

國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

即時的區域性立體視覺比對演算法分析與設計

Analysis and Design of Real-Time Local Stereo Matching

研究生：蔡宗憲

指導教授：張添烜

中華民國 九十七年 九月

即時的區域性立體視覺比對演算法分析與設計

Analysis and Design of Real-Time Local Stereo Matching

研究生: 蔡宗憲

Student: Tsung-Hsien Tsai

指導教授: 張添烜 博士

Advisor: Tian-Sheuan Chang

國立交通大學
電子工程學系 電子研究所碩士班
碩士論文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of Requirements

for the Degree of

Master of Science

In

Electrical Engineering

September 2008

Hsinchu, Taiwan, Republic of China

中華民國 九十七年 九月

即時的區域性立體視覺比對演算法分析與設計

研究生：蔡宗憲

指導教授：張添烜博士

國立交通大學

電子工程學系 電子研究所

摘要

立體視覺廣泛的運用在許多領域，例如自走機器人、自動追蹤的攝影機、甚至於立體電視。由於許多的應用需要即時的立體視覺系統，因此需要設計一個能滿足高運算以及高頻寬的積體電路。

本篇研究提出了一個適合硬體設計的演算法，係基於適應性權重的計算 (Adaptive Weight Generation) 演算法結合微型普查 (Mini-Census) 的比對方式、兩次聚合 (Two-Pass Aggregation) 以及量子化指數曼哈頓距離 (Quantized Manhattan Color Distance) 等技巧。微型普查可以減少運算量，從原來的一個視窗的運算變成只有六個點運算。除此之外，他還加強了原本演算法中對於光線所造成的問題。兩階段資料匯集和量子化指數曼哈頓距離分別減少了 88.7% 和 64.2% 的運算複雜度。相較於原本的權重產生函式，量子化指數曼哈頓距離可以被實現成查表的硬體電路。

最後在聯華電子 90 奈米製程下，提出的設計可以在 100MHz 的工作時脈下達到每秒計算 43 張 CIF 畫面大小及 64 個階層的深度估測。晶片總共需要 562,642 個邏輯閘，以及 21.3K 的晶片記憶體。

Analysis and Design of Real-Time Local Stereo Matching

Student: Tsung-Hsien Tsai

Advisor: Dr. Tian-Sheuan Chang

Department of Electronics Engineering & Institute of Electronics

National Chiao Tung University

Abstract

Stereo matching has been widely used in many fields, such as automatic robots, auto-tracking system, and even the 3D-TV. With these real time application demands, VLSI implementation becomes necessary to fulfill the high complexity and high bandwidth requirements of stereo matching algorithms.

In this thesis, we propose a hardware friendly algorithm, based on adaptive support weight (ADSW), with mini-census, two-pass aggregation, and quantized exponential Manhattan distance techniques. The mini-census reduces the computation complexity from a matching block to only 6 points. Besides, it also improves the capability of ADSW to deal with the radiometric problem. The two-pass aggregation and the quantized Manhattan color distance reduce about 88.7% and 64.2% computation of the cost aggregation respectively. Comparing to the original weight generation function, the quantized Manhattan color distance can be easily implemented by a table based circuit.

The final design implemented by UMC 90nm CMOS technology can achieve 43 frames per second and 64 disparities with CIF image size under 100MHz clock rate. The chip consumes totally 562,642 K gate counts and 21.3K Bytes internal memory.

誌 謝

首先，要感謝我的指導教授—張添烜博士，這兩年來給我的支持和鼓勵，研究方面讓我能想法上能自由發揮，每當遇到困難的時候都能給予適當指導與足夠的資源來解決問題。老師不僅是研究上的良師也是生活上的益友，不僅了解學生的想法也協助學生處理生活上的各種問題。

同時也要感謝我的口試委員們，交大電子王聖智教授和清華電機陳永昌教授，感謝教授們百忙之中抽空來指導我，各位的寶貴意見讓本論文更加完備。

感謝 VSP 實驗室的好伙伴們，特別要謝謝引我入門的張彥中學長，帶領我從零開始，用嚴謹的態度逐步去解決問題，給予我不少中肯有用的建議並協助我在論文方面的寫作。感謝張彥中學長、林佑昆學長，你們傳給我的經驗與知識，讓我受用不盡。感謝李得璋、郭子筠、林嘉俊和吳私璟學長給予我許多 IC 設計的經驗以及研究的建議，也感謝廖英澤學長，在許多採購事物上的經驗傳承。感謝曾宇晟同學，從大學專題開始一直到 IC 競賽及助教，許多的事情少了你我都沒辦法一個人做得好，真的非常謝謝你！感謝詹景竹、戴瑋呈和張瑋城同學，我們各有特色，每天在實驗室一起努力和搞笑是一個難忘的回憶。感謝許博雄及陳奕均學弟，沒有你們的幫忙，我沒有辦法準時畢業！實驗室的黃筱珊、陳之悠、沈孟維、許博淵、蔡政君、廖元歆學弟們當然也不能忘記，和你們相處的日子真的很快樂。

謝謝我的女友，謝謝你不斷的支持與鼓勵，也讓我對未來的學習之路有了全新的轉變。也感謝桌球隊的朋友，跟你們一同練球是我最充實的回憶。

最後要感謝默默支持我的家人們，我的爸媽、姐姐，你們的溫暖是我努力最大的支柱。

在此，把本論文獻給所有愛我與所有我愛的人。

TABLE OF CONTENTS

1.	INTRODUCTION	1
1.1.	BACKGROUND	1
1.2.	MOTIVATION AND CONTRIBUTION	1
1.3.	ORGANIZATION OF THE THESIS	2
2.	INTRODUCTION OF COMPUTATIONAL STEREO	3
2.1.	OVERVIEW	3
2.2.	EPIPOLAR GEOMETRY	3
2.3.	THE GENERAL FLOW OF MATCHING ALGORITHMS.....	4
2.3.1.	<i>Matching Cost Computation</i>	4
2.3.2.	<i>Cost Aggregation</i>	6
2.3.3.	<i>Disparity Computation</i>	6
2.4.	A TAXONOMY EVALUATION.....	6
3.	RELATED WORK	9
3.1.	OVERVIEW	9
3.2.	LOCAL APPROACH.....	9
3.3.	GLOBAL APPROACH	10
3.4.	ADAPTIVE SUPPORT WEIGHT	12
3.5.	REAL-TIME IMPLEMENTATIONS	13
3.5.1.	<i>General Purpose Processor</i>	14
3.5.2.	<i>Graphic Processing Unit</i>	14
3.5.3.	<i>Digital Signal Processing Processor</i>	14
3.5.4.	<i>Application-Specific Integrated Circuit</i>	15
3.6.	SUMMARY	16
4.	PROPOSED MINI-CENSUS ADAPTIVE SUPPORT WEIGHT	17
4.1.	INTRODUCTION	17
4.2.	THE FLOW OF THE PROPOSED ALGORITHM	17
4.3.	MINI-CENSUS.....	18
4.4.	WEIGHT GENERATION AND APPROXIMATION	19
4.4.1.	<i>The Performance with Different Color Space</i>	20
4.4.2.	<i>The Color Distance</i>	21
4.4.3.	<i>The Effect of Proximity Weight</i>	22
4.4.4.	<i>Quantized Exponential Function</i>	22
4.4.5.	<i>The Final Weight Table</i>	24
4.5.	AGGREGATION ITERATION.....	25

4.6.	TWO-PASS COST AGGREGATION APPROXIMATION	28
4.7.	OVERALL SIMULATION RESULT	28
5.	DATA REUSE ANALYSIS OF HARDWARE IMPLEMENTATION	29
5.1.	OVERVIEW	29
5.2.	ARCHITECTURE OVERVIEW	29
5.3.	MATCHING COST COMPUTATION REUSE	31
5.3.1.	<i>Disparity-Order Reuse</i>	31
5.3.2.	<i>Pixel-Order Reuse</i>	32
5.4.	COST AGGREGATION DATA REUSE	33
5.4.1.	<i>Partial Column Reuse (PCR)</i>	33
5.4.2.	<i>Vertically Expanded Row Reuse (VERR)</i>	34
5.5.	COMPARISON	35
5.6.	SUMMARY	35
6.	HARDWARE IMPLEMENTATION	37
6.1.	OVERVIEW	37
6.2.	FUNCTIONAL BLOCK	38
6.2.1.	<i>Mini-Census Transform</i>	38
6.2.2.	<i>Weight Generation</i>	39
6.2.3.	<i>Aggregation and Winner-Takes-All</i>	40
6.2.4.	<i>Input and Output Control</i>	41
6.3.	HANDSHAKING	42
6.4.	ARBITRATION	43
6.5.	MEMORY	45
6.5.1.	<i>Memory Update Mechanism</i>	45
6.5.2.	<i>Memory Size</i>	46
6.6.	IMPLEMENTATION RESULT	48
6.6.1.	<i>External Bandwidth</i>	48
6.6.2.	<i>Area and Gate Counts</i>	49
6.7.	PERFORMANCE RESULT	50
	CONCLUSION	53
	FUTURE WORK	53
	REFERENCE	55

LIST OF FIGURES

FIG. 2-1 THE EPIPOLAR GEOMETRY OF THE BINOCULAR STEREO.....	3
FIG. 2-2 CORRESPONDENCE MATCHING FINDS THE ALL THE MATCHING PENALTIES OVER A DISPARITY RANGE.....	3
FIG. 4-1 THE FLOW OF THE PROPOSED ALGORITHM	17
FIG. 4-2 THE CENSUS TRANSFORM AND MATCHING	18
FIG. 4-3 THE PERFORMANCE COMPARISON WITH DIFFERENT COLOR SPACE	20
FIG. 4-4 THE PERFORMANCE ANALYSI OF PROXIMITY WEIGHTING.....	22
FIG. 4-5 THE WEIGHT FROM QUANTIZED EXPONENTIAL FUNCTION.....	23
FIG. 4-6 THE PERFORMANCE WITH QUANTIZED EXPONENTIAL FUNCTION	24
FIG. 4-7 THE ERROR RATE WITH THE AGGREGATION ITERATION AND WINDOW SIZE.....	26
FIG. 4-8 THE MINIMUM ITERATION WITH DIFFERENT SIZE OF SUPPORT WINDOW	27
FIG. 5-1 THE OVERVIEW OF HARDWARE ARCHITECTURE.....	29
FIG. 5-2 THE TWO DATA REUSE DIRECTIONS WITH DIFFERENT SIZE OF SUPPORT WINDOW	31
FIG. 5-3 THE PARTIAL COLUMN REUSE (PCR) IN 5x5 AGGREGATION WINDOW	33
FIG. 5-4 VERTICALLY EXPANDED ROW REUSE(VERR).....	34
FIG. 5-5 THE AVERAGE ACCESS COUNT VERSUS THE NUMBER OF EXPANDED PIXEL	35
FIG. 6-1 THE OVERVIEW OF THE HARDWARE DESIGN.....	37
FIG. 6-2 THE MODULE OF CENSUS TRANSFORM FOR LEFT AND RIGHT IMAGE.....	38
FIG. 6-3 THE MODULE OF WEIGHT GENERATION OF VERTICAL AND HORIZONTAL WEIGHTS	39
FIG. 6-4 THE MODULE OF COST AGGREGATION AND ITS PROCESSING ELEMENT.....	40
FIG. 6-5 THE PING-PONG BUFFER OF COST AGGREGATION MODULE	41
FIG. 6-6 THE FINITE-STATE-MACHINE OF THE INPUT AND OUTPUT CONTROL	42
FIG. 6-7 THE HANDSHAKING MECHANISM BETWEEN DIFFERENT MODULES	44
FIG. 6-8 THE HYBRID OF ROUND-ROBIN AND FIXED PRIORITY ARBITRATION STRATEGY	45
FIG. 6-9 THE COLUMN BASED CYCLIC BUFFER UPDATE MECHANISM.....	46
FIG. 6-10 THE MEMORY SIZE OF DIFFERENT MODULE	47
FIG. 6-11 THE PERFORMANCE WITH THE BUS ACCESS LATENCY	49
FIG. 6-12 THE PERCENTAGE OF THE MEMORY AREA AND COMBINATIONAL GATE COUNTS	50
FIG. 6-13 THE IMPLEMENTATION RESULT WITH DIFFERENT METHOD	52

LIST OF TABLES

TABLE 2-1 MATCH METRICS FOR CORRESPONDENCE MATCHING [3]	5
TABLE 2-2 THE TEST SEQUENCES OF THE TAXONOMY EVALUATION	8
TABLE 4-1 THE RESULT OF APPROXIMATED COLOR DISTANCE	21
TABLE 4-2 THE WEIGHT TABLE OF PRESERVING 2 MSB BITS.....	25
TABLE 4-3 THE WEIGHT TABLE OF PRESERVING 1 MSB BIT	25
TABLE 4-4 THE EFFECT OF DIFFERENT TECHNIQUES	28
TABLE 5-1 THE RESULT OF APPROXIMATED COLOR DISTANCE	36
TABLE 6-1 THE IMPLEMENTATION RESULT OF AREA AND GATE COUNTS.....	49
TABLE 6-2 THE ERROR RATE COMPARISON OF DIFFERENT METHOD	51
TABLE 6-3 THE PERFORMANCE COMPARISON OF DIFFERENT METHOD	51



1. Introduction

1.1. Background

The stereo vision is one of the most popular topics in computer vision, and still attracts the attention of many researchers. The stereo vision is the process of finding the depth or distance information from a pair of images of the same scene. It can be used for many applications such as the 3D video conference [1], the Z-keying, and the virtual reality [2]. If we obtain the 3D depth map in the high speed, it is possible to merge the real and the virtual world in real time.

The stereo algorithm can be categorized as local and global approach [3]. The local approach focuses on finding the similarities of reference and target windows by using the block matching or feature matching. The global approach uses the global constraints to optimize the result. Since the local approach favors low complexity, they are often adopted by real-time implementation. However, these methods often suffer from incorrect result on occlusion, uniform texture, and ambiguity. The global approach can solve these problems but suffer from the huge processing time. Although some real-time global methods can be implemented through GPU in the graphics card or MMX of CPU, the implementation still cost expensive for embedded applications since GPU and MMX are not dedicated hardware for stereo algorithms.

1.2. Motivation and Contribution

Motivated by the need of high accurate and low cost real-time stereo systems, this thesis proposed hardware friendly algorithm based on a state-of-art local approach. The goal is to build a dedicated hardware for low cost real-time depth estimator with high

accuracy.

The major contribution in this thesis includes:

1. We modified the adaptive support algorithm and make it more hardware friendly. The modified algorithm has much lower complexity and more capability of dealing with radiometric problem.
2. We analyze the pixel-order and disparity-order data reuse strategies with the vertically expanded row and partial column reuse methods.
3. We implemented and verified the real-time hardware of the proposed algorithm (Mini-Census Adaptive Support Weight).

1.3. Organization of the Thesis

In Chapter 2, we briefly introduce background of the computational stereo. In Chapter 3, we briefly introduce the stereo algorithms and real-time implementations. Chapter 4 discusses the detail of the proposed algorithm with the mini-census, two-pass aggregation, and quantized exponential Manhattan distance. In addition, the simulation result is shown in this chapter. Chapter 5 analyzes the data reuse problem of hardware design implemented by aggregation based algorithm. Chapter 6 shows the detail of the hardware design and the implementation result. Finally, the conclusion is given after Chapter 6.

2. Introduction of Computational Stereo

2.1. Overview

The concept of computational stereo is to construct the structure in the three-dimension space from different view point. The fundamental basis is to evaluate the depth of the object by finding the correspondent points of the object projecting on the two unique image pairs. The correspondent points are the feature points visible on both view point. The process of finding the correspondence is referred as *correspondence matching*. The *disparity map* for structure reconstruction can be computed after the correspondence matching.

2.2. Epipolar Geometry

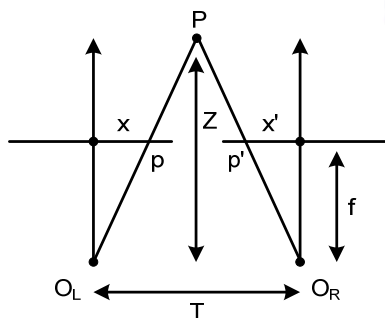


Fig. 2-1 The epipolar geometry of the binocular stereo.

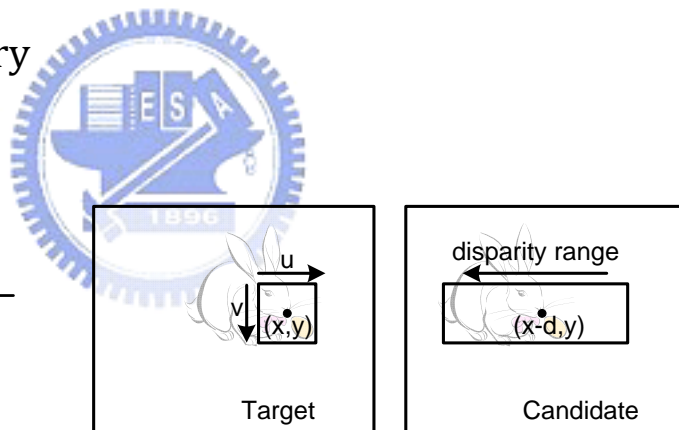


Fig. 2-2 Correspondence matching finds the all the matching penalties over a disparity range.

Fig. 2-1 shows the binocular stereo calibrated with epipolar geometry. O_L , O_R , and f are the two optical centers, and the distance between them is called the baseline. The object P is projected on to two points (p and p'). The depth Z of the object P can be computed by triangulation. As a result, the formula of depth Z can be written as $Z = f/d$, where f is the focal length of the camera, d is the displacement of the two points, $d=x-x'$.

(depicted in Fig. 2-1). All the parameter can be obtained during the setup of the system except the displacement. Therefore, the goal of computational binocular stereo is to estimate the displacement between each corresponding pair of pixels in the target and candidate images (depicted in Fig. 2-2). The displacement is referred as *disparity* and the process is referred as *disparity estimation*. The set of disparity of all the pixels in an image is called the *disparity map* or *disparity image*.

2.3. The General Flow of Matching Algorithms

According to Scharstein and Szeliski [4], the major steps of the stereo algorithms consist of three steps: matching cost computation, cost (support) aggregation, and disparity computation/optimization. The matching cost in the first step represents the dissimilarity of different matching candidates. The cost aggregation is to sums up the result of the dissimilarities together, the concept of this is like exchanging the information of neighboring pixels. The last step is to compute the final disparity map from the matching cost. The details of them will be discussed in the following sections.

2.3.1. Matching Cost Computation

The disparities map can be computed by evaluating the *matching cost* for every disparity candidates. The matching cost represents the matching penalties after the correspondence matching. The range of the disparity candidates is called the *disparity range*. The correspondence matching is based on finding the correspondence of the support region of the reference and candidate pixels. The support region is usually a square window, which is called the *support window*. The match metrics are listed in TABLE 2-1. The details can be referred to [3]. The general formula of matching cost computation can be written as

$$Cost(x, y, d) = Matching(I_1(x, y) - I_2(x - d, y)), \quad (2.1)$$

where I_1 , I_2 , represent the reference and target images. The matching result forms a volume of matching cost in 3D space. The absolute difference (AD) is most commonly used for many stereo algorithms due to its simplicity. However, the AD has poor quality while the test image has the global radiometric changes. The experiment [5] shows that the rank and mutual information performs better than AD for global radiometric changes and noises due to the match metrics compares the difference of their local characteristics rather than absolute difference of luminance.

TABLE 2-1 match metrics for correspondence matching [3]

MATCH METRIC	DEFINITION
Normalized Cross-Correlation	$\frac{\sum_{u,v} (I_1(u, v) - \bar{I}_1) \cdot (I_2(u - d, v) - \bar{I}_2)}{\sqrt{\sum_{u,v} (I_1(u, v) - \bar{I}_1)^2 \cdot (I_2(u - d, v) - \bar{I}_2)^2}}$
Sum of Squared Difference	$\sum_{u,v} (I_1(u, v) - I_2(u - d, v))^2$
Normalized Sum of Squared Difference	$\sum_{u,v} \left(\frac{\sum_{u,v} (I_1(u, v) - \bar{I}_1)}{\sqrt{\sum_{u,v} (I_1(u, v) - \bar{I}_1)^2}} - \frac{\sum_{u,v} (I_2(u, v) - \bar{I}_2)}{\sqrt{\sum_{u,v} (I_2(u, v) - \bar{I}_2)^2}} \right)^2$
Sum of Absolute Difference	$\sum_{u,v} I_1(u, v) - I_2(u - d, v) $
Rank	$\sum_{u,v} I'_1(u, v) - I'_2(u - d, v) $ $I'_k(u, v) = \sum_{m,n} I_k(m, n) < I_k(u, v)$
Census [6]	$\sum_{u,v} HAMMING(I'_1(u, v) - I'_2(u - d, v))$ $I'_k(u, v) = BITSTREAM_{m,n}(I_k(m, n) < I_k(u, v))$
Mutual Information [7]	$\log \left(\frac{P(I_1(u, v) \cdot I_2(u - d, v))}{P(I_1(u, v)) \cdot P(I_2(u - d, v))} \right)$

2.3.2. Cost Aggregation

Cost aggregation is to aggregate the cost of correlated pixels over a support window. The concept of the cost aggregation is that neighboring pixels may be highly correlated to center pixel. The formula of cost aggregation is written as follow

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

where $Cost_{init}$ is the initial matching cost from the match metrics. The ω is the related weight for each cost. The effect of the weight is to limit the influence of unrelated pixels. The cost aggregation helps to improve the quality of low texture area since it is lack of information. However, this work also blurs the edge of the object when the cost of different object is aggregated together. Therefore, the determinant of the weight is of vital important for cost aggregation.

2.3.3. Disparity Computation

The disparity map can be computed from the matching cost or aggregated cost. The simplest way is to select the disparity candidate with minimal cost, and the process of this is called *winner-takes-all* (WTA). The formula of WTA can be expressed as below

$$disparity(x, y) = \{d_m | cost(x, y, d_m) = argmin(cost(x, y, d_n)), m, n \in [0, d_{max}]\}, \quad (2.3)$$

where d_m is the disparity with the minimal cost over a disparity range. The more robust methods with complex disparity optimization will be discussed in 3.2.





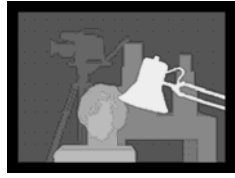

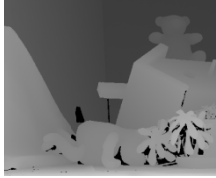
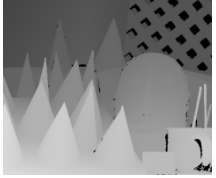
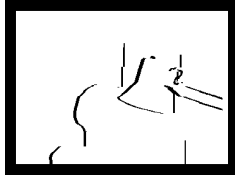











2.4. A Taxonomy Evaluation

For the computational stereo algorithms, the ambiguous match leads to the poor quality for computational result. The ambiguous points include the occlusion, low-texture (non-feature), and repetitive patterns. Hence, a taxonomy evaluation [4] is

proposed. The evaluation includes three parts: non-occluded area, total area, and discontinuous area. The test sequence is shown in TABLE 2-2. The four sequences, tsukuba, venus, teddy, and cones, are the most commonly used for performance evaluation. The gray level of the ground represents the depth of the object. The pixel with brighter gray level means it is closer to the camera or observer, and vice versa. For the images of non-occlusion images, the non-occluded regions and occluded regions are represented with white and black color respectively. In the discontinuities images, the regions near depth discontinuities are represented as white; occluded and unknown regions are represented as black, and other regions are represented as gray. The error for different three parts is only evaluated in white regions.



TABLE 2-2 the test sequences of the taxonomy evaluation

	Tsukuba	Venus	Teddy	Cones
Input				
Ground Truth				
Non-occlusion				
All				
Discontinuities				

3. Related Work

3.1. Overview

The methods of disparity estimation can be roughly categorized into two types: local and global approaches. Local approach determines the disparity of a pixel based on the similarity of a support window. These methods can iteratively aggregate or regularly diffuse the matching cost over the support window. The local methods have low computation complexity and storage requirement, and they are often adopted by real-time implementations [8]. Global methods define objective energy functions which usually include a data term and a neighboring term. The data term is often a transformed version of the matching cost. The neighboring term is represented with a smoothness penalty to enforce disparity smoothness. Sometimes the neighboring term would also include occlusion penalty and segment constraint to improve the disparity estimation result. This is the major difference that set global methods apart from local methods.

3.2. Local Approach

Among the local methods, the matching cost (dissimilarity measure) often is block sum of absolute difference, normalized cross-correlation, census transform, or mutual information. Local methods often suffer from incorrect disparity estimation at occlusion, low texture, and repeating pattern regions. Although larger supporting window and aggregation iteration improve the stereo matching performance at the low texture and repeating pattern regions, it harms the performance at occlusion region. Because of this trade-off between large and small support windows, the reliable variable window size [9-11] was proposed. The window size depends on the reliability

measurement of current window size. The adaptive window size enhances the depth for low texture area but the issue of occluded and border area still remains. To enhance the performance at the occlude and border area, the shiftable window approach is adopted [12][13] and the combination of adaptive size and shiftable window is discussed in [14]. However, the qualitative result [14] shows that it still difficult result on both low texture and border area.

To solve this issue at the both low texture and border area, the concept of adaptive support weight (ADSW) aggregation is proposed by Yoon [15]. This approach adaptively changes the weights in a support window according to the color and spatial distance between the center and neighboring pixels. Consequently, adaptive support weight can achieve the effect of using window with arbitrary size and shape. Once all the weighted sums of costs are computed, they are iteratively recomputed to produce a smoothed dense disparity map. Later, a segmentation support aggregation was proposed [16][17]. The Outlier rejection [16] claimed to have both a very short computation time and good stereo matching performance. Recently, a report [18] shows that Adaptive weight [15] and Segment support [17] outperform than other aggregation based methods. [18-28]. Although adaptive support weight is the state-of-the-art of local methods, the complexity is much more than segment based method [18].

3.3. Global Approach

Global methods assume the disparity map with minimum objective energy should be very similar to the ground truth. Therefore, global methods focus on optimizing the energy function to determine the disparity map. One of the earlier global methods is dynamic programming [29]. This method focuses on optimizing the energy associated with each scanline during disparity estimation. Although dynamic programming takes

the horizontal global information into optimization, vertical correlation between scanlines is not considered. As a result, the disparity map of dynamic programming often exhibit horizontal streaks, thus reducing the quality of the disparity map. Motivated by the need of 2-D optimization during disparity estimation, Roy and Cox [30] proposed to model the disparity-image space as a 3-D grid graph. By finding the min-cut on this graph, the disparity map with optimum energy is found; this optimization algorithm is also known as graph-cut. Unfortunately, the computation and storage requirement for running graph-cut on 3-D grid graph is enormous. Later, Boykov and Kolmogorov proposed the iterative swap and expansion moves [31][32] which also use graph-cut to find the best moves. Unlike Roy and Cox's method, a simpler two-variable graph structure which can be regarded as a 2-D graph was used in swap and expansion moves. This simpler graph reduces the computation loading of graph-cut. However, the extra iterations of moves compensate the benefit.

On the other hand, Scharstein and Szeliski [4] proposed the Bayesian diffusion method which iteratively diffuses support at different disparities according to nonlinear diffusion strength. This is similar to using different weightings within the support window. Later, Sun [15] proposed the belief propagation for disparity estimation based on the concept of the Bayesian diffusion. Essentially, belief propagation is similar to Bayesian diffusion. Both methods propagate information based on probability model between neighboring pixels. However, belief propagation bridges the link of the global energy function with information passing, which is absent in Bayesian diffusion. In addition, belief propagation uses a more complex updating mechanism, which is used to optimize the final energy. As a result, belief propagation has been reported [4][14] to produce disparity maps with much better quality than Bayesian diffusion. Currently, the disparity map produced by the state-of-art methods combine adaptive support weight,

segment constraint, and belief propagation together. Although belief propagation based methods are the leading methods in stereo matching performance, they also suffer from high computational complexity.

3.4. Adaptive Support Weight

Adaptive support weight (ADSW) proposed by Yoon [15] is the state-of-art of local approach, which aggregates the cost with the weight adaptively generated by the color and spatial distance. The concept of ADSW is that the correlation of the neighboring pixels is related to their spatial distance, which is called the *proximity weight*. The correlation of two pixels is related to their color distance, which is called the *color weight*. The weight in the cost aggregation formula (2.2) can be represented as

$$\omega(p, q) = f(\Delta c_{pq}) \cdot f(\Delta s_{pq}), \quad (3.1)$$

where Δc_{pq} and Δs_{pq} represent the color distance and spatial distance between pixel p and q respectively. The $\omega(p, q)$ represents the strength of aggregating the cost. The color distance of two pixels is measured in the CIELab color space due to it is more perceptually uniform. As the distance between two points in color space increases, it is reasonable to assume that the similarity is decreased for perceptual stimuli. Especially, Euclidean distance correlates strongly with human color discrimination performance. Therefore, the perceptual difference between two colors is represented as

$$D(c_p, c_q) = 1 - \exp\left(-\frac{\Delta c_{pq}}{\gamma}\right). \quad (3.2)$$

The strength of aggregating by color similarity is defined as

$$f_c(\Delta c_{pq}) = \exp\left(-\frac{\Delta c_{pq}}{\gamma}\right). \quad (3.3)$$

In the same way, the strength of aggregating by proximity is defined as

$$f_p(\Delta s_{pq}) = \exp\left(-\frac{\Delta s_{pq}}{\gamma}\right). \quad (3.4)$$

According to the (3.3)(3.4), the final weight for aggregating can be rewritten as

$$\omega(p, q) = \exp\left(-\left(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta s_{pq}}{\gamma_s}\right)\right). \quad (3.5)$$

The final weight is the combination of color weight and proximity weight. Hence the cost aggregation can be rewritten as

$$E(p, \bar{p}_d) = \frac{\sum_{q \in N_p, \bar{q}_d \in N_{\bar{p}_d}} \omega(p, q) \omega(\bar{p}_d, \bar{q}_d) e(q, \bar{q}_d)}{\sum_{q \in N_p, \bar{q}_d \in N_{\bar{p}_d}} \omega(p, q) \omega(\bar{p}_d, \bar{q}_d)}, \quad (3.6)$$

where p and q are the corresponding pixels in the reference image, and \bar{p}_d and \bar{q}_d are the corresponding pixels in the target image with disparity value d . $e(q, \bar{q}_d)$ represents the matching cost computed by using the pixels of q and \bar{q}_d . When using the truncated AD (absolute difference), it can be expressed as

$$e(q, \bar{q}_d) = \min \left\{ \sum_{c \in \{R, G, B\}} |I_1(q) - I_2(\bar{q}_d)|, T \right\}, \quad (3.7)$$

where I_1 and I_2 are the reference image and target image respectively. The adaptive support weight gives a quality result on both low texture and border area; the occluded area can be refined by left-right consistent check.

3.5. Real-time Implementations


The real-time stereo is essential part for automatic mobile, robot, or any other tracking system. The issues of implementing the real-time systems are the computing complexity, memory size, and bandwidth. Currently, the implementations can be categorized as four types: general purpose process, graphic processing unit (GPU),

digital signal processor (DSP), and application-specific integrated circuit (ASIC).

3.5.1. General Purpose Processor

With the state-of-art processor, some local approach can be implemented to compute the disparity image in real-time. These implementations [33] cannot give a quality result since they are often simple approach. For a more robust and fast implementation of effective aggregation algorithm [34], it can achieve only 18.9 million disparities per second (MDS), the speed is still far from real-time computing. As for the global approach, the complexity of graph-cut and belief propagation is much higher than local approach. These methods often take several minutes to compute one disparity image. However, a recent implementation [35] shows that dynamic programming can be implemented to compute a good disparity result in real-time.

3.5.2. Graphic Processing Unit



Recently, the configurable graphic hardware gives another solution for parallel computing. The programmer can write CUDA (Compute Unified Device Architecture) code, developed by NVIDIA, to accelerate the software. Currently, the solution of using GPU provides extremely high bandwidth from 6.4GB/sec to 128GB/sec. The number of stream processors is up to 256. (The details of using the GPU can refer to GPGPU <http://www.gpgpu.org/>). With the computing power of GPU and CPU, many algorithms generating high quality result [34] [36] [37] [38] can be implemented in real-time. The programmable graphics hardware is suitable for different stereo algorithms.

3.5.3. Digital Signal Processing Processor

Although the real-time can be implemented by GPU and CPU, the cost is too expensive for embedded applications. For a low cost embedded system, the Digital

Signal Processor (DSP) would be more cost efficient. The DSP provides a SIMD and VLIW instructions, which is very useful for parallel computing for local stereo matching. Some real-time local approach is implemented by using DSP [39][40]. Therefore, the computing power of DSP is limited, and this constraint the development for more accurate disparity estimation algorithms.

3.5.4. Application-Specific Integrated Circuit

Comparing to the GPU, the application-specific integrated circuit (ASIC) has much more flexibility to design the processing element for the algorithms. The matching and data path can be fully customized and achieve high utilization. A simple absolute-difference with variable window size is implemented by hariyama [41], which can achieve high utilization and low. However, the bandwidth issue and internal memory size becomes a bottleneck of designing the hardware. It is a challenge to deal with the intermediate result for the algorithms which requires many times of iteration. The bandwidth requirement of transferring the intermediate result is extremely high and cannot meet the real-time constraint. Besides, the chip area will get large if the intermediate result is stored in the internal memory. The trade-off of the bandwidth and internal memory size becomes the important issue. To solve this problem, the concept of hierarchical approach is proposed. The hierarchical belief propagation (HBP) [42][43] reduces the number of aggregation iteration, and this relaxes the problem of high external bandwidth. Nevertheless, the FPGA implementation of HBP still requires huge block ram. Therefore, although the ASIC design can give a dedicated solution, it is still a challenge to design a low cost real-time architecture with iteratively cost aggregated and disparity optimized algorithms.

3.6. Summary

Considering the real-time problem, the general purpose processor and DSP has its limitation for the more complexity matching algorithms. The acceleration of using GPU has high potential for implementing high complex stereo algorithms since it has extremely high bandwidth and large numbers of streaming processors. Although the GPU solution may be implemented in the embedded system, it still cost expensive. For a low cost embedded system, the DSP or ASIC may be a more proper candidate. However, the issue of dealing with the intermediate result is big challenge for ASIC solution due to the limitation of the external bandwidth. This results in the high internal memory cost for the ASIC solution.



4. Proposed Mini-Census Adaptive Support Weight

4.1. Introduction

In this chapter, we will introduce the proposed algorithm which is modified from the Adaptive Support Weight [15] introduced in 3.4. We simplified the algorithm and make it applicable for hardware design. Besides, we also improve its capability of dealing with the lighting effect by applying census transform [9]. There are three major challenges of designing the hardware for real-time Adaptive Support Weight. The challenges are the adaptive weight generating function, iteratively cost aggregation and data reuse. We will discuss how we solved the problem of the previous two problems in the proposed algorithm, and discuss the data reuse problem in Chap 5.

4.2. The Flow of the Proposed Algorithm

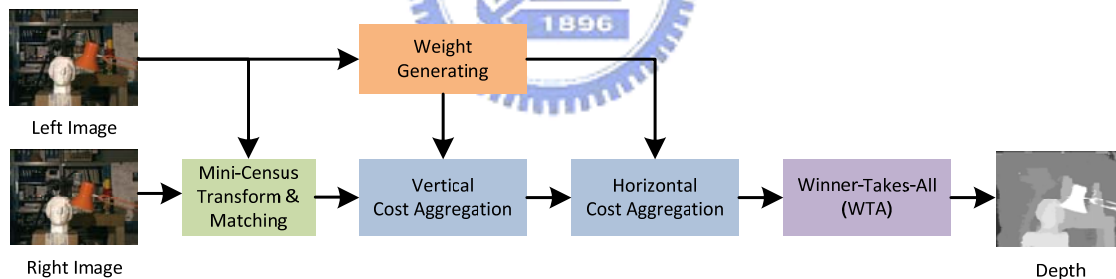


Fig. 4-1 The Flow of the Proposed Algorithm

Fig. 4-1 shows the flow of the proposed algorithm. The proposed algorithm consists of four major steps. First, the mini-census matching cost computation performs mini-census transform on the captured left and right images and computes the initial matching cost of each pixel. The second step is the weight generation which generates the weight coefficients needed in the cost aggregation step. Once the initial matching cost and weight coefficients are available, the matching cost will be aggregate through

a two-pass cost aggregation step. Finally, after the cost aggregation, the disparity map can be obtained by finding the best disparity with the minimum matching cost through a Winner-Takes-All method.

4.3. Mini-Census

The census transform compares the intensity of each pixel within a support window with the center pixel. If a pixel's intensity is larger than the center pixel's intensity, it is given the label 0, otherwise the label 1. The comparison is done in raster-scan order. After the comparison of all pixels within the support window, a binary bitstream is obtained which characterizes the pixel relation between the center pixel and its surrounding pixels. Since the bitstream represents relative information, the census transform is therefore much less sensitive to image bias and gain. In addition, the census transform preserves the depth boundary in disparity maps better than the traditional SAD does.

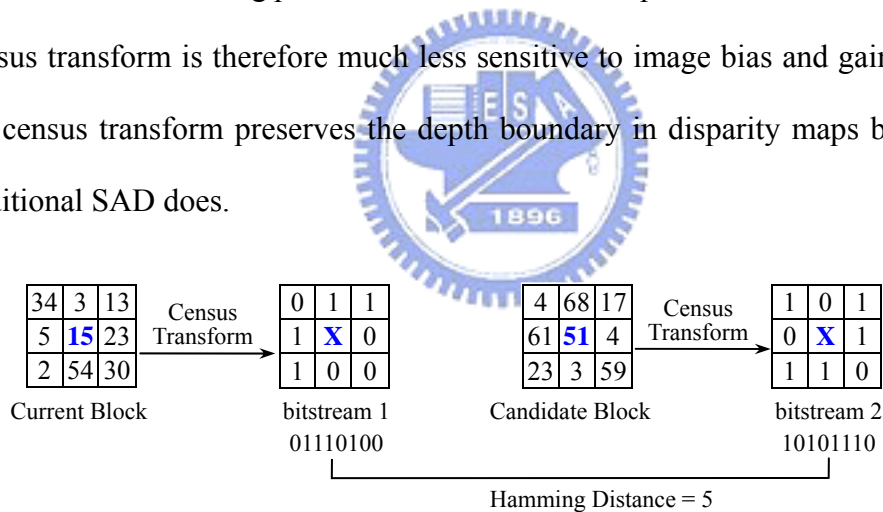


Fig. 4-2 The census transform and matching

To compute the matching cost, the bitstreams b_1 of a pixel in current view and the bitstream b_2 of the candidate corresponding pixel in the other view are obtained first, and then the hamming distance between the two bitstreams is computed and taken as the matching cost. The cost can be defined as

$$Cost(x, y, d) = H(b_1, b_2) \quad (4.1)$$

,where H is the hamming distance function. We would refer the hamming distance as the census cost hereon for brevity. Fig. 4-2 illustrates an example of the census transform with a 3x3 support window. The bitstreams of the current pixel position and the candidate corresponding pixel are 01110100 and 10101110. The hamming distance between the bitstreams is 5; hence, the census cost is 5.

The Mini-Census is a simplified census transform. Instead of a block of pixels, only 6 significant pixels will be transformed into the bitstream. The Mini-Census can help reducing the internal memory size of storing the matching cost with minor matching performance loss.

4.4. Weight Generation and Approximation

The adaptive weight generation is based on the color distance and proximity. The proximity weight is fixed for a constant size of support window, but the color distance term is not fixed as the support aggregation window changes position. In the original Adaptive Support Weight (ADSW), the color distance weight is generated from the CIE-Lab color space, which uses floating-point numbers to represent a color. However, using floating-point numbers is not friendly for hardware design. Besides, the square-root and exponential function used in color distance computation and color weight generation are not hardware friendly either. To improve the algorithm to be more hardware implementation friendly, we adopted integer-valued color space, approximated color distance, and approximated exponential function. Moreover, we also removed the proximity weight to further reduce computational complexity. The performance of these improvements is explained in the following subsections.

4.4.1. The Performance with Different Color Space

The color space has great impact on the performance of many image processing algorithms. We evaluate the impact of using different color spaces (YUV, RGB, and CIE-Lab) on the performance of stereo matching. The best parameters for different size of support aggregation are different. Hence, to eliminate the effect of different parameter, we simulate 100 samples for each size of the window to get the best parameter.

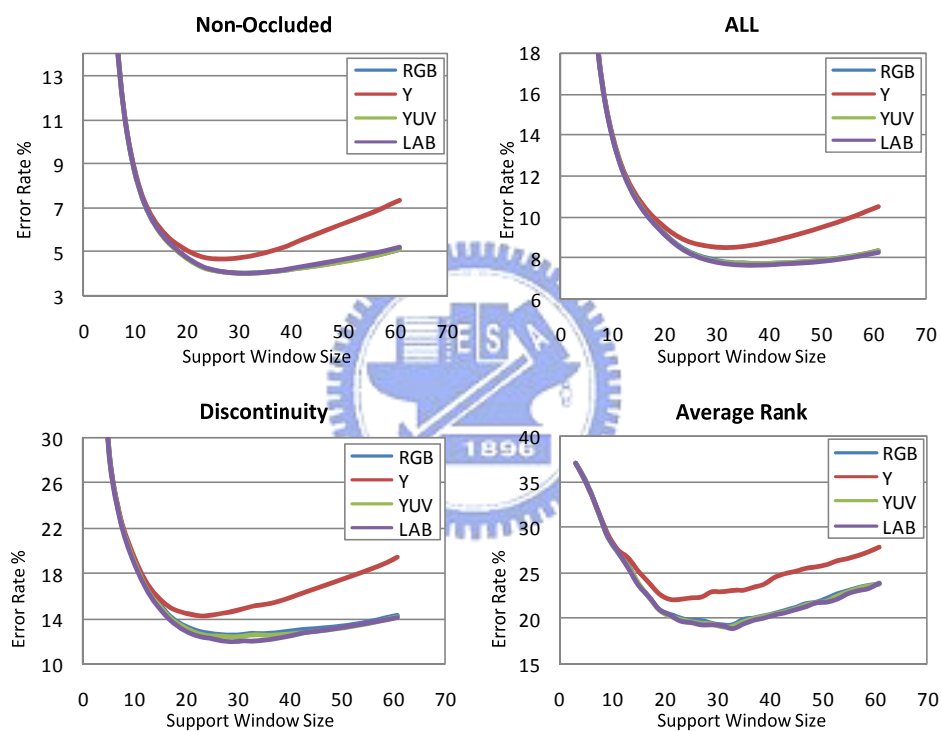


Fig. 4-3 The performance comparison with different color space

From the Fig. 4-3, the performance of using color spaces with three color components (YUV, RGB, CIE-Lab) is almost the same. The color space with only luminance component has the worst performance since it lack the other two dimensions of the color space. It can be seen that for three-component color spaces, the weight generated from using different color spaces does not have significant impact on the

stereo matching performance. Hence, this implies that we can choose to use any three-component color space that is suitable for the design. Since YUV and RGB can be represented using three unsigned integers instead of CIE-Lab's three floating-point numbers, YUV and RGB are more suitable for hardware design. We choose to use YUV in our algorithm because it has been reported to slightly outperform RGB in stereo matching.

4.4.2. The Color Distance

In ADSW, the color distance is defined as the Euclidean distance in the color space, which is written as follow

$$D_{color} = \sqrt{(Y_1 - Y_2)^2 + (U_1 - U_2)^2 + (V_1 - V_2)^2}. \quad (4.2)$$

The square root of the Euclidean distance is a nonlinear operator which is difficult for the hardware design. On the other hand, the Manhattan distance is more hardware efficiency. The formula is written as follow

$$D_{color} = |Y_1 - Y_2| + |U_1 - U_2| + |V_1 - V_2|. \quad (4.3)$$

TABLE I compares the performance of using the Euclidean and Manhattan color distance. The result shows that the Manhattan is distance is little better than Euclidean distance for different error tolerance and different test sequences.

TABLE 4-1 the result of approximated color distance

Method	Error Tolerance	rank	Error Rate %			
			TSUKUB A	VENUS	TEDDY	CONES
Euclidean	0	12.2	7.95	21.4	18.0	12.2
Manhattan		11.1	7.22	21.7	16.8	11.1
Euclidean	1	17.3	3.47	0.91	14.3	11.2
Manhattan		16.3	3.08	0.59	14.0	10.1

4.4.3. The Effect of Proximity Weight

Proximity Weighting reduces the effect of pixels farther from the window center and has been applied to improve the quality of the matching performance. To determine the necessity of applying the proximity weighting, we compare the performance of using and not using proximity weighting. Fig. 4-4 shows the error rate with different support window size. In Fig. 4-4(a), the error rate increased when the window size is too small. The error rate also increases as the window size increases over 27x27. However, the error rate after applying the proximity weighting does not increase while enlarging the window size. This is shown in Fig. 4-4(b). It is the proximity weight that limits the influence of the farther pixels.

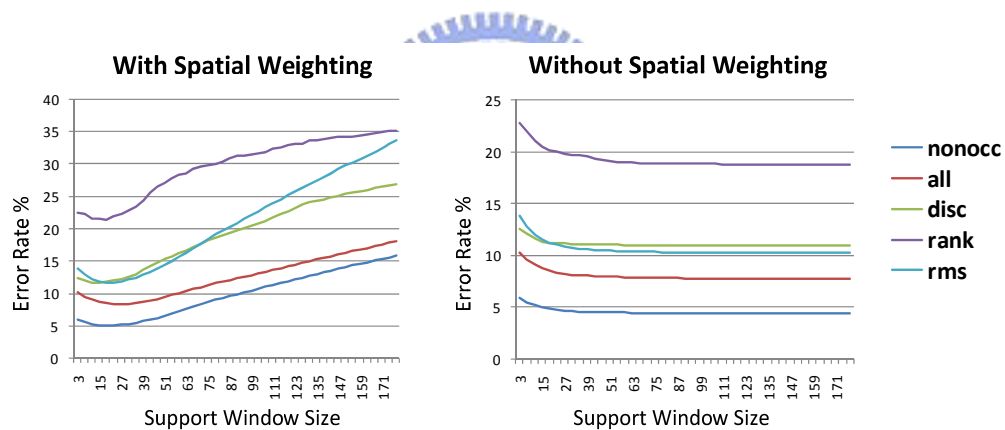


Fig. 4-4 The Performance Analysis of Proximity Weighting

4.4.4. Quantized Exponential Function

The quantized exponential function is the simplification of the original exponential weight generating function and it also helps to reduce the complexity of the aggregation process. The quantized exponential function is a scaled and quantized version of the original function. The quantized exponential function be represented as below.

(4.4)

$$\omega_{color} = \text{quantize} \left[\left(e^{-\frac{D_{color}}{\gamma_c}} \right) \times \text{scaling factor} \right]$$

, the result of the quantized exponential function is acquired by first multiplying the value of the original exponential function with a scaling factor 2^n , and then quantizing it to preserve only a few MSB bits. The scaling maps the floating number to integer number, which is more hardware friendly. The preserving bits help to reduce the complexity of the cost aggregation. In original cost aggregation step, the process is a sum-of-product of the weight vector and cost vector. If the weight is coded with one-hot encoding, the product operator can be simplified to shift operator, which is much more hardware-efficiency. Fig. 4-5 shows the weight from the original and quantized exponential function with different number of preserved bit. The output of the quantized exponential function is multiplied by 64 and the quantized. Fig. 4-5c and Fig. 4-5d are the output of the quantized exponential function with 2 and 1 MSB preserved respectively.

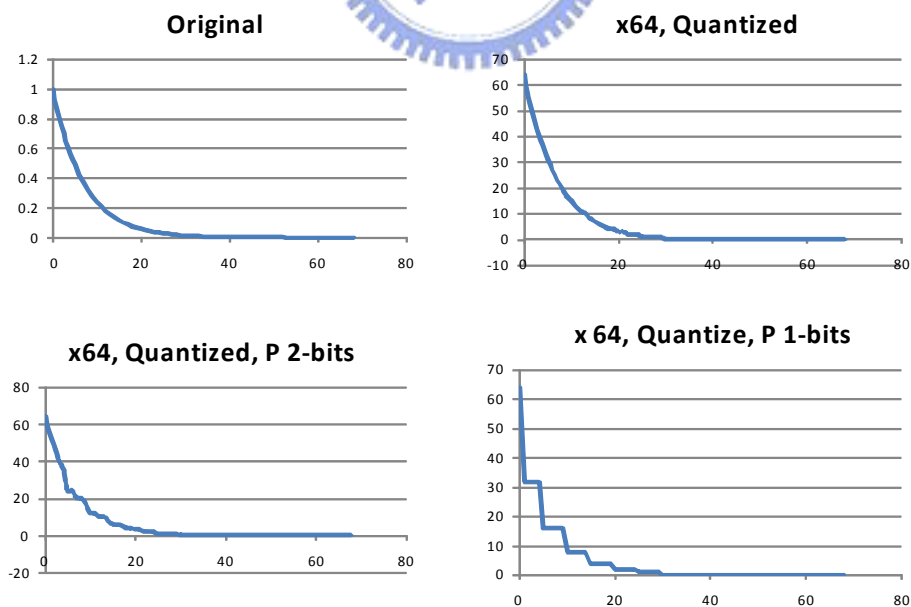


Fig. 4-5 The weight from quantized exponential function

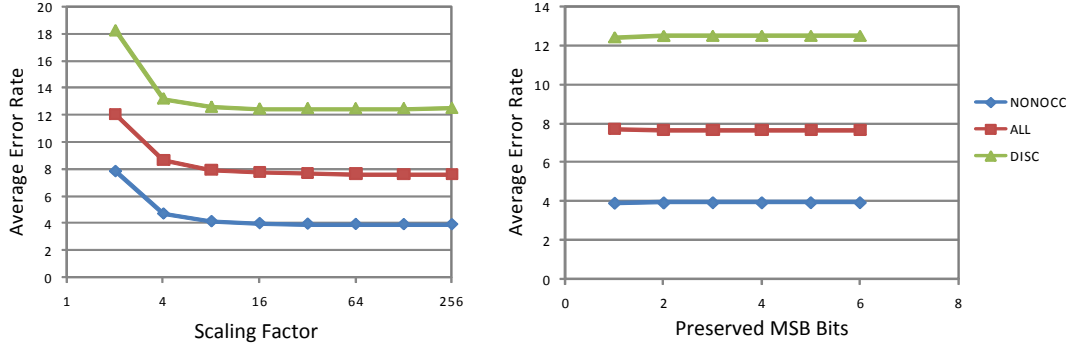


Fig. 4-6 The performance with quantized exponential function

Fig. 4-6 shows the performance of using the quantized exponential function with different scaling factors and number of preserved MSBs. Fig. 4-6a shows that the average error rate is decreasing if the scaling factor is smaller than 32. If the scaling factor is larger than 64, there is no conspicuous difference with the error rate. Hence, with acceptable quality, the smallest scaling factor can be selected as 64. Fig. 4-6b shows that there is no conspicuous difference of all the preserved bits. Therefore, we set the scaling factor as 64 and preserves only one MSB.

4.4.5. The Final Weight Table

After the discussion in 4.1, the weight generating function can be simplified into a mapping table with the YUV color space, discard of the proximity weight, quantized exponential function and Manhattan distance. The table is listed in TABLE 4-2, 4-3. The difference of these two tables is the preserving MSB bits of the quantized exponential function. According to the Fig. 4-6b, TABLE 4-3 would be is more proper for hardware design since the weights of which are all the power of two. As a result, the weight generating Equation (4.4) (4.2) becomes the Equation (4.5) (4.6).

$$\omega_{color}(x, y, i, j) = Table[D_{color}(x, y, i, j)] \quad (4.5)$$

$$D_{color}(x, y, i, j) = |Y_{(x,y)} - Y_{(x+i,y+j)}| + |U_{(x,y)} - U_{(x+i,y+j)}| + |V_{(x,y)} - V_{(x+i,y+j)}| \quad (4.6)$$

TABLE 4-2 The weight table of preserving 2 MSB bits

Distance	Weight	Distance	Weight	Distance	Weight	Distance	Weight
0	64	8	20	16	6	24	2
1	55	9	17	17	5	25	1
2	48	10	12	18	4	26	1
3	40	11	12	19	4	27	1
4	36	12	10	20	3	28	1
5	24	13	10	21	3	29	1
6	24	14	8	22	2		
7	20	15	6	23	2		

TABLE 4-3 The weight table of preserving 1 MSB bit

Distance	Weight	Distance	Weight	Distance	Weight	Distance	Weight
0	64	8	16	16	4	24	2
1	32	9	16	17	4	25	1
2	32	10	8	18	4	26	1
3	32	11	8	19	4	27	1
4	32	12	8	20	2	28	1
5	16	13	8	21	2	29	1
6	16	14	8	22	2		
7	16	15	4	23	2		

4.5. Aggregation Iteration

The aggregation based method refines the depth result by iteratively aggregating the matching cost. The cost aggregation formula is defined as

$$Cost_{t+1}(x, y, d) = \sum_{i=-r}^r \sum_{j=-r}^r Cost_t(x+i, y+j, d) \cdot \omega_{color}(x, y, i, j) \quad (4.7)$$

,where $Cost_t$ and $Cost_{t+1}$ is the aggregated cost at iteration t and $t+1$, and r are the width and height of the aggregation window. The iterative aggregation poses a challenge for

real-time hardware design due to the inter-iteration dependence which limits the parallelism and the huge memory storage and wide bandwidth requirement. Hence, the reduction of aggregation iterations is important issue.

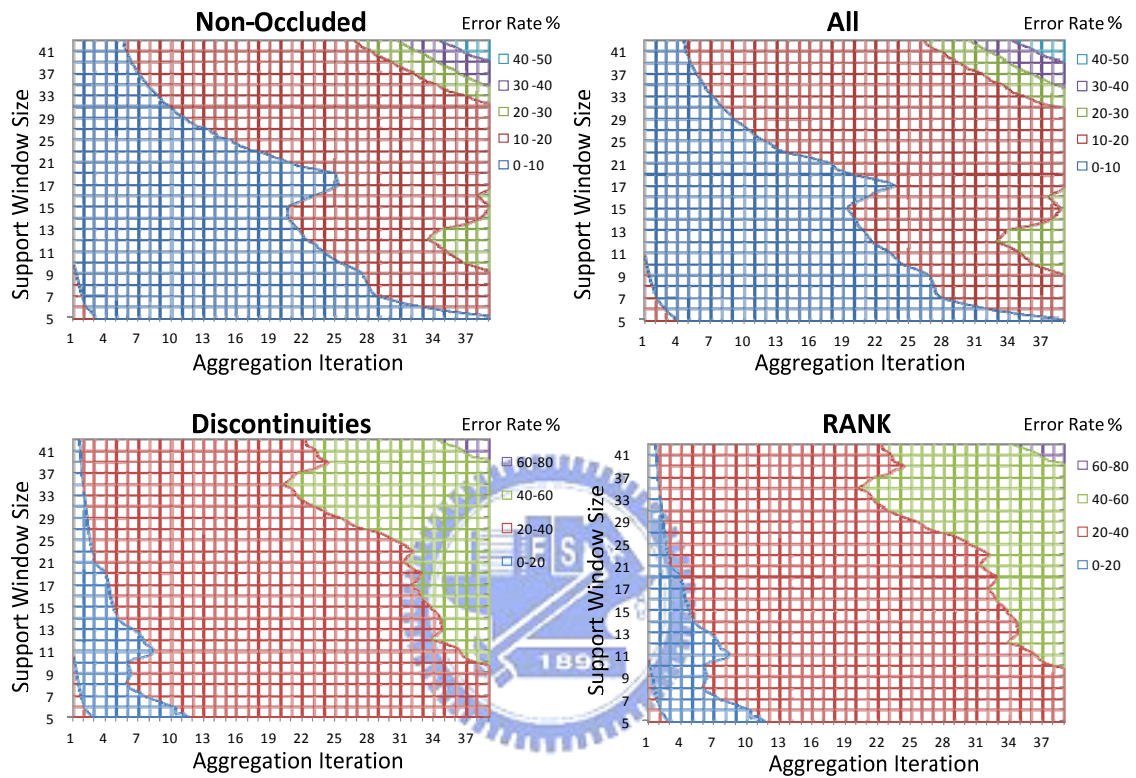


Fig. 4-7 The error rate with the aggregation iteration and window size

The best number of cost aggregation iteration is based on the window size and aggregation algorithm. Fig. 4-7 shows the error rate distribution over the aggregation iteration and window size plane based on the ADSW. From the figure, the best iteration number with the lowest error rate is related to the support window size. The cost aggregation with the smaller window size requires more iterations to achieve lower error rate. On the opposite, the aggregation with larger window size requires fewer iterations. Moreover, the area with lowest error rate exists only with larger window size. Hence, the performance with larger window size is better than smaller size.

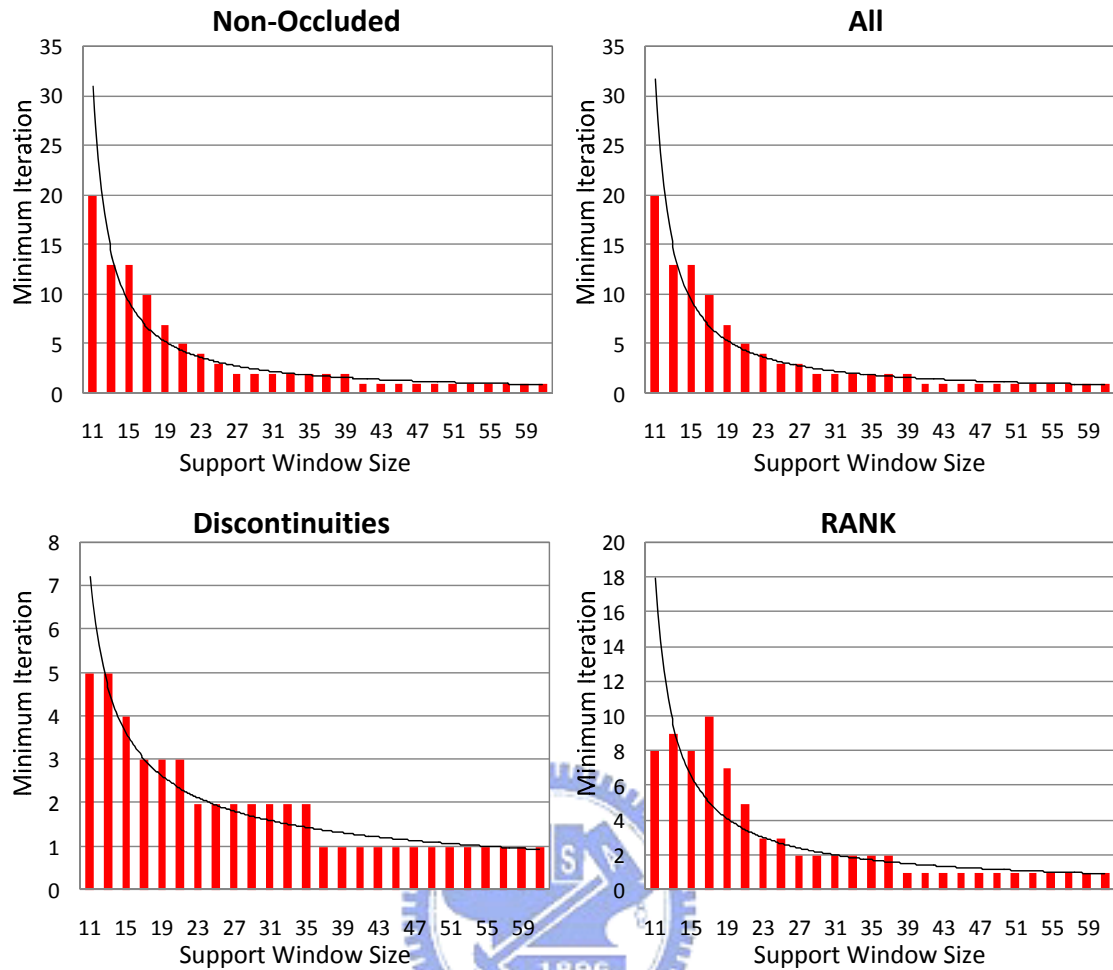


Fig. 4-8 the minimum iteration with different size of support window

Fig. 4-8 shows the minimum iteration to achieve the lowest error rate. The trend of the curve is also plotted on the figure. For the all evaluation regions and the rank, the minimum number of iteration is reduced while the window size increased. Note that if the window size is larger than 39, only one aggregation iteration is required to achieve the lowest error rate. However, it is tough for hardware design to adopt such a larger window size and more than one iteration. Hence, the design must trade some performance with this. As a result, the adopted window size and the number of aggregation iteration are 31 pixels and 1 respectively for this design. The performance is acceptable from Fig. 4-7 and Fig. 4-8.

4.6. Two-Pass Cost Aggregation Approximation

The window based cost aggregation sums up the cost over the support window with related weight. The process requires high computational resources. Fortunately, the process of window based aggregation is separable [44]. The original formula is written as equation (4.7). The separate aggregation is written as equation (4.8) and (4.9). The first aggregation is processed with vertical direction and the second aggregation is with the horizontal direction. The separate cost aggregation can reduce the computation complexity. For instance, if the window size is $(r+1) * (r+1)$ and the disparity range is D . The original complexity is proportional to $O(r^2D)$. For the separate aggregation, the complexity is proportional to $O(2rD)$. Besides, this approximation also helps reducing the internal bandwidth of the hardware design.

$$T(x, y, d) = \sum_{j=-r}^r \text{Cost}_t(x, y + j, d) \cdot \omega(x, y, 0, j) \quad (4.8)$$

$$\text{Cost}_{t+1}(x, y, d) = \sum_{i=-r}^r T(x, +i y, d) \cdot \omega(x, y, i, 0) \quad (4.9)$$

4.7. Overall Simulation Result

TABLE 4-4 the effect of different techniques

Method	ET	Error Rate %				Exec. Time(sec)
		TSUKUBA	VENUS	TEDDY	CONES	
Original	0	1.85	1.19	13.3	9.79	95.65
+MC+2P		3.47	0.91	14.3	11.2	4.75
+MC+2P+ Manhattan		3.08	0.59	14	10.1	3.12
+MC+2P+ Manhattan +Truc(64,2)		3.03	0.61	14	10.1	2.52
+MC+2P+ Manhattan+Truc(64,1)		3.06	0.66	13.9	10.1	1.84
Original	1	18.8	8.40	23.9	19.7	95.65
+MC+2P		12.2	7.95	21.4	18.0	4.75
+MC+2P+ Manhattan		11.1	7.22	21.7	16.8	3.12
+MC+2P +Manhattan +Truc(64,2)		11.0	7.22	21.6	16.8	2.52
+MC+2P +Manhattan +Truc(64,1)		11.2	7.17	21.4	16.7	1.84

5. Data Reuse Analysis of Hardware Implementation

5.1. Overview

External memory bandwidth and internal memory size have been major bottlenecks in designing VLSI architecture for real-time stereo matching hardware because of large amount of pixel data and disparity range. To address these bottlenecks, this chapter explores the impact of data reuse on disparity-order and pixel-order with the partial column reuse (PCR) and vertically expanded row reuse (VERR) techniques we proposed. The analysis result suggests that the disparity-order reuse with both PCR and VERR techniques is suitable for low memory cost and low external bandwidth design, whereas the pixel-order reuse with both techniques is more suitable for low computation resource requirement. However, the implementation of disparity-order requires high internal bandwidth. Hence, our final implementation adopted a hybrid of both the disparity-order and pixel-order reuse with VERR technique.

5.2. Architecture Overview

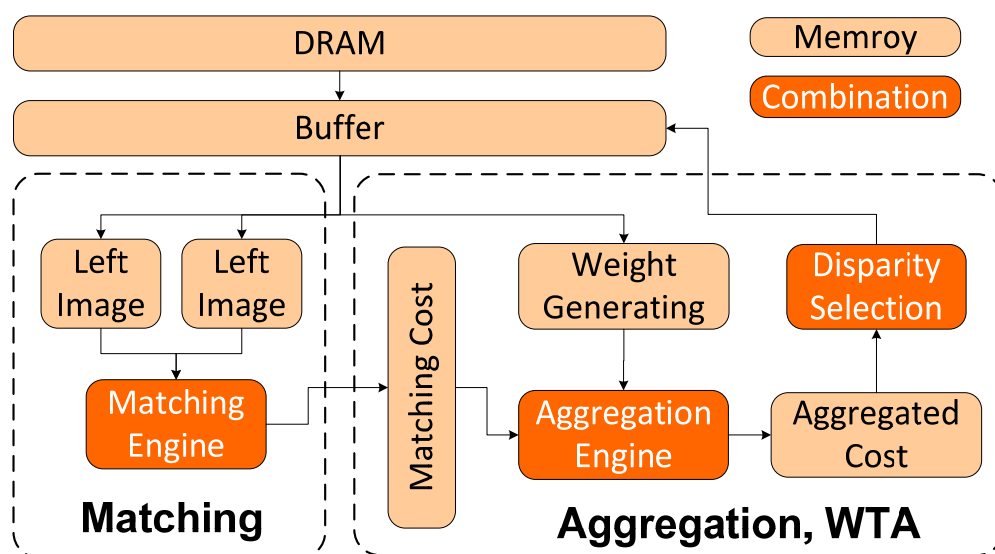


Fig. 5-1 the overview of hardware architecture

On implementing aggregation based method under real-time constraint, there are many solutions to the data reuse issue. We will use the hardware architecture shown in Fig. 5-1 to explain different solutions.

In the matching cost computation, if data reused along the disparity axis is preferred, the computation of all the matching costs of a pixel is computed before jumping to the next pixel. This allows the data within the matching cost support window to be reused. However, the cost aggregation sums the initial matching costs of the same disparity together, which would prefer the initial costs to be output along the spatial X-Y plane than the disparity axis. As a result, to compute the aggregated cost within an aggregation window, all the matching costs at each disparity must be stored before the aggregation can be performed. These initial matching costs form a cuboid in the disparity-spatial D-X-Y space. The volume of this cube represents the memory size needed to store the initial costs. One way to reduce the storage requirement is to avoid the conflict in data reuse direction. For instance, change the reuse direction in the matching cost computation to the X-Y plane so that it meets the processing direction in the cost aggregation. Although doing so removes the conflict between the matching cost computation and the cost aggregation, the conflict between the cost aggregation and the disparity computation exists. To determine the disparity of a pixel, the disparity computation needs to have all the aggregated matching costs at each disparity for that pixel. However, the aggregated costs are generated in the X-Y plane direction, which is different from the direction preferred by the disparity computation. Consequently, additional storage would be required to store the aggregated costs. These conflicts in the data generation and reuse directions play a key role in determining the storage requirement. Therefore, it is important to derive the best data reuse strategy which resolves these conflicts so that the storage requirement can be minimized.

5.3. Matching Cost Computation Reuse

The data reuse in the matching cost computation can be categorized into two types according to the reuse order. The details of these data reuse method are explained below.

5.3.1. Disparity-Order Reuse

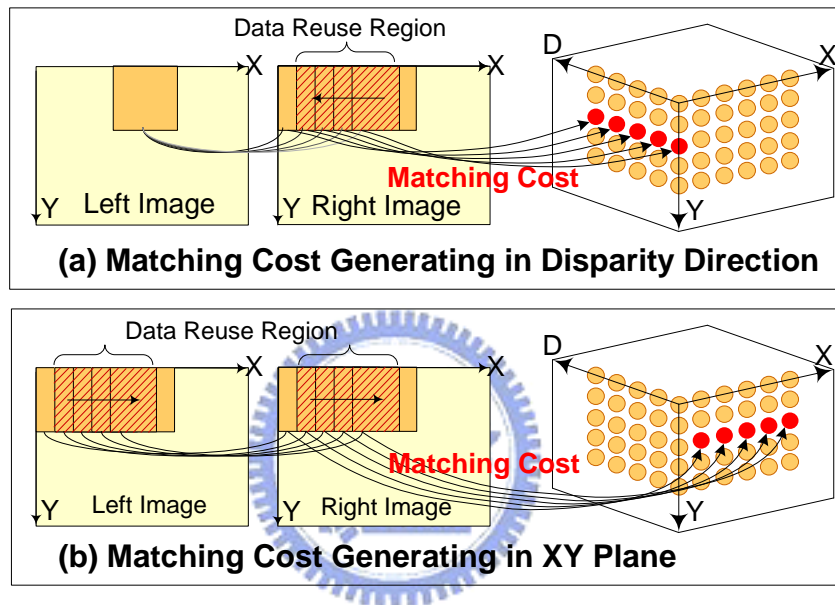


Fig. 5-2 the two data reuse directions with different size of support window

The disparity-order reuse reuses the data in the matching window of different disparities. Fig. 5-2(a) illustrates how disparity-order reuse works. When we compute the disparity of a pixel in the left image, the matching window in the right image would slide leftward within the disparity range. In other words, the matching cost of different disparities for a pixel in the left image is first computed. Then the matching cost computation of the next pixel in the left image is performed. With the disparity-order reuse, the overlapped data within the matching window in the right image shown in Fig. 5-2(a) can be reused to compute the matching cost at different disparities. As a result, if

the pixel data are stored in external memory, there is no need for repeating accesses of the overlapped pixels. Hence, the bandwidth requirement to external memory can be reduced. However, the order of matching cost generation is different from the order of the matching cost consumption in the following cost aggregation step. This would result in additional memory storage requirement.

5.3.2. Pixel-Order Reuse

Comparing to the disparity-order reuse, the pixel-order reuse reuses the data overlapped by the neighboring matching window in both left and right images. Fig. 5-2(b) illustrates the detail of the pixel-order reuse. The matching cost of the same disparity for each pixel is first computed. Then the cost of the next disparity for each pixel is computed. As a result, the matching window in the left and the right images both slides synchronously with the same disparity offset. With the pixel-order reuse, the overlapped data within the matching windows shown in Fig. 5-2(b) can be reused. Therefore, the pixel-order reuse can also reduce the external memory bandwidth requirement. In contrast to the disparity-order reuse, the order of matching cost generation is the same as the order of the cost consumed by the following cost aggregation step. Hence, the buffer size between the two steps can be reduced. However, the data reuse can only be exploited during the cost computation of one single disparity. There is no data reuse between the computations of different disparities. Once all the computation of the previous disparity has been completed for all the pixels in the whole image, pixel data have to be read from the external memory again. Unless all the previously read pixel data could be stored within the internal memory, otherwise repeating external memory accesses are inevitable.

5.4. Cost Aggregation Data Reuse

In addition to the data reuse in the matching cost computation, there are two data reuse methods in the cost aggregation. The details of these two data reuse methods are explained as follows.

5.4.1. Partial Column Reuse (PCR)

The partial column reuse method reduces the local memory size in the cost aggregation by distributing the computation of aggregated cost to each column. Instead of computing the aggregated cost after all the initial costs in an aggregation window are available, the PCR computes the partial sum of a column after the initial costs of this column are available. As a result, the size of the local memory can be reduced from a window to only one column. Moreover, the partial sum of each column can contribute to the aggregated cost of multiple overlapped windows. Storing partial column cost requires less local memory size than storing all the initial matching costs in a column.

Fig. 5-3 illustrates an example of the PCR with a 5x5 aggregation window size. An aggregated cost requires the partial sum of five initial cost columns. With the PCR, the current partial column sum in Fig. 3 can be reused to contribute to the aggregated cost of windows 1 to 5.

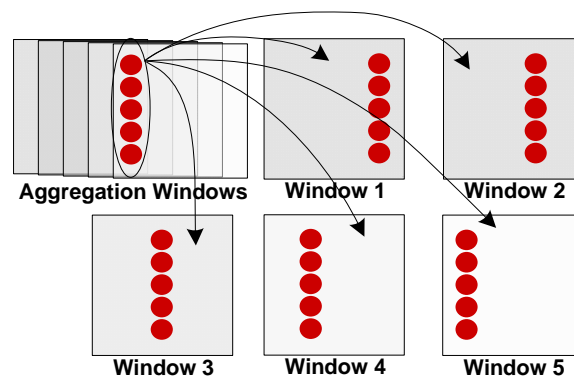


Fig. 5-3 The partial column reuse (PCR) in 5x5 aggregation window

5.4.2. Vertically Expanded Row Reuse (VERR)

The vertically expanded row reuse reduces the bandwidth requirement to the cost aggregation engine by deliberately access additional rows of initial costs. If there's no VERR, when the aggregation finishes processing the current row and jumps to the next row, the overlapped data between the windows at the previous row and the current row have to be read from the cost computation engine again. Fig. 4 shows an example of the situation that the data are overlapped. To avoid accessing the already accessed costs, the VERR vertically expand the rows of initial costs to be read so that they can be reused to compute multiple rows of aggregated cost.

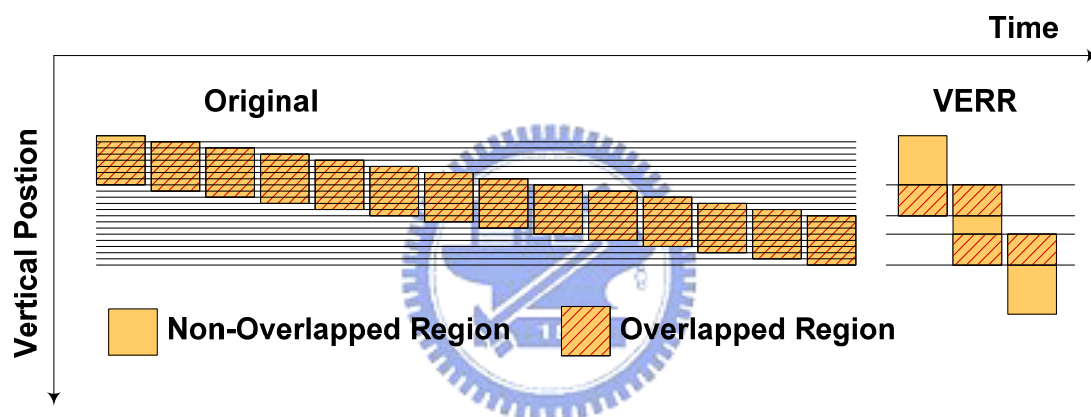


Fig. 5-4 Vertically Expanded row reuse(VERR)

Fig. 5-4 shows how VERR reduces redundant access of the overlapped data. Without the VERR, most of the data in the windows are overlapped for many times. Consequently, these overlapped data are read repeatedly multiple times. In contrast, with the VERR, the portion of overlapped data becomes much smaller than the case without the VERR. Moreover, the overlapped data in the VERR case only overlap once. This implies that with the VERR, the repeating accesses of the overlapped data would be fewer than the case without the VERR.

Fig. 5-5 plots the relationship between the average access count of an initial matching cost and the value k given an aggregation window size of 25×25 . The value k represents the number of expanded rows. It can be observed that the average access count decreases as k increases. This suggests that with more rows expanded, less bandwidth is needed. However, increasing the value of k will also increase the local memory size and computing resource requirement.

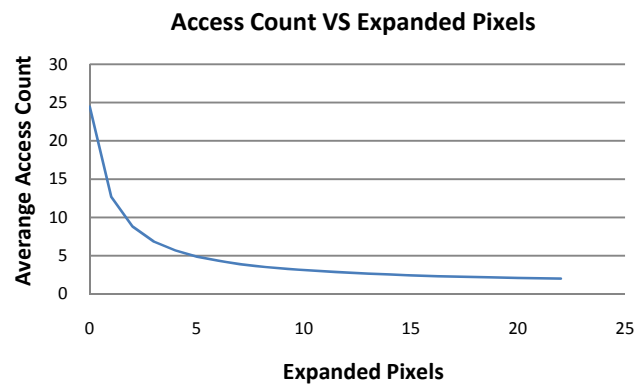


Fig. 5-5 The average access count versus the number of expanded pixel

5.5. Comparison



TABLE I compares the estimated memory size and bandwidth requirement of the disparity-order and pixel-order reuse methods. The target disparity image is 352×288 pixels large with 64 disparity levels. The real-time constraint is 30 fps. The architecture is assumed to operate at 100MHz clock with a 32-bit data port to the external memory. The size of support window in the matching cost computation and cost aggregation are 9×9 and 25×25 pixels respectively.

5.6. Summary

This chapter explores the impact of disparity-order and pixel-order data reuse in the matching cost computation and proposed the partial column reuse (PCR) and

vertically expanded row reuse (VERR) techniques for the cost aggregation. The analysis and comparison conclude that the architecture using the disparity-order reuse with both the PCR and VERR techniques is suitable for the design of low memory cost with high computation resource. On the other hand, the architecture using pixel-order reuse with VERR technique requires less computation resource, but needs large internal memory in storing the aggregated cost.

TABLE 5-1 the result of approximated color distance

Section	Property	Disparity-Order				Pixel-Order			
		Original	+PCR	+VERR	+PCR +VERR	Original	+PCR	+VERR	+PCR +VERR
Step 1	Internal Memory Size (KBytes)	2.4	2.4	2.6	2.6	2.2	2.2	2.4	2.4
	Bandwidth Requirement from External DRAM (MBytes/sec)	3.3	3.2	0.9	0.9	207.9	207.9	10.1	10.1
Step 2	Internal Memory Size (KBytes)	40.0	1.6	44.8	1.8	0.6	0	1.8	0.1
	Bandwidth Requirement from Cost Computation Engine (MBytes/sec)	158.7	158.7	44.3	44.3	158.7	158.7	<u>9.2</u>	<u>9.2</u>
Step 3	Internal Memory Size (KBytes)	0.1	0.1	0.1	0.1	228.1	0.0	228.1	228.1
Total	Internal Memory Size (KBytes)	42.5	<u>4.1</u>	47.6	4.5	230.9	2.2	232.3	230.5
	Bandwidth Requirement from External DRAM (MBytes/sec)	3.3	3.2	<u>0.9</u>	<u>0.9</u>	207.9	207.9	10.1	10.1
	Real-time Constraint (30 fps)	Meet	Meet	Meet	Meet	Fail	Fail	Meet	Meet

6. Hardware Implementation

6.1. Overview

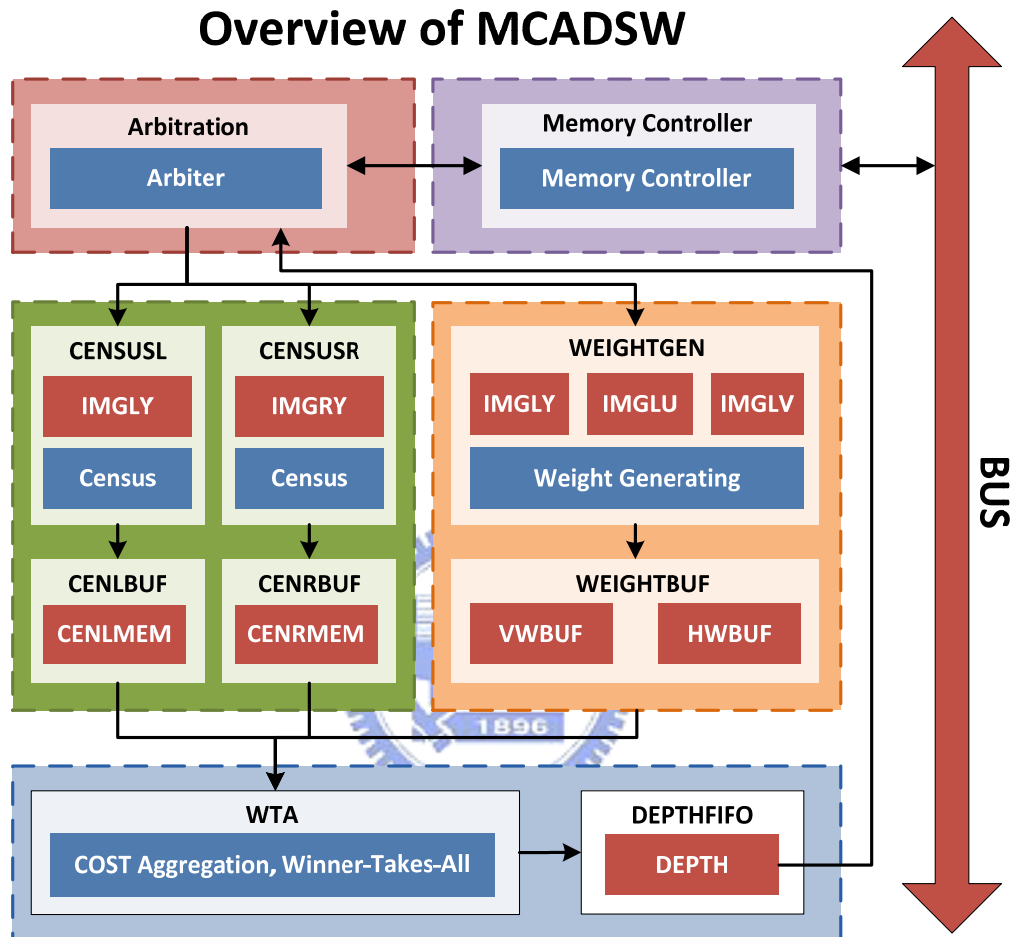


Fig. 6-1 the overview of the hardware design

The architecture of the design is shown in Fig. 6-1. The design of the MCADSW contains five major parts, which are the arbitration, memory controller, census transform, weight generation, and the cost aggregation. The memory controller communicates with the bus and the module granted by the arbiter. For the inside of each part, the blue block is the combinational logic and control of the finite-state-machine (FSM). The red block is the memory buffer used by each part. The detail will be discussed in the rest of this chapter.

6.2. Functional Block

This chapter introduces the details of the hardware implementation, including the input and output control, census transform, weight generating, aggregation, and winner-takes-all.

6.2.1. Mini-Census Transform

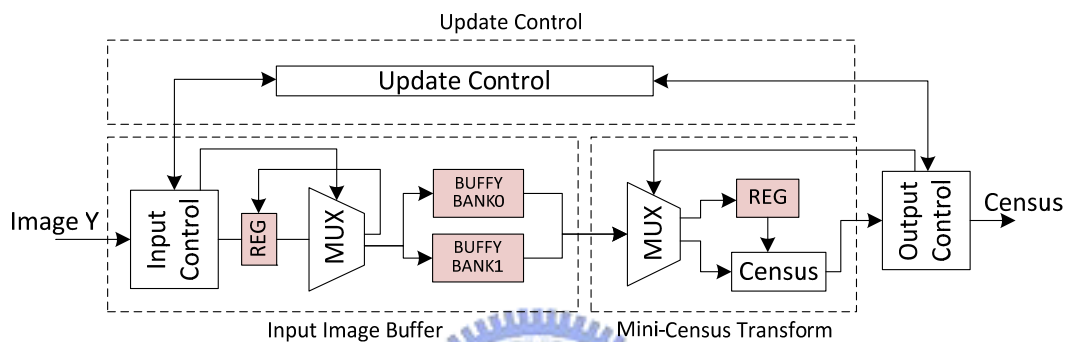


Fig. 6-2 the module of census transform for left and right image

Fig. 6-2 shows the architecture of the mini-census transform. This architecture contains three blocks: input image buffer, update control and mini-census transform. The mini-census transform compares 7 pixels distributed within 5x5 window to calculate one census result. The generation of one census result requires multiple loads from the input image data. Therefore, to reduce the times of data load from the input image, the input is buffered and reused. The input controller stores the input image in the register first. After one word of the data is stored in the register, it will be transferred to the memory buffer. The output control reads the data from the buffer to the register and census block. The register stores the data of center pixels, and the other pixels are transferred to census block. The census block compares the pixels to the center pixels, and then it generates the comparison result. This update control maintains the content of the memory buffer. The update control contains a table storing the validation for each

column of the memory buffer. The access of the memory buffer from input and output control is prohibited without checking the status of the validation table. This favors the synchronization between the input and output control.

6.2.2. Weight Generation

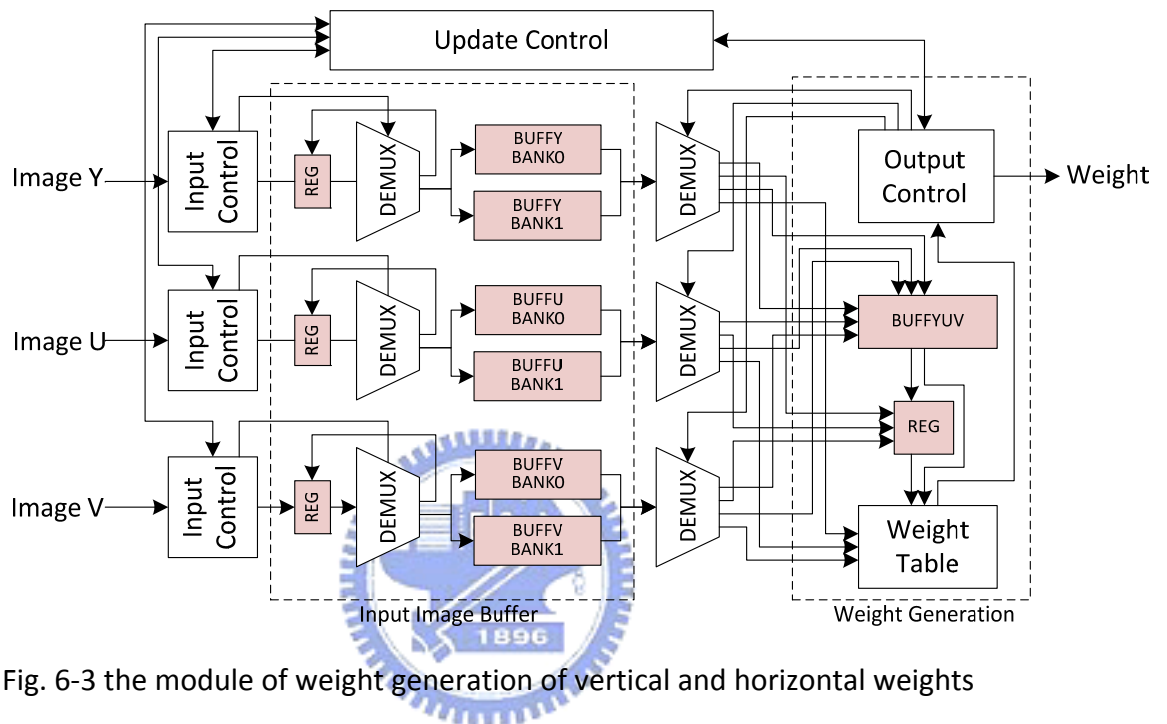


Fig. 6-3 the module of weight generation of vertical and horizontal weights

Fig. 6-3 shows the architecture of the weight generation. The architecture is similar to the architecture of census transform discussed in 6.2.1. However, there are two differences. The first difference is that the input control requires three dimension of color space. Therefore, there should be three input controls and three input buffers. The second difference is that there is additional buffer for output control, which is used for horizontal weight generation. The input control is similar to the one in census transform, only the address control and data size is slight different. After the input buffer is ready, the weight generation block starts to calculate the vertical weight and horizontal weight. The weight generation firstly loads the image data from the input image buffer to

generate the vertical weight by looking up the weight table. The input Y, U, and V images are also stored in the BUFFYUV during the generation of vertical weight. After the vertical weight is generated, the horizontal weight is generated by reading the buffer BUFFYUV.

6.2.3. Aggregation and Winner-Takes-All

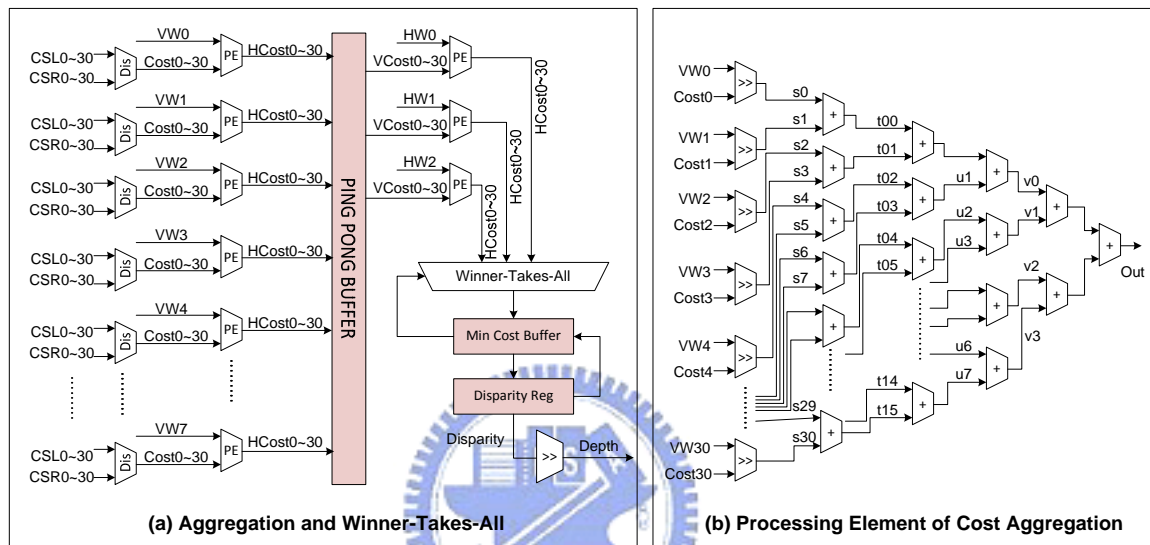


Fig. 6-4 the module of cost aggregation and its processing element

The Fig. 6-4(a) shows the architecture details of the aggregation and winner-takes-all(WTA). At first, the hamming distance is calculated by the left and right census results, which are CSL and CSR on the figure. The 0~30 hamming distances or called initial cost are sent to the processing element. And then the vertical aggregated cost is calculated by the summation of the shifted initial costs. Fig. 6-4(b) shows the detail of the PE. The initial costs are firstly shifted by the associated weights, and then they are summed together. The calculated vertical aggregated cost will be stored in a ping-pong buffer. The second pass aggregation reads the vertical aggregated cost from the ping-pong buffer. The same, the horizontal aggregated cost is shifted and summed. The final cost will be sent to the winner-takes-all block, which compares the

cost with the minimal cost. If the aggregated cost is smaller than the minimum cost, it will replace the minimum cost, and become the disparity candidate, which is stored in the disparity register. The final depth is the shifted disparity normalized to the range of the luminance.

Fig. 6-5 shows the detail of the ping-pong buffer in Fig. 6-4. There are 48 entries for each of the buffer. The figure shows the status of each entry. The color of white, light blue, deep blue and orange means that the entry is empty, being written, ready for reading, and being read respectively. At the first, all the entries are empty, and then the vertical aggregated cost is written into the buffer. After all the entries of the buffer 1 is all ready, the vertical weight will be written into the buffer 2. To generate three horizontal aggregated cost, 33 ready entries are required. Therefore, the vertical weight will be calculated after 40 entries are ready. After that, three entries will be cleared since the data are available anymore. The speed of update and consumption of the buffer are at balanced. Hence, the weight can be calculated continuously.

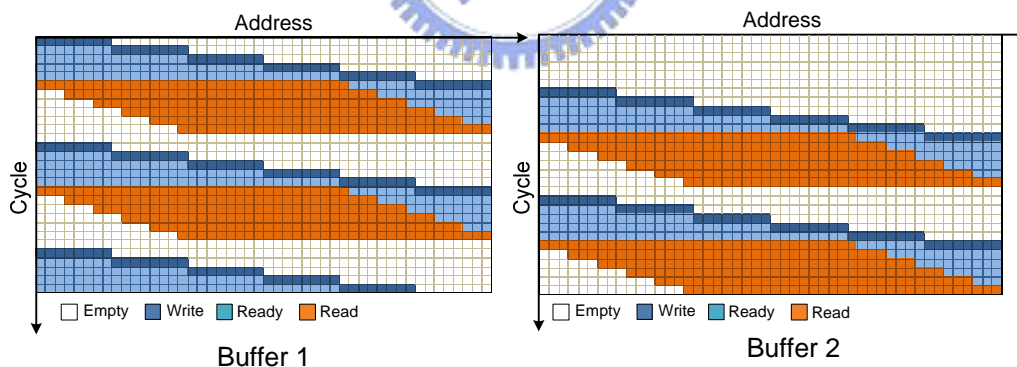


Fig. 6-5 the ping-pong buffer of cost aggregation module

6.2.4. Input and Output Control

Fig. 6-6 shows the concept of the input and output control used by most of the modules in this design. The control deals with the handshaking mechanism which will be discussed in 6.3. Firstly, the state is at WAIT state. The input control waits for the

update of invalid column of internal memory, which will be discussed in 6.5.1. Once the internal memory need an update, the input control sends the request signal to the transmitter, and wait for the data at the REQUEST state. The state changes to SEND state while receiving the data. After all, it will return to WAIT state after a transaction. On the opposite, the output control waits for the validation of internal memory at the WAIT state. Once is the internal memory is valid, it will send the ready signal to receiver, and waits for the request at the READY state. It will switch to SEND state once the request signal is received. The same, it returns to WAIT state after a transaction.

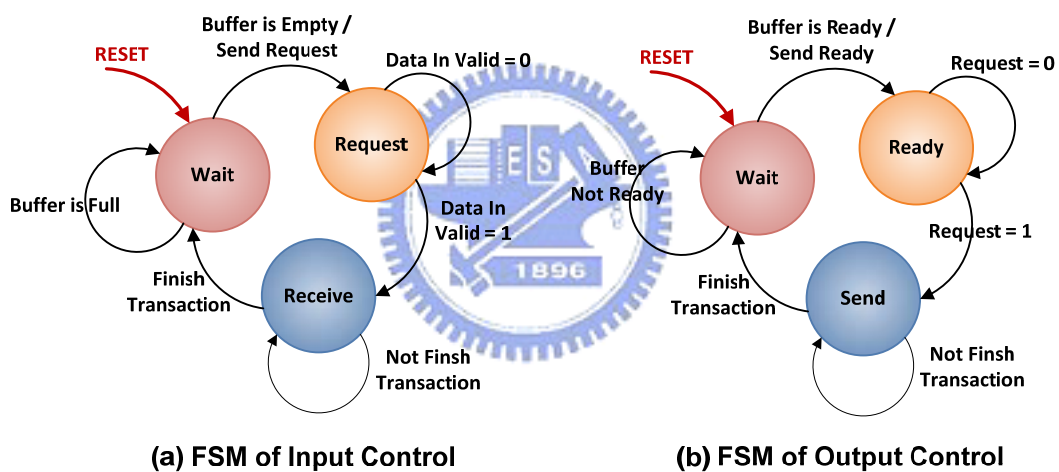


Fig. 6-6 the finite-state-machine of the input and output control

6.3. Handshaking

In this design, there are three handshaking mechanisms, request-valid, request-grant, and ready-request, which are shown in Fig. 6-7. The first request-valid is a one to one communication between two modules. For this mechanism, the receiver sends the request signal to transmitter, and then the transmitter sends a bunch of data with the valid signal. The request-valid mechanism is used for transmission of a bunch data between two modules. The second request-grant is also a one to one

communication mechanism. The mechanism of request-grant is that the transmitter sends the request to receive at first, and then the receiver sends grant signal to the transmitter. Once the transmitter receives one grant signal, it sends one data. The difference from request-valid is that the receiver is not guaranteed that it can receive a bunch of data continuously from the transmitter. Hence, the transmitter must wait for the receiver. The latest one is the ready-request, which is used for many to one data communication between several modules. The transmitters send the ready signals to the receiver. After all the ready signals are received by receiver, the receiver sends the request signals to the transmitters. Once any of the transmitters receives the request signal, it sends a bunch of data continuously for certain cycles, which are 384 cycles in this design. The usage of the ready-request is that it can be used to synchronize the data from different input path.

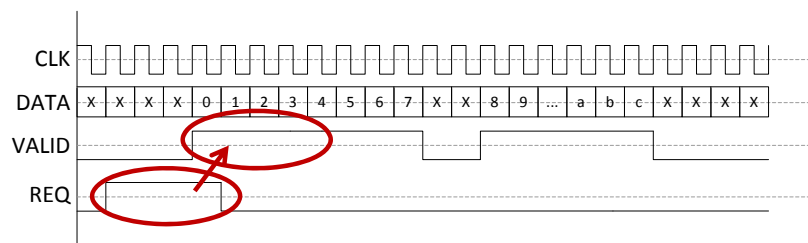
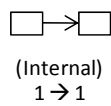
In Fig. 6-1, the handshaking between different modules follows the three handshaking mechanisms we just discussed. The handshaking mechanism between the input image buffer and the arbiter follow the request-valid since the communications between them are all one to one and there is no need for arbiter to wait for the input image buffer. The depth FIFO to the arbiter follows the second request-grant since it has to wait for the grant for each transmission from the arbiter. The latest ready-request mechanism is used between the census transform, weight generating and aggregation modules. In the aggregation module, the input data paths are from different modules. To the guarantee the synchronization of different input path, the ready-request mechanism is applied.

6.4. Arbitration

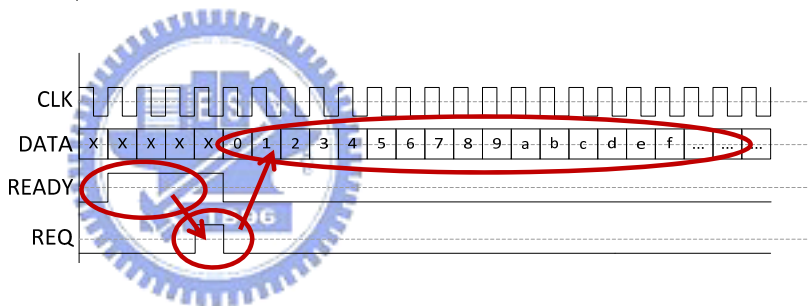
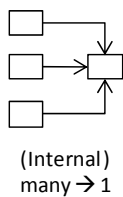
The arbitration of the system is based on the hybrid of round-robin and fixed

priority strategy. There are six modules sending the request to the arbiter to get grant of using the bus. The depth FIFO has a fixed and highest priority to use the bus due to the high penalty of suspending of the aggregation module. If the depth FIFO is full, the aggregation module, which is the kernel of the system, will be suspended. To avoid this suspension of kernel, the data of the depth FIFO must be written out as soon as possible. Hence, it always has the highest priority. As for the other five image buffers follow the round-robin strategy.

(1) Request -> Valid



(2) Ready -> Request



(3) Request -> Grant

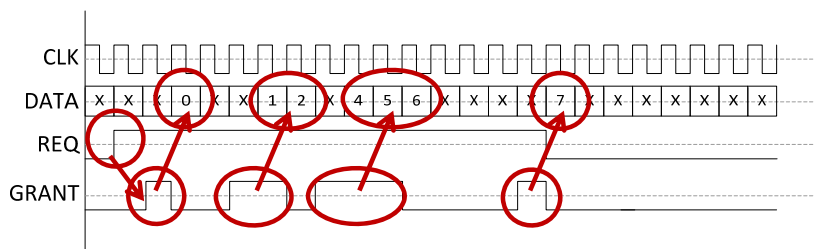
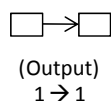


Fig. 6-7 the handshaking mechanism between different modules

Fig. 6-8 demonstrates the change of the priority with the time line. It can be observed that the depth FIFO always has the highest priority under any circumstances. For the other five buffers, the priority rotates if one of them receives the grant from the arbiter. Take Fig. 6-8 for example, the “CENSUS IMGLY” firstly gets the grant, the

priority of “CENSUS IMGLY” becomes the lowest of all at the next time. After the rotation, the next “CENSUS IMGLU” will get the highest priority of the five modules. However, if this module does not send the request, the priority will be also rotated for the grant of any module, except for two conditions. One is the module with the lowest priority gets the grant. The other one is the “DEPTH FIFO” gets the grant.

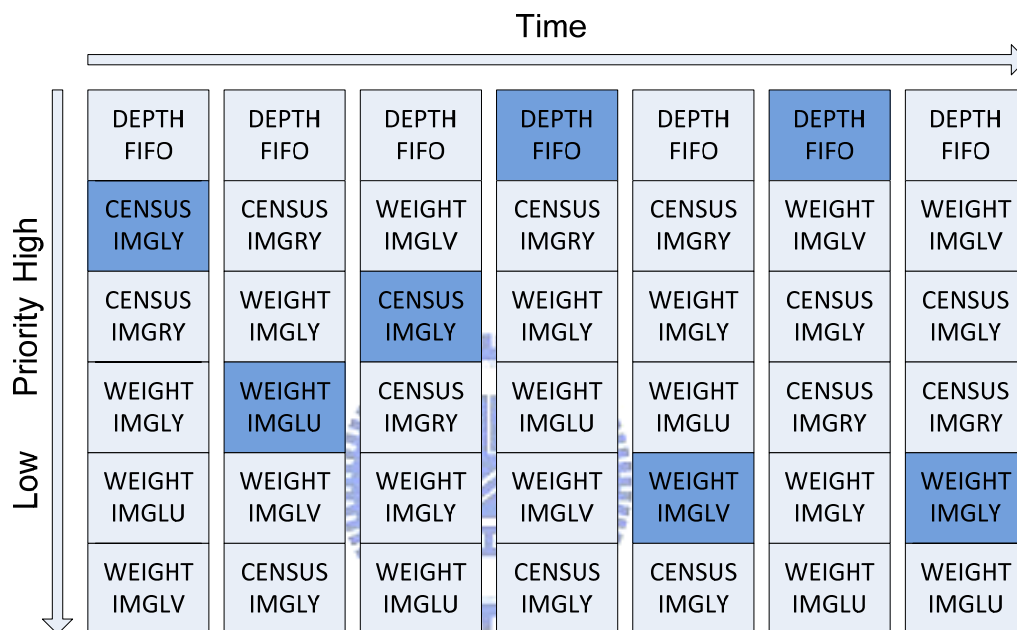


Fig. 6-8 the hybrid of round-robin and fixed priority arbitration strategy

6.5. Memory

6.5.1. Memory Update Mechanism

The update mechanism is a column based cyclic buffer shown in Fig. 6-9. The update is based on an update table which stores the status of each column of the memory. The status represents if the column of the memory is active or inactive. The active column is the column with the data which are being used. The inactive columns wait for the update. The set and clear pointer stores the set and clear position of the column.

Take the figure for example, the active columns 1~5 are being used. During the data processing of the active region, the column 6, 7, and 0 will be updated by order. After the processing window moves toward right direction with 2 columns, the column 1 and 2 will be cleared. This update mechanism works in this design due to the processing region moves orderly in horizontal direction. However, due to the real-time constraint, the implementation of this mechanism requires reading and writing the memory at the same time to speed up the memory update flow.

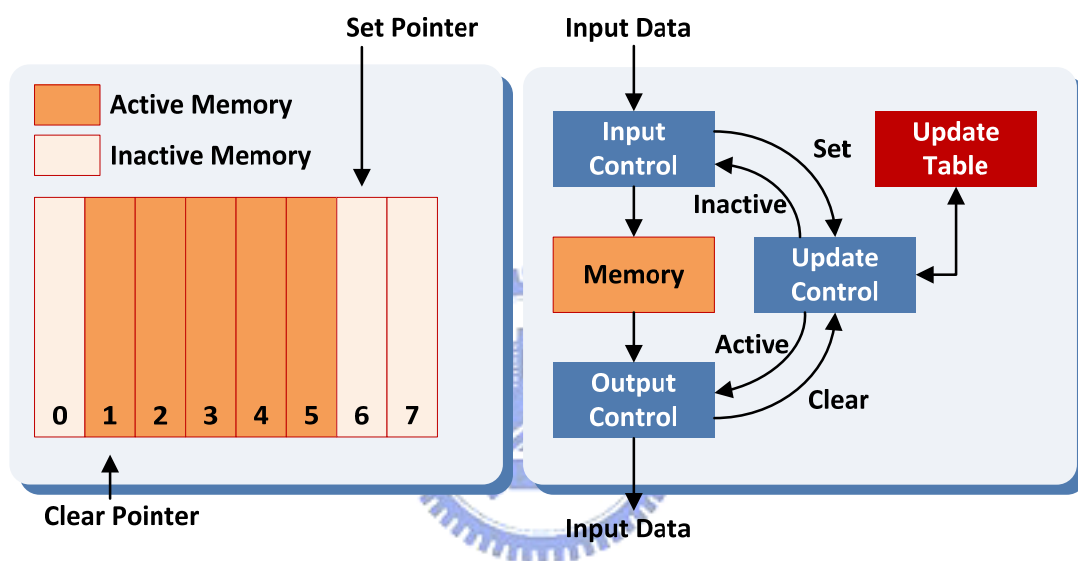


Fig. 6-9 the column based cyclic buffer update mechanism

6.5.2. Memory Size

The memory buffer size is one of the most important issues of this design and the size of it is according to the memory region used in different part shown in Fig. 6-10. In this figure, the block with the color blue, red, and light yellow represent the combinational logic, memory buffer and expanded memory buffer respectively. The memory buffer size is labeled inside the red block. The label represents the buffer size, and the height multiplied with width.

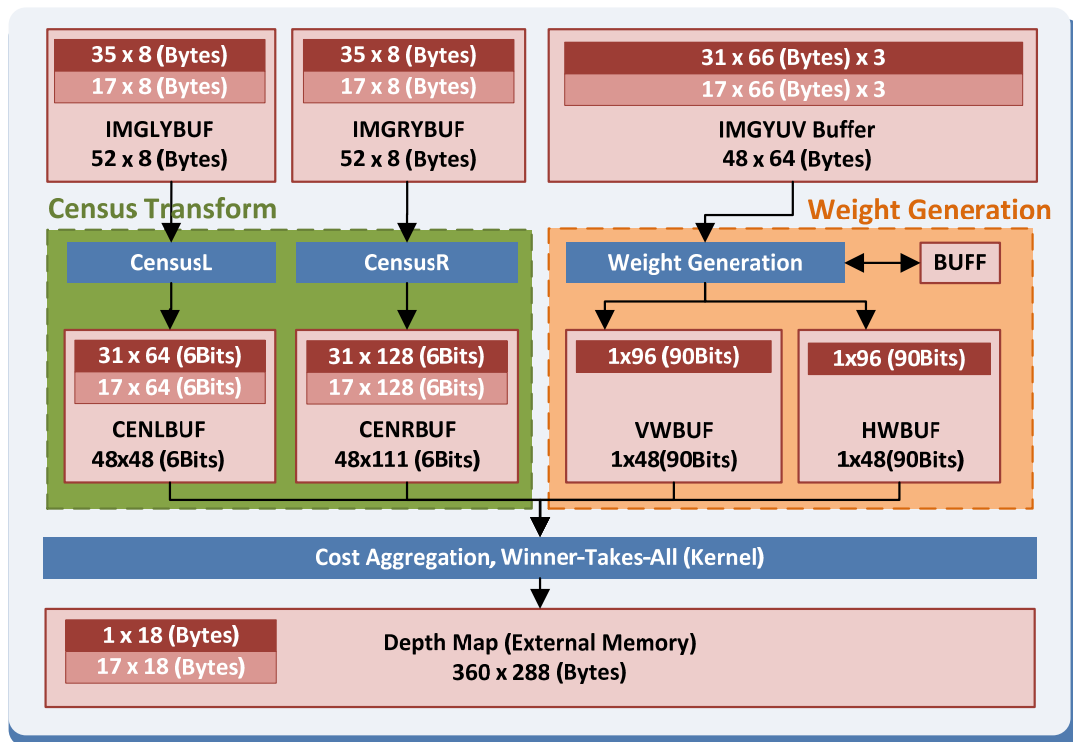
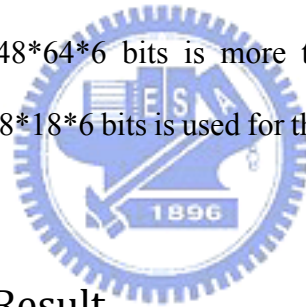


Fig. 6-10 the memory size of different module

The access region of memory buffer is based on the region of depth being calculated and the processing window of the combinational logic. To explain this figure more easily, the detail is discussed from the end to the start. At the right-bottom of the figure, the evaluation of the depth FIFO, the WTA requires 1x18 horizontal aggregated cost. The calculation of the horizontal aggregated cost requires 1x(18+30) vertical aggregated cost. Note that the width of horizontal aggregated cost is extended 30 elements than depth FIFO due to the processing window size is 1x31. Similarly, the Hamming distance will require (30+1)x48 elements due to the window size of vertical aggregation is 31x1. Moreover, the memory size is 63 times more than original size due to the disparity range. Fortunately, the huge memory access region of hamming distances is calculated on the fly. Hence, the memory buffer of the hamming distances does not exist; therefore, the problem will be pass-through to the former census buffer. And this will result in the extension of 63 elements in the censusR buffer. The

discussion of the rest of memory access region is almost the same from above discussion, except for the image buffer used by the census transform. The width of them is much smaller because of the short data life time. Therefore, the data can be discarded after the census transform and result in the reduction of the width.

Fig. 6-10 shows the final result of the memory size. The size is labeled inside the block. The result in this is based on the implementation, such as the memory bank, and the minimal words of the Register-File. The text in the red block is the memory size just discussed. In Fig. 6-10, the light yellow block is the extension for the vertically expanded rows reuse discussed in 5.4.2. Take the CENLBUF in census transform as an example. The $31 \times 64 \times 6$ bits is the memory size without VERR. The $17 \times 64 \times 6$ bits is the extension memory size after applying the VERR. Hence the overall size is $48 \times 64 \times 6$ bits. Note that the memory size $48 \times 64 \times 6$ bits is more than the required memory size $48 \times 48 \times 6$ bits. The additional $48 \times 18 \times 6$ bits is used for the run-time update of the column based cyclic buffer.



6.6. Implementation Result

Our design is targeted at CIF size, 64 disparity range and 30 frames per second. The clock rate of the system is 100MHz and the bus width is 32 bits. The implementation result will be discussed at the rest of this chapter.

6.6.1. External Bandwidth

Fig. 6-11(a) shows the simulation result of the execution cycle with different bus access latency. From the figure, the curve is divided into two segments, both of which are proportional to the access latency. However, the slopes of them are different. After the latency is larger than 5, the buffer for the input latency is not enough. Therefore, the

execution cycle increased faster. Fig. 6-11(b) shows the FPS with difference bus access latency. The system can achieve more than 40 FPS if the average access latency is smaller than 5.

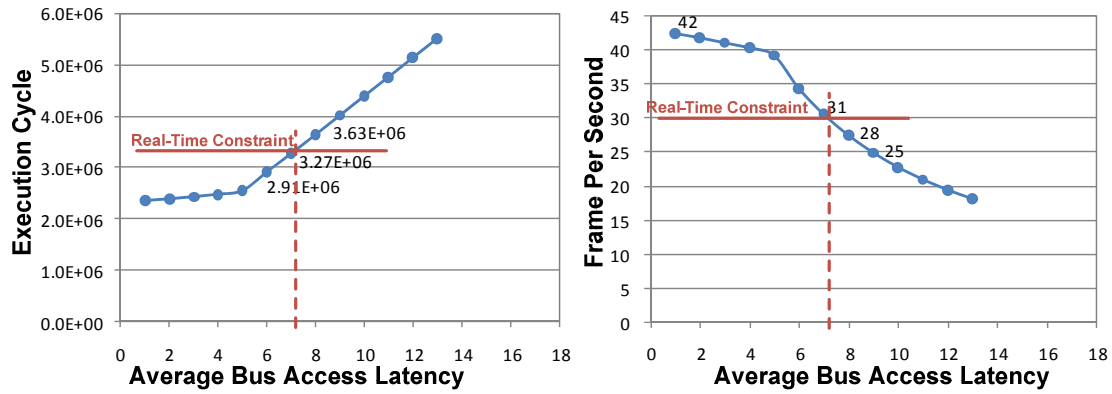


Fig. 6-11 the performance with the bus access latency

6.6.2. Area and Gate Counts

TABLE 6-1 the implementation result of area and gate counts

module name	total area	cell Area	memory size (Byte)	combinational gate count
Weight Generation	7,002,961	700,596	10,170	37,586
Weight Buffer	2,041,062	634,522	1,485	6,127
Census L	1,243,590	100,250	224	5,004
Census R	1,243,590	100,250	224	5,004
Aggregation+WTA	26,102,290	442,254	0	156,716
Arbiter	38,042	475	0	168
Census Lbuffer	26,226,681	1,842,059	4,608	171,842
Census Rbuffer	26,672,820	1,865,630	4,608	180,195
Total	90,571,035	5,686,036	21,319	562,642

The area and gate count of the simulation result is shown in TABLE 6-1 and Fig. 6-12. The result is synthesized with standard cell library of UMC 90 um. It can be observed that the aggregation, left census buffer, and right census buffer are dominated

in this design. The gate count of aggregation and census buffer are large due to the requirement of high computation resource and complex demultiplexing of the memory banks.

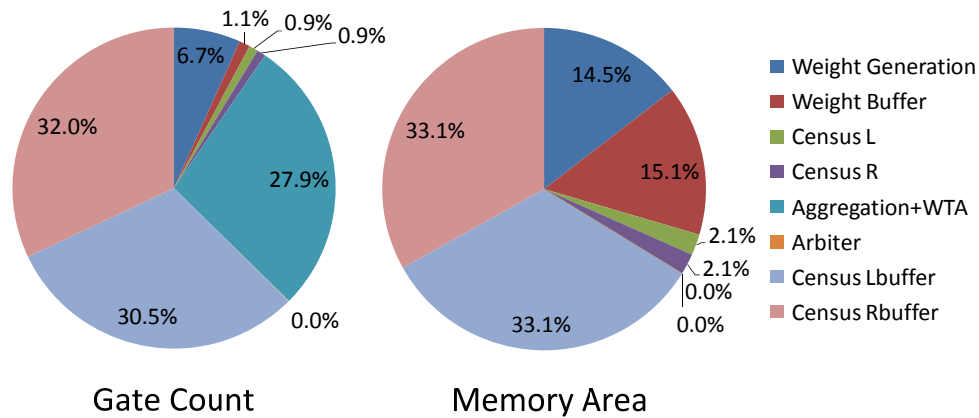


Fig. 6-12 the percentage of the memory area and combinational gate counts

6.7. Performance Result

TABLE 6-2, TABLE 6-3 shows the overall comparison of different implementation result. In TABLE 6-2 most of the implementations are using the programmable GPU. The programmable GPU favors high bandwidth and computation resource. The image size, disparity range, and FPS of all designs are quite different. It is difficult to compare difference implementations. Therefore, the million disparity evaluation (MDE) method has been used. TABLE 6-3 shows the error rate of different implementation result. The test sequences are from the middlebury vision website.

TABLE 6-2 the error rate comparison of different method

Design	Implementation	Image Size	Disparity Range	FPS	MDE/s
Proposed	Hardware	352x288	64	42	272.5
TrellisDP[45]	Hardware (FPGA)	320x240	128	30	294
HBP[43]	Hardware (FPGA)	320x240	32	30	73.7
EffectAggr [46]	Intel C2D 2.14 GHz	320x240 463x370	16 75	5 1.67	18.9
RealDP[35]	AthlonXP 2800	384x288	50, 100	33, 18.9	183, 209
CBiased[36]	Geforce 7900	512x512 256x256	64, 96 64, 96	35, 24 122, 87	588, 605 512, 548
SepLaplacian[37]	Geforce 7900	256x256 512x512	64, 96 64, 96	121, 87 38, 27	507, 547 637, 679
RealTimeBP[42]	Geforce 7900	320x240	16	16	19.6
RealTimeGPU[38]	Radeon 9800, P4 3GHz	320x240	16	16	19.6
ReliableGPU[34]	Radeon 9800	-	-	16.6	-
GradientGuided[24]	Radeon 9800XT	512x384	40	14.7	117

TABLE 6-3 the performance comparison of different method

Design	Publication	TSU	VEN	TED	CON	SAW	MAP
Proposed	-	2.80	0.64	13.7	10.1	2.11	3.21
TrellisDP[45]	MUE 07	2.63	3.44	-	-	1.88	0.91
HBP[43]	Lecture Notes	2.85	1.92	-	-	6.25	6.45
EffectAggr [46]	ICPR 08	2.96	3.53	10.7	4.92	-	-
RealDP[35]	CVPR 04	2.85	6.42	-	-	6.25	6.45
CBiased[36]	ICIP 07	4.77	10.2	-	-	0.82	0.65
SepLaplacian[37]	ICME 07	13.0	-	-	-	-	-
RealTimeBP[42]	BMVC 06	3.40	1.90	13.2	11.6	-	-
RealTimeGPU[38]	3DPVT 06	4.22	2.98	14.4	13.7	-	-
ReliableGPU[34]	CVPR 05	1.36	1.09	-	-	2.35	0.55
GradientGuided[24]	3DIM 05	2.48	3.91	-	-	1.63	0.73

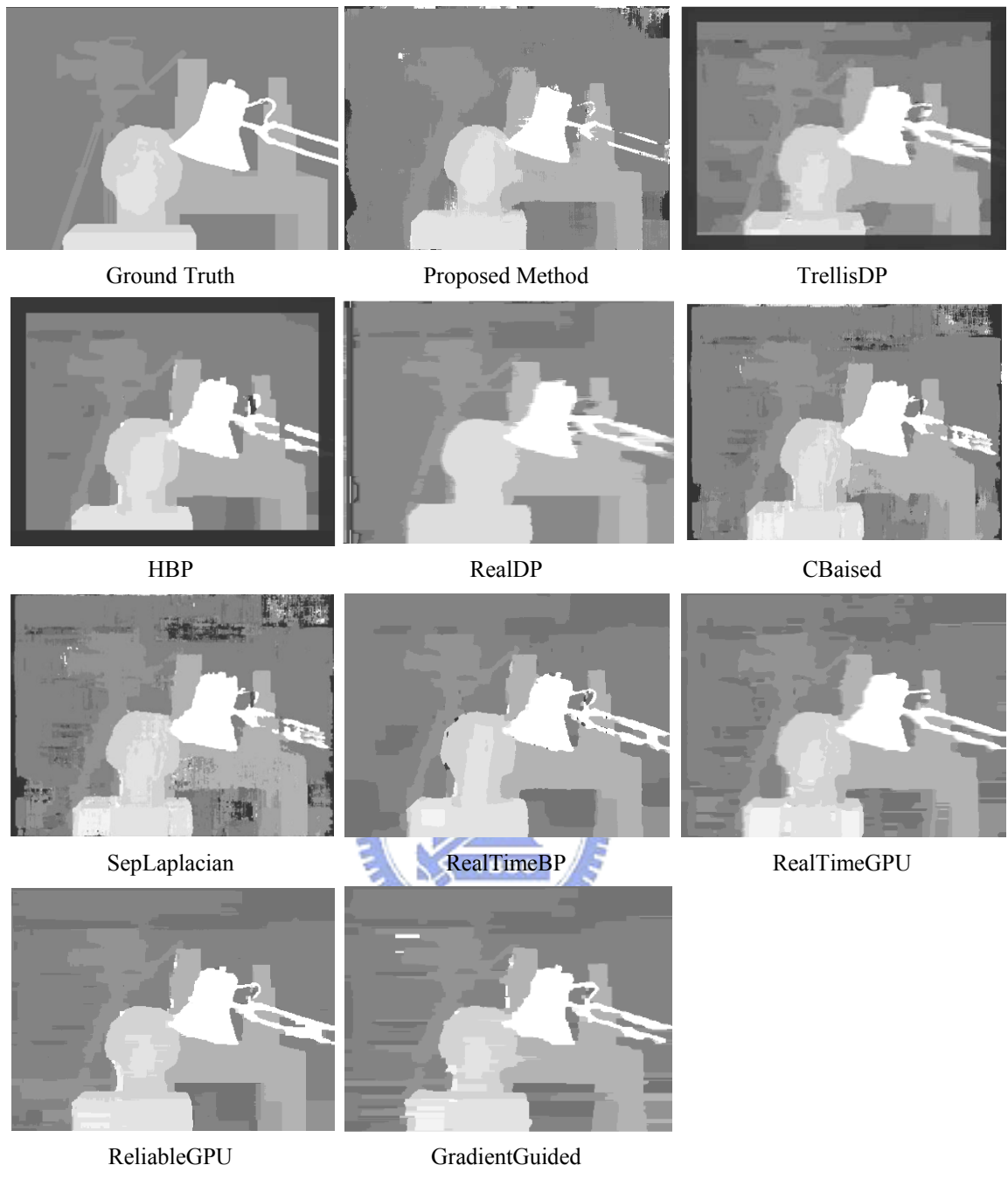
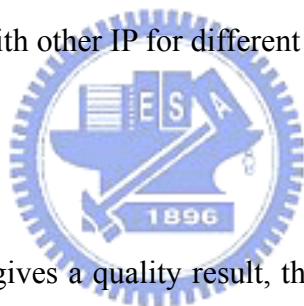


Fig. 6-13 the implementation result with different method

Conclusion

The main contribution of this thesis is to propose a hardware friendly algorithm and an architecture design for real-time local stereo matching. Our design gives a quality depth result for real-time application. The proposed algorithm reduces about 95.14% computation complexity comparing to the original ADSW, and the average quality drop with 1 disparity tolerance is about 0.515%. The implemented design can achieve 43 frames per second and 64 disparities with CIF image size under 100MHz clock rate. The chip consumes totally 562,642 K gate counts and 21.3K Bytes internal memory. Besides, we also consider the bandwidth issue in the system level. The final bandwidth requirement is only 45MB/s, which is about ninth of the total bandwidth, and can be easily integrated with other IP for different kinds of applications.

Future Work



Although our algorithm gives a quality result, the disparity map at the occluded area may be incorrect due to the lack of disparity refinement. Besides, the depth result may be unreliable if the object is tiny or lack of color information. On the other hand, the chip area is large and dominated by the large internal storage and multiple RAM banks. Therefore, the unreliable disparity map area and expensive cost of internal storage size may limit its application.

There are two issues remained in our work. First, the practicability for different applications needs to be investigated, such as the scene reconstruction and 3D-TV, which may require smooth depth on edge and occluded area. The second issue is the expensive cost of internal memory size. To reduce the internal memory size, there are three feasible plans, for example, decreasing the bits of census, truncating the

intermediate result of cost aggregation, and using memory with single port instead of dual port. However, the reduction of the memory area is still limited under the data reuse strategy of the proposed architecture. For a low memory cost implementation, further research for stereo algorithm or architecture is required.



Reference

- [1] P. Kauff, N. Brandenburg, M. Karl, and O. Schreer, "Fast hybrid block- and pixel recursive disparity analysis for real-time applications in immersive teleconference scenarios," in *Proceedings of 9 th International Conference in Central Europe on Computer Graphics Visualization and Computer Vision*, pp. 198-205, 2001.
- [2] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, "A stereo machine for video-rate dense depth mapping and its new applications," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 1996.
- [3] M.Z. Brown, D. Burschka, and G. Hager, "Advances in Computational Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no.8, pp. 993-1008, August 2003.
- [4] 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.
- [5] H. Hirschmuller and D. Scharstein, "Evaluation of Cost Functions for Stereo Matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 17-22 June 2007
- [6] R. Zabih and J. Woodfill, "Non-parametric Local Transforms for Computing Visual Correspondence," in *Proceedings of third European Conference on Computer Vision*, vol. 2, pp. 151-158, 1994.
- [7] G. Egnal, "Mutual information as a stereo correspondence measure," *Computer and Information Science, University of Pennsylvania, Philadelphia, USA, Tech. Rep. MS-CIS-00-20*, 2000.

- [8] M. Hariyama, H. Sasaki, and M.Kameyama, "Architecture of a stereo matching VLSI processor based on hierarchically parallel memory access," *The 2004 47th Midwest Symposium on Circuits and Systems*, vol 2, pp. II245- II247, 2004.
- [9] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," *International Journal of Computer Vision*, vol. 7, pp. 143-162, 1992.
- [10] M. Hariyama, T. Takeuchi, and M. Kameyama, "Reliable stereo matching for highly-safe intelligent vehicles and its VLSI implementation," in *Proceedings of the IEEE Intelligent Vehicles Symposium. IV*, pp. 128-133, 2000.
- [11] P. B. Chou and C. M. Brown, "The theory and practice of Bayesian image labeling," *International Journal of Computer Vision*, vol. 4, pp. 185-210, 1990.
- [12] H. Tao, H. S. Sawhney, and R. Kumar, "A global matching framework for stereo computation," *Proc. Int'l Conf. Computer Vision*, vol. 1, pp. 532-539, 2001.
- [13] A. F. Bobick and S. S. Intille, "Large Occlusion Stereo," *International Journal of Computer Vision*, vol. 33, pp. 181-200, 1999.
- [14] S. B. Kang, R. Szeliski, and J. Chai, "Handling Occlusions in Dense Multi-View Stereo," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001.
- [15] K.J. Yoon and I.S. Kweon, "Adaptive Support-weight Approach for Correspondence search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.
- [16] M. Gerrits, and P. Bekaert, "Local Stereo Matching with Segmentation-based Outlier Rejection," in *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision*, pp. 66-66, 07-09 June 2006.
- [17] F. Tombari, S. Mattoccia, and L. Di Stefano, "Segmentation-Based Adaptive Support for Accurate Stereo Correspondence," *Lecture Notes in Computer Science*, vol. 4872, p. 427, 2007.

- [18] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, June 24-26, 2008
- [19] ISO/IEC JTC1/SC29/WG11 N6501, "Requirements on Multi-view Video Coding," Redmond, USA, July 2004.
- [20] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol.1, pp. I-556-I-561
- [21] S. Kang, R. Szeliski, and J. Chai, "Handling occlusions in dense multi-view stereo," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 103–110, 2001.
- [22] H. Hirschmuller, P. R. Innocent, and J. Garibaldi, "Real-Time Correlation-Based Stereo Vision with Reduced Border Errors," *International Journal of Computer Vision*, vol. 47, pp. 229-246, 2002.
- [23] S. Chan, Y. Wong, and J. Danie, "Dense stereo correspondence based on recursive adaptive size multi-windowing," *Image and Vision Computing New Zealand*, pp. 26-28, 2003.
- [24] M. Gong and R. Yang, "Image-gradient-guided real-time stereo on graphics hardware," in *Proceedings of Fifth International Conference on 3-D Digital Imaging and Modeling*, pp. 548-555, 2005.
- [25] C. Demoulin and M. Van Droogenbroeck. "A method based on multiple adaptive windows to improve the determination of disparity maps," in *Proceedings of IEEE Workshop on Circuit, Systems and Signal Processing*, pp. 615–618, 2005.
- [26] Y. Boykov, O. Veksler, and R. Zabih, "A variable window approach to early vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1283-1294, 1998.

- [27] J. C. Kim, K. M. Lee, B. T. Choi, and S. U. Lee, "A Dense Stereo Matching Using Two-Pass Dynamic Programming with Generalized Ground Control Points," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005.
- [28] M. Okutomi, Y. Katayama, and S. Oka, "A Simple Stereo Algorithm to Recover Precise Object Boundaries and Smooth Surfaces," *International Journal of Computer Vision*, vol. 47, pp. 261-273, 2002.
- [29] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE transactions on pattern analysis and machine intelligence*, vol. 7, pp. 139-154, 1985.
- [30] S. Roy and I. J. Cox, "A Maximum-Flow Formulation of the N-Camera Stereo Correspondence Problem," in *Proceedings of the Sixth International Conference on Computer Vision*, 1998.
- [31] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1222-1239, 2001.
- [32] 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*, pp. 1124-1137, 2004.
- [33] H. Hirschmuller, "Improvements in real-time correlation-based stereo vision," *IEEE Workshop on Stereo and Multi-Baseline Vision*, pp. 141-148, 2001.
- [34] G. Minglun and Y. Yee-Hong, "Near real-time reliable stereo matching using programmable graphics hardware," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol.1, pp. 924-931, 2005.
- [35] S. Forstmann, Y. Kanou, O. Jun, S. Thuering, and A. Schmitt, "Real-Time Stereo by using Dynamic Programming," in *Proceedings of Computer Vision and Pattern Recognition Workshop on Real-Time 3D Sensor and Their Use*, , 2004, pp. 29-29, 2004.

- [36] L. Jiangbo, G. Lafruit, and F. Catthoor, "Fast Variable Center-Biased Windowing for High-Speed Stereo on Programmable Graphics Hardware," in *Proceedings of IEEE International Conference on Image Processing*, pp. VI - 568-VI – 571, 2007
- [37] L. Jiangbo, S. Rogmans, G. Lafruit, and F. Catthoor, "Real-Time Stereo Correspondence using a Truncated Separable Laplacian Kernel Approximation on Graphics Hardware," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 1946-1949, 2007.
- [38] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pp. 798-805, 2006.
- [39] K. Konolige, "Small Vision Systems: Hardware and Implementation," in *Proceedings of Eighth Int'l Symp. Robotics Research*, Oct. 1997.
- [40] N. Chang, T. M. Lin, T. H. Tsai, Y. C. Tseng, and T. S. Chang, "Real-Time DSP Implementation on Local Stereo Matching," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 2090-2093, 2007.
- [41] M. Hariyama, T. Takeuchi, and M. Kameyama, "VLSI processor for reliable stereo matching based on adaptive window-size selection," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 2001.
- [42] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time global stereo matching using hierarchical belief propagation," in *Proceedings of The British Machine Vision Conference*, 2006.
- [43] S Park, C Chen, and H Jeong. "VLSI Architecture for MRF Based Stereo Matching," *Lecture Notes in Computer Science*, vol.4599, no., pp.55-64 2007

- [44] M. Gong, R. Yang, and L. Wang, "A Performance Study on Different Cost Aggregation Approaches Used in Real-Time Stereo Matching," *International Journal of Computer Vision*, vol. 75, pp. 283-296, 2007.
- [45] S. Park, H. Jeong, K. Pohang, and S. Korea, "Real-time Stereo Vision FPGA Chip with Low Error Rate," *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering*, pp. 751-756, 2007.
- [46] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. "Near real-time stereo based on effective cost aggregation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.



作者簡歷

姓名：蔡宗憲

籍貫：台北市

學歷：

台北市立建國高級中學 (民國 88 年 09 月 ~ 民國 91 年 06 月)

國立交通大學電子工程學系 學士 (民國 91 年 09 月 ~ 民國 95 年 06 月)

國立交通大學電子所系統組 碩士 (民國 95 年 09 月 ~ 民國 97 年 09 月)

著作：

國內會議

- [1] T. H. Tsai, Y. C. Chang, and T. S. Chang, "Hierarchical Decision Table for Bad Pixel Detection in Stereo Vision" in *Proceedings of VLSI Design/CAD Symposium*, Spring 2007.
- [2] T. H. Tsai, Y. C. Chang, Y. C. Tseng, and T. S. Chang, "Census diffusion with segmentation constraint for disparity estimation in stereo vision," in *Proceedings of Computer Vision, Graphics, and Image Processing (CVGIP)*, Aug. 2007.

國際會議

- [3] N. Chang, T.M. Lin, T.S. Tsai, Y.C. Tseng, and T.S. Chang, "Real-Time DSP Implementation on Local Stereo Matching," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp.2090-2093, 2-5 July 2007
- [4] T.S. Tsai, N.Y.-C. Chang, and T.S. Chang, "Data reuse analysis of local stereo matching," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp.812-815, 18-21 May 2008