views.

In the second part of this chapter, we describe a novel synthesis method that enables to render a free-viewpoint from multiple existing cameras. The proposed method solves the main problems of depth based synthesis by performing pixel classification to generate an initial new view from stable pixels and using Graph cut to select the best candidate for unstable pixels. By defining the types of pixels and using Graph cuts, the color is consistent and the pixels wrapped incorrectly because of inaccuracy depth maps are removed. The remained disoccluded pixels are inpainted by using depth

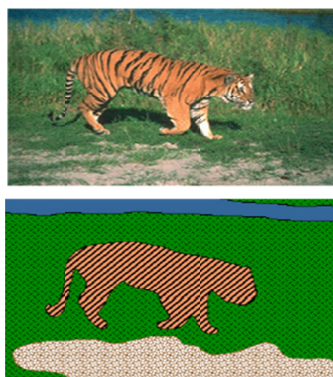
and texture neighboring pixel value. Considering depth information for inpainting, blurring between foreground and background textures are reduced. Experimental results show that the proposed method has strength in artifact reduction. In addition, our smooth term makes the result visually smooth. Objective evaluation has shown that our method get a significant gain in PSNR and SSIM comparing to some other existing methods. Another advantage of our method is that we can use a set of un-rectified images in multi-view system to create a new view with higher quality. The drawback of our method is using Graph Cuts, which is time consuming. However, we just only apply Graph Cuts for unstable pixels, which are a small amount of pixels comparing to whole image, so the time for Graph Cuts can be reduced.

## DEPTH ASSISTED OBJECT SEGMENTATION

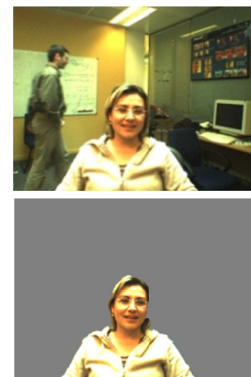
### 5.1 INTRODUCTION

In general, image segmentation can be defined as the process of finding groups of pixels that have similar characteristics. Image segmentation is one of the most common and active research topic in image processing, computer vision for recently decades. The goal of image segmentation is very application- oriented, which emerges in many fields such as object recognition, video monitoring, video indexing, digital entertainment, etc.

Image/video segmentation can be classified into region-based segmentation and object-based segmentation. Region-based segmentation aims to cluster the perceptually similar pixels in the image scene into homogenous regions while object-based segmentation tries to extract the meaningful object and separate foreground from background. Region-based segmentation focuses on the interpreting and understanding of the whole scene which is represented by semantically and geometrically consistent partitions as shown in **Figure 5-1(a)**. On the contrary, object-based segmentation pays more attention to access and manipulate the object-of-interests (OOIs), and the extracted objects are highlighted in the foreground mask as shown in **Figure 5-1(b)**.



(a) Region-based segmentation



(b) Object-based segmentation

**Figure 5-1. An example of region-based and object-based image segmentation.**

Partitioning the image into semantically multi-class regions is a challenging task due to various visual concepts involved. In certain applications, the users are more interested in accessing a specific

object rather than the whole scene, which motivates the development of object-based segmentation. The segmented semantic objects can provide the users with flexibility of object-level manipulation and access, and facilitate many content-related applications such as object recognition and retrieval, object-based entertainment and surveillance. In this chapter, we focus on the techniques of objects based segmentation.

Based on the type of source data, object based segmentation can be classified into image segmentation or video segmentation. Video segmentation may incorporate image segmentation technique to segment each frame into lots of homogenous regions. However, temporal coherence constraint in the sequences results in the difference between video segmentation and the segmentation of series of its single frame. Temporal coherence constraint addresses strong correlation of segmentations overtime, but the results can be quite unstable if segmenting them independently using image segmentation algorithm.

According to the different camera configuration, semantic objects can be segmented from single or multiple views of image/video. Most of interest has been focused on the research of mono-view segmentation leading to many advanced algorithms, theories and technologies. Especially object-based segmentation has drawn great attention from the research and industrial community, resulting in many commercial products with image cutout tools or user interface, such as Video SnapCut [88], Lazy Snapping [89], Ratio cut [90] and GrabCut [91]. These interactive object segmentation tools requires user intervention to provide foreground/background hints by brush strokes or bounding box, which can obtain high accuracy object regions.

On the other hand, multi-view segmentation had not attracted much attention due to the limitation of capture technology. With the recent growing capability of capturing devices, multi-view capture system with dense or sparse camera array can be built with ease, which motivates the development of multi-view techniques and its related applications. Based on the different methodologies involved, the existing algorithms of object detection and extraction from multi-view images can be grouped into three categories: background subtraction, visual hull-based and depth-based. According to the

## 5.2. Current Research on Depth Based Object Segmentation

---

assumption that the depth values over one object vary smoothly and continuously, the depth information recovered from multi-view images serves as an important cue for segmentation. However, due to ill-posed nature of depth estimation, errors may occur in the depth map. To obtain more robust segmentation results for object-level manipulation, integration of depth, color, and other image cues should be considered.

From the perspective of this study, we address the topic of depth-based object segmentation. A review the existing depth based algorithms for image/video segmentation and our proposed algorithm will be discussed in the following sections.

### 5.2 CURRENT RESEARCH ON DEPTH BASED OBJECT SEGMENTATION

Depth information reconstructed from multi-view images (MVI) usually serves as a valuable source in various related techniques such as 3D reconstruction [18], free-viewpoint synthesis (as described detail in *Chapter 4*), object tracking (will be described in next chapter) and object segmentation as showing in this chapter. Comparing with the 2D analysis and processing, the recovered depth information from the multi-view images assists in understanding and visualizing the 3D world in more efficient way. Accurate object segmentation in the clutter scene and complicated scenario is almost impossible or error-prone without any semantic knowledge about the scene or only relying on the 2D information (color, texture, and spatial location) from single-view images, since the semantic object is not always homogenous with these low-level features. By assuming that object locates in the different depth layer in the 3D scene and the depth value over one object forms smooth and consistent distributions, semantic objects can be extracted with known depth and segmentation performance using 2D features can be improved. However, object segmentation only exploiting the depth data is problematic due to the inaccuracy of the depth reconstruction resulting from the inherent difficulties of stereo matching such as the lack of textures and occlusion. Thus, to obtain more precise and robust segmentation for object-level manipulation, intelligent fusion of depth with other features should be taken into account.

In almost public papers, depth estimation and object segmentation from multi-view images are generally addressed in sequential [92], joint [93] [94] or iterative [95] [96] approaches.



The most straightforward approach for depth-based segmentation is to perform depth estimation beforehand, and then incorporate the depth information into the segmentation framework. In [92], authors described models and algorithms for bi-layer segmentation of stereoscopic frames. Stereo disparity is obtained by dynamic programming in Layered Dynamic Programming algorithm, and stereo match likelihood is then probabilistically fused with contrast-sensitive color model to segment each frame by ternary graph cut.

To avoid the propagation of error from depth estimation to foreground extraction in the sequential approaches, depth reconstruction and object segmentation problems can be simultaneously solved by joint optimization. For example, in [93], authors proposed a flexible and homogenous approach to simultaneous depth estimation and background subtraction in a multi-view setting, assisted by a static background image with known depth for each camera. The results of depth reconstruction and background separations algorithm is obtained as minimization of energy functional, to generation a dense depth map and foreground map. In [94], multi-view scene reconstruction and segmentation are dealt with by joint graph-cut optimization for the challenging outdoor environments with moving cameras, such as rugby and soccer scenes. Segmentation and depth labeling field are formulated into the unified energy function, which involves color and contrast term for segmentation, as well as the match and smoothness term for reconstruction.

The iterative depth-based segmentation receives the segmentation feedback from current estimation to improve the depth reconstruction and vice versa. In [95], the estimated depth map and segmentation mask are iteratively computed using an Expectation-Maximization (EM) algorithm. In [96], an iterative algorithm is developed to create the intermediate synthesized view using depth and segmentation information, which continuously performance the disparity estimation and the image segmentation in the iterative circle, and improve the result of each other.

In next section, we propose a method which is an approach in straightforward way. First, depth is estimated and then the depth information is incorporated into the segmentation framework.

### 5.3 PROPOSED SEGMENTATION ALGORITHM USING DEPTH INFORMATION

In this part, we propose a method, which requires no interactive operation, to segment human object from multi-view video. Our method consist of two stages: for initial frame of the video sequence, we automatically extract object based on saliency model and iterated Graph cut. After having segmented object in first frame, we propose the algorithm combining Bayesian estimation and minimizing energy function using Graph cut to segment object. In our energy function, the color, depth and spatial-temporal coherence are integrated in data term. Smooth term is encoded the penalty cost of the neighboring pixels with different labels.

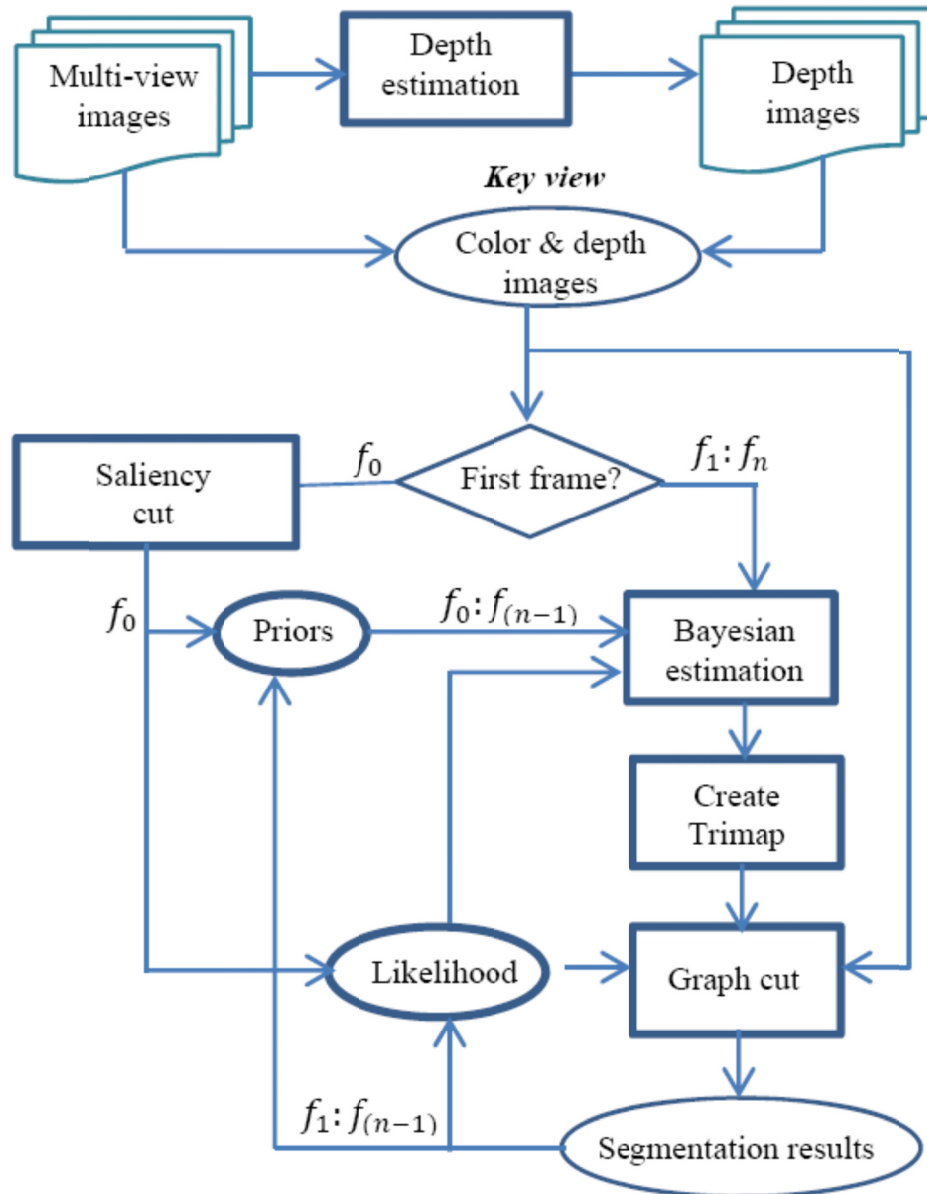
The following section is organized as: *section 5.3.1* will briefly introduce our method; *section 5.3.2* emphasizes on our proposal segmentation algorithm; and finally experiments are presented in *section 5.3.3*.

#### 5.3.1 Alogrithm Overview

Here, we focus on depth-based multi-view object segmentation. Depth information recovered from multi-view image/video serves as an important cue for our segmentation algorithm. We approach in straightforward way. First, depth is estimated and then the depth information is incorporated into the segmentation framework. This approach is reasonable because depth maps are becoming a readily available commodity of the multi-view pipeline. Depth information recovered from multi-view image/video serves as an important cue for our segmentation algorithm. The purpose of this research is to fuse color, with depth to robust object segmentation. The contribution point of our research is automatically created tri-map by Bayesian estimation. Created trimap is initial value to speed-up graph cut optimization algorithm.

In our framework, depth is estimated based on algorithms in [18], detail in *section 5.3.2*. But in many cases, depths are free given with multi-view colors images. For the starting step, if the input is first frames in sequences, we apply the novel method to segment object based Saliency model [97] and GrabCut [91]. From the second frame, we propose a probabilistic model combining Bayesian

estimation and Graph cut algorithm to segment the interested object, discussing in *sub section* 5.3.2.3.



**Figure 5-2. The illustration of proposal algorithm.**

**Figure 5-2** shows the entire algorithm. Our algorithm takes the color image of key view, depth image which is estimated from the multi-views as the inputs, and the segmentation result of the foreground region as the outputs.

### 5.3.2 Segmentation Algorithm

#### 5.3.2.1 Depth Estimation from Multi-view Images

Depth estimation aims at calculating the structure and depth of objects in a scene from a set of multiple views or images. This topic has been introduced in *Chapter 3*. In this chapter, depth is estimated by using graph cut approaches. More detail about depth estimation using graph cut can be found in series of papers [16,18,21].

Here, depth is estimated based on algorithms proposal in [18]. In [18], the data term enforces photo-consistency. Let  $I$  be a set of pair of “nearby” 3D points. These points will come from different view, but they will share the same depth (i.e., the points are of the form  $\langle p, f_p \rangle, \langle q, f_q \rangle$  where  $f_p = f_q$  and  $p, q$  are pixels from different views). Then the data term is:

$$\sum_{\langle (p,I), (q,I) \in I \rangle} D(p,q) T[f_p = f_q = I] \quad (5-1)$$

The smooth term, on the other hand, involves a single camera at a time. It is defined to be:

$$\sum_{\{p,q\} \in N} D_{p,q}(f_p, f_q) \quad (5-2)$$

where,  $N$  is a neighborhood system on pixels in a single view.

The energy function in [18] is minimizing by expansion move algorithm [16], which is efficient approximation graph cut algorithms. **Figure 5-3** shows on example of our depth estimation results.



**Figure 5-3 Stereo depth image.**

Depth map provides vital information for scene interpretation; therefore, depth maps are becoming a readily available commodity of the multi-view pipeline. We can make use of this new free information to our algorithm.

### 5.3.2.2 *Saliency Cut in the First Frame*

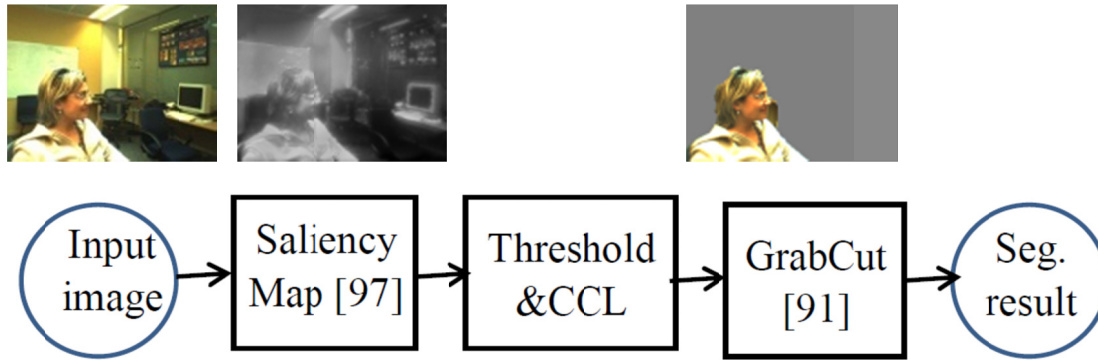
Most of graph cut based object segmentation algorithm need user's intervention to specify the initial foreground and background regions as the hard constraints such as in [89], [91] and [17]. User's interaction is helpful to obtained good segmentation results, but the initialization itself may be annoying the user especially when much guidance is needed.

In our proposal algorithm, the object will be automatically segmented but requiring only the object mask in the first frame in video sequence. To obtain the first frame object mask, we have two ways: manually or automatically locating and extracting object. For manually extracting objects, user can use some tools such as Lazy Snapping [89], GrabCut [91], or using simple background subtraction method with having background of the first frame.

In this section, our purpose is automatically locating and extracting object in the first frame. Visual attention concept gives us with smart mechanism to perceptually attract human's attention toward the location of interesting objects in a complicated scene. Saliency model in [98] is one of the earliest works. Give a static image, this model employs color, intensity and orientation to compute Saliency Map (SM), which encodes the obviousness at each location in the visual input. Until now, there are many saliency models have proposed such as in [97, 99-101].

In this work, we apply the ideal given in [97] and [99] to compute the saliency map. The process is demonstrated in **Figure 5-4**.

After having saliency map, we consider the use of this map to assist in salient object segmentation. Saliency maps have been previously employed for unsupervised object segmentation such as in [102, 103].



**Figure 5-4. Object segment based on SM and GrabCut.**

In our approach, we iteratively apply GrabCut [91] to refine the segmentation result initially obtained by threshold, dilation and erosion operating and Connected Component Labeling (CCL) on the saliency map (see **Figure 5-4**). Instead of manually selecting a rectangular region to initialize the process, as in classical GrabCut, we automatically define the bounding region based on the result of threshold and CCL on SM.

Once initialized, we iteratively run GrabCut to improve the saliency cut result. We apply dilation and erosion operations on the current segmentation result to get a new “seed” for the next GrabCut iteration. Our experiments need 3 or 4 iterations to obtain good result.

Our approach can be the semi-automatic object segment by given some guidance to GrabCut as in classical one when saliency model have failed to locate the desire object.

#### 5.3.2.3 *Probabilistic Model Combining Bayesian Estimation and Graph Cut*

##### 5.3.2.3.1 *Color and Depth Data Models*

This session, we briefly explain the color and depth probability model using in our algorithm.

#### ***Color Data Model***

Gaussian Mixture Model (GMM) in RGB color space is used for color data model. We use two GMM models with  $K$  component, one for background and one for foreground. The likelihood of pixel  $p$  with color  $C_p(r, g, b)$  (r,g,b: color values) belongs to the foreground or background can be written as:

$$P(C_p | f) = \sum_{k=1}^K w_{k,f} G(C_p; \mu_{f,k}, \Sigma_{f,k}), \quad (5-3)$$

where,  $f \in \{F, B\}$  representing foreground and background;  $w_{k,f}$  is a mixture weighting coefficient; and  $G(C_p; \mu_{f,k}; \Sigma_{f,k})$  is the  $k^{th}$  Gaussian component as:

$$G(C_p; \mu_{f,k}, \Sigma_{f,k}) = \frac{1}{(2\pi)^{3/2} |\Sigma_{f,k}|^{1/2}} \exp(-(C_p - \mu_{f,k})^T (\Sigma_{f,k})^{-1} (C_p - \mu_{f,k})). \quad (5-4)$$

Given the GMM model  $\underline{\theta} = \{w_{f,k}, \mu_{f,k}, \Sigma_{f,k}, f = (F, B), k = 1, \dots, K\}$  (i.e. the weight  $w$ , means  $\mu$ , covariance  $\Sigma$ , and  $2K$  Gaussian component for background and foreground), we can calculate the likelihood  $P(C_p | f)$  with  $f = \{F, B\}$  by using Equation (5-3).

### ***Depth Data Model***

Depth image is an array of gray values. Here we use histogram of gray values for depth data model  $h(p; f)$ . The same as color model, we need two histograms, one for foreground and another for background. Histograms are normalized to sum to 1 over gray level range  $\int_p h(p; f) = 1$  and we get likelihood of pixel  $p$  with depth value  $D_p$  as:

$$P(D_p | f) = h(p; f) \quad \text{with } f = \{F, B\}. \quad (5-5)$$

### **5.3.2.3.2 Bayesian Estimation and Trimap Creation**

This section computes the probability of each pixel to be in foreground base on Bayesian estimation and the results are used to create the tri-map, which is used for segmentation object via graph cut.

Let's  $C_p^t, D_p^t$  are color and depth value of pixel  $p$  on color and depth images at time  $t$ . The probability of pixel  $p$  belongs to foreground is calculated based on Bayes' formula as:

$$P^t(F | C_p^t, D_p^t) = \frac{P^t(C_p^t, D_p^t | F) P^t(F)}{P^t(C_p^t, D_p^t | F) P^t(F) + P^t(C_p^t, D_p^t | B) P^t(B)}, \quad (5-6)$$

### 5.3. Proposed Segmentation Algorithm Using Depth Information

---

where,  $P(x)$  is probability of  $x$ ,  $F$  and  $B$  stand for Foreground and Background in which pixel  $p$  belongs to.

We assume that color and depth are independent, so the likelihood:

$$\begin{aligned} P'(C_p, D_p | F) &= P'(C_p | F) P'(D_p | F), \\ P'(C_p, D_p | B) &= P'(C_p | B) P'(D_p | B). \end{aligned} \quad (5-7)$$

The likelihood  $P'(C_p | F)$  and  $P'(C_p | B)$  are calculated from foreground and background Gaussian Mixed Model (GMM) which are constructed from the previous segment result of color frame at  $(t - 1)$ .

Similarly, the likelihood  $P'(D_p | F)$  and  $P'(D_p | B)$  are calculated from gray-level histogram which also constructed from previous segment result of depth image at  $(t - 1)$ .

Because of successive frames in the temporal domain would have strong correlations, so the prior probability  $P'(F)$  and  $P'(B)$  of frame at  $t$  are calculated from the previous image frame at  $(t - 1)$ .

In order to get more accurate result,  $P'(F)$  and  $P'(B)$  can inferred from smooth map the 2D mask of segmentation results in previous frame at  $(t - 1)$  by performing Gaussian filter.

Based on computed prior probability and likelihood probability, the posterior probability  $P'(F | C_p, D_p)$  and  $P'(B | C_p, D_p)$  are calculated by Equation (5-6).

Applying this process for whole image pixel  $p$ , we get the probability image  $I_{prob}(p)$ . Based on probability value of pixel  $p$ ,  $I_{prob}(p)$  the tri-map  $T \{F: \text{Foreground}; B: \text{Background}; U: \text{Uncertainly region}\}$  can be created by:

$$\begin{cases} T(p) = B & \text{if } I_{prob}(p) \leq \text{threshold} \\ T(p) = F & \text{if } I_{prob}(p) > 1 - \text{threshold} \\ T(p) = U & \text{otherwise} \end{cases} \quad (5-8)$$

Here, threshold can be very small real value; in our experiment we choose threshold is 0.005. To remove the noise in trimap  $T$ , a filter to applied to foreground region and background regions.





**Figure 5-5. Tri-map (black color: background region; white color: foreground region; grey color: uncertainly region).**

### 5.3.2.3.3 Object Segmentation by Graph Cut

In this part, we focus on setting the data term and smooth term for our algorithm.

**Data term**  $E_{\text{data}}(\mathbf{f})$ : measures the agreement between the segmentation labels and observer data. We define this term as the weighted sum in following equation:

$$E_{\text{data}}(f) = \alpha \sum_{p \in U} \left\{ -\log(P(C_p | f_p)) - \log(P(D_p | f_p)) \right\} + (1 - \alpha) \sum_{p \in U} -\log(I_{\text{prob}}(p)), \quad (5-9)$$

where, the likelihood  $P(C_p | f_p)$  and  $P(D_p | f_p)$  can be obtained from GMM for color cue and histogram for depth cues as describing in section 5.3.2.3.1. Different from others previous related works, we add new term  $I_{\text{prob}}(p)$  as in the second row of Equation (5-9). This term encodes spatial temporal coherence to improve the segmentation result and can be reused the probability image in the creating tri-map step.

**Smooth term**  $E_{\text{smooth}}(\mathbf{f}_p, \mathbf{f}_q)$ : measures the penalty of two neighboring pixel  $p$  and  $q$  with different labels and is defined as follow:

$$E_{\text{smooth}}(f_p, f_q) = \gamma \sum_{(p,q) \in N} T[f_p \neq f_q] \frac{1}{\| \text{dist}(p, q) \|} \exp(-\beta \|C_p - C_q\|), \quad (5-10)$$

where,

$$T[f_p \neq f_q] = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{otherwise} \end{cases}. \quad (5-11)$$

### 5.3. Proposed Segmentation Algorithm Using Depth Information

---

$\|disp(p, q)\|$  and  $\|C_p - C_q\|$  are Euclidean distance of neighboring pixels in coordinate and color space respectively.  $N$  is the set of pairs of neighboring pixels. In practical experiment, a good results are obtained by defining pixels to be neighbors in 8-way connectivity (horizontal, vertical and diagonally). In [17], they had shown that it is more effective to set  $\beta > 0$  since this relaxes the tendency to smoothness in region of high contrast. We choose  $\beta$  the same in [17] as follow:

$$\beta = \left(2 \langle (C_p - C_q)^2 \rangle\right)^{-1}. \quad (5-12)$$

This choice of  $\beta$  ensures that the exponential term in Equation (11) switches appropriately between high and low contrast. Note that if  $\beta = 0$ , the smooth term is well-known Ising model, which encourages smoothness everywhere [91].

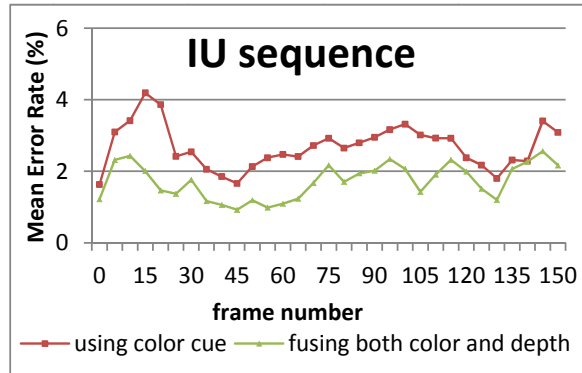
In our algorithm, the automatically generated tri-map  $T = \{T_F, T_B, T_U\}$  (see section 5.3.2.3.2) is set as the initial values for graph cut. This is different point of our algorithms comparing to interactive segmentation method such as GrabCut [91], which requires considerable degree of user interaction for supplying trimap. With our trimap, Graph cut optimization only need perform in uncertainly region  $T_U$ . However,  $T_U$  is smaller region than whole image, so the time for graph cut optimization decreases considerably.

#### 5.3.3 Experimental Results

To evaluate the performance of our method, we compare the segmentation results of our method with respect to the ground truth in the IU sequence, which can be freely downloaded from [104]. We define the Absolute Mean Error Rate (AMRE) of every fifth frame (in the left view) as the number of misclassified pixels over the total number of pixels in the image, which is the same measurement adopted in [92].

$$AMRE = \frac{1}{5} \sum_0^4 \frac{\text{misclassified pixel w.r.t ground trust}}{\text{total number of pixels in images}}. \quad (5-13)$$

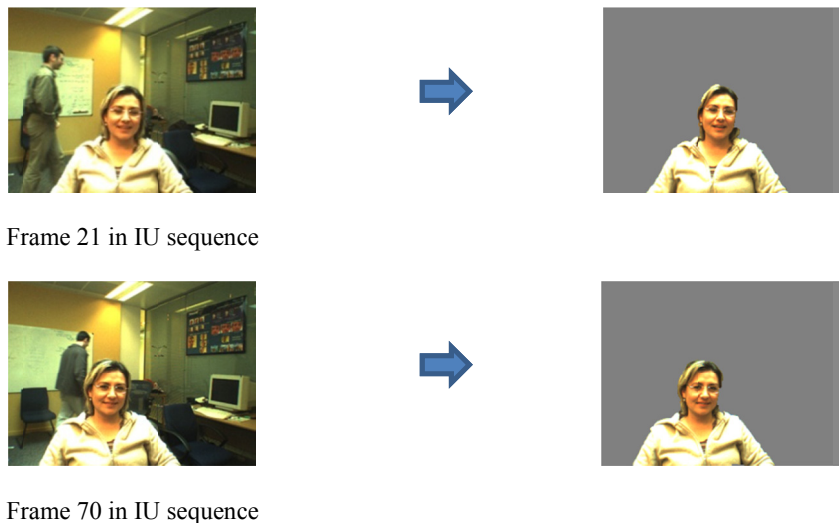
We make a comparison of two cases: 1) using only color/contrast cues, 2) fusing both color and depth cues. The comparison shows in **Figure 5-6**.



**Figure 5-6. Quantitative comparison.**

Using only color/contrast is a simply our algorithm in which the depth is switched off. As in **Figure 5-6** , when depth and color are fused we can get the better segmentation results. The segmentation in the case using both depth and color is more stable than only color/contrast case because absolute mean error rate is not much varying.

Many frames in IU sequence, the background is non-stationary (there are other people moving in the background), but our algorithm is also working well on these frame, as demonstration in **Figure 5-7**.



**Figure 5-7. Segmentation works well with non-stationary background.**

Besides using the IU sequence given in [104], we also set up the system to capture stereo video to show result of our algorithm. A common Minoru 3D webcam [58] is used to capture stereo video.

## 5.4. Conclusions

---

Bouguet's algorithm built in OpenCV [59] is used for camera calibration. Segmentation in our captured video also shows good results as in **Figure 5-8**.



**Figure 5-8. Segmentation on stereo video captured by Minoru 3D webcam**

We are not focusing on optimizing the running time, but we measure the time which Graph cut process is initialized by our trimap comparing with the case the graph cut algorithm is applied over whole image. Our proposal is faster, about 100 milliseconds for each frame. All our implementations use C++ and OpenCV [59].

## 5.4 CONCLUSIONS

In this chapter, we have introduced a framework to automatically segment the human region in multi-view video. Depth is estimated from multi-view image by Graph cut. Saliency model and iterated Graph cut are used to automatically locate and extract the interested object in the first frame to trigger the whole process. Depth is fused with color and spatial-temporal coherence in energy function. By combining Bayesian estimation, trimap is created automatically and used as initial value to speed-up minimizing energy functions via graph cut. Experiment results on various test sequences are encouraging.

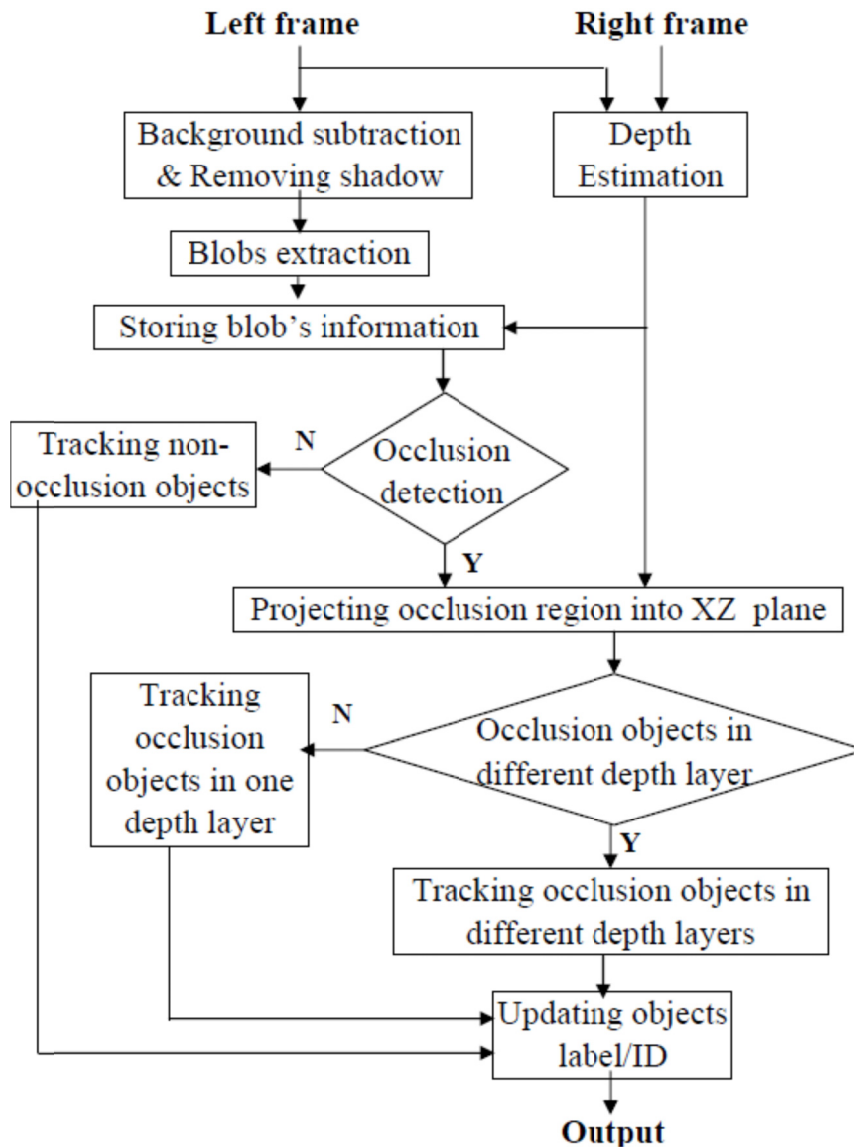
Our future work will be focused on improved this algorithm for multi-objects segmentation and efficient way to project the segmentation result in key-view to another views.

## DEPTH AIDED OBJECT TRACKING

### 6.1 INTRODUCTION

Object tracking in the video sequence has played an important role in a research area of computer vision and a wide range of applications, such as video monitoring and surveillance, video conferencing and video summarization. Based on different camera configurations, objects can be tracked by using a single camera or stereo/multiple cameras. Object tracking with a single camera has studied in many literatures and difference methods have been developed such as tracking by model-based tracking method [105], appearance-based methods [106-108], feature-based tracking [109], and statistical methods [110-112]. Many algorithms can obtain good results in some cases, such as when the targets are separated. However, multiple object tracking is still a challenging task due to the non-rigid motion of deformable object, persistent occlusion and the dynamic change of object attributes, such as color distribution, shape and visibility. In the real scene, occlusion between objects often occurs. For example, in typical surveillance scenario a person is partially or fully occluded by other people. Unfortunately, these occlusions lead to failed tracking. Some classical frameworks have been extended to track multi objects. In the multi-object tracking system [113], level set method is used to handle contour splitting and merging. Extensive methods, i.e. Monte Carlo based probabilistic methods [114], game theory based approaches [115] and appearance model based deterministic methods [116, 117] have been presented to solve the mutual occlusion problem. Another attractive research direction is stereo or multiple camera based method. While object detection and tracking with a single camera is a well-explored topic, the use of multi-cameras technology for this purpose has been attracted much attentions recently due to the availability and low price of new hardware. A multi-camera system observes the scene from two or more different views, and obtains more comprehensive information than a monocular camera system, which can take the advantage of depth information to improve the tracking system performance. Some tracking methods focus on usage of depth information only [118], or usage of depth information on better

foreground segmentation [119], or usage depth information as a feature to be fused in a maximum likelihood model to predict 3D object positions [120].



**Figure 6-1. The flowchart of proposed tracking method.**

In this work, we have presented a novel tracking method aiming at detecting objects and maintaining their label/identification across video frame sequence. The main points of this method are to use depth information and different strategies to track objects under various occlusion scenarios. **Figure 6-1** shows the flowchart of our tracking system.

The rest of this chapter is organized as follows: *section 6.2* presents the proposed tracking method. *Section 6.2.3* shows experimental results; and, finally, *section 0* concludes this chapter.

## 6.2 PROPOSED TRACKING METHOD

### 6.2.1 Method Overview

In this part, we have presented a novel tracking method aiming at detecting objects and maintaining their label/identification over the time. The main key factors of this method are to use depth information and different strategies to track objects under various occlusion scenarios. The foreground objects are detected and refined by background subtraction and shadow cancellation. The occlusion detection is based on information of foreground blobs in successive frames. The occlusion regions are projected to the projection plane XZ to analysis occlusion situation. According to the occlusion analysis results, different objects correspondence strategies are introduced to track object under various occlusion scenarios including tracking occluded objects in similar depth layer and in different depth layers. The experimental results show that our proposed method can track the moving objects under the most typical and challenging occlusion scenarios.

Our proposed tracking system is shown in **Figure 6-1** and it consists of below main steps.

### 6.2.2 Depth-Aided Tracking Multiple Objects under Occlusion

#### 6.2.2.1 Depth Estimation

Depth estimation aims at calculating the structure and depth of objects in a scene from two views or a set of multiple views. This topic has been introduced in *Chapter 3*.



**Figure 6-2. Color image and depth image.**

In this work, depth is estimated based on block matching algorithms proposal in [60]. This block matching technique is a one-pass stereo matching algorithm that uses sliding sums of absolute differences window between pixels in the left image and the pixels in the right image. An example of depth image is shown in **Figure 6-2**.

### 6.2.2.2 Foreground Segmentation and Shadow Cancellation

Our method performs foreground segmentation to speed up the process of object tracking. There are many foreground segmentation algorithms for instance of Gaussian Mixture Model [121, 122]. In our method, we use simple technique based on absolute differences between current image and background image.

In some cases, we have the fixed cameras observing the scene, so we may have an image of the background of the scene. However, in most case this background is not readily available. Moreover, the background scene often evolves over time because for example the light condition might change or because of new object could be added or removed from the background. Therefore, it is necessary to dynamically build the background model by regularly updating it. This can be accomplished by computing moving average using the following formula:

$$\mu_t = (1 - \alpha)\mu_{(t-1)} + \alpha p_t, \quad (6-1)$$

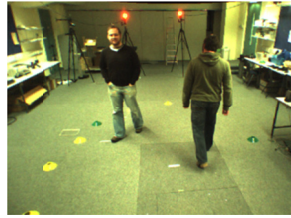
where,  $p_t$  is pixel value at a given time  $t$ ,  $\mu_{(t-1)}$  is the current average value, and  $\alpha$  is called the learning rate and it defines the influence of the current value.

In our method, first a color background model is created by computing a moving average for each channel (R, B and G channels of color image) of each pixel of incoming frames (around 10 frames). The decision to define a foreground pixel is simply based on comparing the current frame with background model and then updating this. Specifically,

$$F(p) = \begin{cases} 0 & \text{if } |I_c(p) - I_{bg}(p)| < t_H, \\ I_c(p) & \text{otherwise} \end{cases}, \quad (6-2)$$

where,  $F(p)$  is value of pixel  $p$  in foreground image,  $|I_c(p) - I_{bg}(p)|$  represents the absolute color difference between the color value at pixel  $p$  of current frame  $I_c(p)$  and the color value of pixel  $p$  of background frame  $I_{bg}(p)$  of R, G, B channels.  $t_H$  is threshold and for the each color channels this threshold can be set to  $0.3 * I_{bg}(p)$ . An example of foreground image is shown in **Figure 6-3**.





(a) input image



(b) segmented foreground image

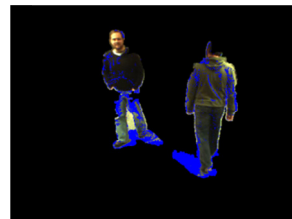
**Figure 6-3. Foreground segmentation.**

However, the segmented foreground image includes noise affected by the shadow. The shadow regions are the parts of moving objects. Shadow detection and removing will be used to refine the foreground. To avoid the effects of shadows, a shadow detection described in [123] is employed. More detail about this method, please refer to [123]. The result of the shadow detection is shown in

**Figure 6-4.**



(a) segmented foreground image



(b) shadow detection (the blue color pixels are the detected shadow)

**Figure 6-4. Shadow regions detection.**

### 6.2.2.3 *Blobs Extraction and Blobs' Information Store*

First, we try to extract the blobs of objects from segmented foreground image. Blob extraction is performed on the foreground binary image by connected component labeling using CvBlobLib library [124]. The foreground binary image can be obtained from simple threshold operation followed by the application of eroded and dilated operation on the segmented foreground image. CvBlobsLib provides two basic functionalities: extracting 8-connected components, referred to as blobs, in binary or grayscale images using Chang's contour tracing algorithm [125], and filtering the obtained blobs to get the objects in the image that we are interested in. In our method, we remove any blobs that have area small than 100 pixels.

## 6.2. Proposed Tracking Method

---

For every frame, after extracting blobs, information about blob is stored in a structured record for later processing steps. The blob's information includes total number of blob ( $NB$ ), blob's center coordination  $(x, y)$ , blob area ( $BA$ ), and the average depth value of blob ( $Z_{\min}$ ). Blobs are also given temporally identification ( $ID$ ).

We define two kinds of distance: the distance between blob  $i$  and blob  $j$  in the same frame  $t$  and the distance between blob  $i$  of frame  $t$  and blob  $j$  of frame  $(t-1)$ .

The distance  $d(i_t, j_t)$  between blob  $i$  and blob  $j$  in the same frame  $t$  is computed by:

$$d(i_t, j_t) = \sqrt{(x_{i_t} - x_{j_t})^2 + (y_{i_t} - y_{j_t})^2}, \quad (6-3)$$

where  $(x_{i_t}, y_{i_t})$  and  $(x_{j_t}, y_{j_t})$  are the center coordination of blob  $i$  and blob  $j$  at frame  $t$ , respectively.

Similarly, a distance  $D(i_t, j_{(t-1)})$  between blob  $i$  of frame  $t$  and blob  $j$  of frame  $(t-1)$  is calculated by:

$$D(i_t, j_{(t-1)}) = \sqrt{(x_{i_t} - x_{j_{(t-1)}})^2 + (y_{i_t} - y_{j_{(t-1)}})^2}, \quad (6-4)$$

where  $(x_{i_t}, y_{i_t})$  and  $(x_{j_{(t-1)}}, y_{j_{(t-1)}})$  are the center coordination of blob  $i$  at frame  $t$  and blob  $j$  at frame  $(t-1)$  correspondingly.

### 6.2.2.4 Occlusion Detection

We detect the occlusion in current frame  $t$  according to blobs' information at frame  $t$  and previous frame  $(t-1)$ . It is based on two clues. The first clue comes from the shortest distance between blobs at the same frame  $(t-1)$  and the second one is the difference of number of blobs at frame  $t$  and  $(t-1)$ . First, we find the shortest distance  $d(i_{(t-1)}, j_{(t-1)})$  between blobs in frame  $(t-1)$ , assuming that it occurs between blob  $m$  and blob  $n$ , i.e.  $d_{\min}(n_{(t-1)}, m_{(t-1)}) = \min\{d(i_{(t-1)}, j_{(t-1)})\}$ .

We define an occlusion flag  $f_{occ\_start}$ . This flag gets value  $t$  if occlusion is found at frame  $t$  and otherwise it gets value  $-1$ . Specially,

$$f_{occ\_start} = \begin{cases} t & \text{if } d_{\min}(n_{(t-1)}, m_{(t-1)}) < d_{threshold} \text{ and } NB_t < NB_{(t-1)}, \\ -1 & \text{otherwise} \end{cases}, \quad (6-5)$$

where,  $NB_{(t-1)}$  and  $NB_t$  are the total number of blobs in frame  $(t-1)$  and frame  $t$  respectively.

$d_{threshold}$  is the threshold of blob distance at the same frame.

Similarly, we also detect when the occlusion terminates. The end of occlusion is checked based on the shortest distance between blobs at the current frame  $t$  and the difference of number of blobs at current frame and previous frame  $(t-1)$ . We define the end of occlusion flag  $f_{occ\_end}$  as following:

$$f_{occ\_end} = \begin{cases} t & \text{if } d_{\min}(n_t, m_t) < d_{threshold} \text{ and } NB_{(t-1)} < NB_t, \\ -1 & \text{otherwise} \end{cases}, \quad (6-6)$$

### 6.2.2.5 Depth Aided Object Tracking

According to the result of occlusion detection, the tracking objects can be dividing into two types: tracking objects without occlusion and tracking objects under occlusion.

#### 6.2.2.5.1 Tracking Objects without Occlusion

The video objects correspondence under non-occlusion is obtained through the shorted distance  $D(i_t, j_{(t-1)})$  between blobs in previous frame and blobs in the current frame. This distance between blobs in previous frame and blobs in the current frame is calculated by Equation (6-4). For instance, once a foreground blob  $m$  at frame  $t$  ( $B_t^m$ ) finds its corresponding blob  $B_{(t-1)}^n$  in frame  $(t-1)$ , its label or identification ( $ID$ ) is updated correspondingly to the  $ID$  of blob  $B_{(t-1)}^n$ . Specially,

$$ID \text{ of } B_t^m \equiv ID \text{ of } B_{(t-1)}^n \\ \text{if } \begin{cases} D(B_t^m, B_{(t-1)}^n) = \min \{D(B_t^m, B_{(t-1)}^j)\}, j = 1, 2, \dots, NB_{(t-1)}, \\ D(B_t^m, B_{(t-1)}^n) < D_{thres} \end{cases}, \quad (6-7)$$

## 6.2. Proposed Tracking Method

---

where  $B_j^k$  denotes the blob  $k$  at frame  $j$ ;  $NB_{(t-1)}$  is number of blobs in frame  $(t-1)$ ;  $D_{thres}$  is distance threshold.

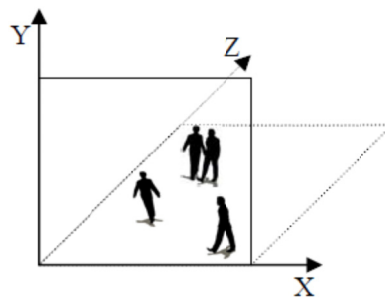
### 6.2.2.5.2 Tracking Objects under Occlusion

The main idea of our tracking method is that the object label or identification ( $ID$ ) is maintained constantly during occlusion and after they switch their positions.

When occlusion occurs, we can detect and extract the occlusion region. We also can detect and separately extract a list of objects that are non-occlusion objects in previous frame but overlaying each other in current frame.

In order to track the objects under occlusion, depth information is used to analysis the occlusion situation. First, the occluded regions are projected to the ground plane  $XZ$  according to their horizontal position and their depth gray level (more detail in the next subsection). Then according to the  $XZ$  plane, the occlusion objects can be divided into two types based on the depth ranges: a) in the different depth layer or b) in the same depth layer. We are dealing with these situations as following parts.

- **Project Occlusion Regions into Ground Plane  $XZ$**

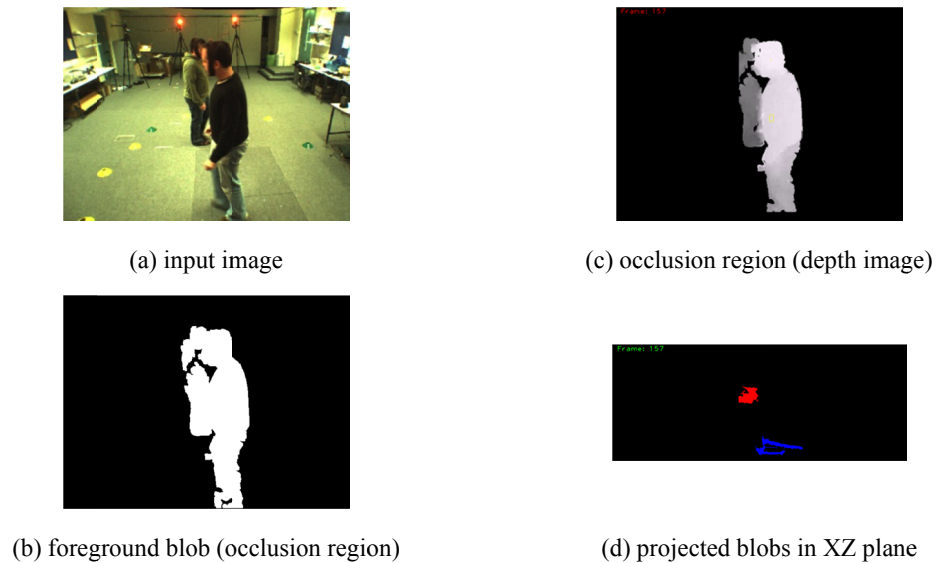


**Figure 6-5. The image plane  $XY$  and ground plane  $XZ$ .**

Each foreground pixel in occlusion region has 3D information obtained from depth map. These pixels are projected to the ground plane  $XZ$  according to their horizontal position and their depth gray level, where  $X$  is the width of the depth map and the range of  $Z$  is  $[0, 255]$ . The projected point, which is located at  $(x, z)$ , is defined as  $p(x, z)$ . The value at position  $p(x, z)$  of projection

plane is the total number of points at position  $x$  in the depth maps that have same gray level (depth value  $z$ ). **Figure 6-5** illustrates the image plane  $XY$  and ground plane  $XZ$ .

In order to remove noisy points, if the value at point  $p(x, z)$  is less than threshold  $T_1$  the point  $p(x, z)$  will be discarded. Then we also apply morphological operations (dilating and eroding operation) to remove noisy points and connect nearby points. The remaining points in  $XZ$  plane are grouped in to the blobs that are based on connected component analysis technique using CvBlobLib library [124]. If a projected blob is small than threshold  $T_2$ , it is consider a noise and it will be removed. The projected blobs are defined as  $\{PB_j | (j = 1, 2, \dots, m)\}$ , where  $m$  is total number of projected blobs. Each projected blobs  $PB_j$  is mask as object regions. **Figure 6-6** shows an example of projected blobs in  $XZ$  plane.



**Figure 6-6. Projected foreground blobs in XZ plane.**

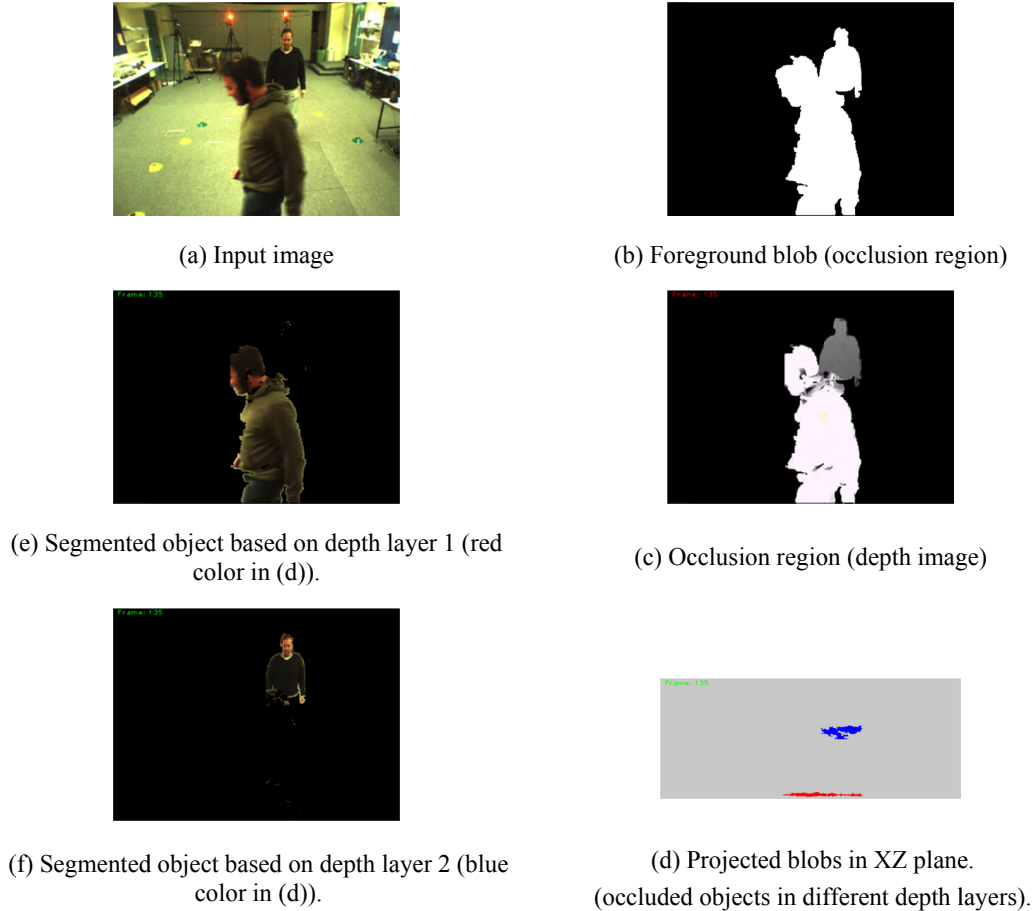
As mention before, according to the projected blobs in  $XZ$  plane, the occlusion objects can be divided into two types based on the depth ranges: a) in the different depth layers or b) in the one depth layer.

- **Tracking Occluded Objects in Different Depth Layers**

## 6.2. Proposed Tracking Method

**Figure 6-6** shows the case of occluded objects is in different depth layers. Once occluded objects have different depth layer, they can be segmented in color image by means of their depth ranges.

An example of color image segmentation by means of depth ranges is shown **Figure 6-7**.



**Figure 6-7. An example of image segmentation by means of depth ranges.**

In our method, object correspondence under different layer is based on Bhattacharyya distance [126] between the color histograms. In statistics, the Bhattacharyya distance measures the similarity of two probability distributions. In our case, Bhattacharyya distance represents the similarity between two normalized histograms. The Bhattacharyya distance is calculated by:

$$BD(H_1, H_2) = \sqrt{1 - \sum_{b=1}^N \sqrt{H_1(b) \cdot H_2(b)}}, \quad (6-8)$$

where,  $BD$  denotes Bhattacharyya distance;  $H_1$  and  $H_2$  are the two normalized color histograms;  $N$  is number of bin in histogram.

Let  $O^k = \{O_1^k, O_2^k, \dots, O_m^k\}$  is the occluded objects at frame  $k$  and  $U^n = \{U_1^n, U_2^n, \dots, U_m^n\}$  denotes the existing non-occluded objects in previous frame  $n$  (the frame before occlusion is found). The color histogram of each occluded objects and existing non-occluded objects are  $H_{O_i^k}$  and  $H_{U_j^n}$ , respectively. In this paper, color histograms are created from hue component of *HSV* color space. **Figure 6-8** shows an example of the color histogram.

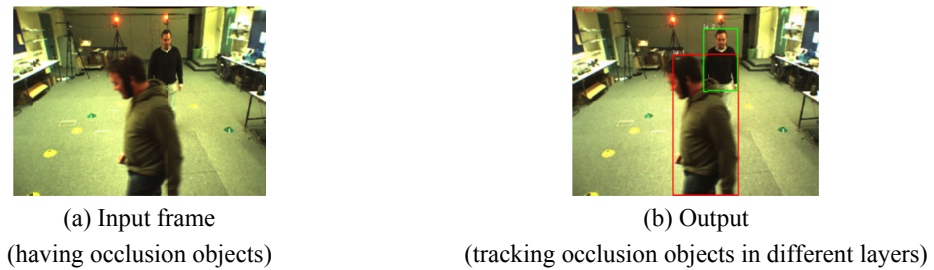


**Figure 6-8. An example of color histogram.**

For each occluded object  $O_i^k$ , we calculate the Bhattacharyya distance  $BD(O_i^k, U_j^n)$  between color histogram of this object and color histogram of every object  $U_j^n$  in  $U^n$  and then find the shortest distance  $BD_{\min}$ . The Bhattacharyya distance  $BD(O_i^k, U_j^n)$  is computed according to Equation (6-8), specially:



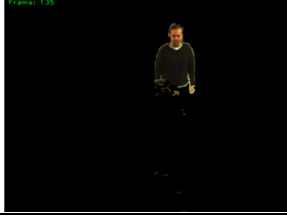

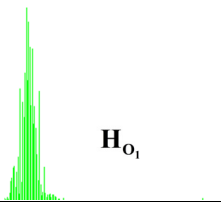
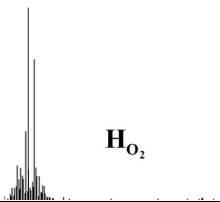

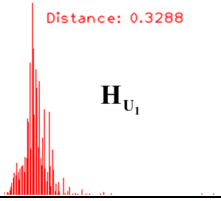
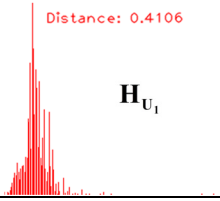

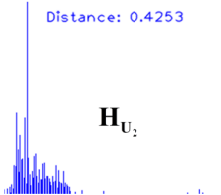
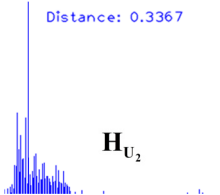
$$BD(O_i^k, U_j^n) = \sqrt{1 - \sum_{b=1}^N \sqrt{H_{O_i^k}(b)H_{U_j^n}(b)}}. \quad (6-9)$$

The occluded object  $O_i^k$  will update its *ID* according to  $U_j^n$  if  $BD(O_i^k, U_j^n) = BD_{\min}$ . **Figure 6-10** shows the numeric results of calculating Bhattacharyya distance between two histograms. **Figure 6-9** illustrates an example of tracking occluded objects in different depth layers.



**Figure 6-9. Tracking occluded objects in different depth layers.**

## 6.2. Proposed Tracking Method

		
<p>(a) Input frame (having occlusion objects)</p>	<p>(b) Segmented obj. 1 in input frame (<math>O_1</math>).</p>	<p>(c) Segmented obj. 2 in input frame (<math>O_2</math>).</p>
	 <p style="text-align: center;"><math>H_{O_1}</math></p>	 <p style="text-align: center;"><math>H_{O_2}</math></p>
<p>(d) Previous frame (Before occlusion happened).</p>	<p>(b1) Histogram of object <math>O_1</math> in (b) (<math>H_{O_1}</math>)</p>	<p>(c1) Histogram of object <math>O_2</math> in (c) (<math>H_{O_2}</math>)</p>
	<p style="text-align: center;"><math>BD(H_{O_1}, H_{U_1})</math></p>  <p style="text-align: center;"><math>H_{U_1}</math></p> <p style="text-align: center;">Distance: 0.3288</p>	<p style="text-align: center;"><math>BD(H_{O_2}, H_{U_1})</math></p>  <p style="text-align: center;"><math>H_{U_1}</math></p> <p style="text-align: center;">Distance: 0.4106</p>
<p>(e) Object 1 before occlusion happened (<math>U_1</math>)</p>	<p style="text-align: center;">(e1) Histogram of object <math>U_1</math> in (e) (<math>H_{U_1}</math>)</p> <hr style="border-top: 1px dashed black;"/> <p style="text-align: center;">distance between <math>H_{O_1}</math> and <math>H_{U_1}</math> is 0.3288</p> <p style="text-align: center;">distance between <math>H_{O_2}</math> and <math>H_{U_1}</math> is 0.4106</p>	
	<p style="text-align: center;"><math>BD(H_{O_1}, H_{U_2})</math></p>  <p style="text-align: center;"><math>H_{U_2}</math></p> <p style="text-align: center;">Distance: 0.4253</p>	<p style="text-align: center;"><math>BD(H_{O_2}, H_{U_2})</math></p>  <p style="text-align: center;"><math>H_{U_2}</math></p> <p style="text-align: center;">Distance: 0.3367</p>
<p>(f) Object 2 before occlusion happened (<math>U_2</math>)</p>	<p style="text-align: center;">(f1) Histogram of object <math>U_2</math> in (f) (<math>H_{U_2}</math>)</p> <hr style="border-top: 1px dashed black;"/> <p style="text-align: center;">distance between <math>H_{O_1}</math> and <math>H_{U_2}</math> is 0.4253</p> <p style="text-align: center;">distance between <math>H_{O_2}</math> and <math>H_{U_2}</math> is 0.3367</p>	

**Figure 6-10. Bhattacharyya distance between two histograms.**



• Tracking Occluded Objects in One Depth Layer

Figure 6-11 shows an example of occluded objects in similar depth layer.

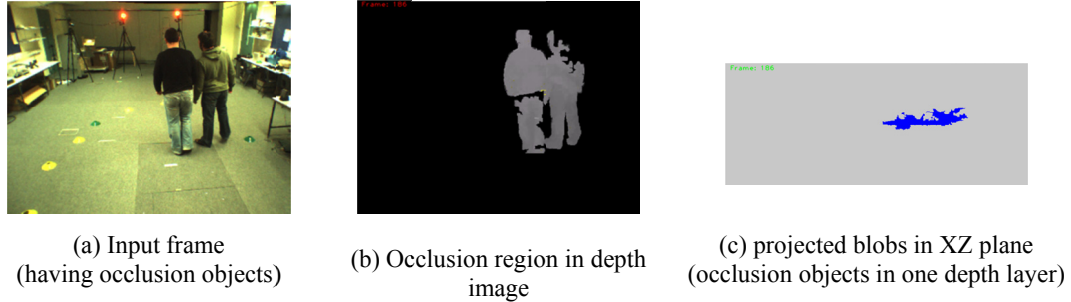


Figure 6-11. Tracking occluded objects in different depth layers.

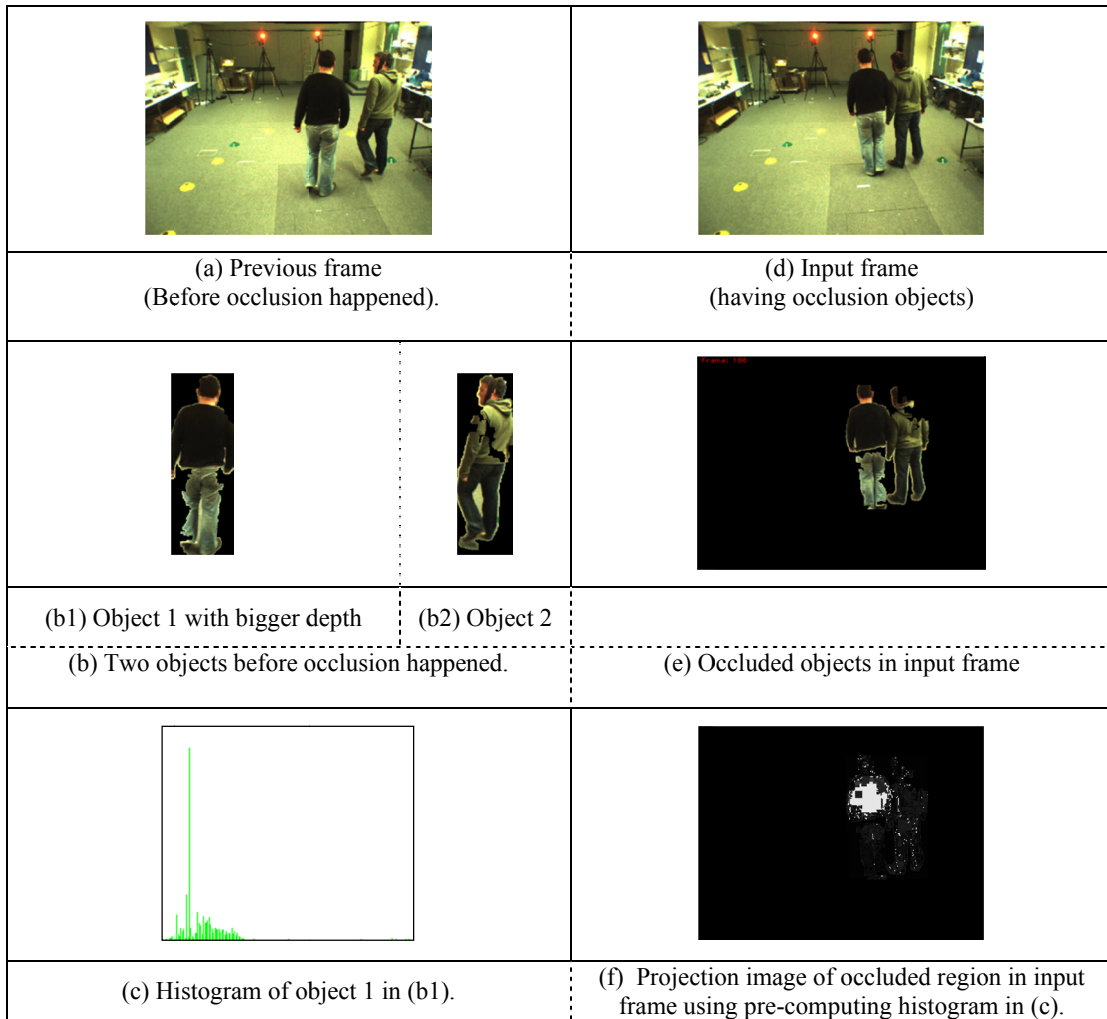
When occlusion objects have similar depth range or full occlusion, it is difficult to segment and track multiple objects as above technique. To deal with this problem, we propose the tracking method based on Camshift (Continuously Adaptive Mean Shift) algorithm [127] as following part.

Assuming that there are  $m$  occluded object  $O_i^k$  ( $i = 1, 2, \dots, m$ ) in the occlusion region  $R_{occ}^k$ , their existing corresponding tracks in the previous frame are  $U_j^{(k-1)}$ . Our algorithm has following steps:

- (i) Pre-computing the color histogram  $H_U$  for every existing track  $U_j^n$ , and the color histogram  $H_R$  for occlusion region. Here we calculate a hue histogram from  $HSV$  color space.
- (ii) Based on the average depth values, sorting the list of object  $U_j^{(k-1)}$  so that the object with biggest depth  $U_{foremost}^{(k-1)}$  (i.e. the shortest distance from camera) goes first.
- (iii) Calculating a back projection of a hue plane (see an example of projection image in **Figure 6-12**) of occlusion region using the pre-computing histogram of  $U_{foremost}^{(k-1)}$ . Based on the back projection image, finding in  $R_{occ}^k$  the corresponding object of  $U_{foremost}^{(k-1)}$  using camsift algorithm. Label it as  $O_{foremost}^k$  with  $ID$  according to the  $ID$  of  $U_{foremost}^{(k-1)}$ .

## 6.2. Proposed Tracking Method

- (iv) Removing the  $O_{foremost}^k$  in  $R_{occ}^k$ . Selecting the next track in the sorted  $U_j^{(k-1)}$  list and running step (iii) to find the next  $O_j^k$  in  $R_{occ}^k$ .
- (v) Repeating step (iv) until all objects in  $O^k$  in  $R_{occ}^k$  finds their corresponding track.

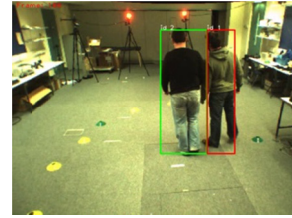


**Figure 6-12. An example of projection image.**

**Figure 6-13** demonstrates a result of tracking occluded objects in one depth layer.



(a) Input frame  
(having occlusion objects)



(b) Output  
(tracking occlusion objects in different layers)

**Figure 6-13. An example of tracking occluded objects in one depth layer.**

In practice, when projecting partly occluded objects or fully occluded objects with similar depth ranges into  $XZ$  plane, we will obtain only one blob/region in  $XZ$  plane. In the case, the objects in similar depth range are partly occluded, the above algorithm will work well (see **Figure 6-13**). However, the fully occluded objects current frame  $t$  will reappear as partial occluded objects bind their occluder in the later frame, so it will be tracked by our method (see example in **Figure 6-17**).

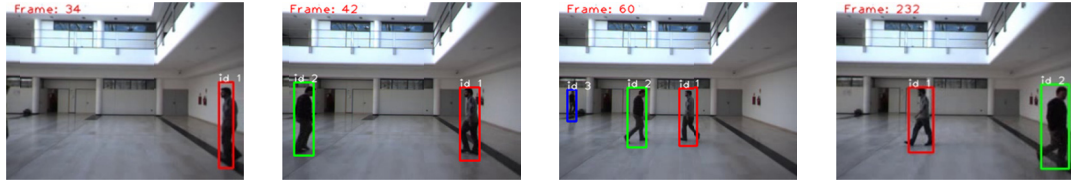
### 6.2.3 Experimental Results

In this section, we show the experimental results to evaluate the proposed tracking method. We evaluate the tracking performance by the capability of detecting and maintaining constant  $ID$  of foreground objects during the occlusion and after the occlusion over.

The proposed tracking method has been test on some video sequences. The input of out method is a pair of video sequence and the output is the left video sequence in which has a set of moving objects labeling with  $ID$  and bounding boxes with different color.

**Figure 6-14** shows results of tracking non-occlusion objects. In the example **Figure 6-14(a)**, in the 3 frames (left to right) each object appears one after another and in the fourth frame, object with  $ID = 3$  has gone out the scene. These results illustrate the ability of our algorithm in detecting object, assigning and maintaining the objects'  $ID$ .

## 6.2. Proposed Tracking Method



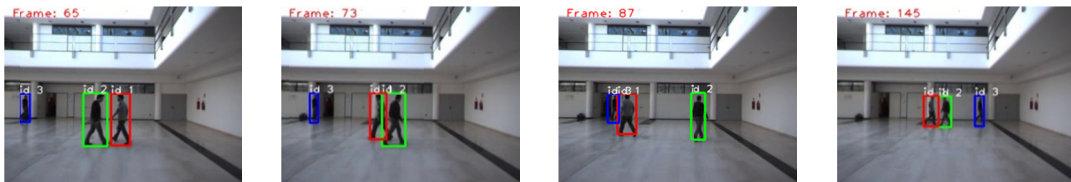
a) Tracking of non-occluded objects in “Gym sequence” (from left to right, the frame numbers are: 34, 42, 60 and 201)



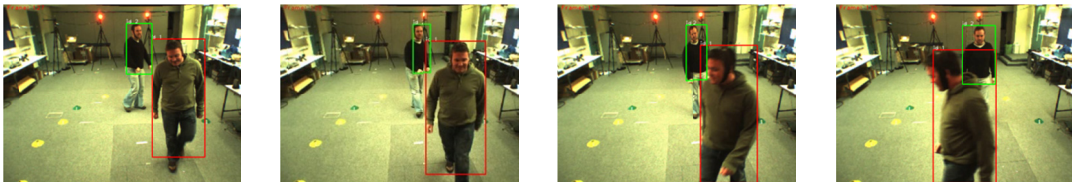
b) Tracking of non-occluded objects in “Room sequence” (from left to right, the frame numbers are: 0, 19, 113 and 250, respectively).

**Figure 6-14. Result of tracking non-occluded objects.**

We demonstrate the result of tracking of occluded objects in different depth layers in **Figure 6-15**. . Our proposed method can successful detect the object under partial occlusion. All of objects have constantly label over the time even they are moving in variety of pose and position.



(a) “Gym sequence”: from left to right, the frames indexes are 65, 73, 87 and 145, respectively.



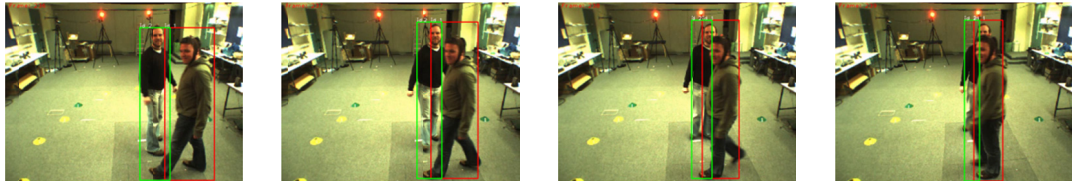
(b) “Room sequence”: from left to right, the frame indexes are 127,129, 132 and 135, respectively.

**Figure 6-15. Tracking occluded objects in different depth layers.**

Tracking the occluded object under the similar depth layer is shown in **Figure 6-16**. These examples show that the proposed algorithm can detect and track a partial occluded object that is equivalent to at least one half a human body.



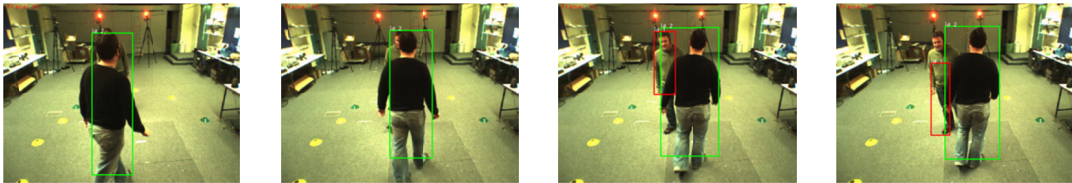
(a) “Gym sequence”: from left to right, the frames indexes are 206, 207, 208 and 209, respectively.



(b) “Room sequence”: from left to right, the frame indexes are 236,237, 238 and 239, respectively.

**Figure 6-16. Tracking occluded objects in one depth layer.**

**Figure 6-17** shows the case an object is fully occluded and in the similar depth layer with its occluder and our system cannot detect it. However, in the future time when this object reappears as partial occluded object, the system can detect and maintain its ID. This example demonstrate the capability of proposed algorithm in term of maintain constant label for object over the video sequence.



“Room sequence”: from left to right, the frame indexes are 67,72, 73 and 74, respectively.

**Figure 6-17. Tracking fully occluded objects.**

### 6.3 CONCLUSIONS

In this chapter, we have presented a novel tracking method aiming at detecting objects and maintaining their ID over the time. The main key factor of this method is to use depth information to help to track objects under various occlusion scenarios. Different object tracking strategies are apply according to occlusion situation including finding correspondence object based on Bhattacharyya distance between two histograms and using camshift based algorithm with the help of object depth ordering. The experimental results presented have confirmed the capability of our proposed objects tracking algorithm under the most typical and challenging occlusion scenarios.

However, the proposed algorithm can work only in an indoor or medium sized environment since the reliability of depth information diminished in proportion to the distance from camera and only when the moving velocity of objects are slow. In the future work, to construct a robust moving object tracking system in both indoor and outdoor environment, we will study to use more object's features to classify and track the objects.

## CONCLUSIONS AND OUTLOOK

### 7.1 SUMMARY AND CONCLUSIONS

This thesis devotes to firstly study depth estimation from stereo/multi-view images and then use this useful information for one of the key application in 3DTV, namely free viewpoint synthesis, and for object segmentation and multiple moving object tracking applications. The achievements are summarized as follow parts.

- **Chapter 3: Depth Estimation**

We have proposed a method that allows the use of several un-rectified images simultaneously to estimate a consistency and reliability depth image. We have introduced three constraints, i.e. intra-line, inter-line and inter-view smoothness constraint, which enforce smooth variations of depth value in the scanline, across scanline and consistent depth value across the views. The proposed algorithm combines two stages: the first stage serves as a calculation of initial depth images and the second stage enhances the depth initial depth images in the first step by enforcing consistent depth across the views. The three smooth constraints can be efficiently integrated into one dimensional optimization dynamic program algorithm. Experiments have shown that the proposed smooth constrains yield reasonably depth image quality for various multi-view data sets. Although, depth estimation has been widely studied, we have some small contributions as:

- Using several uncertified images simultaneously. As we known, in the multi-view camera configuration, multiple images are available so that the algorithm employ all view could yields an accurate depth map comparing the common case using only two views. Furthermore, with uncertified images we do not need the image rectification pre-processing or image de-rectification post-processing;
- Adding the smoothness constraints which enforce smooth variations of depth value in the scanline, across scanline and consistent depth value across the views.

## 7.1. Summary and Conclusions

---

This proposed method was published in the Proceeding of the fourth International Conference on Communication and Electronics (ICCE), 1<sup>st</sup> -3<sup>rd</sup> August 2012, Hue, Vietnam.

- **Chapter 4: Depth Based View Synthesis**

After investigating and presenting the depth estimation algorithm, the Chapter 4 of this thesis focus on the depth based image rendering for 3D video and 3DTV systems. As we have introduced, view synthesis is one of key techniques in the near future 3DV/3DTV system.

For synthesizing high quality of virtual view, we have proposed a novel method employing multiple color and depth images. The proposed method solves the main problems of depth based synthesis by applying forward depth map following with inverse warping texture, performing pixel classification to generate an initial new view from stable pixels and using Graph cut to select the best candidate for unstable pixels. The remained disoccluded pixels are inpainted by using depth and texture neighboring pixel value. Experimental results show that the proposed method has strength in artifact reduction. Objective evaluation has shown that our method get a significant gain in PSNR and SSIM comparing to some other existing methods. Another advantage of our method is that we can use a set of un-rectified images in multi-view system to create a new view with higher quality.

Depth imaged based view synthesis is one of active research field, but we have added some contributions as:

- Solving the cracks and holes due to sampling rate by applying forward depth map following with inverse warping texture. This allows a simple and accurate re-sampling of synthetic pixel.
- Presenting the procedure to classify the pixel as stable, unstable and disocclusion from the multiple images and providing an energy function to select the unstable pixels by Graph Cuts. By defining the types of pixels and using Graph cuts, the incorrectly wrapped pixels because of inaccuracy depth maps are removed in the synthetic view.
- Using both color and depth information to fill the disocclusion regions. This will help to reduce the blurring between foreground and background textures.



This work was published in the Proceeding of the 19<sup>th</sup> Korea-Japan Joint Workshop on Frontiers of Computer Vision; and in the Proceeding of the 17<sup>th</sup> International Conference on Image Processing, Computer Vision, & Pattern Recognition. The revised version was published in Journal of Signal and Information Processing.

- **Chapter 4: Depth Assisted Object Segmentation**

As the estimated depth information available, our concern is to apply the usefully estimated 3D information for the object segmentation method. We have proposed a method using both depth and color cues, which requires no interactive operation, to segment human object. Our method consist of two stages: for initial frame of the video sequence, the interested object is automatically extracted based on saliency model and iterated Graph cut. After having segmented object in first frame, we have propose a method combining Bayesian estimation and minimizing energy function using Graph cut to segment object. We use Gaussian Mixture Model (GMM) in RGB space for the color cue and histogram for depth cue. Based on these probabilistic models, the probability of each pixel to be in foreground is computed based on Bayesian estimation and the results are used to create the tri-map including foreground (F), background (B) and uncertain region (U). Graph cut is then performed on the uncertain region. In our energy function for Graph cut optimization, the color, depth and spatial-temporal coherence are integrated in data term and the penalty cost of the neighboring pixels with different labels is encoded in smoothness term. Experiment results on test sequences are encouraging and showed that our method is more effective than the case using only color cue.

Many researchers have investigated object tracking and different approaches have been presented, but our research has added some contributions as:

- Proposing the object segmentation method for video using both depth and color cues, which requires no interactive operation;
- Using probabilistic models for depth and color cues with Bayesian estimation to create tri-map; with tri-map, Graph cut only performed on the uncertain region, which can speed up the Graph cut process.

## 7.2. Outlook

---

This concept of integrating depth and color into the object segmentation was published in ICGST International Journal on Graphics, Vision and Image Processing GVIP (ICGST-GVIP).

- **Chapter 5: Depth Aided Object Tracking**

The final work in this thesis is to using the estimated depth information for object tracking. We have presented a novel tracking method aiming at detecting objects and maintaining their label/identification over the time. The main key factors of this method are to use depth information and different strategies to track objects under various occlusion scenarios. The foreground objects are detected and refined by background subtraction and shadow cancellation. The occlusion detection is based on information of foreground blobs in successive frames. The occlusion regions are projected to the projection plane XZ to analysis occlusion situation. According to the occlusion analysis results, different objects correspondence strategies are introduced to track object under various occlusion scenarios including tracking occluded objects in similar depth layer and in different depth layers. The experimental results show that our proposed method can track the moving objects under the most typical and challenging occlusion scenarios.

Many researchers have investigated object tracking and different approaches have been presented, but our research has added some contributions as:

- Using depth information to help to track objects under various occlusion scenarios;
- Applying different object tracking strategies according to occlusion situation with the help of object depth ordering.

This proposed method was published in Journal of Signal and Information Processing and in the International Journal of Computer Science and Network Security.

## 7.2 OUTLOOK

This thesis has contributes to several areas of depth estimation, view synthesis for 3DTV and object tracking and segmentation, however there are still a number of issues need to be address. For future work, that further improves techniques proposed in this thesis as well as potential new researches are listed below:

- **Depth Estimation**

- The proposed depth estimation algorithm is only performed independently for each frame of views. We can modify this algorithm to utilize the temporal consistency in successive video frames to obtain a more accurate depth image;
- To improve the quality of depth maps, we could concern about other approaches such as Graph models, two dimension optimization, or depth map refinement techniques;
- In this thesis, we just concern the narrow baseline multiple camera system, thus the future work can investigate for the wide based line case.

- **Free viewpoint synthesis for 3DTV**

- The similar in depth estimation, the proposed view synthesis is just performed for each frame. The extension with utilizing temporal information in successive video frames could yield a higher quality of synthesis view;
- We think that improving the disocclusion filling technique can further enhance the perceptive rendering quality. For example, edge, depth and segmentation features can use to correctly fill in the disocclusion pixels;
- The DIBR brings new challenges such as the question of synthesized view evaluation. We could be interested in the protocols of subjective assessment and the reliability of object quality metrics in the context of DIBR based view synthesis, because it may lead to improving the quality of synthesize views.
- To use in real time system, it should consider to reduce the algorithm complexity or to implement in hardware.

- **Depth based object segmentation**

- We had assumed that color and depth are independent, however there could be some correlations between color and depth information. We think that modeling the correlation of color and depth in some ways such as 4 dimensions histogram (3 dimensions are

## 7.2. Outlook

---

assigned for 3 color channels and another dimension for depth) could give a better results than the case when these two cues are modeled independently;

- The segmentation algorithm only deals with object segmentation in the key view, so finding efficient way to project the segmentation result in key-view to another views is an interested topic.
- We could extent the proposed method to segment multi-objects;
- **Depth based multiple object tracking**
  - The proposed algorithm can work well only in an indoor or medium sized environment since the reliability of depth information diminished in proportion to the distance from camera and only when the moving velocity of objects are slow. In the future work, to construct a robust moving object tracking system in both indoor and outdoor environment, we will study to use more object's features to classify and track the objects.

# A

## APPENDICES

### A.1 CALIBRATION PARAMETERS OF THE MULTI-VIEW SEQUENCES “BREAK-DANCERS” AND “BALLET”

These test sequences were acquired by Microsoft Research [69]. This data includes a sequence of 100 images captured from 8 cameras. The resolution of both sequences is  $1024 \times 768$  pixels and the frame rate is 15 frames per second. **Table 7-1** and **Table 7-2** show each camera along with the calibration parameters of the multi-view sequences “Break-dancers” and “Ballet”, respectively.

Note that the calibration parameters are a left-handed coordinate system with the origin (0,0) of the image located in the bottom left corner.

**Table 7-1. Calibration parameters of the multi-view sequence “Break-dancers”.**

	<b>K</b>			<b>R</b>			<b>T</b>
Camera 0	1884.19	-0.654998	513.7	0.962107	-0.005824	0.272486	-14.832727
	0.0	1887.49	395.609	0.004023	0.999964	0.007166	0.093097
	0.0	0.0	1.0	-0.272519	-0.005795	0.962095	-0.005195
Camera 1	1898.03	0.282128	517.91	0.97581	-0.02601	0.216939	-11.315863
	0.0	1900.81	382.815	0.022983	0.999598	0.016432	-0.167907
	0.0	0.0	1.0	-0.217280	-0.011048	0.976016	0.701363
Camera 2	1904.87	0.437636	497.954	0.987	-0.009204	0.160317	-7.554977
	0.0	1908.15	385.047	0.007599	0.999912	0.010582	0.000823
	0.0	0.0	1.0	-0.1604	-0.009226	0.986984	1.245294
Camera 3	1872.93	0.680911	546.988	0.996735	-0.00745	0.08025	-3.841023
	0.0	1877.1	380.224	0.00641	0.999888	0.013222	-0.089977
	0.0	0.0	1.0	-0.080339	-0.012665	0.996671	-0.083842
<b>Camera 4</b>	<b>1877.36</b>	<b>0.415492</b>	<b>579.467</b>	<b>1.0</b>	<b>0.0</b>	<b>0.0</b>	<b>0.000006</b>
	<b>0.0</b>	<b>1882.43</b>	<b>409.612</b>	<b>0.0</b>	<b>1.0</b>	<b>0.0</b>	<b>0.000001</b>
	<b>0.0</b>	<b>0.0</b>	<b>1.0</b>	<b>0.0</b>	<b>0.0</b>	<b>1.0</b>	<b>-0.000003</b>
Camera 5	1871.23	0.747826	540.106	0.998897	-0.017983	-0.043130	3.858103
	0.0	1877.30	412.656	0.017587	0.999799	-0.009367	0.069365
	0.0	0.0	1.0	0.043289	0.008599	0.999013	0.606667
Camera 6	1873.25	1.0738	578.641	0.991407	-0.015086	-0.129693	7.647271
	0.0	1880.06	386.506	0.016454	0.999816	0.009545	-0.012308
	0.0	0.0	1.0	0.129526	-0.011597	0.991473	-0.270987
Camera 7	1876.87	2.04	580.624	0.982395	0.005137	-0.186558	11.306122
	0.0	1883.93	395.399	-0.003610	0.999954	0.008587	-0.146099
	0.0	0.0	1.0	0.186594	-0.007762	0.982368	-1.040691

A.1. Calibration Parameters of the Multi-view Sequences “Break-Dancers” and “Ballet”

**Table 7-2. Calibration parameters of the multi-view sequence “Ballet”.**

	<b>K</b>			<b>R</b>			<b>T</b>
Camera 0	1918.27	2.48982	494.085	0.949462	0.046934	0.310324	-15.094651
	0.0	1922.58	447.736	-0.042337	0.998867	-0.021532	0.189829
	0.0	0.0	1.0	-0.310985	0.007308	0.950373	1.383263
Camera 1	1913.69	-0.14361	533.307	0.972850	0.010365	0.231187	-11.58932
	0.0	1918.17	398.171	-0.012981	0.999864	0.009794	-0.355771
	0.0	0.0	1.0	-0.231056	-0.012528	0.972852	1.045534
Camera 2	1914.07	0.343703	564.645	0.98923	0.003946	0.146295	-7.784865
	0.0	1918.5	428.422	-0.004391	0.999983	0.002724	-0.431597
	0.0	0.0	1.0	-0.146283	-0.003337	0.989230	1.392058
Camera 3	1909.91	0.571503	545.069	0.996415	0.026023	0.08048	-3.903715
	0.0	1915.89	394.306	-0.026884	0.999591	0.009614	-0.040429
	0.0	0.0	1.0	-0.080197	-0.011743	0.996707	0.168691
<b>Camera 4</b>	<b>1908.25</b>	<b>0.335031</b>	<b>560.336</b>	<b>1.0</b>	<b>0.0</b>	<b>0.0</b>	<b>-0.000002</b>
	<b>0.0</b>	<b>1914.16</b>	<b>409.596</b>	<b>0.0</b>	<b>1.0</b>	<b>0.0</b>	<b>0.0</b>
	<b>0.0</b>	<b>0.0</b>	<b>1.0</b>	<b>0.0</b>	<b>0.0</b>	<b>1.0</b>	<b>0.0</b>
Camera 5	1915.78	1.210910	527.609	0.998175	0.028914	-0.0530	3.849864
	0.0	1921.73	394.455	-0.028594	0.999567	0.006786	0.041657
	0.0	0.0	1.0	0.053173	-0.005258	0.998570	0.428967
Camera 6	1910.57	0.786148	578.134	0.988494	0.037674	-0.146458	7.602324
	0.0	1916.27	404.469	-0.037105	0.999288	0.006622	-0.045578
	0.0	0.0	1.0	0.146603	-0.001111	0.989188	-0.044837
Camera 7	1929.09	0.831916	585.52	0.975422	0.032363	-0.217910	11.142041
	0.0	1937.21	416.944	-0.033721	0.999425	-0.002516	0.200655
	0.0	0.0	1.0	0.217705	0.009803	0.975952	-0.230057

## B

### ABBREVIATIONS

MRF	Markov Random Fields	2D	Two Dimension
3DTV	Three Dimension Television	3D	Three Dimension
FTV	Free-viewpoint Television	3DV	Three Dimensional Video
BM	Block Matching	fps	Frames Per Second
GPU	Graphic Processing Unit	HSV	Hue, Saturation, Value
SVD	Singular Value Decomposition	RGB	Red, Green, Blue
SAD	Sum of Absolute Difference	GMM	Gaussian Mixture Model
SSD	Sum of Square Difference	MAP	Maximum A Posteriori
CC	Cross Correlation.	OOI	Object Of Interest
NCC	Normalized Cross Correlation	F	Foreground region
WTA	Winner – Take – All	B	Background region
MSE	Mean Square Error	U	Uncertain region
PSNR	Peak Signal Noise Ratio	CMOS	Complementary Metal Oxide Semiconductor
MVD	Multiple View plus Depth	CCL	Connected Component Labeling
MVC	Multi-view Video Coding	SM	Saliency Map
SSIM	Structure Similarity Index		
DIBR	Depth Image Based Rendering		

## C

### LIST OF PUBLICATIONS BY AUTHOR

#### Reviewed Conference Papers

- [C1] Anh Tu Tran, Koichi Harada, "Multi-view Consistency Depth Image Estimation," *the fourth International Conference on Communication and Electronics (ICCE)*, 1<sup>st</sup> – 3<sup>rd</sup> August 2012, Hue, Vietnam, pp.584-589.
- [C2] Anh Tu Tran, Koichi Harada, "View synthesis with depth information based on graph cuts for FTV," *in 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, Jan. 30 – Feb. 1, 2013, Incheon, Korea, pp. 289-294.
- [C3] Anh Tu Tran, Koichi Harada, "View Synthesis Based on Depth Information and Graph Cuts for 3DTV," *the 17th International Conference on Image Processing, Computer Vision, & Pattern Recognition*, July 22-25, 2013, Las Vegas, USA, pp. 56-62.

#### Journal Papers

- [J1] Anh Tu Tran, Koichi Harada, "Multi-view Video Segmentation based on Bayesian Estimation and Graph Cut," *ICGST International Journal on Graphics, Vision and Image Processing GVIP (ICGST-GVIP)*, Vol.12, Issue 1, April 2012, pp. 9-14.
- [J2] Anh Tu Tran, Koichi Harada, "Depth assisted Tracking Multiple Moving Objects under Occlusion," *International Journal of Computer Science and Network Security*, Vol. 13, No. 5, May 2013, pp. 49-56.
- [J3] Anh Tu Tran, Koichi Harada, "Depth based View Synthesis using Graph Cuts for 3DTV," *Journal of Signal and Information Processing (JSIP)*, Vol. 4, No. 3, August 2013, pp. 327-335. doi: 10.4236/jsip.2013.43041.
- [J4] Anh Tu Tran, Koichi Harada, "Depth-aided Tracking Multiple Objects under Occlusion", *Journal of Signal and Information Processing (JSIP)*, Vol. 4, No. 3, August 2013, pp. 299-307. doi: 10.4236/jsip.2013.43038.



## REFERENCES

- [1] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," ed: Cambridge University Press, 2003.
- [2] G. Bradski and A. Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library," ed: O'Reilly Media, Incorporated, 2008.
- [3] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*: Prentice Hall, 1998.
- [4] D. C. Brown, "Close-Range Camera Calibration," *Photogrammetric Engineering*, vol. 37, pp. 855-866, 1971.
- [5] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 323-344, 1987.
- [6] J. Heikkilä and O. Silvén, "A Four-step Camera Calibration Procedure with Implicit Image Correction," in *IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997.
- [7] T. A. Clarke and J. G. Fryer, "The Development of Camera Calibration Methods and Models," *The Photogrammetric Record*, vol. 16, pp. 51-66, 1998.
- [8] P. F. Sturm and S. J. Maybank, "On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999, pp. 432-437.
- [9] J. Heikkilä, "Geometric camera calibration using circular control points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1066-1077, 2000.
- [10] Z. Zhengyou, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330-1334, 2000.
- [11] W. Sun and R. Cooperstock, "An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques," *Machine Vision and Applications*, vol. 17, pp. 51-67, 2006.
- [12] J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*. vol. 630, G. A. Watson, Ed., ed: Springer Berlin Heidelberg, 1978, pp. 105-116.
- [13] R. Hartley, "Theory and Practice of Projective Rectification," *International Journal of Computer Vision*, vol. 35, pp. 115-127, 1999.
- [14] J.-Y. Bouguet. (8/2013). *Camera Calibration Toolbox for Matlab (Fifth calibration example - Calibrating a stereo system, stereo image rectification and 3D stereo triangulation)*. Available: [http://www.vision.caltech.edu/bouguetj/calib\\_doc/htmls/example5.html](http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html)
- [15] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact Maximum A Posteriori Estimation for Binary Images," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 51, pp. 271-279, 1989.
- [16] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1222-1239, Nov. 2001.
- [17] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in *the 8th IEEE International Conference on Computer Vision (ICCV)*, 2001, pp. 105-112.

## References

---

- [18] V. Kolmogorov and R. Zabih, "Multi-camera Scene Reconstruction via Graph Cuts," in *the 7th European Conference on Computer Vision-Part III*, 2002, pp. 82-96.
- [19] Q. Zhang and K. N. Ngan, "Multi-view video based multiple objects segmentation using graph cut and spatiotemporal projections," *Journal of Visual Communication and Image Representation*, vol. 21, pp. 453-461, 2010.
- [20] L. R. Ford and D. R. Fulkerson, *Flows in Networks*: Princeton University Press, 1962.
- [21] Y. Boykov and V. Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1124-1137, 2004.
- [22] (5/2013). *MESA Imaging (The leading provider of 3D Time of Flight cameras)*. Available: <http://www.csem.ch/fs/imaging.htm>
- [23] (May 21, 2013). *PrimeSense (3D sensing technology)* Available: <http://www.primesense.com/>
- [24] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005, pp. 593-600.
- [25] C. Chao-Chung, L. Chung-Te, and C. Liang-Gee, "A novel 2D-to-3D conversion system using edge information," *IEEE Transactions on Consumer Electronics*, vol. 56, pp. 1739-1745, 2010.
- [26] T. Nagai, T. Naruse, M. Ikehara, and A. Kurematsu, "HMM-based surface reconstruction from single images," in *International Conference on Image Processing*, 2002, pp. II-561-II-564, vol.2.
- [27] A. Maki, M. Watanabe, and C. Wiles, "Geotensity: Combining Motion and Lighting for 3D Surface Reconstruction," *International Journal of Computer Vision*, vol. 48, pp. 75-90, 2002.
- [28] T. Lindeberg and J. Garding, "Shape from texture from a multi-scale perspective," in *Fourth International Conference on Computer Vision*, 1993, pp. 683-691.
- [29] A. Hertzmann and S. M. Seitz, "Example-based photometric stereo: shape reconstruction with general, varying BRDFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1254-1264, 2005.
- [30] A. Saxena, A. Ng, and S. Chung, "Learning Depth from Single Monocular Images," *NIPS*, vol. 18, 2005.
- [31] A. Saxena, S. Chung, and A. Ng, "3-D Depth Reconstruction from a Single Still Image," *International Journal of Computer Vision*, vol. 76, pp. 53-69, 2008.
- [32] T. Wa James and L. Zhang, "3D-TV Content Generation: 2D-to-3D Conversion," in *IEEE International Conference on Multimedia and Expo*, 2006, pp. 1869-1872.
- [33] T. Yi-Min, Y.-L. Chang, and C. Liang-Gee, "Block-based Vanishing Line and Vanishing Point Detection for 3D Scene Reconstruction," in *International Symposium on Intelligent Signal Processing and Communications (ISPACS)*, 2006, pp. 586-589.
- [34] I. Ideses, L. Yaroslavsky, and B. Fishbain, "Real-time 2D to 3D video conversion," *Journal of Real-Time Image Processing*, vol. 2, pp. 3-9, 2007.
- [35] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *International Journal of Computer Vision*, vol. 47, pp. 7-42, 2002.

## References

---

- [36] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 401-406, 1998.
- [37] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 920-932, 1994.
- [38] K. Sing Bing, R. Szeliski, and C. Jinxiang, "Handling occlusions in dense multi-view stereo," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. I103 - I110, vol.1.
- [39] T. Hai, H. S. Sawhney, and R. Kumar, "A global matching framework for stereo computation," in *the Eighth IEEE International Conference on Computer Vision (ICCV)*, 2001, pp. 532-539, vol.1.
- [40] A. Criminisi, A. Blake, C. Rother, J. Shotton, and P. H. S. Torr, "Efficient dense stereo with occlusions for new view-synthesis by four-state dynamic programming," *International Journal of Computer Vision*, vol. 71, pp. 89-110, Jan 2007.
- [41] M. Bleyer and M. Gelautz, "A layered stereo matching algorithm using image segmentation and global visibility constraints," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 59, pp. 128-150, 2005.
- [42] T. Hai and H. S. Sawhney, "Global matching criterion and color segmentation based stereo," in *the Fifth IEEE Workshop on Applications of Computer Vision*, 2000, pp. 246-253.
- [43] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, 1984.
- [44] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," in *the Sixth International Conference on Computer Vision*, 1998, pp. 1073-1080.
- [45] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo Matching Using Belief Propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 787-800, 2003.
- [46] V. Kolmogorov, R. Zabih, and S. Gortler, "Generalized Multi-camera Scene Reconstruction Using Graph Cuts," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*. vol. 2683, A. Rangarajan, M. Figueiredo, and J. Zerubia, Eds., ed: Springer Berlin Heidelberg, 2003, pp. 501-516.
- [47] (5/2013). *Middlebury Stereo Vision Page*. Available: <http://vision.middlebury.edu/stereo/>
- [48] T. Kanade, P. J. Narayanan, and P. W. Rander, "Virtualized reality: concepts and early results," in *IEEE Workshop on Representation of Visual Scenes (In Conjunction with ICCV'95)*, 1995, pp. 69-76.
- [49] T. Kanade, H. Saito, and S. Vedula, "The 3D Room: Digitizing Time-Varying 3D Events by Synchronized Multiple Video Streams," Technical Report, The Robotics Institute, Carnegie Mellon University (CMU-RI-TR-98-34),1998.
- [50] Z. Cha and C. Tsuhan, "Multi-View Imaging: Capturing and Rendering Interactive Environments," in *Computer Vision for Interactive and Intelligent Environment*, 2005, pp. 51-67.
- [51] (8/2013). *MPEG-FTV Project, FTV test sequences*. Available: <http://www.tanimoto.nuee.nagoya-u.ac.jp/MPEG-FTVProject.html>
- [52] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Transactions on Graphics*, vol. 23, pp. 600-608, 2004.

## References

---

- [53] L. Sang-Beom and H. Yo-Sung, "View-consistent multi-view depth estimation for three-dimensional video generation," in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2010, pp. 1-4.
- [54] L. Sang-Beom, O. Kwan-Jung, and H. Yo-Sung, "Segment-Based Multi-View Depth Map Estimation Using Belief Propagation from Dense Multi-View Video," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, 2008, pp. 193-196.
- [55] E. S. Larsen, P. Mordohai, M. Pollefeys, and H. Fuchs, "Temporally Consistent Reconstruction from Multiple Video Streams Using Enhanced Belief Propagation," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007, pp. 1-8.
- [56] P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger, "Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability," *Signal Processing: Image Communication*, vol. 22, pp. 217-234, 2007.
- [57] M. Tanimoto, T. Fujii, and K. Suzuki, "Multi-view depth map of Rena and Akko & Kayo," MPEG Doc, M14888, Oct.2007.
- [58] (11/2012). *Minoru 3D webcam*. Available: <http://www.minoru3d.com/>
- [59] (8/2013). *OpenCV (Open Source Computer Vision Library)*. Available: <http://opencv.org/>
- [60] K. Konolige, "Small Vision Systems: Hardware and Implementation," in *Robotics Research*, Y. Shirai and S. Hirose, Eds., ed: Springer London, 1998, pp. 203-212.
- [61] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 328-341, 2008.
- [62] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *the 8th IEEE International Conference on Computer Vision (ICCV)*, 2001, pp. 508-515 vol.2.
- [63] (7/2013). *Middlebury Stereo Datasets*. Available: <http://vision.middlebury.edu/stereo/data/>
- [64] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," in *the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000, pp. 307-318.
- [65] A. F. Bobick and S. S. Intille, "Large Occlusion Stereo," *International Journal of Computer Vision*, vol. 33, pp. 181-200, 1999.
- [66] J. B. T. M. Roerdink and A. Meijster, "The Watershed Transform: Definitions, Algorithms and Parallelization Strategies," *Fundamental Informaticae*, vol. 41, pp. 187-228, 2001.
- [67] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603-619, 2002.
- [68] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communication of the ACM*, vol. 24, pp. 381-395, 1981.
- [69] (8/2013). *Microsoft Research, Interactive Visual Media Group*. Available: <http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload/>
- [70] C. Cigla, X. Zabulis, and A. A. Alatan, "Region-Based Dense Depth Extraction from Multi-View Video," in *IEEE International Conference on Image Processing (ICIP)*, 2007, pp. V213 - V216.
- [71] M. Tanimoto, "Overview of FTV (free-viewpoint television)," in *IEEE International Conference on Multimedia and Expo*, New York, NY, USA, 2009, pp. 1552-1553.

## References

---

- [72] (8/2013). *3D Autostereoscopic Display*. Available: <http://www.usa.philips.com/>
- [73] Y. Chen, Y.-K. Wang, K. Ugur, M. M. Hannuksela, J. Lainema, and M. Gabbouj, "The Emerging MVC Standard for 3D Video Services," *EURASIP Journal on Advances in Signal Processing* vol. 2009, Jan. 2009.
- [74] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient Prediction Structures for Multiview Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, pp. 1461-1473, 2007.
- [75] C. Fehn, P. Kauff, M. O. D. Beeck, F. Ernst, and e. al., "An Evolutionary and Optimised Approach on 3D-TV " *In Proceedings of International Broadcast Conference*, pp. 357-365, 2002.
- [76] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *SPIE Stereoscopic Displays and Virtual Reality Systems XI*, 2004, pp. 93-104.
- [77] L. McMillan, "An Image-Based Approach to Three-Dimensional Computer Graphics," PhD thesis, University of North Carolina, Chapel Hill, USA, April 1997.
- [78] K. Muller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems " in *15th IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 2448 - 2451
- [79] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "View generation with 3D warping using depth information for FTV," *Signal Processing: Image Communication*, vol. 24, pp. 65-72, Jan 2009.
- [80] A. C. Telea, "An image inpainting technique based on the Fast Marching Method," *Journal of Graphics Tools*, vol. 9, pp. 25-36, 2004.
- [81] S. Zinger, L. Do, and P. H. N. de With, "Free-viewpoint depth image based rendering," *Journal of Visual Communication and Image Representation*, vol. 21, pp. 533-541, 2010.
- [82] K.-J. Oh, S. Yea, and Y.-S. Ho, "Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-D video " in *Picture Coding Symposium*, 2009.
- [83] V. Kolmogorov and R. Zabih, "What Energy Functions Can Be Minimized via Graph Cuts?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 147-159, 2004.
- [84] Z. Wang, A. C. Bovik, and H. R. Sheikh, "Image Quality Assessment: From Error Measurement to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600-612, 2004.
- [85] (8/2013). *The SSIM Index for Image Quality Assessment*. Available: <http://www.cns.nyu.edu/~lcv/ssim/#usage>
- [86] S. M. Rhee, Y. J. Yoon, I. K. Shin, Y. G. Kim, Y. J. Choi, and S. M. Choi, "Stereo Image Synthesis by View Morphing with Stereo Consistency," *Applied Mathematics & Information Sciences*, vol. 6, pp. 195-200, 2012.
- [87] M. Solh and G. AlRegib, "Hierarchical Hole-Filling For Depth-Based View Synthesis in FTV and 3D Video," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, pp. 495-504, Sep 2012.
- [88] X. Bai, J. Wang, D. Simons, and G. Sapiro, "Video SnapCut: robust video object cutout using localized classifiers," *ACM Transactions on Graphics*, vol. 28, pp. 1-11, 2009.
- [89] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," *ACM Transactions on Graphics*, vol. 23, pp. 303-308, 2004.

## References

---

- [90] W. Song and J. M. Siskind, "Image segmentation with ratio cut," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 675-690, 2003.
- [91] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, pp. 309-314, 2004.
- [92] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Probabilistic Fusion of Stereo with Color and Contrast for Bilayer Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1480-1492, 2006.
- [93] B. Goldlücke and M. A. Magnor, "Joint 3D-Reconstruction and Background Separation in Multiple Views using Graph Cuts," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 683--694.
- [94] J. Y. Guillemaut, J. Kilner, and A. Hilton, "Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes," in *IEEE 12th International Conference on Computer Vision*, 2009, pp. 809-816.
- [95] N. Grammalidis, L. Bleris, and M. G. Strintzis, "Using the expectation-maximization algorithm for depth estimation and segmentation of multi-view images," in *First International Symposium on Data Processing Visualization and Transmission*, 2002, pp. 686-689.
- [96] M. A. A. Medeiros and L. A. D. S. Cruz, "Iterative disparity estimation and image segmentation," in *the 8th Conference on Signal Processing, Computational Geometry and Artificial Vision*, Rhodes, Greece, 2008, pp. 67-72.
- [97] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: A Bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, 2008.
- [98] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1254-1259, 1998.
- [99] L. Zhang, M. H. Tong, and G. W. Cottrell, "SUNDAy: Saliency Using Natural Statistics for Dynamic Analysis of Scenes," in *Thirty-first Annual Cognitive Science Society Conference*, Amsterdam, Netherlands.
- [100] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1597-1604.
- [101] Y.-F. Ma and H.-J. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *the Eleventh ACM International Conference on Multimedia*, Berkeley, CA, USA, 2003, pp. 374-381.
- [102] B. C. Ko and J.-Y. Nam, "Object-of-interest image segmentation based on human attention and semantic region clustering," *Journal of the Optical Society of America A (JOSA A)*, vol. 23, pp. 2462-2470, 2006.
- [103] H. Junwei, K. N. Ngan, L. Mingjing, and Z. Hong-Jiang, "Unsupervised extraction of visual attention objects in color images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 141-145, 2006.
- [104] (8/2013). *Microsoft Research, Free research data*. Available: <http://research.microsoft.com/en-us/projects/i2i/data.aspx>
- [105] D. Koller, K. Danilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, vol. 10, pp. 257-281, 1993.

## References

---

- [106] T. Hai, H. S. Sawhney, and R. Kumar, "Object tracking with Bayesian estimation of dynamic layer representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 75-89, 2002.
- [107] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1296-1311, 2003.
- [108] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564-577, 2003.
- [109] J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593-600.
- [110] Y. Rui and Y. Chen, "Better Proposal Distributions: Object Tracking Using Unscented Particle Filter," in *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 786-793.
- [111] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, pp. 5-28, 1998.
- [112] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 495-501.
- [113] N. K. Paragios and R. Deriche, "A PDE-based level-set approach for detection and tracking of moving objects," in *Sixth International Conference on Computer Vision*, 1998, pp. 1139-1145.
- [114] Z. Tao, R. Nevatia, and W. Bo, "Segmentation and Tracking of Multiple Humans in Crowded Environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1198-1211, 2008.
- [115] X. Zhou, Y. F. Li, and B. He, "Game-Theoretical Occlusion Handling for Multi-Target Visual Tracking," *Pattern Recognition*, 2012.
- [116] V. Papadourakis and A. Argyros, "Multiple objects tracking in the presence of long-term occlusions," *Computer Vision and Image Understanding*, vol. 114, pp. 835-846, 2010.
- [117] M. Wu, X. Peng, Q. Zhang, and R. Zhao, "Segmenting and tracking multiple objects under occlusion using multi-label graph cut," *Computers & Electrical Engineering*, vol. 36, pp. 927-934, 2010.
- [118] E. Parvizi and Q. M. J. Wu, "Multiple Object Tracking Based on Adaptive Depth Segmentation," in *Canadian Conference on Computer and Robot Vision (CRV)*, 2008, pp. 273-277.
- [119] S. J. Krotosky and M. M. Trivedi, "On Color-, Infrared-, and Multimodal-Stereo Approaches to Pedestrian Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, pp. 619-629, 2007.
- [120] R. Okada, Y. Shirai, and J. Miura, "Object tracking based on optical flow and depth," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems (IEEE/SICE/RSJ)*, 1996, pp. 565-571.
- [121] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *the 17th International Conference on Pattern Recognition (ICPR)*, 2004, pp. 28-31, vol.2.
- [122] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection," in *Video-Based Surveillance Systems*, P. Remagnino, G. Jones, N. Paragios, and C. Regazzoni, Eds., ed: Springer US, 2002, pp. 135-144.

## References

---

- [123] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 918-923, 2003.
- [124] (8/2013). *OpenCV Wiki, cvBlobLib*. Available: <http://opencv.willowgarage.com/wiki/cvBlobsLib>
- [125] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding*, vol. 93, pp. 206-220, 2004.
- [126] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99-109, 1943.
- [127] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *the Fourth IEEE Workshop on Applications of Computer Vision (WACV)*, 1998, pp. 214-219.