

國立交通大學

電子工程學系 電子研究所

碩士論文

使用於立體視差估算之快速圖形切割

演算法

Fast Graph Cuts Algorithm for Disparity
Estimation

研究生：周正偉

指導教授：杭學鳴

中華民國九十九年六月

使用於立體視差估算之快速圖形切割

演算法

Fast Graph Cuts Algorithm for Disparity Estimation

研究生：周正偉

Student: Cheng-Wei Chou

指導教授：杭學鳴博士

Advisor: Dr. Hsueh-Ming Hang

國立交通大學

電子工程學系 電子研究所



A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Electronic Engineering

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

使用於立體視差估算之快速圖形切割

演算法

研究生：周正偉

指導教授：杭學鳴 博士

國立交通大學

電子工程學系 電子研究所碩士班



視差估算在 3D 視頻處理系統中是其中一個關鍵的因素。許多技術已經被提出來計算視差圖，圖形切割演算法是一種公認較好的視差估算計畫。然而，圖形切割演算法具有很高的計算複雜度。

在這篇論文中，我們提出了一個用於視差估算的快速圖形切割演算法，有兩個加速的技巧被提出：一個是提前終止規則，另一個是排出 α - β 交換對的搜索的優先順序。我們的模擬結果表現，當我們跟原始方法比較，該演算法可以加速 68% 的平均運算時間。同時，視差圖的品質可以保持在幾乎跟原始方法一樣。

另一個加速技術，我們是採用多解析度的方法。一開始我們先對原始影像降頻取樣，並針對低解析度的影像作視差估算，產生低解析度的視差圖。接著，我

們再對低解析度的視差圖做升頻取樣，並以此視差圖作為初始值去做原始解析度的視差估測，我們去測試幾種降頻取樣及升頻取樣的方式，並找到最佳的組合。我們的模擬顯示，多解析度的圖形切割演算法只使用原始計算時間的 16%，而壞像素的升幅只有 1%。我們研究的最後一個主題是使用多相機拍照的視差估測，初步觀察顯示了一些有趣的結果，我們需要進一步的實驗才能發揮這主題的優勢。



Fast Graph Cuts Algorithm for Disparity Estimation

Student: Cheng-Wei Chou

Advisor: Dr. Hsueh-Ming Hang

Department of Electronic Engineering &
Institute of Electronics
National Chiao Tung University

Abstract

Disparity estimation is one of the critical elements in a 3D video processing system. Many techniques have been proposed to calculate the disparity map from a pair of images and the graph cut (GC) algorithm is one of the recognized better disparity estimation schemes. However, GC has a very high computational complexity.

In this thesis, we propose a fast GC algorithm for disparity estimation purpose. Two accelerating techniques are suggested: one is the early termination rule and the other is prioritizing the α - β swap pair search order. Our simulations show that the proposed fast GC algorithm can reduce 68% computing time on the average, when compared with the original GC scheme. Meanwhile, its disparity estimation performance is about the same as that of the original GC.

Another speed-up technique we adopt is the multi-resolution approach. The

original images are down-sampled and a low-resolution disparity map is first estimated. Then, the low-resolution disparity map is up-sampled as the initial values for estimating the disparity map of the original images. Several down-sampling and up-sampling filters are tested to find the best combination. Our simulation shows that the multi-resolution GC (MRGC) algorithm uses only 16% of the original computing time and the bad pixel probability increases only by 1%. The last topic we investigate is disparity estimation using multi-camera pictures. The initial exploration shows some interesting results. Further investigation is needed to fully take the advantage of multiple images recoded by a camera array.



誌謝

能夠完成碩士論文，我首先要感謝的是指導教授杭學鳴老師，在研究過程中，經由老師耐心指導，學習到做研究的方法和應有的態度。除了在專業領域的教導，老師也時常關心我們的身體健康。這兩年來，從老師身上不僅僅只學到專業知識，更學到待人處事的方法。相信這些經驗將成為我人生中重要的瑰寶。

研究過程中，非常感謝蔡彰哲學長的指導。從確定研究目標到完成實驗，學長總是費盡心思的與我討論並給予意見。在我遇到問題的時候，帶著我找尋答案，真的非常感謝蔡彰哲學長。非常感謝大師，無論在專業領域上，或是論文寫作技巧，都給予我許多資訊與幫助，在大師循循善誘的教導下，讓我在碩士生涯成長不少。此外，也非常感謝家揚、雄哥的幫助，以及眾多學長學弟同學們的陪伴，讓我的碩士班生活更多采多姿。

最後要感謝我的爸媽以及姊姊，在我求學生涯中默默的支持我，你們的溫暖是我前進的動力。

Table of Contents

摘要	i
Abstract	iii
致謝	v
Table of Contents	vi
List of Figures	viii
List of Tables	x
Chapter 1 Introduction.....		1
1.1 Background.....		1
1.2 Motivation and Contributions.....		2
1.3 Organization of the Thesis.....		3
Chapter 2 Introduction of Computational Stereo.....		5
2.1 Overview.....		5
2.2 Epipolar Geometry.....		5
2.3 The General Structure of Matching Algorithm.....		6
2.3.1 Initial Matching Cost Computation.....		7
2.3.2 Cost Aggregation.....		8
2.3.3 Disparity Computation and Optimization.....		10
2.3.4 Disparity Refinement.....		13
2.4 A Taxonomy Evaluation.....		13
2.4.1 Overview of the Platform.....		13
2.4.2 Quality Metrics.....		15
2.4.3 Test Data.....		16
Chapter 3 Energy Minimization by Graph Cuts.....		18
3.1 Overview.....		18
3.2 Max-Flow and Min-Cut Problem.....		18
3.3 Push-Relabel Algorithm.....		20
3.4 Energy Minimization using Graph Cuts.....		23
3.4.1 The General Form of Energy Function.....		23
3.4.2 The α - β Swap Method.....		25
3.4.3 Multiway Cut Algorithm.....		29
Chapter 4 The α-β Swap Algorithm Speed-Up and Early Termination.....		32
4.1 Overview.....		32
4.2 Early Termination of Energy Minimization Process.....		32
4.3 Prioritizing the α - β Swap Pair Sequence.....		34

4.4	Simulation Results and Discussions	37
4.4.1	Experiment Environment Setting.....	37
4.4.2	Simulation Results	38
4.4.3	Analysis and Discussions.....	46
Chapter 5	Multi-Resolution Graph Cuts and Disparity Estimation for Multi-Camera Array.....	48
5.1	Overview	48
5.2	Disparity Estimation using Multi-Resolution Graph Cuts.....	48
5.2.1	Image Down-Sampling	50
5.2.2	Disparity Map Up-sampling and Scaling.....	51
5.2.3	Neighborhood Graph Cuts	53
5.3	Disparity Estimation in Multi-Camera Array	54
5.4	Simulation Results and Discussions	56
5.4.1	Multi-Resolution Graph Cuts.....	56
5.4.2	Disparity Estimation in Multi-Camera Array	65
Chapter 6	Conclusions and Future Work.....	68
6.1	Conclusions.....	68
6.2	Future Work	69



List of Figures

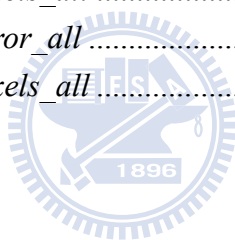
Fig. 2-1 Stereo image geometry.....	6
Fig. 2-2 Process of the general stereo correspondence algorithms.....	7
Fig. 2-3 Disparity space image	7
Fig. 2-4 An illustration of the shiftable window	10
Fig. 2-5 Stereo matching using dynamic programming	12
Fig. 2-6 Program structure of the middlebury platform	14
Fig. 3-1 A simple example of the graph \mathcal{G} and the minimum cut (the red line).....	19
Fig. 3-2 A simple example of push-relabel algorithm	22
Fig. 3-3 An example of a directed weighted graph	26
Fig. 3-4 An example of the graph for a 1D image.....	26
Fig. 3-5 Properties of a cut on the graph	28
Fig. 3-6 The change of disparity map after an α - β swap	28
Fig. 3-7 (a) An example of the graph with multiple terminals $\mathcal{L} = \{0, 1, \dots, k\}$ (b) An induced graph by a multiway cut (dotted lines indicate cut edges).....	30
Fig. 3-8 Flowchart of GC.....	31
Fig. 4-1 Disparity distribution of the test image pair ‘sawtooth’ after the i -th iteration of the outer loop (P^{i-th})	33
Fig. 4-2 E and θ in the energy minimization process of the test image pair ‘sawtooth’	34
Fig. 4-3 RMS_error and Bad_pixel in the energy minimization process of ‘sawtooth’.....	34
Fig. 4-4 Disparity distribution of ‘sawtooth’ after the 1 st outer-loop iteration	36
Fig. 4-5 Flow chart of our proposed fast GC.....	37
Fig. 4-6 Plot of computing time.....	39
Fig. 4-7 Plot of rms_error_all	40
Fig. 4-8 Plot of bad_pixels_all	41
Fig. 4-9 Disparity maps of ‘Map’	44
Fig. 4-10 Disparity maps of ‘Sawtooth’	44
Fig. 4-11 Disparity maps of ‘Tsukuba’	45
Fig. 4-12 Disparity maps of ‘Venus’	45
Fig. 5-1 Flowchart of MRGC	49
Fig. 5-2 An example of 4 to 1 pixel-skip down-sampling.....	50
Fig. 5-3 An example of disparity map up-sampling and scaling.....	51
Fig. 5-4 The up-sampling method of H.264	52
Fig. 5-5 The disparity pair candidates in neighborhood graph cuts (a)+1 (b) \pm 1.....	53
Fig. 5-6 Multi-camera array.....	54

Fig. 5-7 Flowchart of GC for multi-camera pictures	55
Fig. 5-8 The scaling and shifting moves.....	55
Fig. 5-9 Plot of computing time comparison	58
Fig. 5-10 Plot of <i>rms_error_all</i>	59
Fig. 5-11 Plot of <i>bad_pixels_all</i>	60
Fig. 5-12 Disparity maps of ‘Map’	61
Fig. 5-13 Disparity maps of ‘Sawtooth’	62
Fig. 5-14 Disparity maps of ‘Tsukuba’	63
Fig. 5-15 Disparity maps of ‘Venus’	64
Fig. 5-16 Disparity maps of “Sawtooth”	66
Fig. 5-17 Disparity maps of “Venus”	67



List of Tables

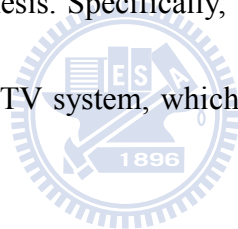
Table 2-1 Match metrics for correspondence matching [6].....	9
Table 2-2 Descriptions of error metrics [5]	16
Table 2-3 Test data [5]	17
Table 3-1 Weight information of the edges	27
Table 4-1 The experiment environment setting.....	38
Table 4-2 Comparison of computing time.....	39
Table 4-3 Comparison of <i>rms_error_all</i>	40
Table 4-4 Comparison of <i>bad_pixels_all</i>	41
Table 4-5 Comparison of <i>bad_pixels_nonocc</i>	42
Table 4-6 Comparison of <i>bad_pixels_textureless</i>	42
Table 4-7 Comparison of <i>bad_pixels_discont</i>	43
Table 5-1 Computing time comparison	58
Table 5-2 Comparison of <i>rms_error_all</i>	59
Table 5-3 Comparison of <i>bad_pixels_all</i>	60
Table 5-4 Comparison of <i>rms_error_all</i>	65
Table 5-5 Comparison of <i>bad_pixels_all</i>	65



Chapter 1 Introduction

1.1 Background

Recently the ISO/IEC Moving Picture Expert Group (MPEG) initiated a standardization process on free viewpoint television (FTV) [1]. As a new type of interactive video system, FTV can synthesize 3 dimensional (3D) scenes at nearly any (virtual) viewpoint and thus receives its name. An FTV system typically consists of modules of multi-view video capture, image correction, depth map estimation, data coding/decoding, and view synthesis. Specifically, the disparity estimation module is an inevitable component of an FTV system, which is used in both multi-view scene analysis and synthesis.



With the help of epipolar geometry [2], the general stereo correspondence problem is simplified to disparity estimation (DE) [3][4][5] under the assumption of dense camera array. Here, disparity refers to the location difference of the corresponding objects along the epipolar lines on the two recorded images. For years, most researchers have focused on improving the accuracy of DE, not on the speed acceleration. With the emerging 3D TV and FTV, we focus on fast DE algorithms in this study.

DE algorithm generally is divided into 4 stages: 1) initial matching cost

calculation, 2) cost aggregation, 3) disparity computation and optimization, and 4) disparity refinement. In this contribution, we use the absolute intensity difference (with the Birchfield and Tomasi's sampling insensitive dissimilarity measure) [5] between the corresponding feature points as the initial matching cost. The cost aggregation collects the initial matching cost by using a moving average filter in a square window (box filter). Once the aggregated costs are computed, the disparity computation and optimization module determines which discrete set of disparities best represents the scene surface depth. Finally the disparity refinement step increases the disparity accuracy to sub-pixel precision.

The stereo algorithm can be categorized as local and global approaches [5]. The local approach focuses on the cost calculation and aggregation. The winner-take-all method (WTA), which chooses the lowest aggregated cost as the selected disparity at each pixel, is a simple disparity computation and optimization method used in the local approach. In the global approach, we further consider the disparity smoothness among neighboring pixels. More sophisticated algorithms, such as dynamic programming, simulated annealing, belief propagation (BP) and graph cut (GC), has been suggested to offer better DE results.

1.2 Motivation and Contributions

Among various global DE algorithms, GC and BP offer better DE results. The

qualities of them are similar. According to Tappen and Freeman [7], the computational time of the synchronous BP is much larger than GC, but the accelerated BP uses only 80% of the GC's computational time. In this study, we choose GC because it can be accelerated and may have advantages in computation. In addition, to our knowledge, the fast graph cuts algorithms are rare in literature. Therefore, we propose methods to accelerate GC algorithm.

GC and its variations [8]-[11] generally show admirable performance in their disparity estimation quality. However, GC suffers from the huge amount of processing time. Owing to its good DE performance, GC is chosen as the target algorithm for speed-up. The major contributions in this thesis include the following items.

- (1) An early termination process is proposed to save the computing time.
- (2) We save computing time by prioritizing the α - β swap pair sequence.
- (3) The computing time greatly increases with the image size and the disparity range.

We use multi-resolution graph cuts to reduce the computational complexity.

- (4) We attempt to improve the disparity map for the multi-camera array.

1.3 Organization of the Thesis

In chapter 2, we briefly introduce the background of computational stereo. In chapter 3, we describe the graph cuts process that minimizes the energy function in DE problem. Chapter 4 describes our proposal of the early termination and optimizing

the α - β swap pair sequence. Simulations are conducted and the significant amount of computing time saving is shown. Chapter 5 discusses the multi-resolution graph cuts algorithm and our initial investigation on the disparity estimation on the multi-camera array picture. Finally, brief summary and remarks on future work are given in chapter 6.



Chapter 2 Introduction of Computational

Stereo

2.1 Overview

The concept of stereo correspondence is to find the correspondent point in the image of the other view. Based on epi-polar geometry, the general stereo correspondence problem is simplified to disparity estimation under the assumption of dense camera array. The search region of the corresponding features between the left and the right images can thus be reduced to the epi-polar lines. The goal of a stereo correspondence algorithm is to produce a univalued function in disparity space $d(x, y)$ that best describes the depth information of the surfaces in the scene.

2.2 Epipolar Geometry

Fig. 2-1 shows a typical stereo image geometry in the 3D space. There are two pinhole cameras viewing a 3D scene from different view points. We place a virtual image plane in front of each camera. The intersection between the baseline connects two cameras and the two image planes are called *epipole*. The plane formed by any point in the space and the base line is *epipolar plane*. The *epipolar plane* intersects each camera's image plane, the intersection forms lines—the *epipolar lines*. Any point \mathbf{P} in the epipolar plane corresponds to points \mathbf{P}_1 and \mathbf{P}_2 which are the projections of

point P onto the epipolar line. Therefore, if the two epipolar lines belong to the same epipolar plane, for each point observed on one epipolar line must be observed on the other epipolar line. We can use this property to reduce the search range from the whole image to an epipolar line. This geometry property is called epipolar constraint.

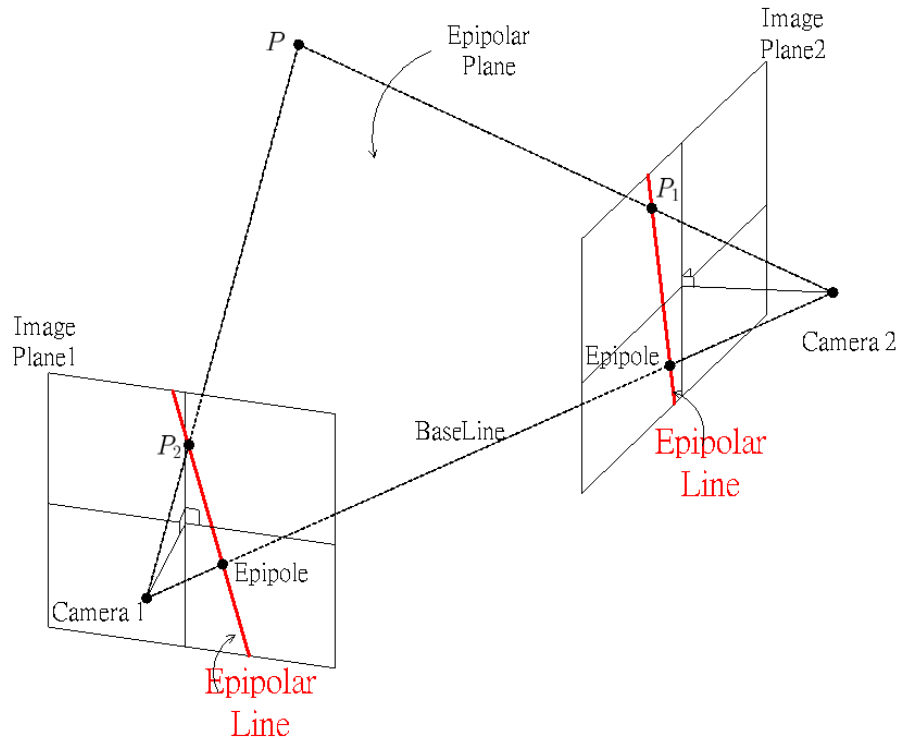


Fig. 2-1 Stereo image geometry

2.3 The General Structure of Matching Algorithm

According to Scharstein and Szeliski [5], the stereo correspondence algorithms generally consist of four parts:

1. Initial matching cost computation,
2. Cost aggregation,
3. Disparity computation and optimization,
4. Disparity refinement,

Fig. 2-2 shows the general procedure of a stereo correspondence algorithm. The output is the disparity map of an image pair input. The details of the four parts will be discussed in the following sections.

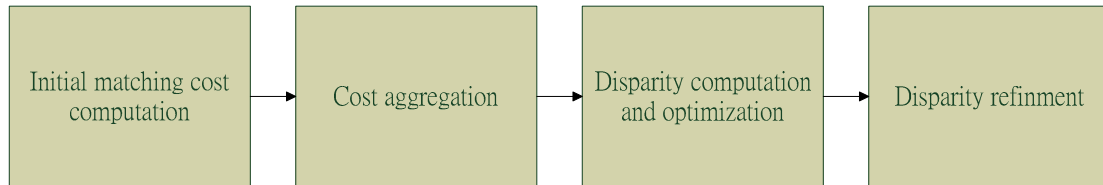


Fig. 2-2 Process of the general stereo correspondence algorithms

2.3.1 Initial Matching Cost Computation

The matching cost represents the dissimilarity between two pixels in our correspondence problem. The range of the disparity candidates is called the disparity range. The initial disparity space image $C_0(x, y, d)$ (Fig. 2-3) consists of the matching cost values over all pixels and all disparities.

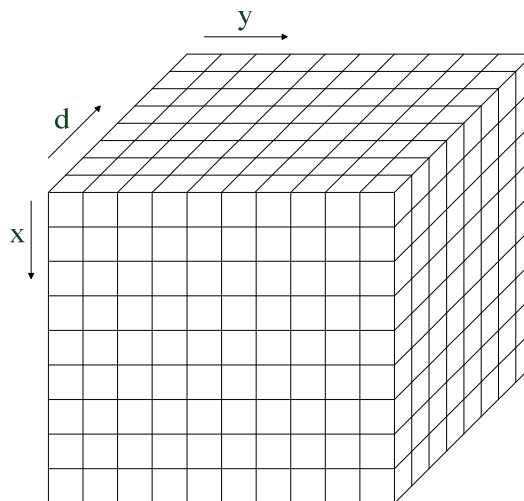


Fig. 2-3 Disparity space image

The most popular pixel-based match metrics are the squared intensity difference (SD) and the absolute intensity difference (AD). The correspondence problem to find

the best match between the candidate pixel and the reference pixel in the support region \mathcal{W} . In addition, Birchfield and Tomasi proposed a matching cost sensitive to image sampling [12][13]. The several match metrics are listed in Table 2-1 [6]. In our platform, the parameter *match_fn* selects the matching cost function we use. The general formula of matching cost computation can be written as

$$Cost(x, y, d) = Matching(I_L(x, y) - I_R(x - d, y)), \quad (2.1)$$

where I_L and I_R represent the left (reference) and the right images, respectively.

2.3.2 Cost Aggregation

After computing matching cost, we aggregate nearby pixel costs. Because the disparity values of the neighboring pixels should often be consistent, we select a support window to add up their costs. The cost aggregation can be formulated as

$$Cost_{aggr}(x, y, d) = \sum_i \sum_j Cost_{init}(x + i, y + j, d) \cdot \omega(x, y, i, j), \quad (2.2)$$

where $Cost_{init}$ is the initial matching cost calculated in the previous step. The ω function indicates the related weight of neighboring pixels contributing to the aggregated cost. Although the cost aggregation can reduce the noise effect, it blurs the edge of the object when aggregating the cost of difference objects. Therefore, how to design a good aggregation scheme is an important topic. In our platform, the parameter *aggr_fn* selects the aggregation method we use. Several aggregation methods are described below:

- Box filter: Use a separable moving average filter (add one right/bottom value, subtract one left/top). The decision of the window size will affect the performance and the computation time. If we want to implement real-time matcher, we should consider the window size as a vital factor.

Table 2-1 Match metrics for correspondence matching [6]

Match Metric	Definition
Normalized Cross-Correlation	$\frac{\sum_{(u,v) \in \mathcal{W}} (I_L(u, v) - \bar{I}_L) \cdot (I_R(u - d, v) - \bar{I}_R)}{\sqrt{\sum_{(u,v) \in \mathcal{W}} (I_L(u, v) - \bar{I}_L)^2 \cdot (I_R(u - d, v) - \bar{I}_R)^2}}$
Sum of Squared Difference	$\sum_{(u,v) \in \mathcal{W}} (I_L(u, v) - I_R(u - d, v))^2$
Normalized Sum of Squared Difference	$\sum_{(u,v) \in \mathcal{W}} \left(\frac{(\sum_{u,v} (I_L(u, v) - \bar{I}_L))}{\sqrt{\sum_{u,v} (I_L(u, v) - \bar{I}_L)^2}} - \frac{(\sum_{u,v} (I_R(u, v) - \bar{I}_R))}{\sqrt{\sum_{u,v} (I_R(u, v) - \bar{I}_R)^2}} \right)^2$
Sum of Absolute Difference	$\sum_{(u,v) \in \mathcal{W}} I_L(u, v) - I_R(u - d, v) $
Mutual Information	$\log \left(\frac{p(I_L(u, v) \cdot I_R(u - d, v))}{p(I_L(u, v)) \cdot p(I_R(u - d, v))} \right)$
<p>$I_L(u, v)$ and $I_R(u, v)$ represent the intensity value of left and right image where (u, v) is the index of pixel. \bar{I}_L represents the mean of intensity value in the support window \mathcal{W}. $p(\cdot)$ is the probability density function and d represents the disparity value.</p>	

- Binomial filter: The ω function is a separable FIR (finite-duration impulse response) filter. We use the coefficients $1/16\{1, 4, 6, 4, 1\}$ proposed by Burt and Adelson's [14] Laplacian pyramid.
- Minimum filter: The ω function is a sliding window with a location bias (Fig. 2-4). We can use a box filter and its center is not the candidate pixel. But the candidate pixel should be included in the shifted windows, it is called shiftable window [15]. We choose the minimum aggregation cost among all the shiftable window [15]. We choose the minimum aggregation cost among all the shiftable windows in the pre-selected range. The shiftable window can avoid aggregating the cost near object boundary.

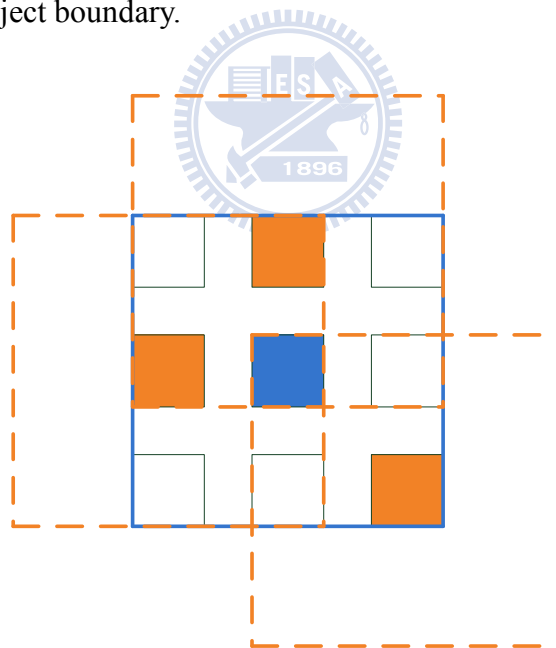


Fig. 2-4 An illustration of the shiftable window

2.3.3 Disparity Computation and Optimization

The disparity map can be obtained from the original matching cost or the aggregated cost. In addition, the disparity computation method can be categorized into

two types: the local and the global approaches. These two approaches are described as follows.

In a local method, the matching cost or the cost aggregation are the key components. The simplest local method is the Winner Take All (WTA) algorithm, in which the disparity of each pixel $d(x, y)$ is determined by minimizing the matching cost or aggregated cost in the disparity search range $[0, d_{max}]$, that is,

$$d(x, y) = \arg \min_{d_n \in [0, d_{max}]} Cost(x, y, d_n), \quad (2.3)$$

Moreover, the disparity of each pixel is independently calculated.

In a global method, we define an energy function, which includes a data term and a smooth term. The data term includes the cost function that we discuss previously, and the smooth term represents the smoothness penalty of the disparity map. The detail of the energy function will be discussed in section 3.1. One of the earlier proposed global optimization methods is the dynamic programming (DP) [16]. The dynamic programming scheme optimizes the energy function of each scanline independently. Fig. 2-5 shows a stereo matching using dynamic programming for a pair of corresponding scanline. We select the minimum path through the matrix of all pairwise matching costs. The lowercase array represents the intensities along a scanline of the left image. The uppercase array represents the intensities along a corresponding scanline of the right image. The matches are indicated by M, and the

partially occluded points are indicated by L or R, which based on the points only visible in the left image or the right image. In this example, the disparity range is 0-4, which indicated by the non-shaded boxes. The shaded boxes are disparities outside this range. Although dynamic programming can optimize the horizontal global information, the vertical correlation is not considered. The disparity maps produced by dynamic programming may exhibit horizontal streaks, and it reduces the subjective quality of the synthesized image.

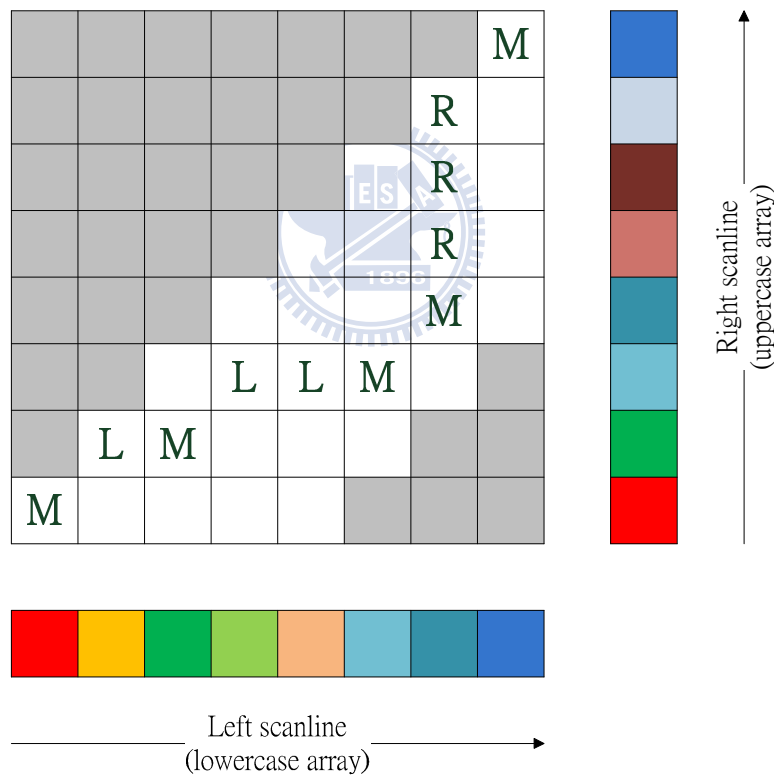


Fig. 2-5 Stereo matching using dynamic programming

For considering the vertical and horizontal information simultaneously, the energy minimization using the so-called graph-cuts technique has been proposed. This optimization algorithm performs well in disparity estimation. Unfortunately, the

computation and the storage requirement for graph-cuts algorithm are enormous. The detail of graph-cuts will be discussed in sections 3.2 and 3.3, and we will propose our algorithm for reducing the computation time in chapter 4.

2.3.4 Disparity Refinement

Most stereo correspondence algorithms produce an integer disparity map. The integer disparity is not good enough for some applications, which require good-quality synthesized images. To improve the synthesis result, many algorithms apply a disparity refinement stage in the procedure of stereo correspondence algorithms. In the disparity refinement stage, the sub-pixel disparity map can be computed. We can increase the resolution of the disparity map with a little additional computation. However, the goal of this research is to decrease the computation and maintain the quality of the disparity map. Because we do not focus on the synthesis result, we do not use the disparity refinement stage in the reference software (platform) by setting the Boolean variable *refine_subpix* false.

2.4 A Taxonomy Evaluation

2.4.1 Overview of the Platform

The source codes downloaded from the middlebury website are used as our platform [5]. Fig. 2-6 depicts the program structure of the middlebury platform. In the beginning, the *main* file calls the *interpretCommandLine* function which is included in

the *StereoIO* file. In the *interpretCommandLine* function, the first step initializes the entire parameters (*StereoParameter.cpp*), and then we adjust the values of parameters specified by the configuration file or the command lines (*ParameterIO.cpp*). The next step executes the stereo matching program (*StereoMatcher.cpp*), which is the primary part in the platform. The stereo matching can be divided into four components as depicted Fig. 2-2. They are initial matching cost computation, cost aggregation, disparity computation (optimization), and disparity refinement. At the end, the program evaluates the quality by comparing the computed disparity map and the ground truth disparity map (*StcEvaluate.cpp*).

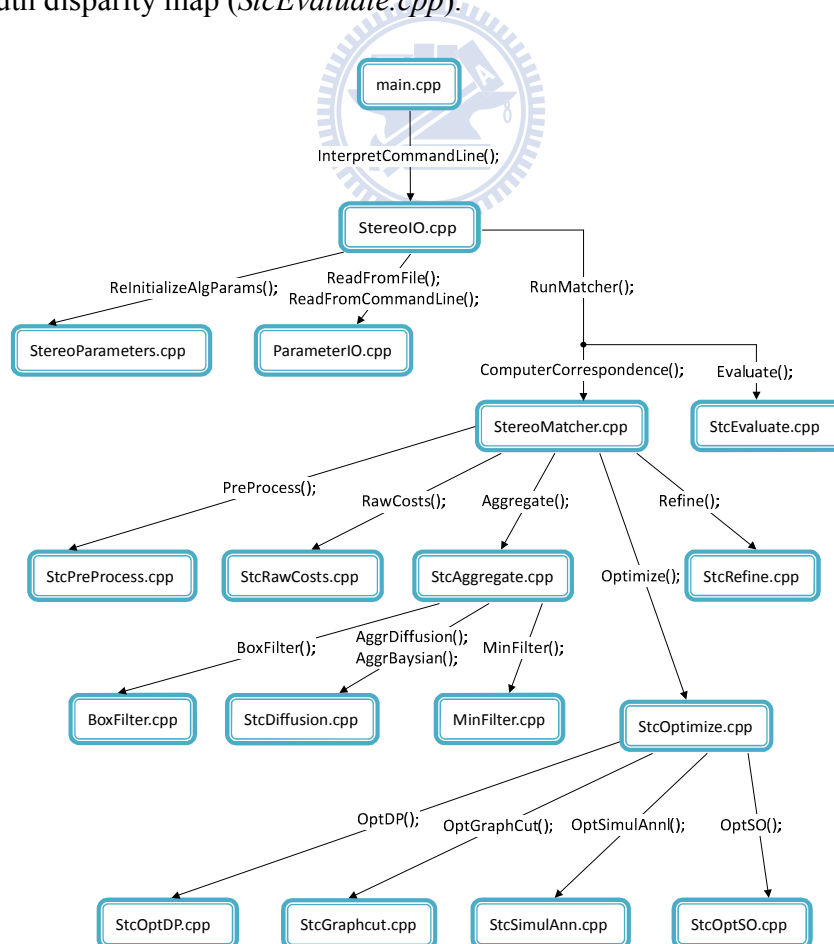


Fig. 2-6 Program structure of the middlebury platform

2.4.2 Quality Metrics

To evaluate the quality of a disparity map computed by a stereo algorithm, we compute the following two quality metrics in our platform:

1. RMS (root-mean square) error between the computed disparity map $d_C(x, y)$ and the ground truth map $d_T(x, y)$ can be written as

$$R = \sqrt{\frac{1}{N} \sum_{(x,y) \in \mathcal{A}} (d_C(x, y) - d_T(x, y))^2} \quad (2.4)$$

where R is the RMS error, N is the total number of pixels and \mathcal{A} denotes the image area.

2. Percentage of bad matching pixels can be written as

$$B = \frac{1}{N} \sum_{(x,y) \in \mathcal{A}} f(\delta_d), \text{ where } f(\delta_d) = \begin{cases} 1, & \text{if } |d_C(x, y) - d_T(x, y)| > \delta_d \\ 0, & \text{otherwise} \end{cases} \quad (2.5)$$

where B is the percentage of bad pixels and δ_d is a disparity error tolerance. We simply adopt the default setting of the published platform and set $\delta_d = 1$ in our experiment.

Besides computing the two quality metrics over the whole image, we also compute the two quality metrics over three difference kinds of regions, which are textureless regions \mathcal{T} , occluded regions \mathcal{O} and depth discontinuity regions \mathcal{D} . Their symbols and descriptions are listed in Table 2-2.

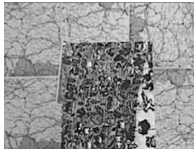




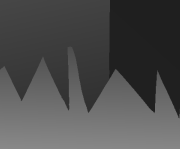


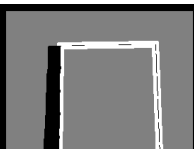



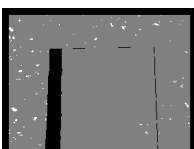
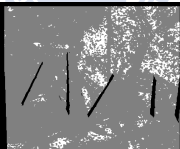


Table 2-2 Descriptions of error metrics [5]

PARAMETER NAME	SYMBOL	DESCRIPTION
<i>rms_error_all</i>	R	RMS disparity error
<i>rms_error_nonocc</i>	$R_{\overline{\mathcal{O}}}$	RMS disparity error (no occlusions)
<i>rms_error_occ</i>	$R_{\mathcal{O}}$	RMS disparity error (at occlusions)
<i>rms_error_textureless</i>	$R_{\overline{\mathcal{T}}}$	RMS disparity error (textureless)
<i>rms_error_textured</i>	$R_{\mathcal{T}}$	RMS disparity error (textured)
<i>rms_error_discont</i>	$R_{\mathcal{D}}$	RMS disparity error (discontinuities)
<i>bad_pixels_all</i>	B	Bad pixel percentage
<i>bad_pixels_nonocc</i>	$B_{\overline{\mathcal{O}}}$	Bad pixel percentage (no occlusions)
<i>bad_pixels_occ</i>	$B_{\mathcal{O}}$	Bad pixel percentage (at occlusions)
<i>bad_pixels_textureless</i>	$B_{\overline{\mathcal{T}}}$	Bad pixel percentage (textureless)
<i>bad_pixels_textured</i>	$B_{\mathcal{T}}$	Bad pixel percentage (textured)
<i>bad_pixels_discount</i>	$B_{\mathcal{D}}$	Bad pixel percentage (discontinuities)

2.4.3 Test Data

We download the test data on the middlebury website. The test data set we used is shown in Table 2-3. The four sequences, which are map, tsukuba, sawtooth, and venus, are the most commonly used ones for quality evaluation.

Table 2-3 Test data [5]

	Map	Sawtooth	Tsukuba	Venus
Image size <i>(width × height)</i>	284 × 216	434 × 380	384 × 288	434 × 383
Input				
Ground Truth				
Occlusion and discontinuities				
Occlusion and textureless				

Chapter 3 Energy Minimization by Graph Cuts

3.1 Overview

In disparity estimation, the graph cuts and the belief propagation algorithms are generally recognized as the better global optimization methods. Unfortunately, their computation time is very high. Because we only focus on the graph cuts method, we only describe the procedure of the graph cuts in this chapter. The maximum flow/minimum cut (max-flow/min-cut) problem and an algorithm which solves the max-flow/min-cut problem are introduced [17][18]. Based on the graph cuts, two algorithms have been proposed for solving the stereo correspondence problem by minimizing the energy function, namely α - β swap and α -expansion [8].

3.2 Max-Flow and Min-Cut Problem

First, we glance at the graph theory. In Fig. 3-1, we show a simple example of a graph \mathcal{G} . A directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined as a set of nodes (vertices) \mathcal{V} and a set of directed edges \mathcal{E} that connect the nodes. In the graph, a source terminal as s and a sink terminal as t are denoted. A cut is a set of edges $\mathcal{C} \subset \mathcal{E}$ such that the two terminals become separated on the induced graph $\mathcal{G}' = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$. A cut can also be represented by $(\mathcal{S}, \mathcal{T})$ which produces a partition of \mathcal{V} into \mathcal{S} and $\mathcal{T} = \mathcal{V} - \mathcal{S}$, such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$.

A minimum cut is a cut whose cost is the minimum over all possible cuts of \mathcal{G} .

The minimum cut problem can be solved by finding a maximum flow from the source s to the sink t . In other words, the maximum source-to-sink flow is equal to the cost of the minimum cut in \mathcal{G} . Maximum flow can be considered the maximum “amount of water” that can be sent from the source to the sink, and the cost of edge can be considered the capacity of a directed “pipe”.

The algorithms to solve the maximum flow problem can be classed into two groups: Ford-Fulkerson algorithm [19] and push-relabel algorithm [20]. The Ford-Fulkerson algorithm examines the whole residual network to find an augmenting path. The algorithm begins with no flow and runs iteratively. At each iteration, the flow is increased by finding the augmenting path from the source to the sink in the residual network. The process repeats until no further augmenting path we can find, and the flow is the maximum flow.

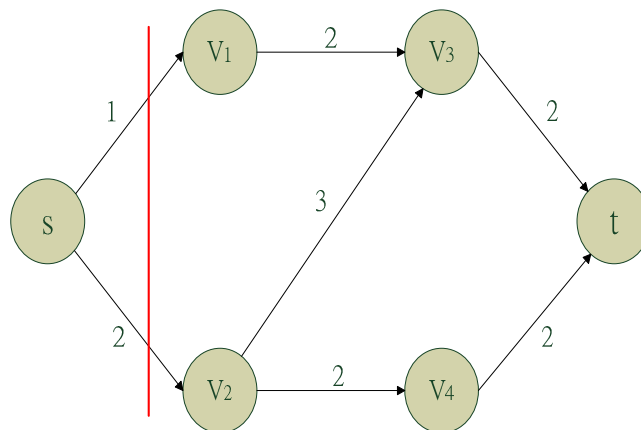


Fig. 3-1 A simple example of the graph \mathcal{G} and the minimum cut (the red line)

If we used a breath-first search to implement the augmenting path calculation in the Ford-Fulkerson algorithm, the bound of running time can be improved to $O(N_V N_E^2)$, where N_V is the number of nodes and N_E is the number of edges in the graph. We call the Ford-Fulkerson algorithm so implemented the Edmonds-Karp algorithm. Push-relabel algorithms look only at the node's neighbors in the residual network and process one node at a time. Compared with the Ford-Fulkerson algorithm, the push-relabel algorithms are local methods and a simple implementation that runs in $O(N_V^2 N_E)$ time. Besides, unlike the Ford-Fulkerson, the push-relabel algorithms do not maintain the flow-conservation property throughout their execution. Therefore, we use push-relabel algorithms in this research instead of the Ford-Fulkerson method. We will describe the detail of push-relabel algorithms in the next section.

3.3 Push-Relabel Algorithm

In a push-relabel algorithm, the directed edges correspond to pipes as the Ford-Fulkerson algorithm, but the intuition of nodes is different from the Ford-Fulkerson algorithm. The nodes, which can be regarded as the pipe junctions, have two parameters. One is the accumulation of the fluid, and the other one is the node heights. The node heights determine how flow is pushed. We only push flow from a higher node to a lower node.

In a push-relabel algorithm, there are two basic operations, the *push operation*

and the *relabel operation*. The *push operation* performs pushing flow excess from a node to one of its neighbors. The *relabel operation* can increase height of a node. We will describe the flow of push-relabel algorithm by a simple example, which is shown in Fig. 3-2. In Fig. 3-2 (a), a graph with nodes and directed edges is shown. The number at the edge represents the cost (capacity of the pipe). In the beginning, the parameters of each node are initialized as Fig. 3-2 (b). The height of the source s is set to the number of the nodes in the graph. The height of other nodes is initialized to zero. In Fig. 3-2 (c), we employ the first *push operation* that saturate all outgoing edges from the source s . The flow follows the direction of the edges. When an edge is saturated with flow along its original direction, we then change the direction of the edge. Also, the flow accumulations, e , of node V_1 and V_2 are changed to one and two. Because the flow accumulations of node V_1 and V_2 cannot be pushed to the other node after the first *push operation*, we must increase their height and this processing is called *relabel operation*. The *relabel operation* is depicted in the Fig. 3-2 (d). After the *relabel operation*, we can push the flow accumulations of V_1 and V_2 to the lower nodes as shown in Fig. 3-2 (e). We perform *push operation* and *relabel operation* repetitively until there is no accumulation of flow in the each node except for the sink t , and the final graph is depicted in Fig. 3-2 (f).

The maximum flow problem is solved when the final graph is obtained. The final

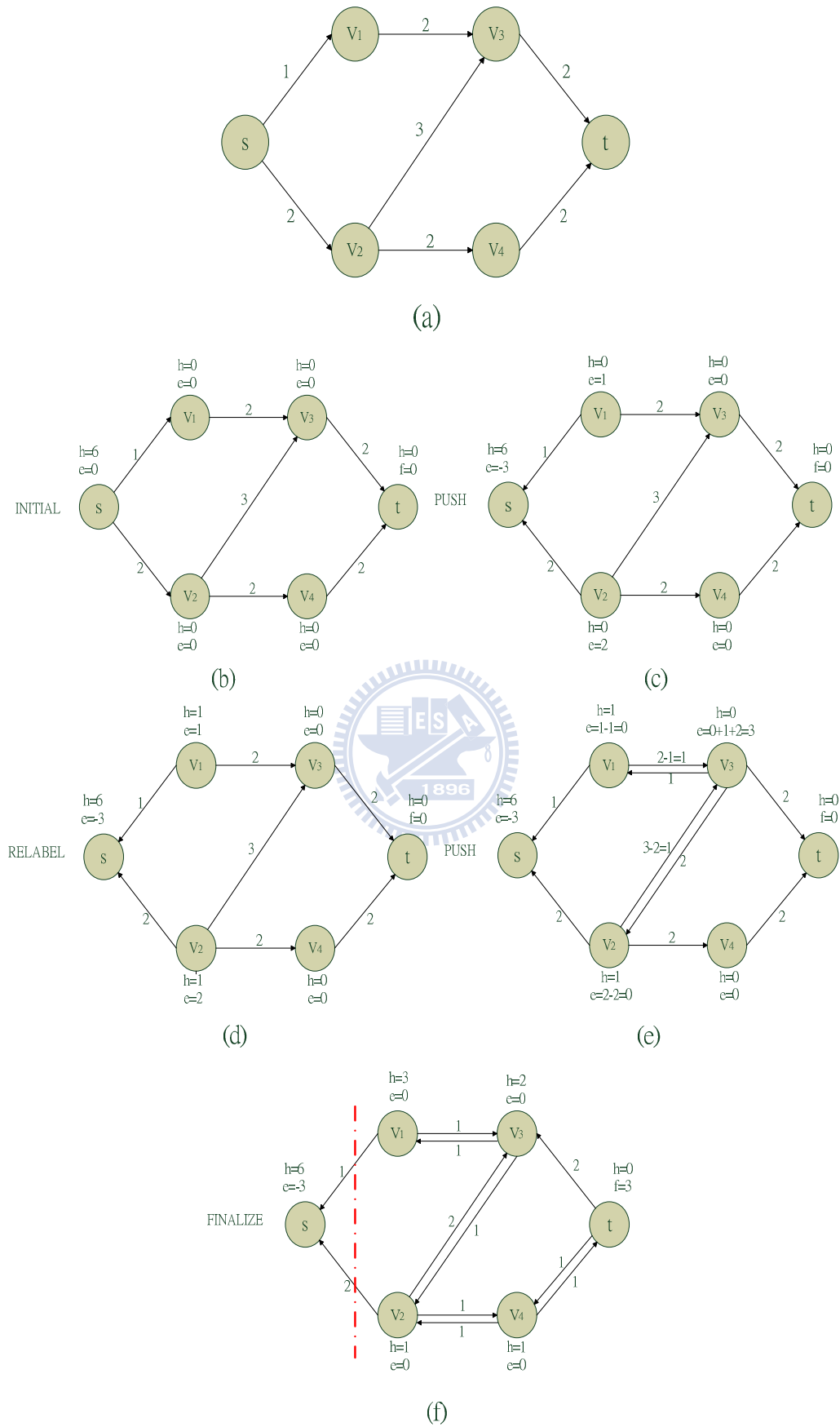
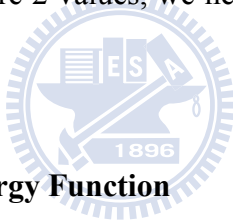


Fig. 3-2 A simple example of push-relabel algorithm

graph provides the minimum cut. A minimum cut separates the original graph \mathcal{G} into two parts \mathcal{S} and \mathcal{T} , such that $\mathcal{S} = \{\text{all nodes reachable from source } s\}$ and $\mathcal{T} = \mathcal{G} - \mathcal{S}$.

3.4 Energy Minimization using Graph Cuts

Disparity estimation using graph cuts outperforms many other optimization methods. Before explaining how to compute the disparity map using graph cuts, we introduce the general form of energy function. In addition, we will introduce a method to minimize the energy function by graph cuts, α - β swap [8]. Finally, because the disparity values are generally more 2 values, we need to build the graph many times, which is called multiway cuts [9].



3.4.1 The General Form of Energy Function

The general form of energy function can be written as

$$E(f) = E_{data}(f) + \lambda E_{smooth}(f) \quad (3.1)$$

where λ is a parameter, which controls the effect of the smooth energy term.

The data energy term, $E_{data}(f)$, represents the dissimilarity between the left and the right images when the disparity map is f . The form of data energy term is typically

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \quad (3.2)$$

where p is a pixel, \mathcal{P} is the set of total pixels in the left image, f_p is the disparity

value of pixel p , and $D_p(f_p)$ represents the cost when the disparity value of pixel p is f_p . We can select different matching cost functions and different aggregation methods to generate the cost array that was explained in section 2.3.1 and 2.3.2. The data energy term is the summation of all costs when a disparity map is given.

The smooth energy term, $E_{smooth}(f)$, measures the extent to which f is not piecewise smooth. If there are some non-smooth regions on the disparity map, we should add some penalties on total energy. The smooth data term will make the disparity map much smooth everywhere. The smooth energy term typically has the form

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \quad (3.3)$$

where \mathcal{N} is the set of interacting pairs of neighboring pixels, and $V_{p,q}(f_p, f_q)$ is the an interacting function, which has many different forms. In our platform, we use the Potts model as our interacting function. The Potts model can be represented as

$$V_{p,q}(f_p, f_q) = K \times T(f_p \neq f_q), \quad (3.4)$$

where $T(\cdot)$ is 1 if the argument is true, and otherwise, 0, and K is a penalty constant. This model encourages disparity values consisting of several regions, where pixels in the same region have equal disparity value. We also call it a piecewise constant model.

According to the previously introduced energy function in this section, we know

the impact of energy function. We next describe how we minimize the energy function by graph cuts in section 3.4.2.

3.4.2 The α - β Swap Method

First, we review some basic fact about graphs which is used to minimize the energy function of disparity estimation. A directed weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ consists of a set of nodes \mathcal{V} and a set of directed edge \mathcal{E} that connect them. The nodes are usually composed of the pixel nodes and the terminal nodes. The pixel nodes and the terminal nodes respectively correspond to the pixels in the image and the disparity values which we can assign to pixels. In Fig. 3-3, we show a simple example of a 3×3 image with two disparity values. The two terminals are usually called the *source*, s , and the *sink*, t . For more clearly seeing, we provide a simpler illustration in Fig. 3-4, whose pixel nodes are arranged in 1D. The set of pixels in the 1D image is $P_{\alpha\beta} = P_{\alpha} \cup P_{\beta}$, where $P_{\alpha} = \{p, r, s\}$ and $P_{\beta} = \{q, \dots, w\}$. The weight information of the edges is shown in Table 3-1. The edges can be classed into two groups: *t-links* and *n-links*. The *t-links* (terminal links) connect each pixel node to the terminals α and β , which is called t_p^{α} and t_p^{β} , respectively. The *n-links* (neighbor links) connect each pair of pixels $\{p, q\} \subset P_{\alpha\beta}$, that are neighbors ($\{p, q\} \in \mathcal{N}$). The symbol of the *n-links* is $e_{\{p,q\}}$.

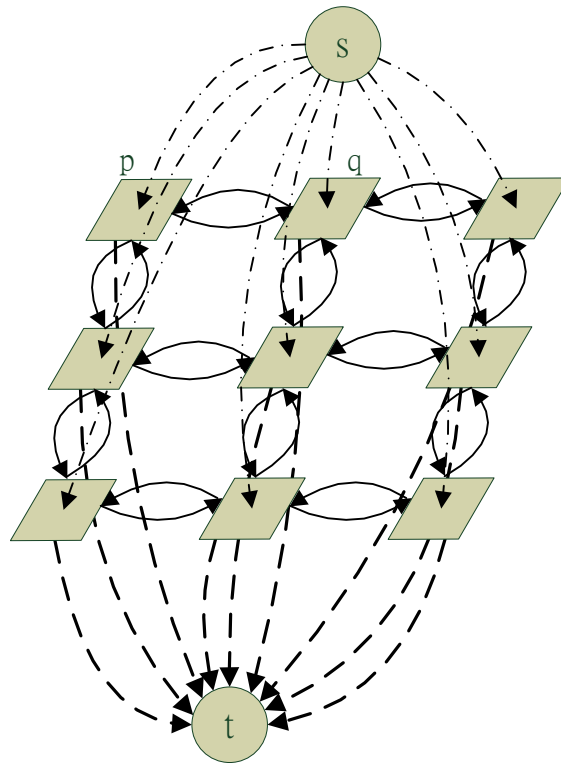


Fig. 3-3 An example of a directed weighted graph

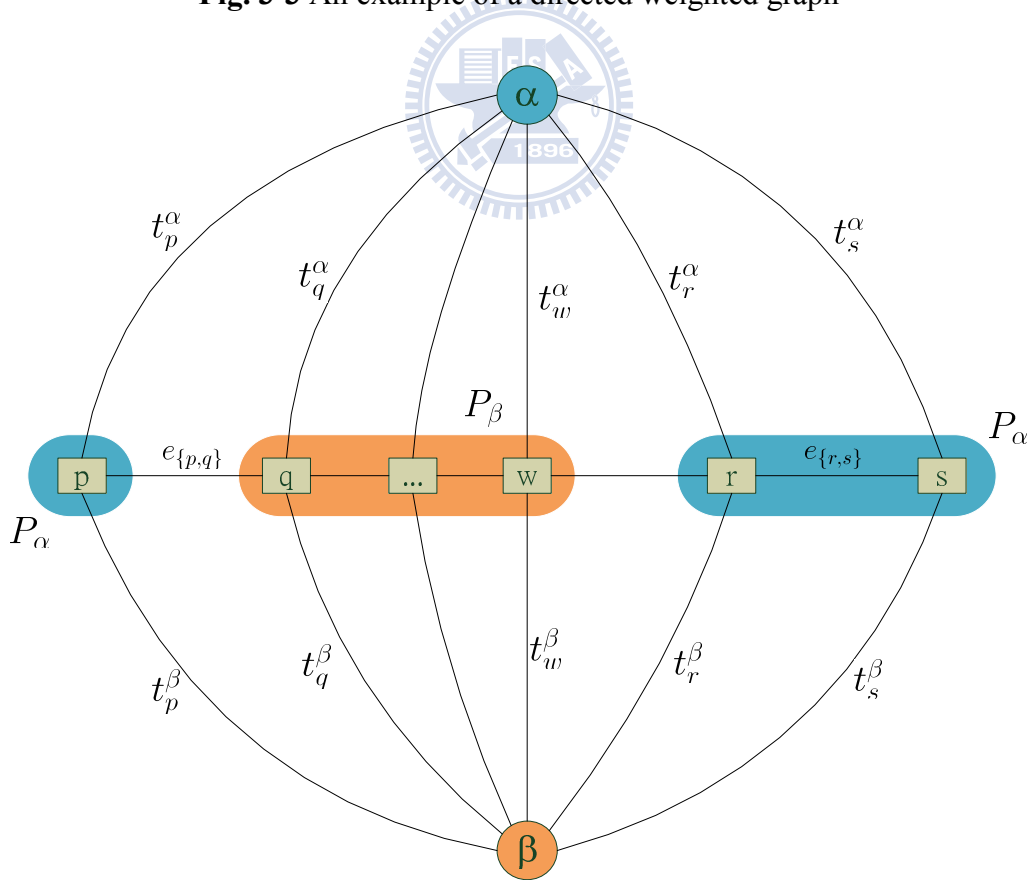


Fig. 3-4 An example of the graph for a 1D image

Table 3-1 Weight information of the edges

edge	weight	for
t_p^α	$D_p(\alpha)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V_{p,q}(\alpha, \beta)$	$\{p, q\} \in \mathcal{N}, p \in \mathcal{P}_{\alpha\beta}, q \in \mathcal{P}_{\alpha\beta}$

From the Table 3-1, we know that the weight of edge t_p^α is $D_p(\alpha)$, which is the data energy term. $D_p(\alpha)$ is the cost value of pixel p , which is assigned a disparity value α . The edge t_p^β is similarly defined. The weight of edge $e_{\{p,q\}}$ is $V_{p,q}(\alpha, \beta)$, which is the smooth energy term. $V_{p,q}(\alpha, \beta)$ is an interacting function as we said previously. When the graph is constructed, we find the minimum cut of the graph by the push-relabel algorithm. We discuss the properties of a cut by a simple example in Fig. 3-5. If the cut of Fig. 3-5 (a) is the minimum cut, we will assign disparity value α to pixels p and q . The case of Fig. 3-5 (b) is similar to (a). If the cut of Fig. 3-5 (c) is the minimum cut which includes the n -link, $e_{\{p,q\}}$, we will assign disparity value β and α to the pixels p and q , respectively.

After finding the minimum cut of the graph, we can assign the disparity value to each pixel in $\mathcal{P}_{\alpha\beta}$. The method that we adjust the disparity map is called α - β swap. Fig. 3-6 (a) is an example of the initial disparity map. Although the disparity map is always a gray level picture, we fill the region of disparity value α with red (darkest) color for illustration. After an α - β swap, the disparity map is modified to Fig. 3-6 (b). The disparity map in (b) is much smoother than (a) in the regions of α and β ,

because we minimize the energy function by graph cuts. However, a disparity map may contain more than two disparity values. We cannot simply employ the graph cuts once and, therefore, we will introduce the multiway cut algorithm to solve this problem.

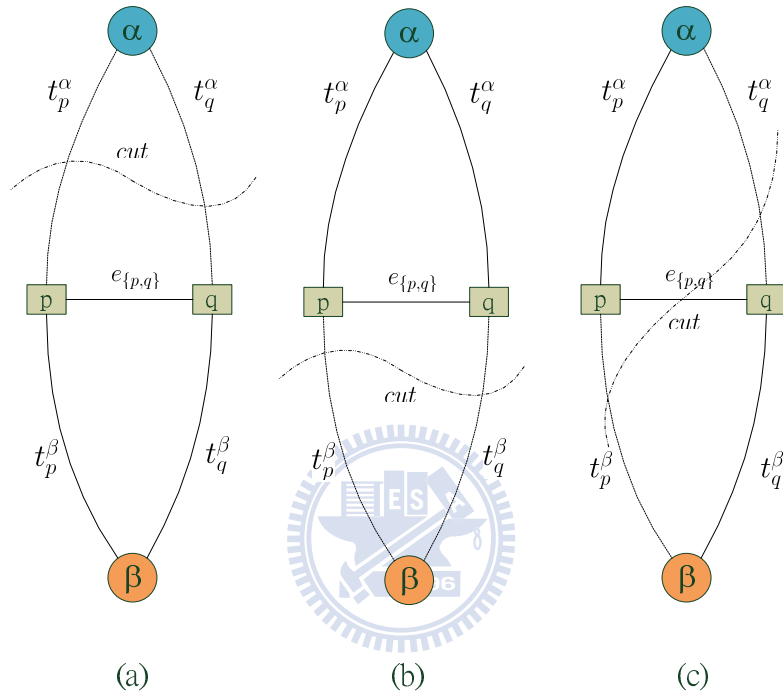


Fig. 3-5 Properties of a cut on the graph

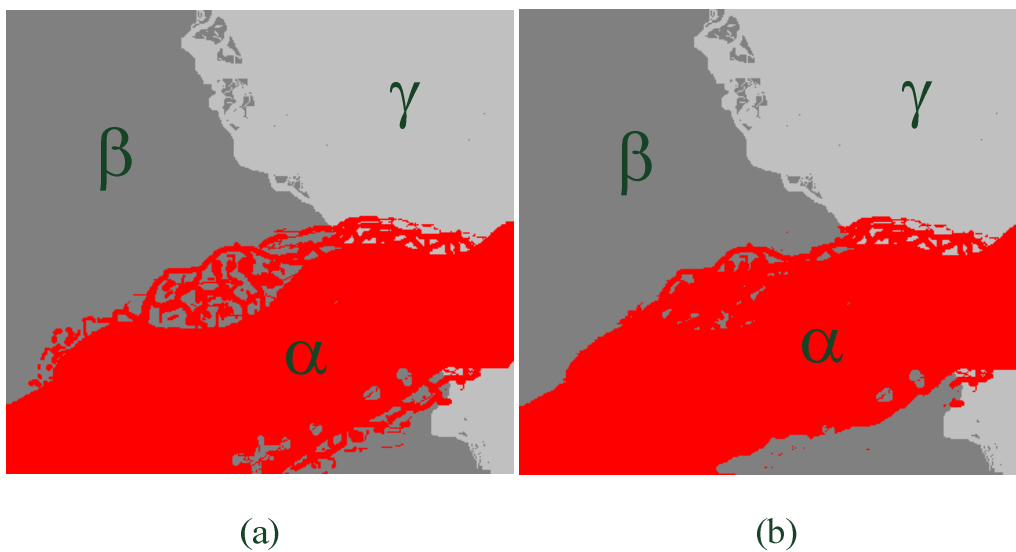


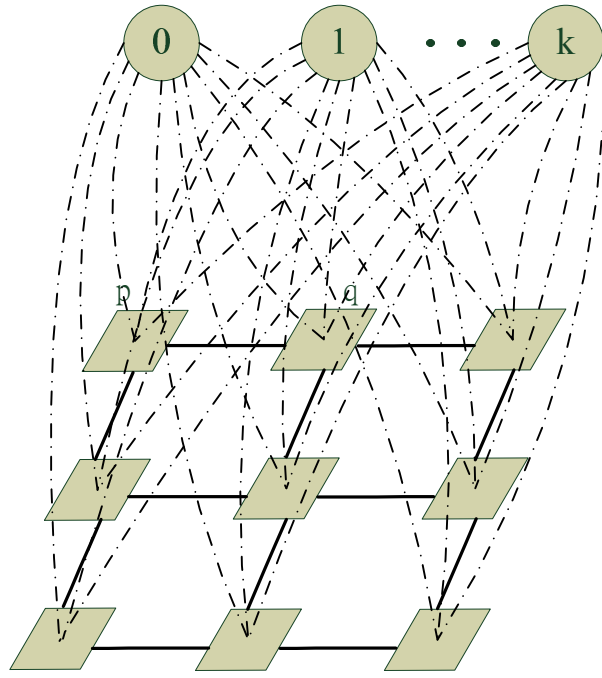
Fig. 3-6 The change of disparity map after an α - β swap

3.4.3 Multiway Cut Algorithm

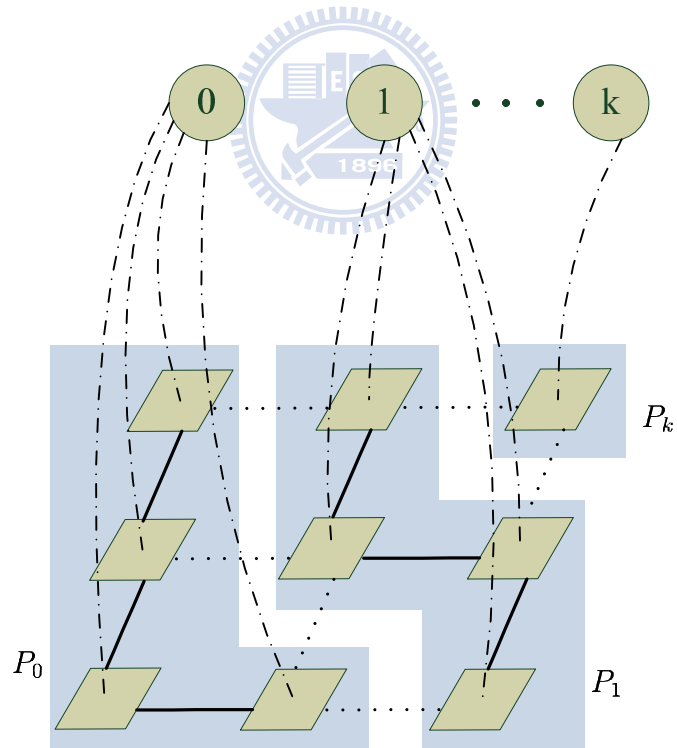
Because the number of disparity values is greater than two, there are multiple terminals in the graph as shown in Fig. 3-7. The α - β swap method processes only two terminals at a time, and, therefore, we must use multiway cut algorithm to solve the problem. We describe the detail of multiway cut algorithm as follows.

- (1) Start with an initial disparity map which may be given by the WTA method.
- (2) Randomize the disparity sequence, and pick up two disparity values as the terminals of the graph. Note that only the selected pixels can be the pixel nodes in the graph.
- (3) Employ the algorithm which we introduce previously to find the minimum cut.
- (4) Repeat step (3) for every possible pair of disparity values.
- (5) Repeat the step (3) and step (4) until the energy does not change.

After the total flow of multiway cut algorithm flow is computed, we reach a sub-optimal total energy of the disparity map. Fig. 3-8 shows the flowchart of the GC algorithm. (1) corresponds of to the S1, and (2) is composed of S2 and S3. (3) is equal to S4. In addition, (4) and (5) are the inner and outer loop.



(a)



(b)

Fig. 3-7 (a) An example of the graph with multiple terminals $\mathcal{L} = \{0, 1, \dots, k\}$ (b) An induced graph by a multiway cut (dotted lines indicate cut edges)

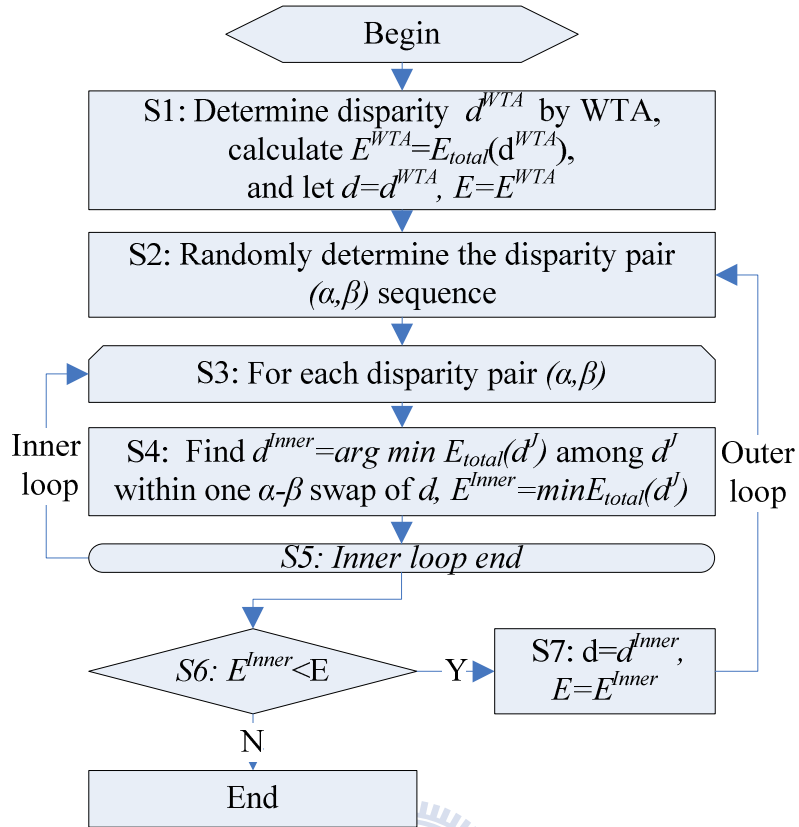


Fig. 3-8 Flowchart of GC

Chapter 4 The α - β Swap Algorithm Speed-Up and Early Termination

4.1 Overview

Among the global optimal algorithms, graph cuts (GC) and its variations [8]-[11] generally show very good performance in their disparity estimation quality. Therefore, GC is chosen as the target algorithm for speed up. The flowchart of the original GC is shown in Fig. 3-8 which we was discussed in Chapter 3 previously. In this chapter, we propose a fast GC algorithm for disparity estimation purpose. Two accelerating techniques are suggested: one is the early termination rule, and the other prioritizes the α - β swap pair search order.

4.2 Early Termination of Energy Minimization Process

In this section, we examine the energy minimization process of GC to determine an early termination threshold. First, we define a terminology to be used in the following discussions. Fig. 4-1 shows the probability distribution of all possible disparity values for a test image pair after the i -th outer-loop iteration. This probability distribution (sequence) forms a vector: $\mathbf{P}^{i\text{-th}}$. We measure the similarity between two vectors by their inner product and thus theta θ , the angle between $\mathbf{P}^{i\text{-th}}$ and $\mathbf{P}^{(i-1)\text{-th}}$, is defined by (4.1).

$$\theta = \cos^{-1} \left(\frac{\mathbf{P}^{i\text{-th}} \cdot \mathbf{P}^{(i-1)\text{-th}}}{|\mathbf{P}^{i\text{-th}}| \cdot |\mathbf{P}^{(i-1)\text{-th}}|} \right) \quad (4.1)$$

Fig. 4-2 shows the values E and θ after each outer-loop iteration on the test image ‘sawtooth’, and Fig. 4-3 shows the RMS_error and Bad_pixel of the corresponding disparity map. In the energy minimization process, GC monotonically decreases E . However, the quality metrics slightly fluctuate when E reaches its minimum. The other test image pairs show similar results. When the decrease in E is small, further iterations may not necessarily improve the quality, even though the energy level can be further lowered slightly. Therefore, we suggest an early termination mechanism to save computation. The optimization process terminates when the angle θ between $\mathbf{P}^{i\text{-th}}$ and $\mathbf{P}^{(i-1)\text{-th}}$ is smaller than a given threshold $\theta_{\text{threshold}}$. That is, when the change between $\mathbf{P}^{i\text{-th}}$ and $\mathbf{P}^{(i-1)\text{-th}}$ is small, the iteration stops.

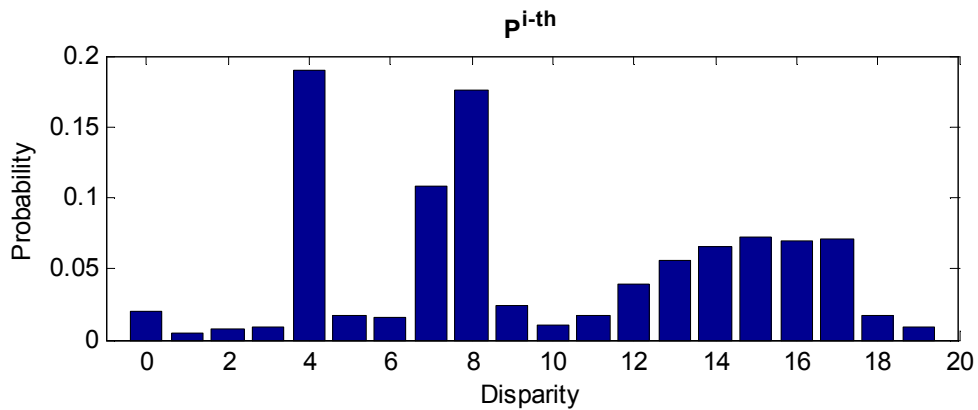


Fig. 4-1 Disparity distribution of the test image pair ‘sawtooth’ after the i -th iteration of the outer loop ($\mathbf{P}^{i\text{-th}}$)

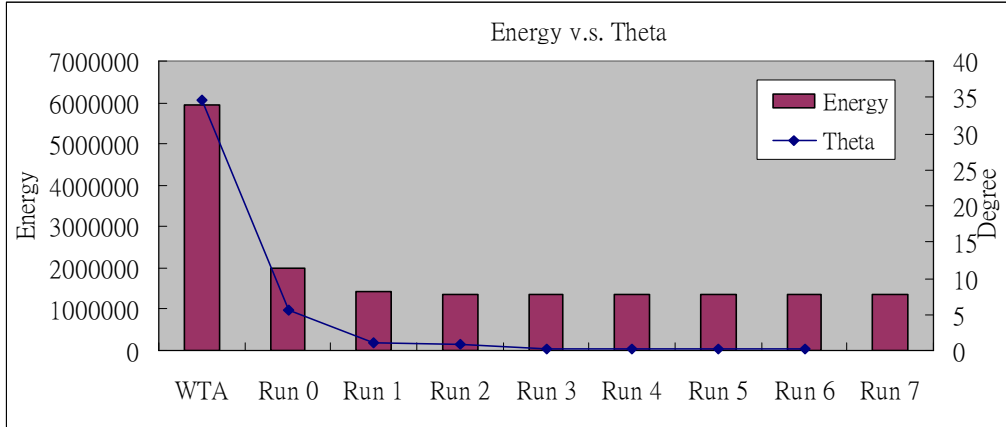


Fig. 4-2 E and θ in the energy minimization process of the test image pair ‘sawtooth’



Fig. 4-3 RMS_error and Bad_pixel in the energy minimization process of ‘sawtooth’

4.3 Prioritizing the α - β Swap Pair Sequence

The original GC scheme randomly selects a disparity pair (α, β) from the disparity candidate set \mathcal{L} and then does the α - β swap for all possible disparity values. However, let α' and β' be a chosen disparity pair. If (α', β') is the best disparity pair for only a few nodes, this specific α' - β' swap has a limited impact on minimizing the total energy but it consumes the computing power. Therefore, if we find an effective strategy to prioritize the α - β swap pairs, i.e., the better matched

pairs are tested first, then the total energy converges faster.

After each run of the outer loop, we obtain $d(x, y)$, E_{data} , E_{smooth} , and E_{total} for each node. Consequently, we have the disparity probability (the number of nodes with a specific disparity divided by the number of all pixel nodes), E_{data} , E_{smooth} , and E_{total} for each disparity value. Fig. 4-4 shows the disparity distribution of the test image pair 'sawtooth'. Part (a) is the probability after the first iteration (denoted by \mathbf{P}^{1-st}). Part (b) is the probability difference between the first and the final iterations (denoted by $\mathbf{P}^{final} - \mathbf{P}^{1-st}$), which represents the probability difference needed to be adjusted by the iterative algorithm. Typically, there are only a few objects in an image; therefore, the disparity distribution is dominated by a few disparity values. Often, the dominated disparity values show up after the first couple of iterations. That is, their probabilities are higher than the other disparities. Hinted by this observation, the disparity probability distribution can be used as clues for choosing the final disparity values. In this section, we prioritize the disparity pairs (α, β) according to their probability although the other attributes such as E_{data} , E_{smooth} , and E_{total} may also be used for prioritization purpose. We will discuss the difference between them in the next section.

The benefits of prioritizing the α - β swap pair search order mainly come from the early iterations of the outer loop. With the correctly prioritized disparity

candidates, we reach the final goal much faster and thus save computation.

Fig. 4-5 shows the flow chart of our proposed fast GC, which is the combination of prioritizing the α - β swap pair sequence and the early termination (section 4.2) technique. The modifications to the original GC scheme are steps 2, 7, and 8. In Step 2, we prioritize the disparity pair (α, β) search order based on their disparity probabilities.

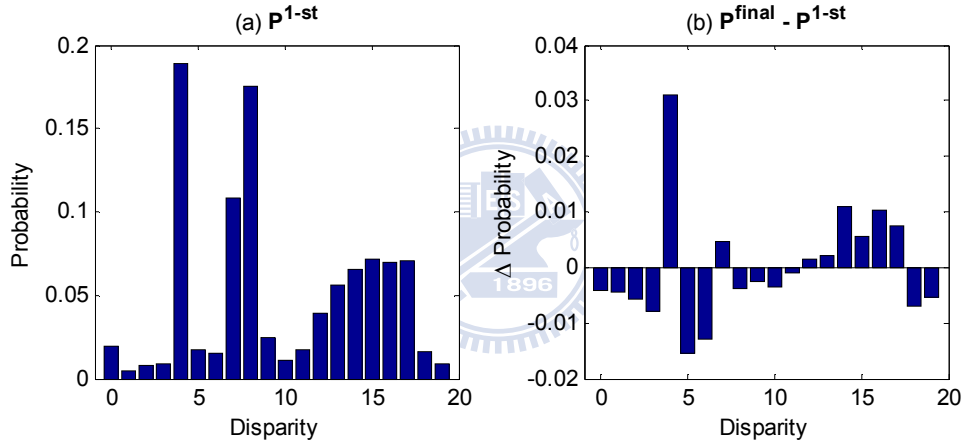


Fig. 4-4 Disparity distribution of ‘sawtooth’ after the 1st outer-loop iteration

We then perform the α - β swap sequentially according to the priority order (probability) of disparity pairs. In step 7, we also calculate θ value. In the extra step 8, we check whether θ is larger than a given $\theta_{\text{threshold}}$. If so, we run another iteration of the outer loop. Otherwise, we terminate the optimization process. The value of $\theta_{\text{threshold}}$ is 1° (degree) in our experiment. It is empirically determined.

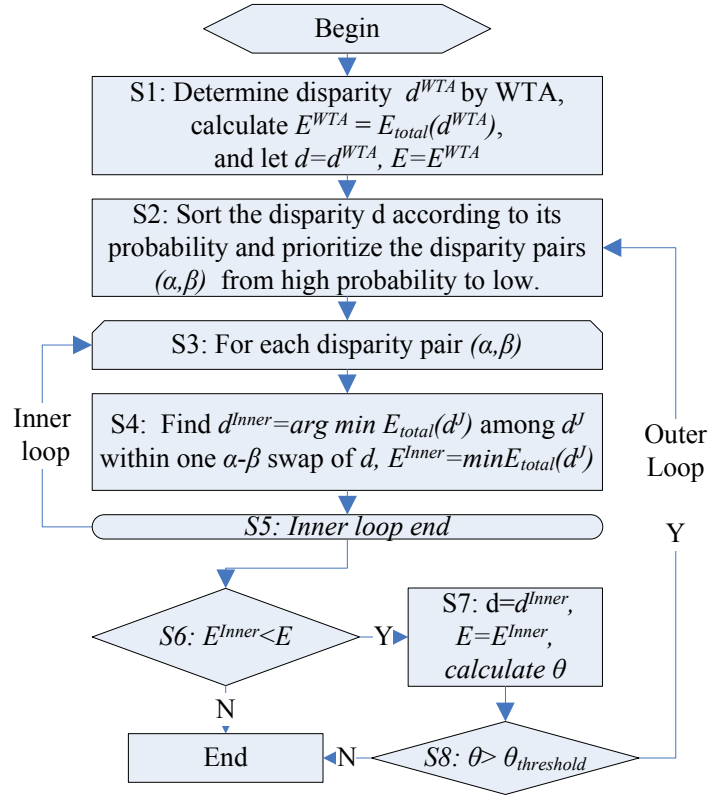


Fig. 4-5 Flow chart of our proposed fast GC

4.4 Simulation Results and Discussions

In this section, we will show the experiment environment setting and the simulation results for different criterions. The improvement of our proposed algorithm is also shown in this section. In addition, we will analyze and discuss the simulation results.

4.4.1 Experiment Environment Setting

We implement our proposed algorithm and test it on the test bed retrieved from the Middlebury stereo vision web page. Four test image pairs – ‘Map’, ‘Sawtooth’, ‘Tsukuba’, and ‘Venus’ (all with ground truth disparity maps) – are in use. Our simulation platform is a PC with Intel Core2Quad Q6600 and 4G RAM. The

performances of the original GC and our proposed fast GC are measured by six metrics: computing time, R (rms_error_all), B (bad_pixels_all), $B_{\mathcal{O}}$ (bad_pixels_nonocc), $B_{\mathcal{T}}$ ($bad_pixels_textureless$), and $B_{\mathcal{D}}$ ($bad_pixels_discont$). Table 4-1 shows the important parameters in our implementation. The simulation results of the original GC with the same setting are close to those in the Middilebury web page.

Table 4-1 The experiment environment setting

Parameter	Value	Meaning
$match_fn$	1	Absolutely difference
$match_max$	1000	No truncation
$match_interval$	1	BT
opt_fn	4	GC
$aggr_iter$	0	No aggregation
$opt_smoothness$	20	Weight of smoothness term (λ)
opt_grad_thresh	8	Threshold for magnitude of intensity gradient [22]
$opt_grad_penalty$	4 (Map, Tsubuka) 2 (Sawtooth, Venus)	Smoothness penalty factor if gradient is too small [22]

4.4.2 Simulation Results

The simulation results of prioritizing the α - β Swap pair sequence according to the disparity probability (the number of nodes with a specific disparity divided by the number of all nodes), E_{data} , E_{smooth} , and E_{total} for each disparity value are shown in detail in this section. In addition, we also represent the simulation results with early termination in energy minimization process (ET), separately.

Table 4-2 Comparison of computing time

Method	Computing time (sec)				Average Improvement (%)
	Map	Sawtooth	Tsukuba	Venus	
Original	83.52	156.39	93.88	131.16	
Probability	41.45	121.13	38.02	96.86	36.02
E_{data}	55.66	127.50	73.98	117.86	19.35
E_{smooth}	50.83	101.89	39.27	85.97	40.22
E_{total}	48.75	110.24	44.63	84.45	38.04
ET	38.31	65.30	75.78	71.58	46.02
Probability+ET	20.61	50.75	24.73	53.88	67.74
E_{data} +ET	30.38	60.30	38.25	81.30	54.78
E_{smooth} +ET	21.97	40.41	25.67	70.99	65.79
E_{total} +ET	21.19	39.45	24.72	69.83	66.62

Improvement is $1 - \frac{T_x}{T_{original}}$ where T_x denotes the computing time of method "X" and $T_{original}$ denotes the computing time of the original method.

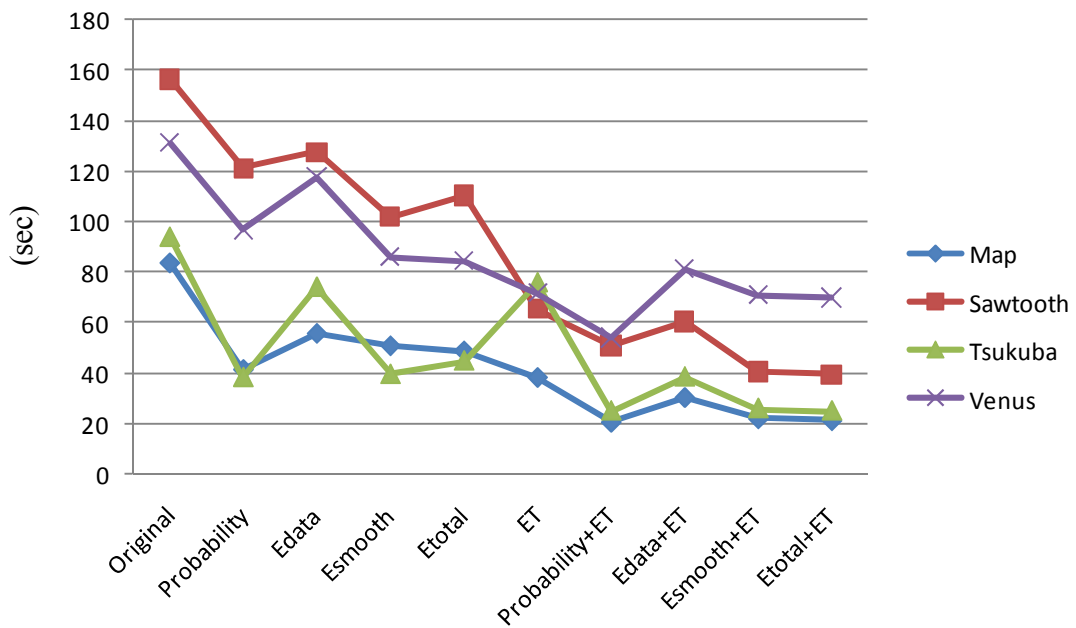


Fig. 4-6 Plot of computing time

Table 4-3 Comparison of *rms_error_all*

Method	<i>R</i>				Average ΔR (%)
	Map	Sawtooth	Tsukuba	Venus	
Original	4.10	1.48	1.30	1.48	
Probability	4.11	1.49	1.28	1.45	-0.75
E_{data}	3.85	1.49	1.28	1.45	-7.25
E_{smooth}	4.02	1.48	1.28	1.45	-3.25
E_{total}	3.85	1.47	1.28	1.49	-6.75
ET	4.05	1.49	1.30	1.39	-3.25
Probability+ET	4.12	1.49	1.28	1.45	-0.50
E_{data} +ET	3.85	1.48	1.28	1.44	-7.75
E_{smooth} +ET	4.01	1.48	1.28	1.45	-3.50
E_{total} +ET	3.85	1.43	1.28	1.49	-7.75

$\Delta R = R_X - R_{original}$ where R_X denotes the RMS error of method "X" and $R_{original}$ denotes the RMS error of the original method.

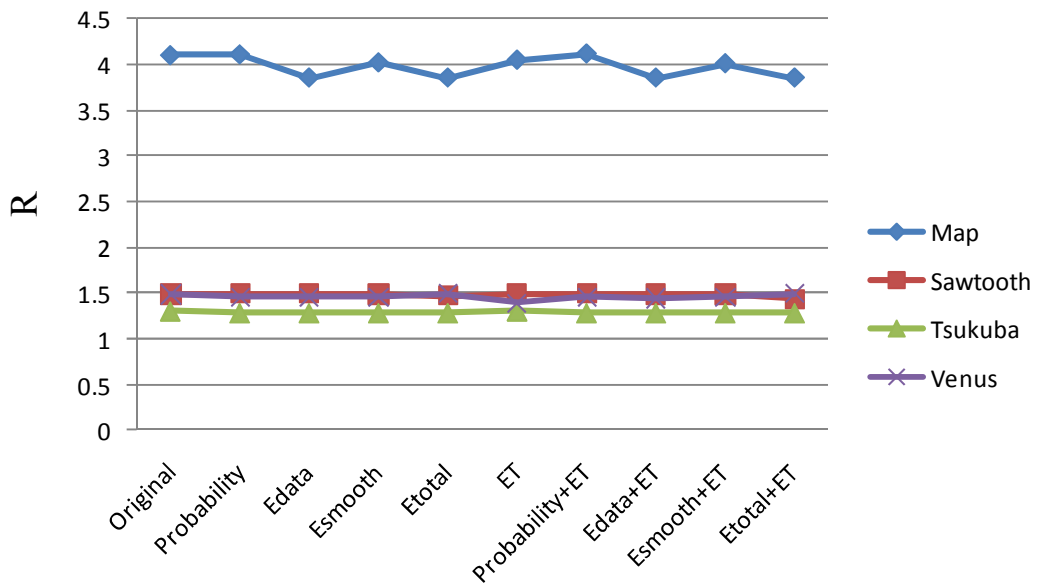


Fig. 4-7 Plot of *rms_error_all*

Table 4-4 Comparison of *bad_pixels_all*

Method	<i>B</i> (%)				Average ΔB
	Map	Sawtooth	Tsukuba	Venus	
Original	5.45	3.94	4.16	3.50	
Probability	5.53	3.93	4.20	4.50	0.28
E_{data}	5.48	4.03	4.18	3.43	0.02
E_{smooth}	5.48	3.91	4.12	3.62	0.02
E_{total}	5.40	3.98	4.31	4.41	0.26
ET	5.43	3.96	4.16	3.50	0.00
Probability+ET	5.47	3.96	4.20	4.50	0.27
E_{data} +ET	5.47	4.03	4.18	3.43	0.02
E_{smooth} +ET	5.47	3.94	4.12	3.62	0.03
E_{total} +ET	5.40	4.01	4.31	4.40	0.27

$\Delta B = B_X - B_{\text{original}}$ where B_X denotes the percentage of bad pixels of method “X” and B_{original} denotes the percentage of bad pixels of the original method.

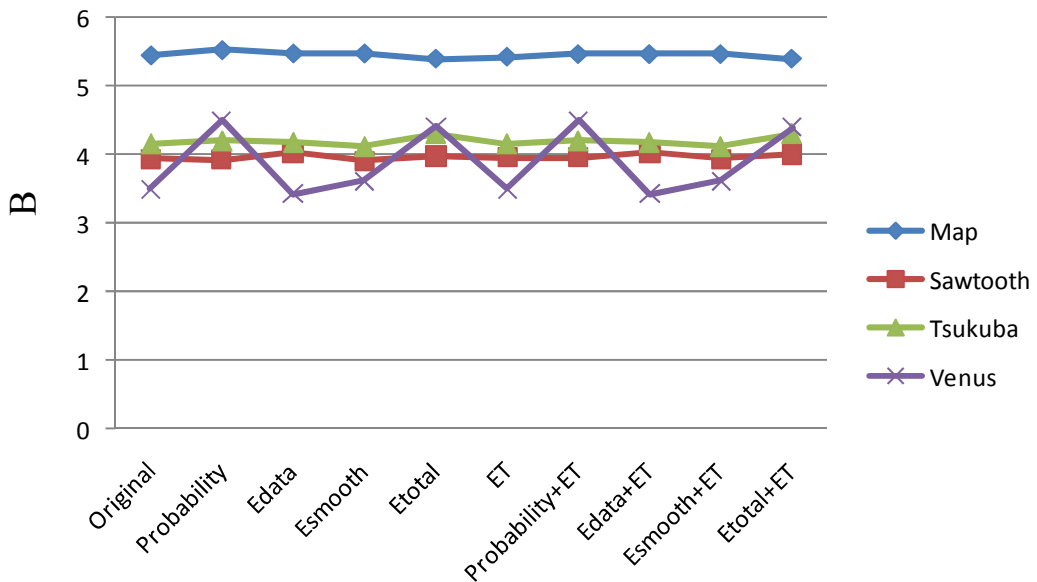


Fig. 4-8 Plot of *bad_pixels_all*

Table 4-5 Comparison of *bad_pixels_nonocc*

Method	$B_{\overline{\mathcal{O}}} (\%)$				Average $\Delta B_{\overline{\mathcal{O}}}$
	Map	Sawtooth	Tsukuba	Venus	
Original	0.38	1.34	2.00	1.87	
Probability	0.41	1.34	2.03	2.81	0.250
E_{data}	0.35	1.42	2.03	1.77	-0.005
E_{smooth}	0.38	1.34	1.96	1.97	0.015
E_{total}	0.32	1.38	2.16	2.77	0.260
ET	0.38	1.36	2.00	1.85	0.000
Probability+ET	0.40	1.36	2.04	2.82	0.258
E_{data} +ET	0.35	1.43	2.02	1.77	-0.005
E_{smooth} +ET	0.38	1.38	1.96	1.97	0.025
E_{total} +ET	0.33	1.41	2.15	2.77	0.268

$\Delta B_{\overline{\mathcal{O}}} = B_{\overline{\mathcal{O}},X} - B_{\overline{\mathcal{O}},original}$ where $B_{\overline{\mathcal{O}},X}$ denotes the percentage of non-occlusion bad pixels of method “X” and $B_{\overline{\mathcal{O}},original}$ denotes the percentage of non-occlusion bad pixels of the original method.

Table 4-6 Comparison of *bad_pixels_textureless*

Method	$B_{\overline{\mathcal{T}}} (\%)$				Average $\Delta B_{\overline{\mathcal{T}}}$
	Map	Sawtooth	Tsukuba	Venus	
Original	0.00	0.24	1.09	2.76	
Probability	0.00	0.26	1.16	5.04	0.79
E_{data}	0.00	0.26	1.15	2.63	-0.02
E_{smooth}	0.00	0.25	1.11	3.83	0.37
E_{total}	0.00	0.30	1.40	5.14	0.92
ET	0.00	0.24	1.09	2.50	-0.06
Probability+ET	0.00	0.26	1.16	5.04	0.79
E_{data} +ET	0.00	0.26	1.15	2.63	-0.02
E_{smooth} +ET	0.00	0.28	1.11	3.83	0.38
E_{total} +ET	0.00	0.35	1.39	5.14	0.93

$\Delta B_{\overline{\mathcal{T}}} = B_{\overline{\mathcal{T}},X} - B_{\overline{\mathcal{T}},original}$ where $B_{\overline{\mathcal{T}},X}$ denotes the percentage of textureless bad pixels of method “X” and $B_{\overline{\mathcal{T}},original}$ denotes the percentage of textureless bad pixels of the original method.

Table 4-7 Comparison of *bad_pixels_discont*

Method	$B_{\mathcal{D}}$ (%)				Average $\Delta B_{\mathcal{D}}$ (%)
	Map	Sawtooth	Tsukuba	Venus	
Original	3.82	6.23	9.87	7.22	
Probability	4.55	6.55	10.15	7.28	0.35
E_{data}	3.76	6.42	10.03	6.80	-0.38
E_{smooth}	3.85	6.48	9.77	6.63	-0.07
E_{total}	3.62	6.53	10.79	6.76	0.24
ET	3.85	6.28	9.87	6.62	-0.27
Probability+ET	4.49	6.54	10.15	7.28	0.46
E_{data} +ET	3.76	6.43	10.00	6.66	-0.40
E_{smooth} +ET	3.94	6.65	9.76	6.57	0.02
E_{total} +ET	3.79	6.49	10.79	6.80	0.24
$\Delta B_{\mathcal{D}} = B_{\mathcal{D},X} - B_{\mathcal{D},original}$ where $B_{\mathcal{D},X}$ denotes the percentage of non-occlusion bad pixels of method "X" and $B_{\mathcal{D},original}$ denotes the percentage of non-occlusion bad pixels of the original method.					



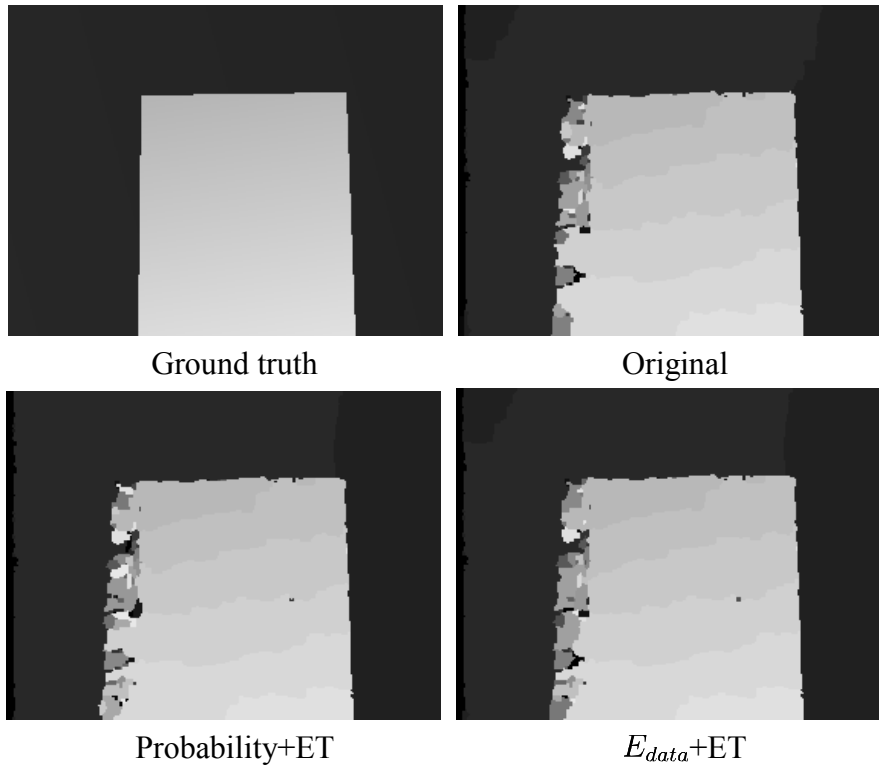


Fig. 4-9 Disparity maps of 'Map'

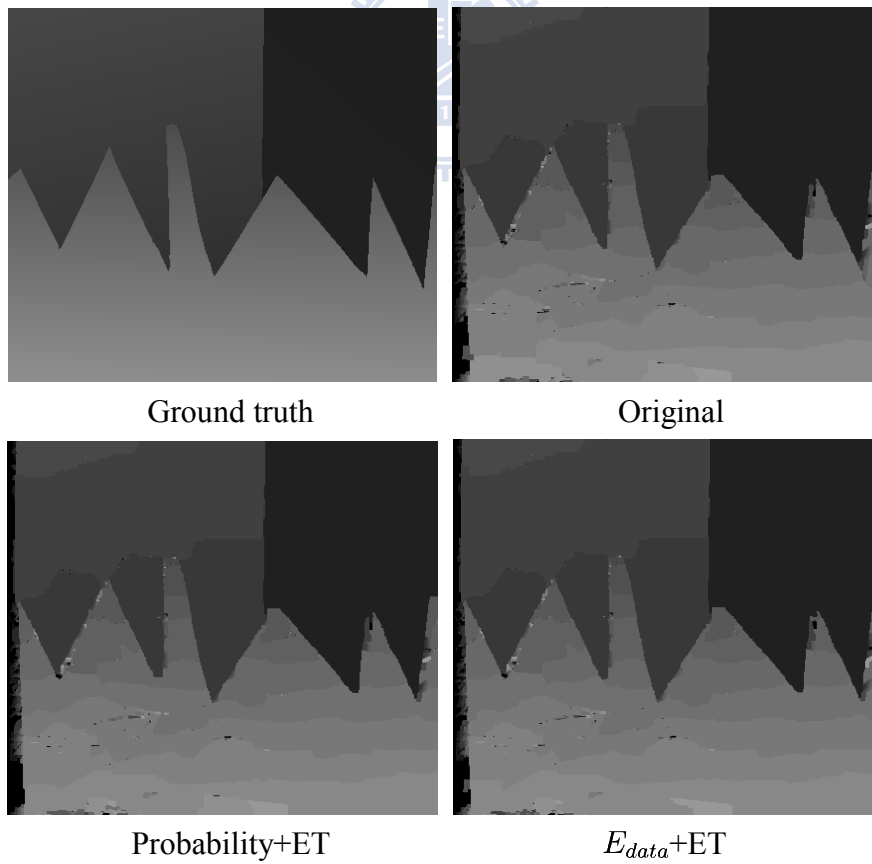


Fig. 4-10 Disparity maps of 'Sawtooth'

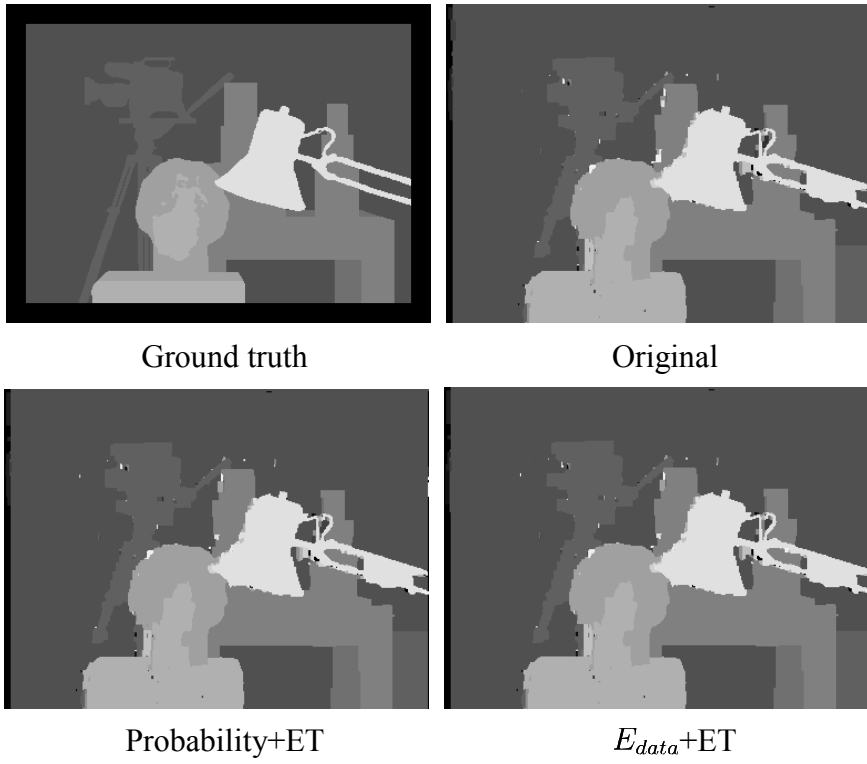


Fig. 4-11 Disparity maps of 'Tsukuba'

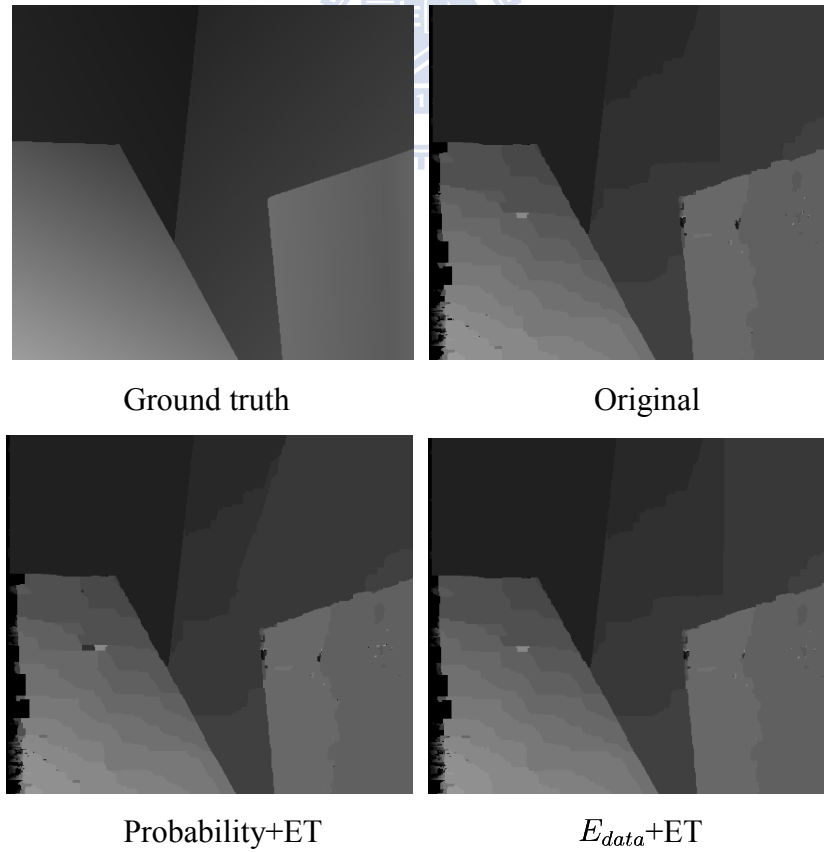


Fig. 4-12 Disparity maps of 'Venus'

4.4.3 Analysis and Discussions

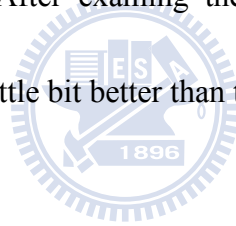
Table 4-2 and Fig. 4-6 show the computing time of the original GC and the proposed methods. Improvement is calculated from one minus their ratio. In the methods without early termination, the improvement of prioritizing the α - β swap pair according to E_{smooth} is the best. However, if the early termination is employed, prioritizing the α - β swap pair according to probability has the least computing time. In addition, we find that the improvement due to early termination only can reach 46%.

Table 4-3 and Fig. 4-7 show the rms_error_all of the original GC and the proposed methods. The improvement is calculated from their average performance on 4 image pairs. The rms_error_all slightly decreases in our methods. Prioritization according to E_{data} and E_{total} can improve rms_error_all more than others with or without the early termination. Table 4-4 and Fig. 4-8 show the bad_pixels_all of the original GC and the proposed methods. Prioritization according to probability and E_{total} are worse than the others. The degradation is dominated by the ‘Venus’. However, its performance is still about the same as that of the original GC.

Table 4-5, Table 4-6, and Table 4-7 show that different methods have different performance for different types of images. For example, prioritization according to E_{data} is the best performer in the textureless region. In conclusion, if we can tolerate

the slight degradation of quality, we choose the probability prioritized method which saves most computing time. On the other hand, if we allow a slightly higher computing time to exchange for a better quality, we can choose the E_{data} prioritized method. In addition, the E_{smooth} prioritized method is a good choice, because it decreases a lot of computing time and only slightly degrades the quality.

Fig. 4-9, Fig. 4-10, Fig. 4-11, and Fig. 4-12 show the disparity maps of the ground truth, original GC, probability+ET, and E_{data} +ET. The disparity maps generated by the proposed methods generally are very close to the disparity map produced by the original GC. After examining them closely, we find out that the disparity map of E_{data} +ET is a little bit better than that of probability+ET.



Chapter 5 Multi-Resolution Graph Cuts and Disparity Estimation for Multi-Camera Array

5.1 Overview

In disparity estimation, the graph cuts and the belief propagation algorithms provide better disparity map quality. Unfortunately, their computation time is very high. In this chapter, we use multi-resolution graph cuts to reduce the computing time. Then, we estimate the disparity maps when the multi-camera array is in use.

5.2 Disparity Estimation using Multi-Resolution Graph Cuts

In section 3.3, we describe the push-relabel algorithm which can solve the max-flow/min-cut problem. The worst-case running time for this algorithm is $O(N_V^2 N_E)$, where N_V is the number of nodes and N_E is the number of edges in the graph. Because N_V and N_E increase with the image size, the running time of the graph cuts algorithm greatly increases with the image size. In addition, the α - β swap method constructs a graph of two terminals (disparity values) at a time. If the disparity range is $0, 1, 2, \dots, n - 1$, we construct C_2^n graph in total (all combinations). Therefore, the running time also greatly increases with the disparity range when we

use the α - β swap method.

We can employ the multi-resolution graph cuts (MRGC) technique for reducing the computation time. Fig. 5-1 shows the flowchart of MRGC. We first use the image down-sampling technique to generate the low-resolution images. The right-side path in the Fig. 5-1 is same as the original GC method except the image size and the disparity range. If the disparity range of the original GC is $0, 1, 2, \dots, n - 1$ in the original resolution, the disparity range is $0, 1, 2, \dots, \frac{n}{2}$ in the low-resolution image. After the low-resolution disparity map is obtained, we come back to the original resolution image size. We up-sample the disparity map. Then, it becomes the initial disparity map for the neighborhood graph cuts.

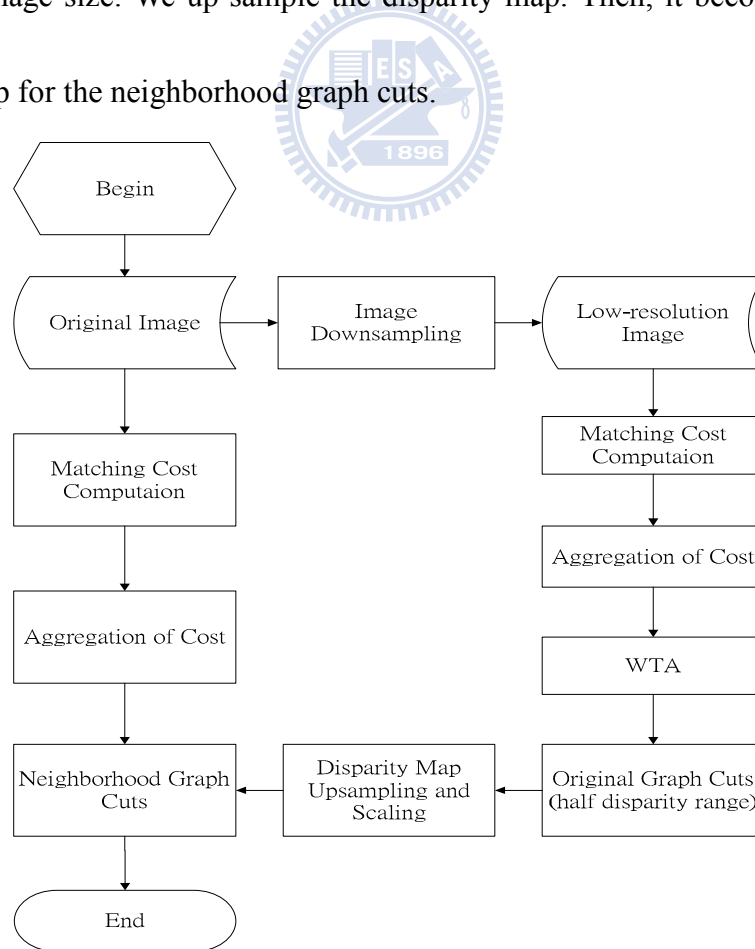
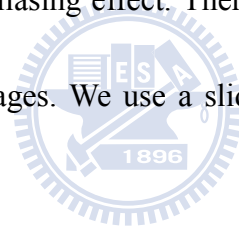


Fig. 5-1 Flowchart of MRGC

The main components that are different from the original GC method are the image down-sampling, disparity map up-sampling and scaling, and neighborhood graph cuts operations. We describe the detail in the following sub-sections.

5.2.1 Image Down-Sampling

In this section, we describe two down-sampling methods that we use in MRGC. Fig. 5-2 shows an example of a simple down-sampling method. The sampling factor is 2 for the width and the height. This method simply skips every other pixel in one-dimension. Because of lacking prefitting, the low-resolution images after the down-sampling may suffer the aliasing effect. Therefore, we attempt another method to down-sample the original images. We use a sliding window whose coefficient is depicted below for pre-fitting.



$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{3 \times 3} \quad (5.1)$$

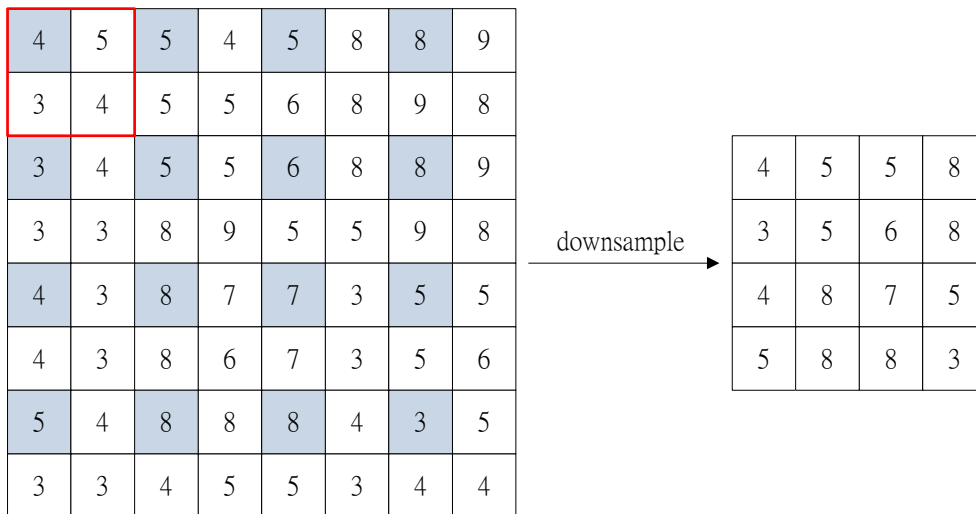


Fig. 5-2 An example of 4 to 1 pixel-skip down-sampling

Because the coefficients of the window are all integer and the sum of the coefficients is 16, we can shift right four bits in calculation instead of using division. This filter does not increase much computing time comparing to the simplest method. We will compare the two methods in section 5.4 by simulation.

5.2.2 Disparity Map Up-sampling and Scaling

In this section, we also describe two methods to up-sample the low-resolution disparity map. Fig. 5-3 shows a simple up-sampling method that duplicates the pixel value to its neighbors directly and multiplies the disparity value by 2. Thus, the derived disparity map becomes the initial disparity map for the neighborhood graph cuts. Because the disparity maps produced by the simple method may produce blocky images, we can reduce artifacts by performing some types of linear or bilinear interpolation in up-sampling. However, the interpolation process increases the computation time and its quality improvement on the disparity map is uncertain.

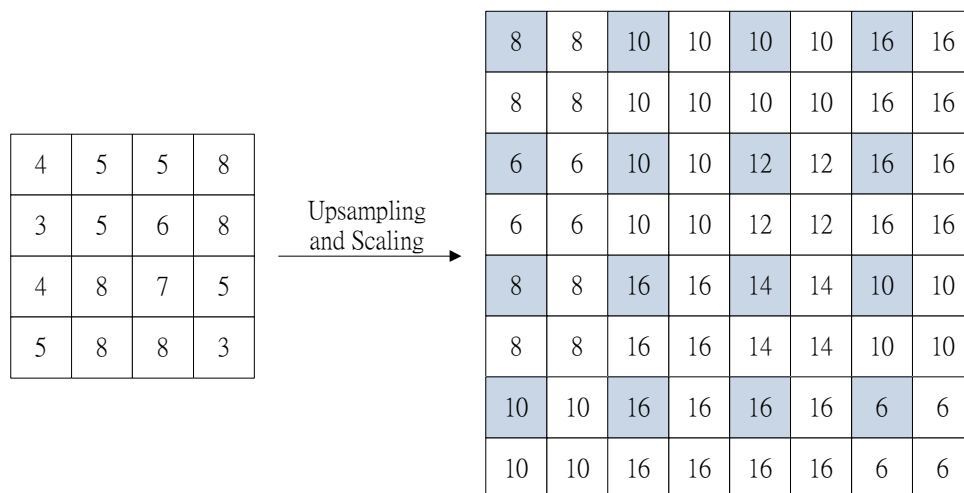


Fig. 5-3 An example of disparity map up-sampling and scaling

We attempt to employ the up-sampling method of H.264 [23], which is shown in Fig. 5-4. The pixel i can be obtained from pixels a , b , c , d , e , and f by the formula below.

$$i = \text{round}((1 \times a - 5 \times b + 20 \times c + 20 \times d - 5 \times e + 1 \times f) \gg 5) \quad (5.2)$$

We can use pixels g , h , i , j , k , and l , to interpolate the pixel m similarly.

$$m = \text{round}((1 \times g - 5 \times h + 20 \times i + 20 \times j - 5 \times k + 1 \times l) \gg 5) \quad (5.3)$$

The coefficients of the interpolation filter are $(\frac{1}{32}, \frac{-5}{32}, \frac{5}{8}, \frac{5}{8}, \frac{-5}{32}, \frac{1}{32})$, which mimic the sinc function. After the up-sampling interpolation process, we multiply the disparity values by 2. In section 5.4, we will compare the two methods based on the simulation results.

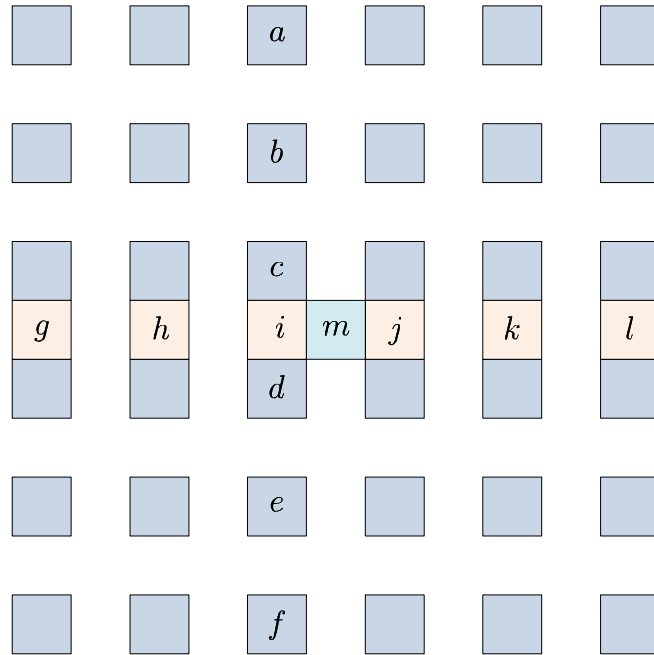
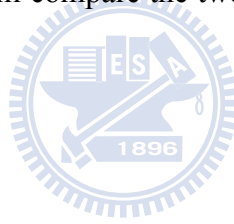
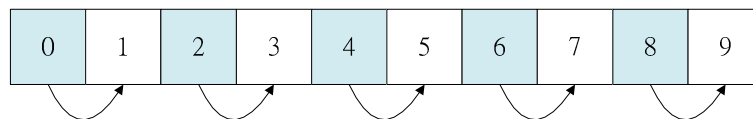


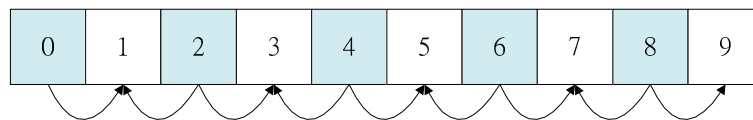
Fig. 5-4 The up-sampling method of H.264

5.2.3 Neighborhood Graph Cuts

In the original graph cuts, the number of graphs needed for constructing α - β swap is the total combinations of disparity pairs selected from the disparity range. The neighborhood graph cuts method reduces the number of constructing graph. Unlike the original graph cuts, we use the disparity map obtained from the up-sampling and scaling process as the initial disparity map. We assume that the disparity value f_p of each pixel only differ to its neighborhood disparity values by 1. Therefore, we try to reduce the number of combinations of disparity pairs in α - β swap to reduce the computing time. Fig. 5-5 shows the disparity pair combination of neighborhood graph cuts. The gray nodes are the disparity values obtained from the scaling. The arrow shows the value that the disparity value can change to. That is, we select two disparity values to do the α - β swap.



(a)



(b)

Fig. 5-5 The disparity pair candidates in neighborhood graph cuts (a)+1 (b) \pm 1

Here, we depict two cases of the neighborhood graph cuts. In Fig. 5-5(a), the search

range of neighborhood graph cuts is ± 1 . In Fig. 5-5(b), the search range is ± 1 and their combination is nearly two times more than ± 1 . Note that we cannot do α - β swap with the same value. In section 5.4, we will compare the performance of these two methods and show the computing time saved by using MRGC.

5.3 Disparity Estimation in Multi-Camera Array

In this section, we propose a method for disparity estimation for multi-camera picture. We pick up “sawtooth” and “venus” as our test data for the multi-camera experiment.. The two test data sets both include nine images captured by nine cameras (Fig. 5-6). We call them $im_0, im_1, \dots,$ and im_8 . These images are captured by $cam_0, cam_1, \dots,$ and cam_8 , respectively.

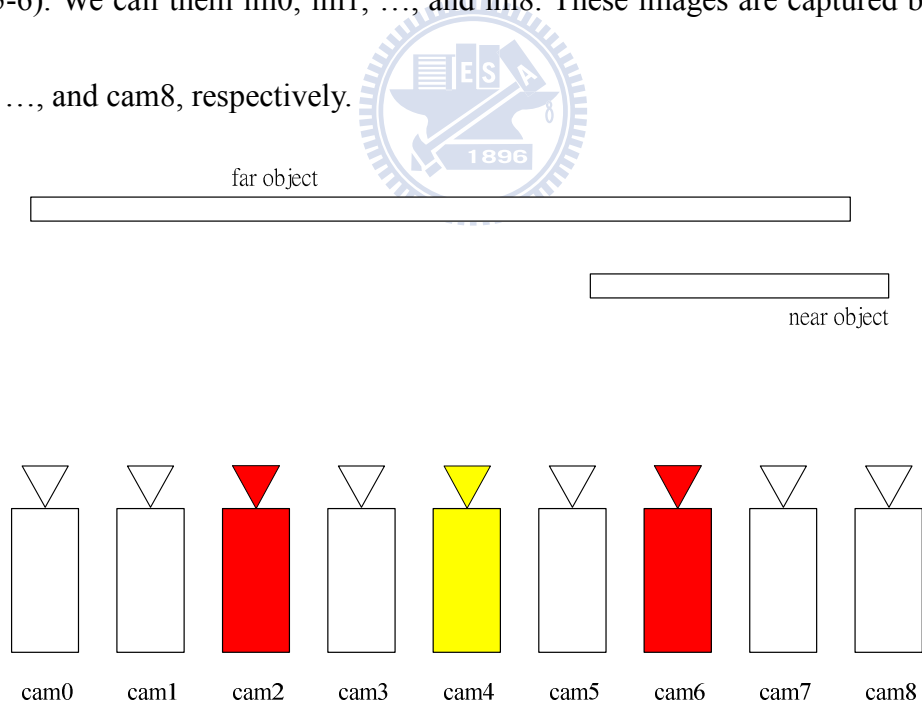


Fig. 5-6 Multi-camera array

In the original method, we compute im_2 's disparity map which is relative to im_6 . Now, im_4 is added into the proposed method. Fig. 5-7 shows the flowchart of our GC

algorithm for multi-camera pictures. First, the im4's disparity map relative to im6 is computed by using the graph cuts algorithm. We use this to predict the disparity map of im2 relative to im6, because the optical geometry tell us that the im2's disparity map relative to im6 is a shifted and scaled version of the im4's disparity map relative to im6. The results of scaling and shifting are shown in Fig. 5-8. This predicted disparity map is need as the initial disparity map and refine it by the graph cuts algorithm. In section 5.4, we will show the simulation results. The improvement of this method is not significant.

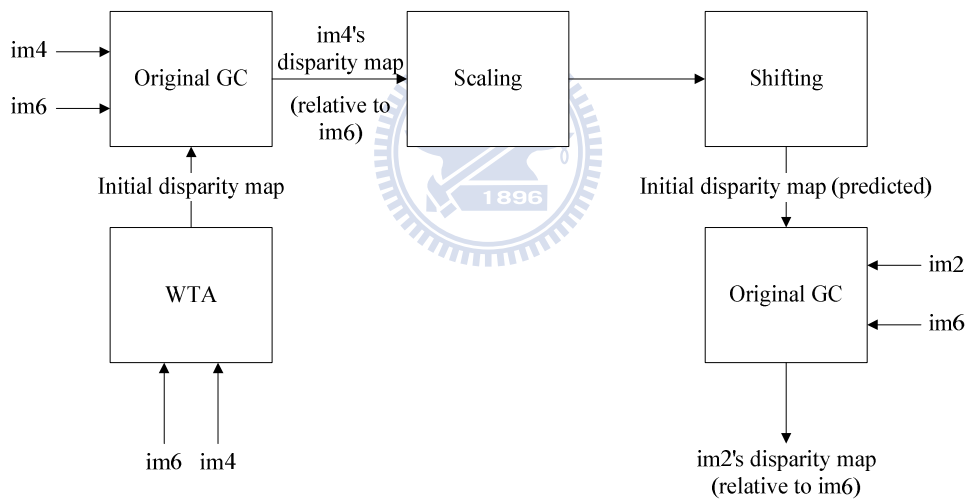


Fig. 5-7 Flowchart of GC for multi-camera pictures

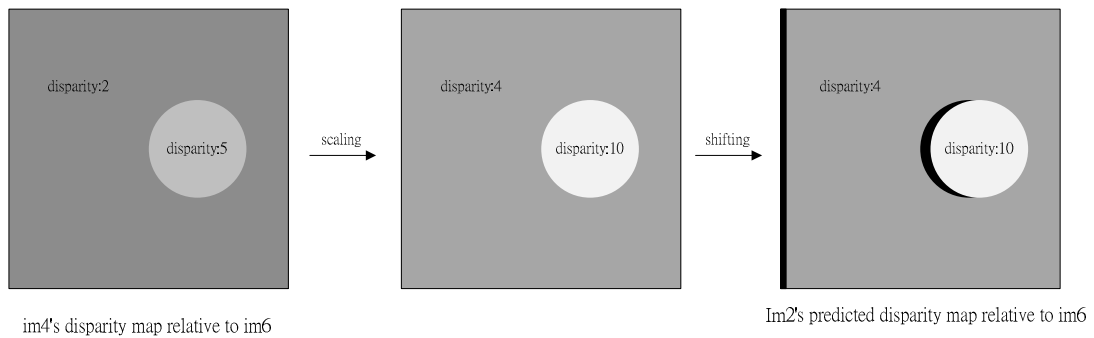


Fig. 5-8 The scaling and shifting moves

5.4 Simulation Results and Discussions

In this section, we will show the simulation results of MRGC and multi-camera pictures. Thus, we will discuss the possible causes leading to the simulation results. The experiment environment setting is the same as section 4.4. In MRGC, the scaling factor is 2.

5.4.1 Multi-Resolution Graph Cuts

We first explain the symbols to appear in the following tables. MRGC(+1) indicates the multi-resolution graph cuts whose search range of neighborhood graph cuts (NGC) is +1. Similarly, the search range of NGC of MRGC(± 1) is ± 1 . In MRGCD, the down-sampling method indicates the low-pass filter describe in formula (5.1). Likewise, MRGCU denotes that its up-sampling method uses the H.264 up-sampling filter. In MRGCDU, both down-sampling and up-sampling processes adopt the before mentioned filters.

Table 5-1 shows the computing time improvement by MRGC ranges from 81% to 92%. MRGC(± 1) runs a little longer than MRGC(+1), because the search range is wider. Table 5-2 and Table 5-3 show the image quality comparison of different methods. Although the computation time of MRGC(± 1) is slightly larger than MRGC(+1), its quality is much better. In addition, the down-sampling method of (5.1) is better than the simple sample-skip method. However, if we replace the simple

disparity map duplication method by the H.264 interpolation the method, the computing time gets higher and the *bad_pixels_all* becomes worse. This may due to the fact that the discontinuous region is critical for initial disparity map, and the H.264 up-sampling method blurs the initial disparity map.

Fig. 5-12, Fig. 5-13, Fig. 5-14, and Fig. 5-15 show the disparity maps of different methods. Obviously, the disparity map looks much smoother by down-sample the original image by the formula (5.1). In addition, the disparity map of MRGC(± 1) is much better than MRGC(+1). The MRGC(± 1) produces a disparity may close to the original GC method.

Hierarchical graph cuts [24] is one of the few fast graph cuts algorithms found in the literature. According to [24], the computing timing of the hierarchical graph cuts is about 25% of the original GC on the test image “Tsukuba”. However, MRGC is faster than the hierarchical graph cuts. Our method takes about 16% of the computing time of the original GC. In addition, the quality of the hierarchical graph cuts is not discussed in the paper. We are not sure about the quality degradation of this method.

Table 5-1 Computing time comparison

Method	Computing time (sec)				Average Improvement (%)
	Map	Sawtooth	Tsukuba	Venus	
Original	83.52	156.39	93.88	131.16	
MRGC(+1)	5.27	10.28	6.83	14.52	92.06
MRGC(±1)	7.84	23.45	18.78	27.14	83.39
MRGCD(±1)	6.98	20.61	18.09	25.16	84.76
MRGCU(±1)	13.97	25.44	25.20	30.75	79.49
MRGCDU(±1)	13.13	24.81	22.20	29.61	80.70

Improvement is $1 - \frac{T_x}{T_{original}}$ where T_x denotes the computing time of method “X” and $T_{original}$ denotes the computing time of the original method.

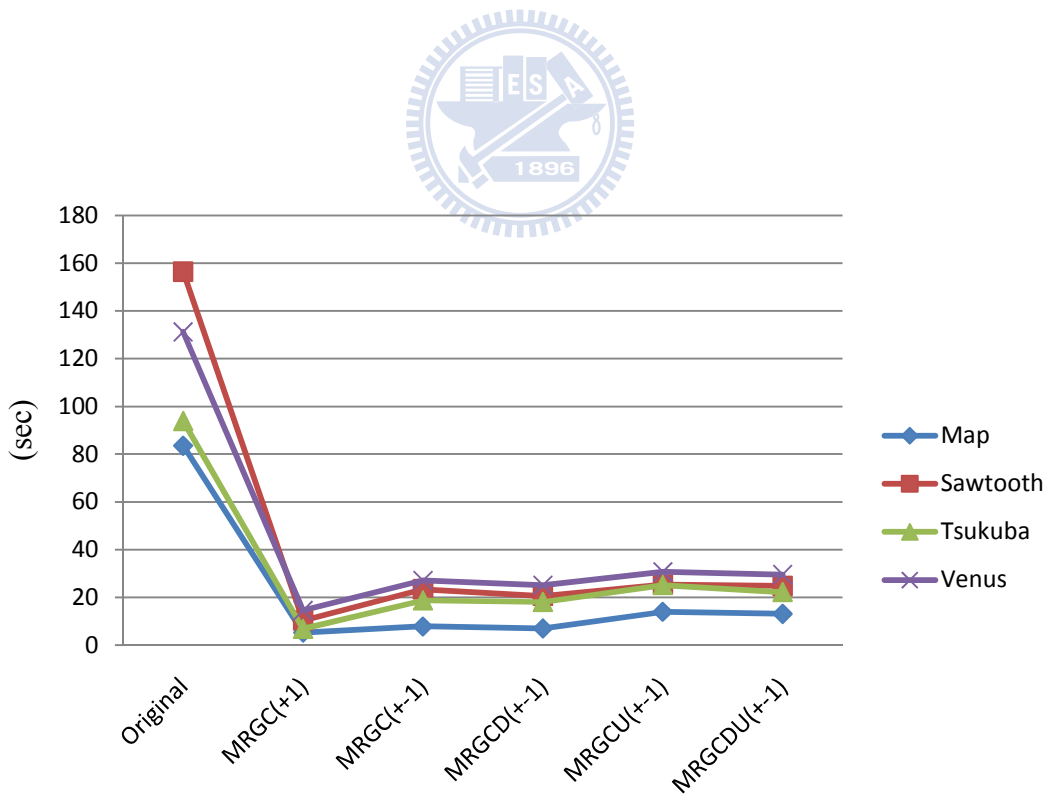


Fig. 5-9 Plot of computing time comparison

Table 5-2 Comparison of *rms_error_all*

Method	<i>R</i>				Average ΔR (%)
	Map	Sawtooth	Tsukuba	Venus	
Original	4.10	1.48	1.30	1.48	
MRGC(+1)	4.31	1.69	1.60	1.63	21.75
MRGC(± 1)	4.19	1.62	1.53	1.54	13.00
MRGCD(± 1)	4.24	1.49	1.30	1.63	7.50
MRGCU(± 1)	4.17	1.59	1.54	1.51	11.25
MRGCDU(± 1)	4.28	1.46	1.26	1.60	6.00

$\Delta R = R_X - R_{\text{original}}$ where R_X denotes the RMS error of method "X" and R_{original} denotes the RMS error of the original method.

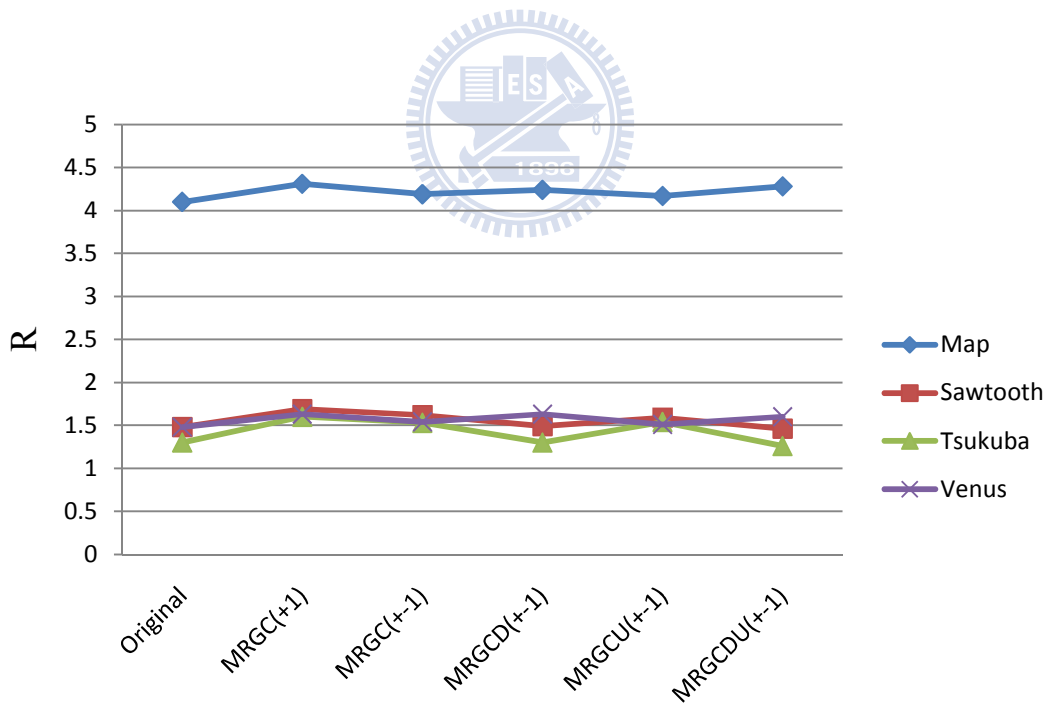


Fig. 5-10 Plot of *rms_error_all*

Table 5-3 Comparison of *bad_pixels_all*

Method	<i>B</i> (%)				Average ΔB
	Map	Sawtooth	Tsukuba	Venus	
Original	5.45	3.94	4.16	3.50	
MRGC(+1)	8.38	10.09	7.75	7.15	4.08
MRGC(± 1)	6.16	4.87	7.07	5.22	1.57
MRGCD(± 1)	5.88	4.08	6.36	5.26	1.13
MRGCU(± 1)	7.32	5.42	7.50	5.77	2.24
MRGCDU(± 1)	7.31	4.22	5.94	4.69	1.28

$\Delta B = B_X - B_{\text{original}}$ where B_X denotes the percentage of bad pixels of method "X" and B_{original} denotes the percentage of bad pixels of the original method.

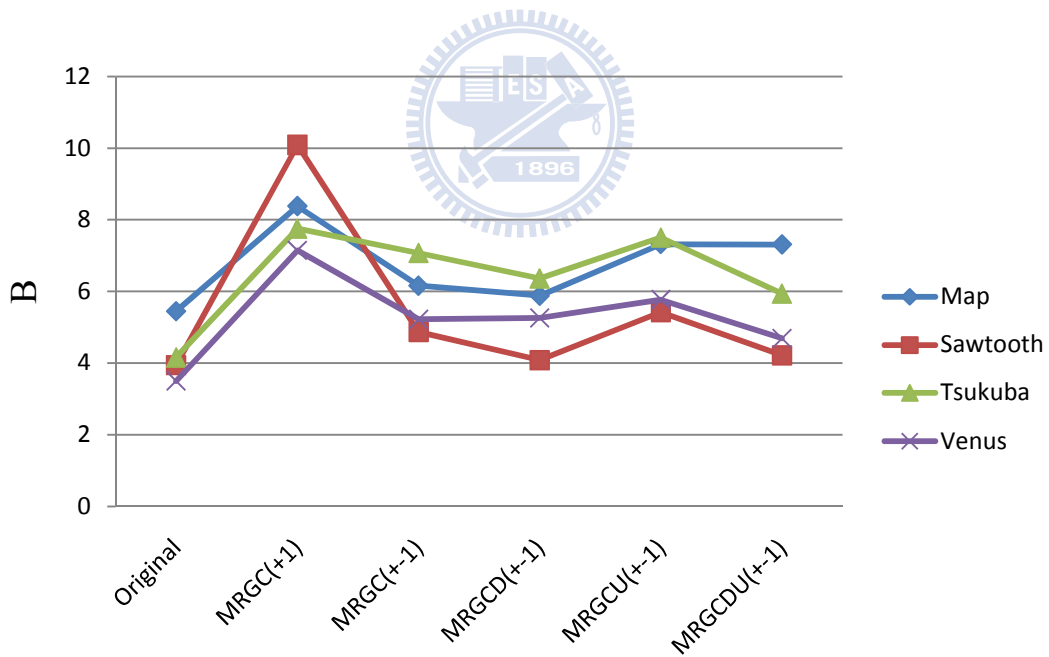
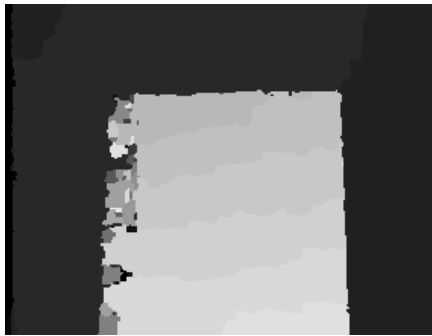


Fig. 5-11 Plot of *bad_pixels_all*



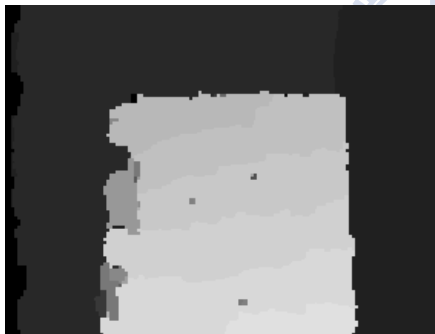
Ground truth



Original



MRGC(+1)



MRGC(±1)



MRGCD(±1)



MRGCU(±1)

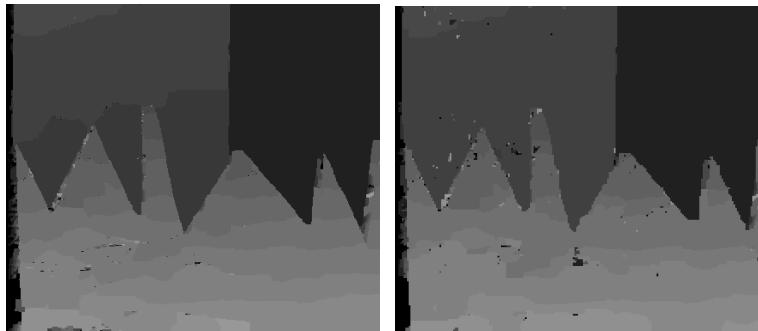


MRGCDU(±1)

Fig. 5-12 Disparity maps of 'Map'

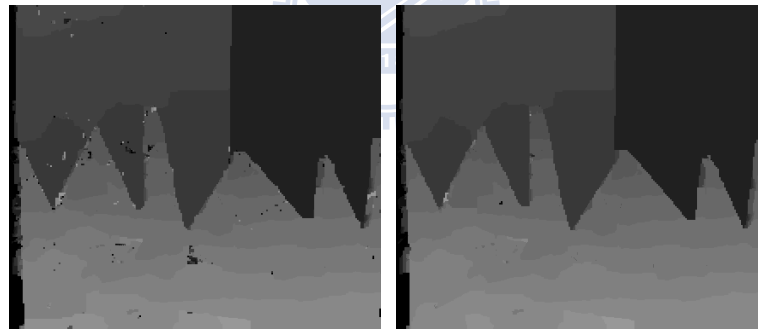


Ground truth



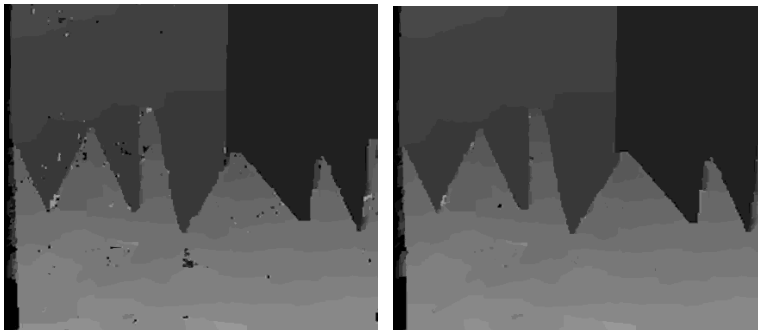
Original

MRGC(+1)



MRGC(±1)

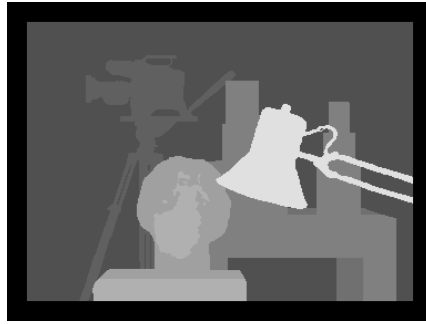
MRGCD(±1)



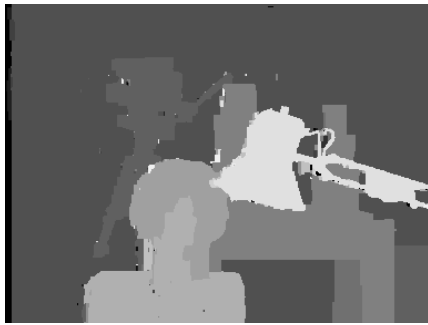
MRGCU(±1)

MRGCDU(±1)

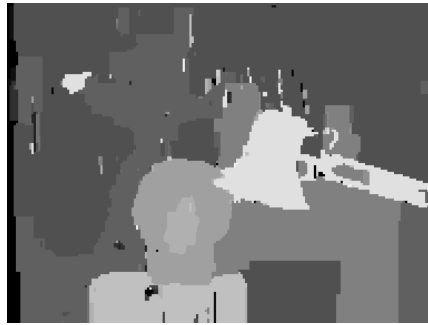
Fig. 5-13 Disparity maps of 'Sawtooth'



Ground truth



Original



MRGC(+1)



MRGC(±1)



MRGCD(±1)



MRGCU(±1)

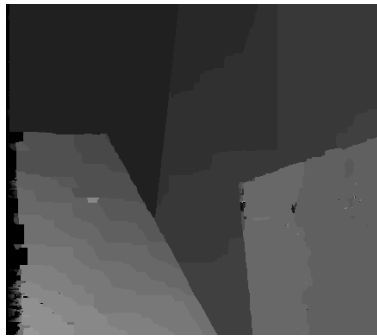


MRGCDU(±1)

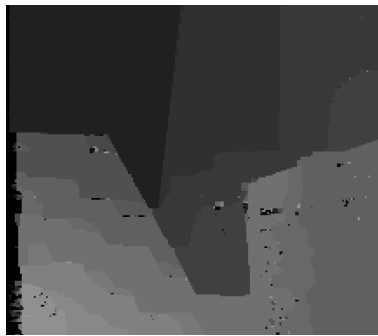
Fig. 5-14 Disparity maps of ‘Tsukuba’



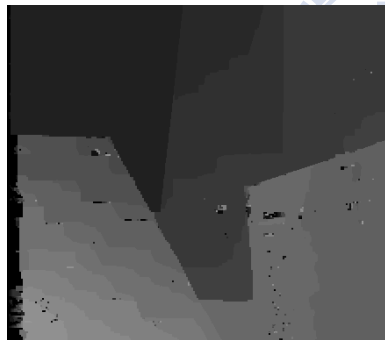
Ground truth



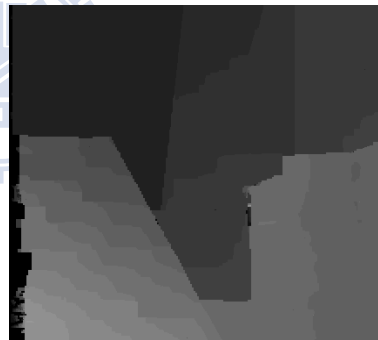
Original



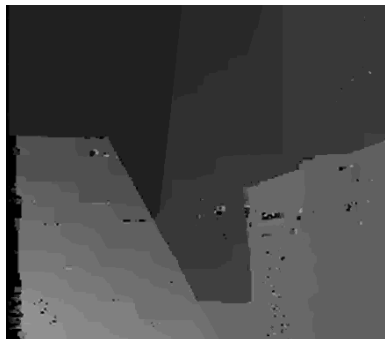
MRGC(+1)



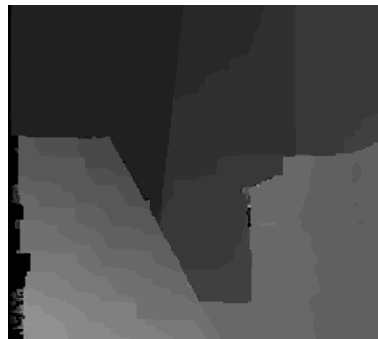
MRGC(±1)



MRGCD(±1)



MRGCU(±1)



MRGCDU(±1)

Fig. 5-15 Disparity maps of ‘Venus’

5.4.2 Disparity Estimation in Multi-Camera Array

Table 5-4 and Table 5-5 show that the disparity estimation using our proposed multi-camera scheme does method cannot improve the quality of disparity maps. Fig. 5-16 and Fig. 5-17 show the change of disparity map. The predicted disparity map computed by scaling and shifting the im4's disparity map relative to im6. We run the graph cuts algorithm by using the predicted disparity map to be the initial disparity map.

Table 5-4 Comparison of *rms_error_all*

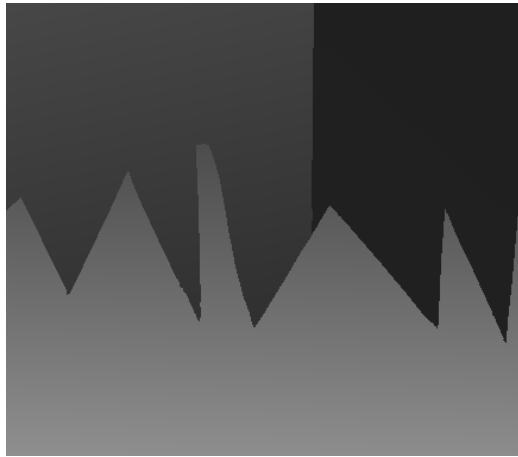
Method	R		Average ΔR (%)
	Sawtooth	Venus	
Original	1.48	1.48	
Multi-cam	1.48	1.49	0.5

$\Delta R = R_X - R_{\text{original}}$ where R_X denotes the RMS error of method "X" and R_{original} denotes the RMS error of the original method.

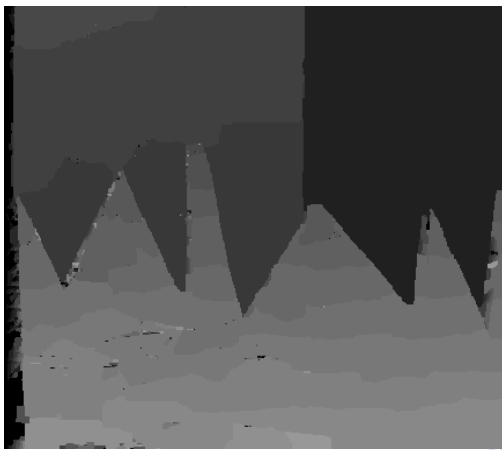
Table 5-5 Comparison of *bad_pixels_all*

Method	B (%)		Average ΔB
	Sawtooth	Venus	
Original	3.94	3.50	
Multi-cam	3.91	3.68	0.08

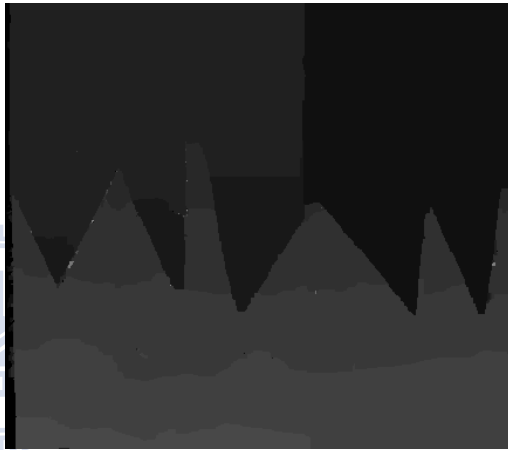
$\Delta B = B_X - B_{\text{original}}$ where B_X denotes the percentage of bad pixels of method "X" and B_{original} denotes the percentage of bad pixels of the original method.



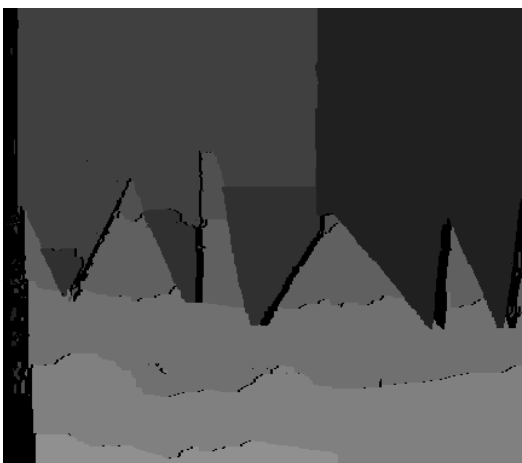
Ground truth



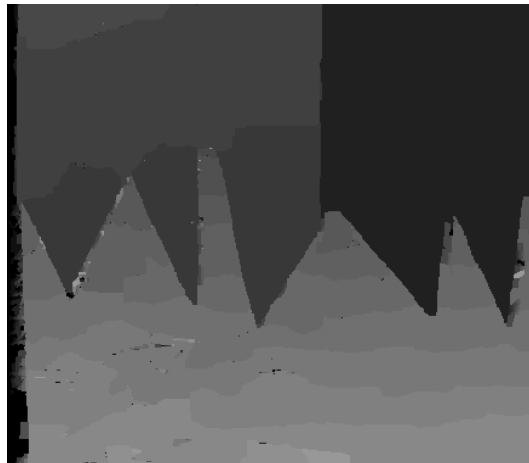
Original



im4's disparity map relative to im6



predicted by scaling and shifting

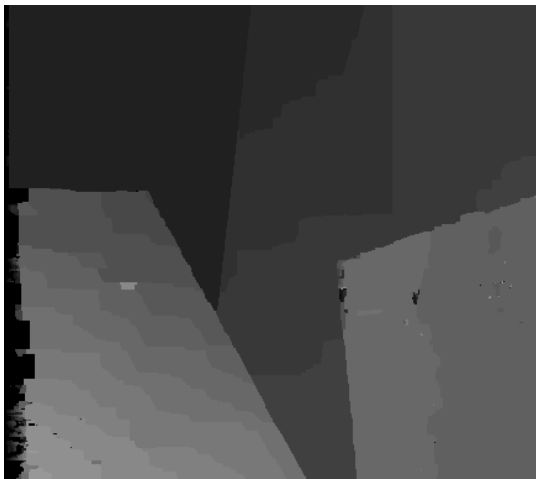


im2's disparity map relative to im6

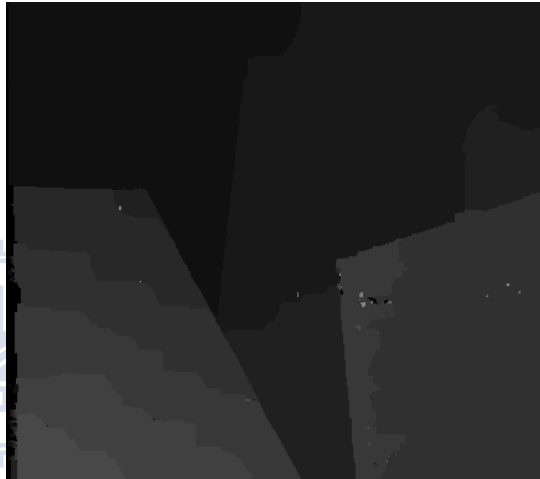
Fig. 5-16 Disparity maps of "Sawtooth"



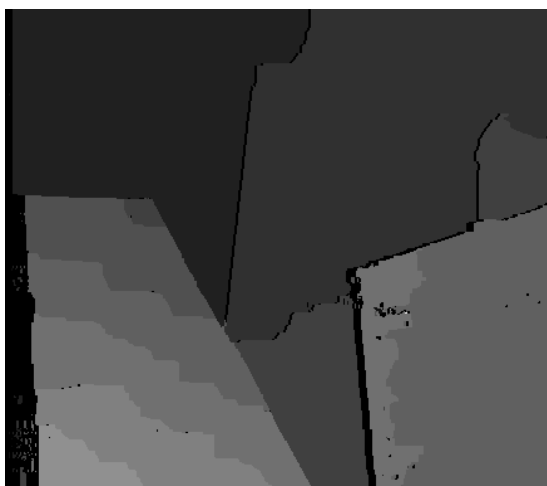
Ground truth



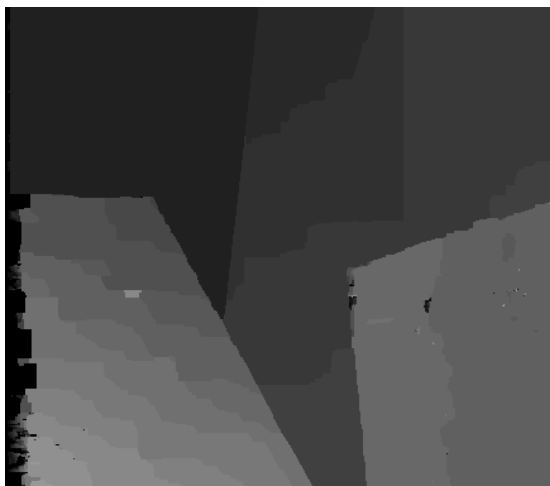
Original



im4's disparity map relative to im6



predicted by scaling and shifting



im2's disparity map relative to im6

Fig. 5-17 Disparity maps of "Venus"

Chapter 6 Conclusions and Future Work

6.1 Conclusions

The Graph Cut (GC) algorithm is an effective disparity estimation algorithm. Yet, it consumes a huge amount of computations due to its high complexity. The original GC scheme randomizes the α - β swap pairing in the inner iteration loop and terminates the iteration in outer loop when no further energy reduction is possible. Observing the energy minimization process of GC, we propose two techniques to speed up GC. One is the inclusion of an early termination mechanism in the outer iteration loop, and the other is prioritizing the α - β swap pair search order in the inner iteration loop. Simulation results show that our proposed fast GC can achieve up to 68% speed-up (reduce 68% computing time) in computation while it preserves the high accuracy of disparity map as measured by the RMS disparity error and the bad pixels probability.

The worst case running time for the GC algorithm we use is $O(N_V^2 N_E)$, where N_V is the number of nodes and N_E is the number of edges. The running time greatly increases with the image size and disparity range. We propose a multi-resolution graph cuts (MRGC) to reduce the computing time, but it slightly decreases the quality of disparity map. Simulation results show that the MRGC method may achieve up to 84% speed-up and increases 1% of bad pixel ratio. In addition, we attempt to improve

the quality of disparity map by using the multi-camera picture. However, the simulation results show that simple method has no contribution on disparity estimation the improvement.

6.2 Future Work

This thesis concentrates on reducing the computing time of GC algorithms, so that real-time application and multi-camera application become possible. The other reason we study the GC algorithm is its good performance. Potentially, we can further improve the DE quality of the occlusion regions by modifying the energy function designed for multi-camera image, since we have much more information in hands. Furthermore, the 2D camera array that takes pictures of objects from different angles (both horizontally and vertically) may help in both disparity estimation and new view synthesis. This topic can be explored in the future.

Bibliography

- [1] M. Tanimoto and M. Wildeboer, “Frameworks for FTV coding,” *Proc. Picture Coding Symposium*, pp. 1–4, 2009.
- [2] M. Tanimoto, T. Fuji, and K. Suzuki, “Data format for FTV,” *ISO/IEC JTC1/SC29/WG11 M16093*, Feb. 2009.
- [3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, Mar. 2004.
- [4] StereoMatcher software. [Online] Available: <http://vision.middlebury.edu/stereo/>.
- [5] 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, Apr. 2002.
- [6] M.Z. Brown *et al.*, “Advances in computational stereo,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 993–1008, Aug. 2003.
- [7] Tappen, M.F. and Freeman, W.T., “Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters,” *IEEE International Conference on Computer Vision*, 2003.
- [8] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*,

vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

- [9] S. Birchfield and C. Tomasi, “Multiway cut for stereo and motion with slanted surfaces,” *Proc. Int’l Conf. Computer Vision*, pp. 489–495, 1999.
- [10] V. Kolmogorov and R. Zabih, “What energy functions can be minimized via graph cuts?,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, Feb. 2004.
- [11] 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, no. 9, pp. 1124–1137, Sep. 2004.
- [12] S. Birchfield and C. Tomasi, “Depth discontinuities by pixel-to-pixel stereo,” *Proc. Int’l Conf. Computer Vision*, pp. 1073–1080, 1998.
- [13] S. Birchfield and C. Tomasi, “A pixel dissimilarity measure that is insensitive to image sampling,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401–406, 1998.
- [14] P. J. Burt and E. H. Adelson, “The Laplacian pyramid as a compact image code,” *IEEE Transactions on Communications*, vol. COM-31, no. 4, pp. 532–540, 1983.
- [15] A. F. Bobick and S. S. Intille, “Large occlusion stereo,” *International Journal of Computer Vision*, vol. 33, no. 3, pp. 181–200, 1999.

- [16] G. Van Meerbergen, M. Vergauwen, M. Pollefeys and L. Van Gool, “A hierarchical symmetric stereo algorithm using dynamic programming,” *International Journal of Computer Vision*, vol. 47, no. 1-3, pp 275–285, 2002.
- [17] S. Roy, “Stereo without epipolar lines: A maximum-flow formulation,” *International Journal of Computer Vision*, vol. 34, no. 2-3, pp. 147–162, Aug. 1999.
- [18] T. Cormen, C. Leiserson, R. Rivest and C. Stein, “Maximum Flow,” Chapter 26 of *Introduction to algorithms*, 2nd edition, pp. 643-698, McGraw-Hill, 2005
- [19] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.
- [20] A. V. Goldberg and R. E. Tarjan, “A new approach to the maximum-flow problem,” *Journal of the Association for Computing Machinery*, vol. 35, no. 4, pp. 921–940, Oct. 1988.
- [21] G. Egnal, “Mutual information as a stereo correspondence measure,” *Tech. Rep. MS-CIS-00-20*, University of Pennsylvania, 2000.
- [22] O. Veksler. *Efficient Graph-based Energy Minimization Methods in Computer Vision*. Ph.D. dissertation, Cornell University, 1999.
- [23] I.E.G. Richardson, *H.264 and MPEG-4 video compression: video coding for next-generation*, Wiley, 2003.
- [24] S.B. Kang, R. Szeliski, and J. Chai. “Handling occlusions in dense multi-view

stereo,” *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

Expanded version available as MSR-TR-2001-80.



自傳

周正偉，1986 年 6 月 27 日出生於台北市。2008 年畢業於國立成功大學電機工程學系，之後進去國立交通大學電子研究所攻讀碩士學位，研究方向是視差估算，論文題目為「使用於立體視差估算之快速圖形切割演算法」。

