

國立交通大學

電子工程學系 電子研究所

博士論文

適用於高畫質立體電視應用之視差估測設計研究



The Study of Disparity Estimation Design for High Definition 3DTV
Applications

研究生：曾宇晟

指導教授：張添烜 教授

中華民國一百年八月

適用於高畫質立體電視應用之視差估測設計研究

The Study of Disparity Estimation Design for High Definition 3DTV Applications

研究生：曾宇晟

Student : Yu-Cheng Tseng

指導教授：張添烜

Advisor : Tian-Sheuan Chang

國立交通大學



A Dissertation

Submitted to Department of Electronics Engineering and
Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electronics Engineering

August 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年八月

適用於高畫質立體電視應用之視差估測設計研究

學生：曾宇晟

指導教授：張添烜

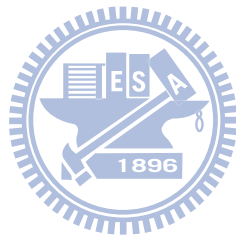
國立交通大學電子工程學系暨電子所博士班

摘要

隨著立體電視的問世，人們可以藉由立體視訊獲得新的視覺經驗。立體視訊可以立體攝影機擷取，並經由影像處理技術運算後，可支援多視角與自由視點之立體電視應用。在立體視訊的處理中，視差估測為最重要的技術之一。視差估測可產生拍攝場景之視差圖，可用於虛擬視角視訊的合成。動態影像壓縮標準組織的立體視訊編碼團隊已提出目前最先進視差估測演算法。其演算法可針對立體電視的應用產生高品質的視差圖，但因採用圖形切割演算法導致高運算複雜度與低平行運算的問題。特別對於高畫質視訊，其問題更為嚴重。

為解決以上問題，本論文首先提出初階視差估測演算法，採用訊息傳遞演算法以提高視差估測的運算平行度，並搭配聯合雙邊上取樣演算法以減少運算的畫面大小。其硬體設計面臨之問題，可藉由所提出之硬體架構方法解決。以此初階演算法為基礎，我們進一步提出一高品質視差估測演算法，可改善時間軸一致性與遮蔽之問題，並產生高品質的視差圖。針對高品質視差演算法，我們提出適用於不同實作方法的二快速視差估測演算法。針對軟體程式設計，所提出的稀疏運算之快速演算法可藉由時間軸與空間軸的分析選擇稀疏像素，僅針對稀疏像素更新視差值，達到降低運算時間至 62.9%。另一方面，針對超大型積體電路設計，所提出的高硬體效率之快速演算利用新的比對資訊擴散方法可降低運算時間至 57.2%，並大幅降低原演算的記憶體成本至 0.00029%。客觀評比的結果顯示針對虛擬視角視訊合成之應用，我們所提出的演算法可達到近於現今最先進演算法的高品質。

最後，我們化簡高硬體效率之快速演算法，進而提出高輸出效能的架構設計。其硬體實作結果顯示所提出的視差估測引擎可支援視差範圍 128，同時產生三視角 HD1080p 視差圖，並達到每秒 95 畫面的輸出速度，也就是每秒 75.64G 像素視差。總言之，本論文所提出的視差估測設計可滿足高畫質度立體電視應用的需求。



The Study of Disparity Estimation Design for High Definition 3DTV Applications

Student: Yu-Cheng Tseng

Advisor: Dr. Tian-Sheuan Chang

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

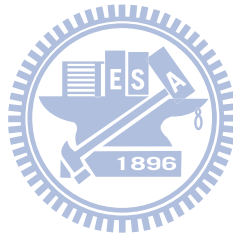
ABSTRACT

With emerging 3DTVs, human can have new visual experience from 3D videos that can be captured by new stereo camera and further processed by image processing techniques for the 3DTV applications of multi-view or free viewpoint. In the 3D video processing, one of the most important techniques is the disparity estimation that could generate disparity maps for synthesizing virtual-view videos. The state-of-the-art disparity estimation algorithm proposed by the MPEG 3D Video Coding team could deliver high-quality disparity maps, but suffers from high computational complexity and low parallelism due to its graph-cut algorithm, especially for high definition videos.

To address the problems, this dissertation first proposes the baseline disparity estimation algorithm that adopts the belief propagation algorithm to increase the parallelism of disparity estimation, and the joint bilateral upsampling algorithm to reduce the computational resolution. Their design challenges could be solved by our proposed architectural design methods. Based on the baseline algorithm, we further propose the high-quality algorithm that could well improve the temporal consistency and occlusion problems, and deliver high performance disparity maps. To accelerate the high-quality algorithm, we propose the two fast algorithms for different implementation method. The sparse-computation fast algorithm could decrease the processed pixels in the spatial and temporal domains to reduce the execution time to 62.9% for the software implementation. On the

other hand, for the hardware implementation, we propose the hardware-efficient fast algorithm that could reduce the execution time of high-quality algorithm to 57.2%, and decrease the memory cost of belief propagation to 0.00029% by the proposed cost diffusion method. The objective evaluation results show that our disparity quality is similar to the quality of state-of-the-art algorithm for view synthesis applications.

Moreover, we further simplify the hardware-efficient algorithm and propose a high-throughput architectural design. The implementation results shows that the proposed disparity estimation engine could achieve the throughput of 95 frames/s for three view HD1080p disparity maps with 128 disparity levels (i.e. 75.64G pixel-disparities/s). It could satisfy the requirement of high definition 3DTV applications.



謝誌

從大學四年到研究所五年，交大給予我許多成長與回憶。在此，藉由博士論文的完成以感謝所有的人。首要感謝的是指導老師張添烜教授，自大三的專題研究至博士班研讀，不論在研究方法、論文撰寫與投稿皆給予我耐心的指導與建議。接著要感謝王聖智教授，在我大學四年級時指導我和嘉賓進行色盲專題研究，更推薦我得以逕讀博士班，並擔任我的博士學位口試委員。另外也感謝其他口試委員，包含楊家輝教授、李鎮宜教授、杭學鳴教授、蔣迪豪教授、蔡宗漢教授及林嘉文教授，願意撥空給予指導。

工程四館 427 實驗室是我博士班在交大停留最久的地方，首要感謝的是張彥中學長，教導我良好的研究方法與態度，並引領我進入博士論文的研究題目。也感謝作最久實驗室同學的國龍，在實驗室的五年裡與我分享研究及生活，並一同朝著取得博士學位努力。接著要感謝實驗室的學長們：佑昆、朝鐘、君偉、裕仁、錦木、國亘、旻奇、子筠、嘉俊、英澤、得瑋、私璟，傳授我硬體設計基本觀念，並營造實驗室和樂的氣氛。當然也感謝實驗室的同學：宗憲、景竹、瑋呈、瑋城，因為有你們實驗室總是歡笑不斷。另外，感謝和我一起合作的實驗室學弟妹們：之悠、博淵、政君、孟維、筱珊、博雄、奕君、瑩蓉、宥辰、元歆、英佑、克嘉、亮齊、孟勳。

最後要感謝我的家人和女友，從攻讀博士班的決定，資格考的準備，期刊論文審稿的等待，到博士論文的撰寫與口試，一路上有你們的支持與陪伴使我能夠取得此學位。

此博士論文獻給以上所有感謝的人。

Table of Contents

摘要.....	i
ABSTRACT	iii
謝誌.....	v
List of Symbols.....	xi
I Introduction	1
1.1 Background.....	1
1.2 Motivation	1
1.3 Contribution.....	2
1.4 Dissertation Organization	3
II Background	5
2.1 Disparity Estimation	5
2.1.1 Epipolar Geometry.....	5
2.1.2 General Algorithm Flow	7
2.2 View Synthesis	19
2.2.1 Warping.....	20
2.2.2 Blending.....	20
2.2.3 Hole Filling.....	21
2.3 Review of DERS Algorithm from 3DVC	22
2.3.1 Input and Output View Configuration	22
2.3.2 DERS Algorithm.....	23
2.3.3 Reference Software for 3-View Configuration	26
2.3.4 Evaluation Method for Disparity Quality	27
2.3.5 Design Challenges	31
2.4 Summary.....	33
III Baseline Disparity Estimation with Belief Propagation and Joint Bilateral Filter for High Definition 3DTV Applications	34
3.1 Introduction	34
3.1.1 Baseline Belief Propagation	34
3.1.2 Joint Bilateral Upsampling	36
3.2 Analysis and Design of Baseline Belief Propagation	38
3.2.1 Analysis of Belief Propagation	38
3.2.2 Proposed Low Memory Cost Access Approach	41
3.2.3 Proposed Efficient PE.....	50

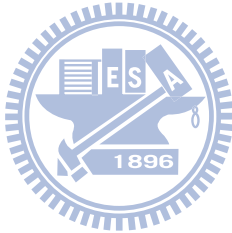
3.2.4	Implementation Result	54
3.3	Analysis and Design of Joint Bilateral Filtering	58
3.3.1	Related Acceleration Approaches.....	58
3.3.2	Analysis of Integral Histogram Approach.....	62
3.3.3	Proposed Memory Reduction Methods	66
3.3.4	Proposed Architecture	69
3.3.5	Implementation Result	75
3.4	Baseline Disparity Estimation Algorithm.....	77
3.4.1	Baseline Algorithm.....	77
3.4.2	Comparison	79
3.5	Summary	84
IV	Advanced Disparity Estimation Algorithms for High Definition 3DTV Applications	86
4.1	High-Quality Disparity Estimation Algorithm	86
4.1.1	Related Work	86
4.1.2	Observation in DERS and Baseline Algorithms.....	89
4.1.3	Proposed Algorithm Flow.....	91
4.1.4	Downsampled Disparity Estimation for Full Range Disparity.....	93
4.1.5	Joint Bilateral Upsampling.....	98
4.1.6	Occlusion Handling.....	99
4.1.7	Temporal Consistency Enhancement.....	102
4.2	Sparse-Computation Disparity Estimation Algorithm	106
4.2.1	Related Work	107
4.2.2	Proposed Algorithm Flow.....	107
4.2.3	Sparse Pixel Selection	111
4.2.4	Sparse-Computation Steps.....	115
4.2.5	Computational Reduction.....	116
4.3	Hardware-Efficient Disparity Estimation Algorithm	117
4.3.1	Design Challenges in High-Quality Algorithm	118
4.3.2	Proposed Algorithm Flow.....	121
4.3.3	Cost Diffusion Algorithm	122
4.3.4	Image Buffer Reduction Methods	126
4.3.5	Small Filter Window Size.....	127
4.3.6	Regular Occlusion Handling	128
4.3.7	Simple Region Detection.....	129
4.4	Summary	131
V	Experimental Results	132
5.1	Experiment Setting.....	132
5.1.1	Test Sequences.....	132

5.1.2	Input and Output Configuration.....	134
5.2	Comparison.....	136
5.2.1	Execution Time.....	136
5.2.2	Objective Quality Evaluation	138
5.3	Summary.....	159
VI	Design of Disparity Estimation Engine for High Definition 3DTV Applications.....	161
6.1	Architectural Analysis	161
6.1.1	Analysis of Hardware-Efficient Disparity Estimation Algorithm.....	161
6.1.2	Proposed Hardware-Based Algorithm	164
6.2	Overview of Disparity Estimation Engine.....	168
6.2.1	Proposed Three-Stage Pipelining Architecture.....	168
6.2.2	Schedule of Main Core	169
6.3	Detailed Architectural Design	170
6.3.1	Low-Resolution Disparity Estimation Stage	171
6.3.2	Occlusion Handling Stage	179
6.3.3	High-Resolution Disparity Estimation Stage.....	184
6.4	External Memory Access.....	189
6.4.1	Bandwidth Requirement.....	189
6.4.2	External Memory Architecture	191
6.4.3	Data Configuration in External Memory	192
6.4.4	External Memory Access Schedule	194
6.5	Implementation Result.....	195
6.5.1	Hardware Cost	195
6.5.2	Disparity Quality	199
6.6	Summary.....	203
VII	Conclusion.....	204
7.1	Contribution.....	204
7.2	Future Work.....	205
	Bibliography.....	206
	作者簡歷.....	215

List of Tables

Table II-1 Various match metrics for computing $C_o(x, y, d)$	10
Table III-1 Comparison of memory cost in memory access approaches for the iteration count of 30..	55
Table III-2 Logic cost comparison of PE architectures	57
Table III-3 Implementation results of various BP-based algorithms.....	58
Table III-4 Comparison of BF acceleration approach in computational complexity and memory cost	59
Table III-5 Computational flow and analysis for a pixel in the integral histogram approach.....	63
Table III-6 Modified computational flow and analysis for a pixel in the integral histogram approach	70
Table III-7 Example implementation result of the proposed architecture	76
Table III-8 Comparison of hardware cost per frame	76
Table III-9 Previous VLSI implementations of bilateral filtering	77
Table III-10 Comparison of different implementations.....	77
Table IV-1 Simulation results with different sampling factors in Y-PSNR (dB).....	94
Table IV-2 Comparison of execution time of HQ-DE and SC-DE algorithms	117
Table IV-3 Window sizes of filter-based processes in HQ-DE algorithm.....	120
Table IV-4 Comparison of memory requirement between BP-M and cost diffusion methods	124
Table IV-5 Window sizes of filter-based processes in HE-DE algorithm	128
Table V-1 Test sequences	134
Table V-2 Input and output views for 2-view configuration [71].....	135
Table V-3 Input and out views for 3-view configuration [71].....	135
Table V-4 Experiment setting in our evaluation.....	136
Table V-5 Average execution time of proposed algorithms on PC for one frame	137
Table V-6 Average execution time scaled to HD1080p resolution and disparity range of 128.....	137
Table V-7 Evaluation results of Y-PSNR for View0.....	139
Table V-8 Evaluation results of Y-PSNR for View8.....	139
Table V-9 Evaluation results of SSIM for View0.....	140
Table V-10 Evaluation results of SSIM for View8.....	141
Table V-11 Evaluation results of T_PSPNR (dB) for View0.....	142
Table V-12 Evaluation results of T_PSPNR for View8.....	142
Table VI-1 Estimated average external bandwidth for computing four disparity rows.	191
Table VI-2 Performance of the proposed disparity estimation engine	196
Table VI-3 Internal SRAM usage in the proposed disparity estimation engine	196
Table VI-4 Internal registers in the proposed disparity estimation engine.....	197
Table VI-5 Area of the computational logic	197
Table VI-6 Comparison of our design and previous implementation	198
Table VI-7 Evaluation results of Y-PSNR for View0.....	200
Table VI-8 Evaluation results of Y-PSNR for View8.....	200
Table VI-9 Evaluation results of SSIM for View0	201

Table VI-10 Evaluation results of SSIM for View8..... 201
Table VI-11 Evaluation results of T_PSPNR (dB) for View0 202
Table VI-12 Evaluation results of T_PSPNR (dB) for View8 202



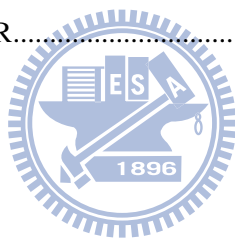
List of Figures

Figure II-1 Epipolar geometry.....	6
Figure II-2 Image planes with rectification.....	6
Figure II-3 Relation between disparity and depth for a pair of correspondences	7
Figure II-4 A general framework for disparity estimation algorithms	8
Figure II-5 Matching costs of a target pixel and its correspondence candidates.....	9
Figure II-6 Illustration of a cost cube.....	9
Figure II-7 Block-based matching cost with the block radius r	10
Figure II-8 Various cost aggregation approaches	12
Figure II-9 Concept of dynamic programming approach.....	14
Figure II-10 Graph model of graph-cut algorithm	15
Figure II-11 Graph model of belief propagation approach.....	16
Figure II-12 General flow of view synthesis.....	19
Figure II-13 Warping methods in view synthesis.....	20
Figure II-14 Blending step in view synthesis.....	21
Figure II-15 Input and output view configuration defined by the 3DVC.....	23
Figure II-16 Flow of the DERS algorithm	24
Figure II-17 Data flow for 3-view configuration	27
Figure II-18 Example of temporal noise changing successive frames [76]	30
Figure II-19 Example of block matching in the DERS algorithm.....	32
Figure III-1 Illustrations of BP.....	35
Figure III-2 Configuration of the message passing PEs.....	40
Figure III-3 Traditional fixed memory access approach in a 1-D node line for node $n3$ computation .	43
Figure III-4 Proposed spinning-message approach	44
Figure III-5 Proposed spinning-message approach in a 2-D node plane for node $n3$ computation	45
Figure III-6 Comparison of memory access approaches in different node planes	45
Figure III-7 Sliding node plane in different directions.....	46
Figure III-8 Sliding node plane with the spinning-message approach.....	47
Figure III-9 Bipartite node plane with the spinning-message approach.....	48
Figure III-10 Proposed sliding-bipartite node plane	49
Figure III-11 Pseudo code of the message passing for calculating a new message	51
Figure III-12 Architecture of Park's PE	51
Figure III-13 Proposed architecture	53
Figure III-14 Ratio of memory cost in different node planes with spinning-message approach	56
Figure III-15 Classification of BF acceleration approaches.....	59
Figure III-16 Concept of histogram-based approaches	61
Figure III-17 Concept of integral histogram approach.....	64
Figure III-18 Runtime updating method (RUM).....	67

Figure III-19 Stripe-based method (SBM)	68
Figure III-20 Sliding origin method (SOM)	69
Figure III-21 Proposed architecture of JBF.	71
Figure III-22 Schedule of the proposed architecture	72
Figure III-23 Selected-bin adder in the histogram calculation engines	73
Figure III-24 Proposed architectures of histogram calculation engines hi_c and hc_c	73
Figure III-25 Proposed architecture of (a) convolution engine and (b) its table selection modules.....	75
Figure III-26 Flow of the proposed baseline disparity estimation algorithm	78
Figure III-27 Experimental results of the baseline algorithm and the DERS algorithm	80
Figure III-28 Center disparity maps and synthesized View8 of baseline algorithm at the 100th frame	82
Figure III-29 Center disparity maps and synthesized View8 of DERS algorithm at the 100th frame...	84
Figure IV-1 Flow of the adaptive-BP algorithm [39]	87
Figure IV-2 Flow of the double-BP algorithm [40]	88
Figure IV-3 An example of flicker artifact of the baseline algorithm in BookArrival	90
Figure IV-4 An example of foreground copy artifact of the DERS algorithm in BookArrival	90
Figure IV-5 An example of occlusion problem at the 44th frame of BookArrival.....	91
Figure IV-6 Flow of the HQ-DE algorithm for a center-view disparity map	92
Figure IV-7 Flow of the HQ-DE algorithm for a side view disparity map	93
Figure IV-8 Comparison of different sampling factors in the average Y-PSNR of two frames	94
Figure IV-9 Simulation results using the sampling factors of $1/2 \times 1/4$ and $1/4 \times 1/4$	95
Figure IV-10 Illustration of downsampled disparity estimation for full disparity range	96
Figure IV-11 Comparison between the original regional vote [6] and the proposed window vote	99
Figure IV-12 Illustration of the proposed occlusion detection method	100
Figure IV-13 Results with and without the proposed occlusion handling method in BookArrival.....	102
Figure IV-14 Results of the HQ-DE algorithm in BookArrival compared to Figure IV-5	102
Figure IV-15 Concept of the proposed no-motion registration (NMR) method.....	104
Figure IV-16 Results of the proposed NMR method in BookArrival.....	105
Figure IV-17 Results of the proposed NMR method in the 32th, 34th, 36th, 38th frames.....	105
Figure IV-18 Results of the proposed SEP method in BookArrival	106
Figure IV-19 Profiling of the HQ-DE algorithm on PC	108
Figure IV-20 Flow of the SC-DE algorithm for center-view disparity map.....	109
Figure IV-21 Flow of the SC-DE algorithm for side-view disparity maps.....	111
Figure IV-22 Flow of region detection for sparse pixel selection	112
Figure IV-23 Example of edge maps in BookArrival.....	113
Figure IV-24 Example of occlusion maps in BookArrival	113
Figure IV-25 Example of motion maps in BookArrival	115
Figure IV-26 Concept of sparse SSAD and sparse ADSW methods	115
Figure IV-27 Concept of sparse BP-M method	116
Figure IV-28 Profiling of the SC-DE algorithm on PC	117

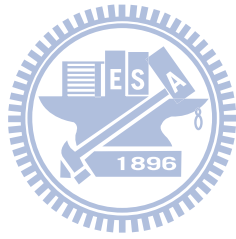
Figure IV-29 Image buffer required by the SSAD and ADSW steps	119
Figure IV-30 Flow of the HE-DE algorithm for center view	122
Figure IV-31 Concept of BP-M computation.....	123
Figure IV-32 Concept of the proposed window-based SSAD method.....	127
Figure IV-33 Flow of proposed occlusion handling method in HE-DE algorithm	128
Figure IV-34 Flow of edge detection and motion detection in HE-DE algorithm	130
Figure V-1 Clips of test sequences in center view.....	133
Figure V-2 Evaluation results of Y-PNSR.....	140
Figure V-3 Evaluation results of SSIM	141
Figure V-4 Evaluation results of T_PSPNR.....	143
Figure V-5 Disparity maps and view synthesized images in the 50 th frame of BookArrival	145
Figure V-6 Disparity maps and view synthesized images in the 50 th frame of LoveBird1	147
Figure V-7 Disparity maps and view synthesized images in the 100 th frame of Newspaper.....	149
Figure V-8 Disparity maps and view synthesized images in the 50 th frame of Café.....	149
Figure V-9 Disparity maps and view synthesized images in the 50 th frame of Kendo.....	150
Figure V-10 Disparity maps and view synthesized images in the 100 th frame of Balloons	151
Figure V-11 Disparity maps and view synthesized images in the 50 th frame of Champagne.....	153
Figure V-12 Disparity maps and view synthesized images in the 50 th frame of Pantomime	155
Figure V-13 Disparity maps and view synthesized images in the 50 th frame of Hall1.....	156
Figure V-14 Disparity maps and view synthesized images in the 50 th frame of Hall2.....	157
Figure V-15 Disparity maps and view synthesized images in the 167 th frame of CarPark	158
Figure V-16 Disparity maps and view synthesized images in the 50 th frame of CarPark	159
Figure VI-1 Data dependency of the HE-DE algorithm.....	162
Figure VI-2 Required row buffers in filter-based processes for pipelining architecture.....	163
Figure VI-3 Memory buffers in the motion detection	164
Figure VI-4 Flow of the proposed HW-DE algorithm	165
Figure VI-5 Proposed motion detection in the HW-DE algorithm.....	167
Figure VI-6 Overview architecture of the proposed disparity estimation engine	169
Figure VI-7 Proposed computational schedule for main core.....	170
Figure VI-8 Architecture of the low-resolution disparity estimation stage	172
Figure VI-9 Data access of the motion detection module in the frame coordinate system.....	173
Figure VI-10 Architecture of the motion detection module	173
Figure VI-11 Input and required data in matching cost calculation for three target views	174
Figure VI-12 Architecture of the window-based SSAD and DPotts modules	175
Figure VI-13 Architecture of the temporal cost calculation module.....	176
Figure VI-14 Architecture of vertical cost diffusion module	177
Figure VI-15 Fully parallel architecture of the horizontal cost diffusion module.....	178
Figure VI-16 Architecture of the horizontal cost diffusion module	179
Figure VI-17 Architecture of the occlusion handling stage.....	180

Figure VI-18 Architecture of the disparity cross warping module	181
Figure VI-19 Architecture of the occlusion detection PE and the warp filling PE.....	182
Figure VI-20 Architecture of the good disparity detection module.....	183
Figure VI-21 Architecture of border filling and inside filling modules	184
Figure VI-22 Architecture of the high-resolution disparity estimation stage	185
Figure VI-23 Memory configuration in the high-resolution disparity estimation stage	186
Figure VI-24 Architecture of the joint bilateral upsampling module	187
Figure VI-25 Architecture of the window vote module.....	188
Figure VI-26 Architecture of the mask and vote PEs for the window vote module.....	188
Figure VI-27 Architecture of the still-edge preservation module.....	189
Figure VI-28 Rough schedule for external memory access.....	190
Figure VI-29 Architecture of external memory in our design	191
Figure VI-30 Read and write latency in the SDRAM model [110].....	192
Figure VI-31 Data configuration in external memory	194
Figure VI-32 Schedule of external memory access for one HD1080p frame at 800MHz.....	195
Figure VI-33 Evaluation results of Y-PNSR.....	200
Figure VI-34 Evaluation results of SSIM.....	201
Figure VI-35 Evaluation results of T_PSPNR.....	202



List of Symbols

Symbols	Descriptions
H, W	Frame height, frame width
DR	Disparity range
$I_{S0,S1}^{S2}$	Image frame where $S0$ could be H for high resolution, and L for low resolution, $S1$ could be L for left view, C for center view, and R for right view, and $S2$ could be t for current frame, and $t-1$ for previous frame
$D_{S0,S1}^{S2}$	Disparity map where $S0$ could be H for high resolution, and L for low resolution, $S1$ could be L for left view, C for center view, and R for right view, and $S2$ could be t for current frame, and $t-1$ for previous frame
C_0	Initial cost cube computed by the matching cost calculation step
C_{aggr}	Cost cube computed by the cost aggregation step
C_{view}	Cost for inter-view consistency constraint
C_{temp}	Cost for temporal consistency enhancement
C_{vert}	Cost for vertical diffusion method
C_{total}	Final Cost for disparity optimization
T	Iteration count in belief propagation
$D(d)$	Data term in disparity optimization process
$V(d_i, d_j)$	Smoothness term in disparity optimization process



I Introduction

1.1 Background

With the prompt development of 3-D display techniques, people could obtain the new visual experience from 3-D videos, which have multi-view videos for left and right eyes. Compared to traditional 2-D videos, 3-D videos could make human have the distance feeling of scene with the additional video processes: calibration and rectification, multi-view video coding, disparity estimation, and virtual view synthesis. For these 3-D video processes, the Moving Picture Experts Group (MPEG) 3-D Video Coding (3DVC) has delivered a basic 3DTV framework that consists of the depth estimation reference software (DERS) [63], view synthesis reference software (VSRS) [64], and Multi-view Video Coding (MVC) standard [107]. They also provide the multi-view video sequences [71] for the performance evaluation. The basic 3DTV framework can be extended to various systems such as the stereoscopic TV for multiple viewers and the free-viewpoint TV for a larger viewing zone [100], [101].

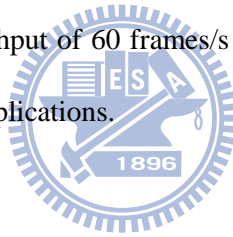
For the basic 3DTV framework, the previous VLSI implementation of VSRS and MVC decoder [61], [62] can reach the real-time performance for high definition videos. On the other hand, the DERS could deliver high quality disparity maps but suffers from high computational complexity due to its graph-cut optimization, especially for high definition videos. Therefore, it is necessary to develop a disparity estimation engine that could deliver high quality disparity maps and achieve the real-time performance for high definition videos.

1.2 Motivation

Many disparity estimation algorithms have been developed in computer vision for different applications, such as medical image analysis, augmenting reality, robot, 3DTV, and etc. The disparity

accuracy evaluation [72] shows that the graph-cut and the belief propagation approaches could perform better than other kinds of approaches. Based on the graph-cut approach, the state-of-the-art DERS algorithm delivered by MPEG 3DVC could generate high quality disparity maps for 3DTV applications, but it still encounters the following problems. First, the temporal consistency problem is not addressed well due to the foreground copy artifact. Second, its execution time will be dramatically increased with the increasing video resolution and disparity range. For one HD1080p frame, it takes more than 20 minutes in average on a personal computer. Third, the computation of graph-cut is irregular and iterative, so that it is not suitable to be accelerated by the parallel computing PEs of VLSI design or multi-core platform.

Motivated by the problems in the state-of-the-art disparity estimation algorithm, the goal of this dissertation is to develop a new disparity estimation engine that could not only generate high quality disparity maps, but also achieve the throughput of 60 frames/s for the HD1080p resolution to satisfy the requirement of high definition 3DTV applications.



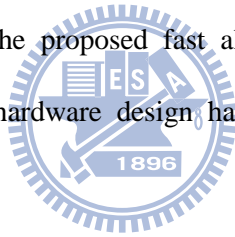
1.3 Contribution

To achieve the above goal, this dissertation develops a disparity estimation engine from algorithm level to architectural design level. The main achievement of this dissertation includes a baseline and an advanced disparity estimation algorithms, and two fast algorithms for the advanced one, and a high throughput disparity estimation design.

The contributions in each achievement are as follows. First, the baseline disparity estimation algorithm combines the belief propagation approach to increase the computational parallelism of disparity estimation, and the joint bilateral upsampling approach to decrease the computational space. In addition, we also solve their memory cost problems by architectural design techniques. Second, based on the baseline algorithm, we propose the advanced disparity estimation algorithm that could solve the temporal consistency and occlusion problems, and deliver better disparity maps than the

DERS algorithm. Third, we also propose two fast disparity estimation algorithms to accelerate the high-quality algorithm by different strategies for different implementation methods. For the processor-based platform, the sparse-computation algorithm could reduce the original execution time to 62.9% by reducing the processed pixels from dense to sparse space. On the other hand, for the hardware design, the hardware-efficient algorithm could reduce the original memory cost to 0.00029% by replacing the belief propagation with the proposed cost diffusion method. Finally, we propose a high throughput disparity estimation engine for the hardware-efficient algorithm with three-stage row-based pipelining architecture. The dedicated design could achieve the throughput of 95 frames/s for three HD1080p view disparity maps, using 1,645K gate counts and 59.4-Kbyte memory.

In the objective quality evaluation, the experimental results show that our proposed advanced disparity estimation algorithm could perform better than the DERS algorithm, especially for the temporal consistency. In addition, the proposed fast algorithms have similar performance to the advanced algorithm, and the final hardware design has slight quality degradation because of its simplification.

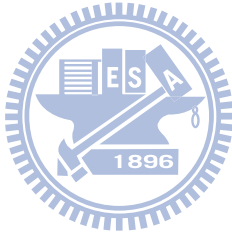


To sum up, the proposed disparity estimation design could deliver the disparity maps with the high throughput and high quality to satisfy the requirement of high definition 3DTV applications.

1.4 Dissertation Organization

This dissertation is organized as follows. Chapter II introduces the general framework of a disparity estimation algorithm, and the existing approaches of each step in the framework. Chapter III analyzes the algorithm and architecture of the belief propagation and the joint bilateral upsampling, and presents the baseline disparity estimation algorithm. To improve the quality and speed of baseline algorithm, Chapter IV proposes the high-quality disparity estimation algorithm and its two fast algorithms: sparse-computation and hardware-efficient. Then, Chapter V compares the disparity results of our proposed algorithms with the 3DVC's DERS algorithm by the objective evaluation

methods. With the hardware-efficient algorithm, Chapter VI proposes the architecture of disparity estimation engine, and demonstrates our implementation results. Finally, Chapter VII concludes this dissertation and future work.



II Background

In this chapter, the background of disparity estimation and its application to view synthesis are introduced. This chapter is organized as follows. First, we present the concept of disparity estimation, and review the existing disparity estimation algorithms. Then, we illustrate the view synthesis technique, depth-image-based rendering (DIBR), which is our target application of disparity estimation. Finally, we introduce the state-of-the-art disparity estimation algorithm [63] developed by MPEG 3-D Video Coding (3DVC), and point out its quality and design problems.

2.1 Disparity Estimation

In 3DTV applications, the disparity estimation is to extract the disparity information from source videos and generate a disparity map for each frame. The disparity map can describe the relative distance of objects in scene, and be further used to generate virtual-view videos. For different number of input video view, the disparity estimation has different approach. The *2-D to 3-D conversion approach* is for traditional single-view videos, while the *stereo correspondence approach* is for two-view and multiple-view videos. The former one recognizes the disparity map from various disparity cues, such as texture, defocus, vanish point, and etc. [102], [103], [104]. On the other hand, the latter one finds the pairs of correspondences to compute disparity maps. The dissertation focuses on the stereo correspondence approach.

2.1.1 Epipolar Geometry

The disparity estimation for multi-view videos could be constrained by the epipolar geometry to reduce the correspondence search range from 2-D space to 1-D space. Figure II-1 shows the concept of epipolar geometry with two-view configuration. In which, the object P_b is watched by the target viewpoint C and projected into the 2-D image plane at the pixel p . For the reference viewpoint C' , the

correspondence candidates with p would be located on the ray from C to Pb , whose projected line in the reference image plane is called *epipolar line*. In other words, the correspondence with p could be searched on the epipolar line, and the search range is restricted in 1-D space.

Furthermore, the image planes could be rectified and translated into the new positions with parallel epipolar lines as shown in Figure II-2. In which, the correspondence search range is on a horizontal line, instead of an oblique line in the original image plane. In other words, the pair of correspondences is at the identical y -coordinate in two views. Thus, the computation of disparity estimation can be regular in the raster-scan order.

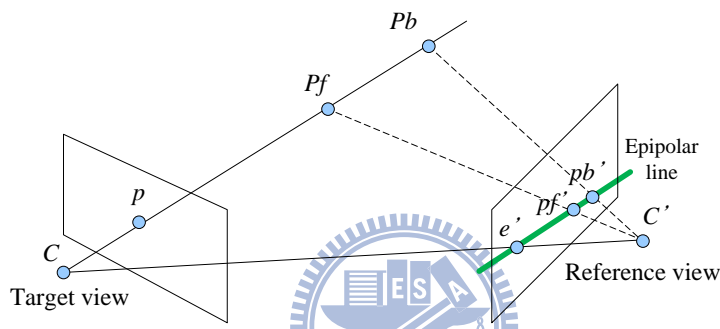


Figure II-1 Epipolar geometry

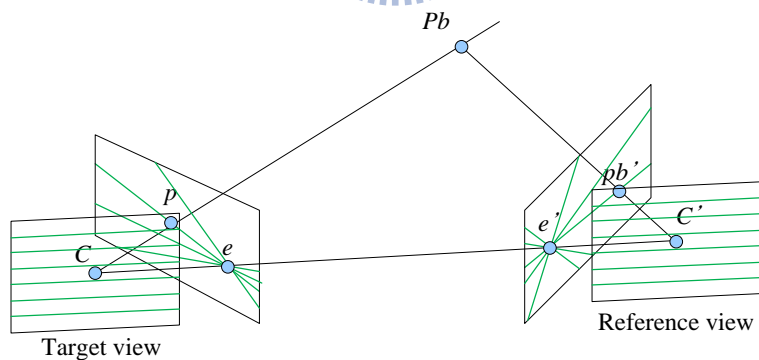


Figure II-2 Image planes with rectification

With the rectified image planes, Figure II-3 shows the relation between depth and disparity for a pair of correspondences. In which, the two cameras at the viewpoints C and C' capture the object point Pb and project it to the pair of correspondences on the epipolar line. The correspondences are located

at the coordinates of X and X' based on their camera centers. Given the focal length f and the baseline B of the cameras, if we could estimate the disparity $X-X'$, the object depth Z can be acquired by

$$Z = \frac{f \times B}{X - X'} . \quad (\text{II-1})$$

Therefore, the disparity estimation is to find the pair of correspondences, and use their x -coordinates to compute disparity value of depth value for each pixel.

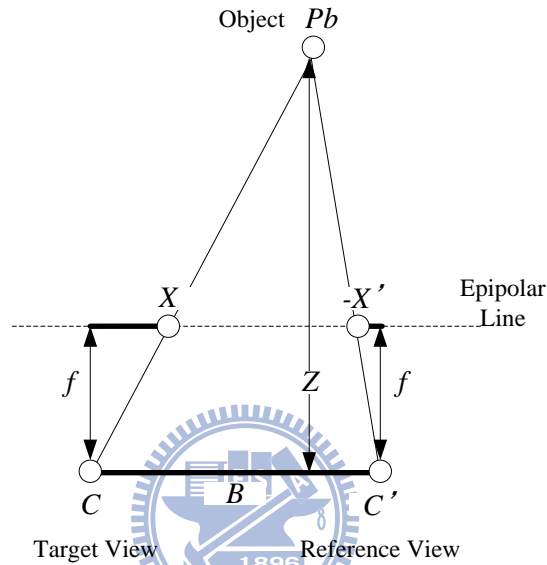


Figure II-3 Relation between disparity and depth for a pair of correspondences

2.1.2 General Algorithm Flow

For disparity estimation algorithms, a general framework is proposed by Scharstien and Szeliski [105] as shown in Figure II-4. In this framework, two images are captured and rectified as inputs, and a disparity map is the target result. By this framework, disparity estimation algorithms can be classified into the two categories: local approach and global approach [105], [106]. The local approach only consists of the matching cost calculation and the cost aggregation, and the global approach additionally performs the optimization process. The last disparity refinement step is an optional process for computing fractional disparity and other post-processing. The existing approaches for each step are reviewed as follows.

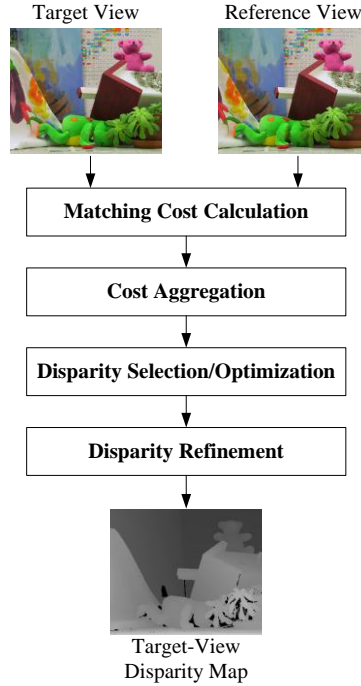


Figure II-4 A general framework for disparity estimation algorithms

1. Matching Cost Calculation

Matching cost is a quantitative dissimilarity measure to find the best pair of correspondences. Figure II-5 shows the concept of the matching cost calculation. In which, a target pixel has multiple reference pixels as correspondence candidates, and each correspondence candidate has a matching cost. The number of correspondence candidates is equal to the disparity range DR , which is related to the nearest and farthest objects in scene. Hence, each target pixel has DR matching costs. To determine a whole disparity map, the matching costs of all target pixels are calculated and form a disparity image space (DSI), which is called *cost cube* in this dissertation. As shown In Figure II-6, a cost cube contains the spatial dimensions X , Y and the disparity dimension d . The size of this cube for whole frame is $H \times W \times DR$ where H and W are the frame height and width. The initial values of the cost cube are computed by the matching cost calculation.

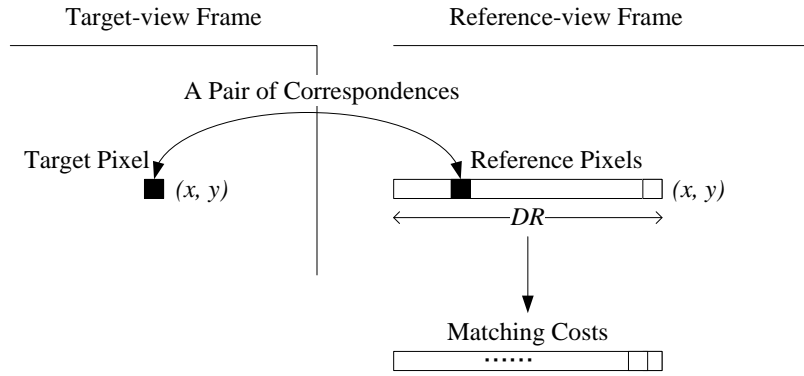


Figure II-5 Matching costs of a target pixel and its correspondence candidates

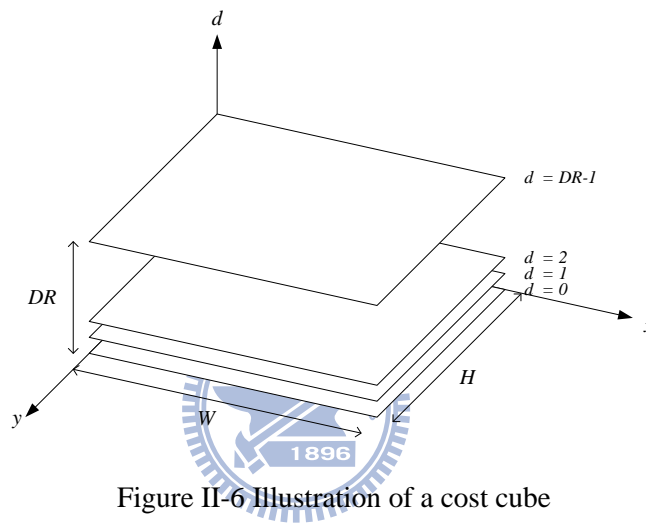


Figure II-6 Illustration of a cost cube

To compute the initial cost cube C_0 , one of the various match metrics [105]-[3] could be adopted. Table II-1 lists the commonly used match metrics, which can be classified into pixel base and block base. For the pixel-based match metric, the absolute difference (AD) and the square difference (SD) are computed using a target pixel and a reference pixel. The pixel dissimilarity measure (PDM) additionally considers the half pixels to lessen the sampling sensitivity [1].

On the other hand, the block-based match metric is computed using a target block and a reference block with support pixels as illustrated in Figure II-7. In Table II-1, the normalized cross correlation (NCC) is a statistical method that uses the block mean and variance to reduce the sensitivity to radiometric gain and bias. The Rank transforms the pixel color into the rank value, which is the relative order of center pixel in the block, and computes the matching cost by the rank difference. On the other hand, the Census transforms the pixel intensity into census bit stream, which consists of the

intensity comparison results between the center pixel and the support pixels. The matching cost of two census bit streams is computed by the Hamming distance. Because the Rank and Census transform original pixel from color to different domains, they could better resist the radiometric distortion between views.

To sum up, the initial cost cube C_0 is computed in this matching cost calculation step, and the computational complexity of this step is $O(H \times W \times DR)$.

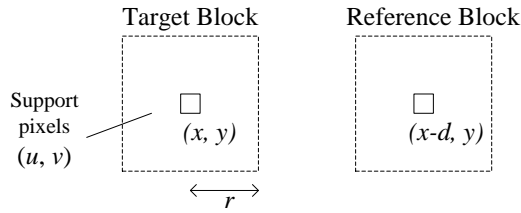


Figure II-7 Block-based matching cost with the block radius r

Table II-1 Various match metrics for computing $C_0(x, y, d)$

Pixel-based metric	
Absolute Difference (AD)	$ I_{tar}(x, y) - I_{ref}(x - d, y) $
Square Difference (SD)	$[I_{tar}(x, y) - I_{ref}(x - d, y)]^2$
Pixel Dissimilarity Measure (PDM)	$\min\{ I_{tar}(x, y) - I_{ref}(x - d, y) , I_{tar}(x, y) - I_{ref}^+ , I_{tar}(x, y) - I_{ref}^- \}$ where I_{ref}^+ and I_{ref}^- are the neighboring half pixel of $I_{ref}(x - d, y)$
Block-based metric	
Normalized Cross Correlation (NCC)	$\frac{\sum_{ x-u \leq r, y-v \leq r} [I_{tar}(u, v) - \bar{I}_{tar}][I_{ref}(u - d, v) - \bar{I}_{tref}]}{\sqrt{\sum_{ x-u \leq r, y-v \leq r} [I_{tar}(u, v) - \bar{I}_{tar}]^2 [I_{ref}(u - d, v) - \bar{I}_{tref}]^2}}$
Rank	$ I'_{tar}(x, y) - I'_{ref}(x - d, y) $, where $I'(m, n) = \sum_{ m-u \leq r, n-v \leq r} I(m, n) > I(u, v)$
Census	$Hamming(I'_{tar}(x, y), I'_{ref}(x - d, y))$, where $I'(m, n) = bitstream_{ m-u \leq r, n-v \leq r}(I(m, n) > I(u, v))$

2. Cost Aggregation

The main idea of cost aggregation step is to gather the costs of neighboring pixels to the center pixel in a window. It implies that the neighboring pixels have the same disparity as the center pixel, and gather the costs of neighbors could increase the reliability of matching cost. Thus, the cost aggregation step accumulate the neighboring costs for the center pixel by the general equation,

$$C_{aggr}(x, y, d) = \frac{\sum_{(u,v) \in win(x,y)} C_0(u, v, d) \cdot W_{aggr}(u, v)}{\sum_{(u,v) \in win(x,y)} W_{aggr}(u, v)}, \quad (\text{II-2})$$

where C_0 is the initial cost cube, and C_{aggr} is the aggregated cost cube. In this equation, each initial cost $C_0(u, v, d)$ in an aggregation window with radius r is accumulated with the weight $W_{aggr}(u, v)$ for the target cost $C_{aggr}(x, y, d)$. In addition, the accumulated value is normalized by the sum of weights. The computational complexity of this step is $O(H \times W \times DR \times r^2)$, which is proportional to the aggregation window size.

Figure II-8 shows the various existing cost aggregation approaches with different weight distributions. In Figure II-8 (a), the uniform weight has constant weight for each support pixels and the fixed r . Its disparity map would be over-blurred for thin objects if r is too large, while it would be incorrect for textureless regions if r is too small. Therefore, for better disparity quality, the radius of uniform weight need to be adaptively adjusted according to image content as shown in Figure II-8 (b). The other common-used is the Gaussian weight approach that makes the pixel near window center has higher weight. However, these three approaches could not obtain accurate disparity due to their fixed window shape, (i.e. square or circle).

To control the window shape, the adaptive polygon weight approach [4], [5] uses the 8-direction or 4-direction configuration to fit the object shape as shown in Figure II-8 (d). Then, the cross-based weight approach [6] uses multiple cross lines to fit the object shape as shown in Figure II-8 (e). In the two approaches, a support region grows from the window center until its boundary touches a dissimilar pixel. However, the two approaches could not perform well for the highly texture regions because of their continuous support regions.

The adaptive support-weight (ADSW) approach [7] can avoid their problem, because all support pixels are considered and their weight is determined by the kernels of bilateral filter. Its weight is defined as

$$W_{aggr}(u, v) = W_{tar}(u, v) \times W_{ref}(u - d, v) \quad , \quad (\text{II-3})$$

where W_{tar} is the weight from target-view window, and W_{ref} is the weight from reference-view window.

Both the weights W_{tar} and W_{ref} are computed by the kernels of bilateral filter,

$$W(u, v) = f(\|(x, y) - (u, v)\|)g(\|I(x, y) - I(u, v)\|) \quad . \quad (II-4)$$

where f is the spatial kernel with the position distance, and g is the range kernel with the color distance.

With the two kernels, the aggregation weight would be large if the support pixel is near the center pixel or the support pixel is similar to center pixel. Figure II-8 (f) illustrates the adaptive support-weight. In which, the aggregation weight could fit object shape better than the adaptive polygon weight and cross-based weight approaches for highly texture regions. However, the main disadvantage of ADSW approach is high computational complexity. Nevertheless, it can be addressed by the integral histogram approach [8], the iterative aggregation with small window approach [9], and the data reuse approach in VLSI design [10].

In summary, the aggregation cost step processes the initial cost cube C_0 to a more reliable cost cube C_{aggr} by the well-define weights.

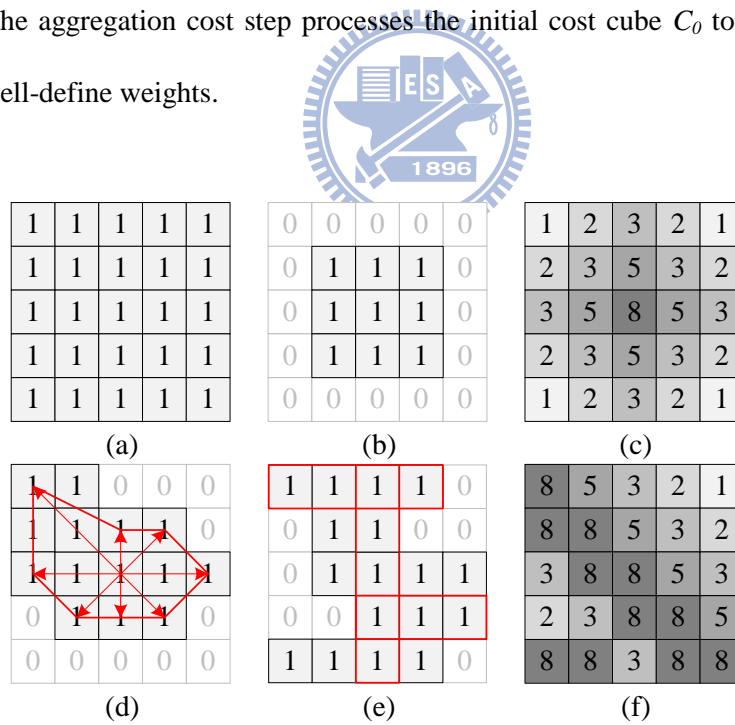


Figure II-8 Various cost aggregation approaches

(a) uniform weight, (b) uniform weight with adaptive window radius, (c) Gaussian weight, (d) adaptive polygon weight, (e) cross-based weight, (f) adaptive support-weight.

3. Disparity Selection/Optimization

With the aggregated cost cube C_{aggr} , two optional methods can be applied to compute the disparity map. One is the winner-take-all manner (WTA) which directly determines the disparity result by selecting the reference pixel with minimum cost as the best correspondence for each target pixel. The other one is the disparity optimization method which considers the aggregated costs of whole frame to compute the disparity map by the energy minimization. The latter can acquire more accurate disparity maps as shown in the evaluation results [72].

The common-used disparity optimization approaches are dynamic programming (DP), graph-cut (GC), and belief propagation (BP). Their main concept is to convert the disparity estimation problem into an energy minimization problem. The energy function is generally formulated by

$$E(\mathbf{d}) = E_{data}(\mathbf{d}) + \lambda E_{smooth}(\mathbf{d}) \quad (\text{II-5})$$

where E_{data} is data term to penalize the dissimilarity of a correspondence pair, and E_{smooth} is smoothness term to penalize the disparity inconsistency of two neighboring pixels. In addition, \mathbf{d} is a selected disparity set for whole frame. The optimization approaches attempt to find a disparity set \mathbf{d} by the way of minimizing the total energy E .

The concept of the common-used optimization approaches are reviewed as follows.

(1) Dynamic Programming

The main idea of DP approach is to convert the disparity estimation to a finding shortest path problem. The optimization process is performed row by row. Figure II-9 (a) shows the graph model for finding shortest path problem. In which, the position of node is corresponding to the coordinate in the $x-d$ plane, and the shortest path will be from x of 0 to $W-1$. The path would suffer from matching penalty on a node, and smoothness penalty on an edge. The DP approach is to find the path with minimum penalty by the two steps: forward accumulating and backward tracing. In Figure II-9 (b), first step accumulate the penalty in the forward direction to select the moving direction for each node. In Figure II-9 (c), with the moving direction map, the second step trace the path with minimum penalty in the backward direction.

However, the DP approach suffers from streak artifact in the disparity map because of its row-by-row process. To address this problem, Ohta and Kanade [11] perform the DP in a 3-D space that consists of the original intra-scanline space and the additional inter-scanline space. In addition, the tree-based DP algorithms [12]-[14] use the tree structure to connect scanlines and remove the streak artifacts.

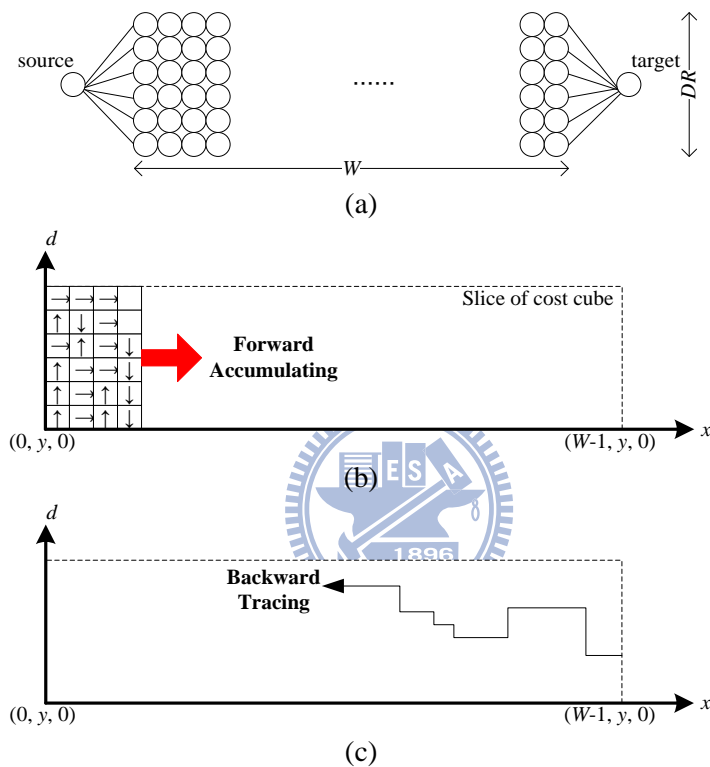


Figure II-9 Concept of dynamic programming approach

(a) graph model in DP approach, (b) forward accumulating, (c) backward tracing

(2) Graph-Cut

The main idea of GC approach is to convert the disparity selection problem to the min-cut/max-flow problem [15], and the associated optimization techniques could be adopted. The GC approach can generate accurate disparity maps.

Figure II-10 shows the graph model of min-cut/max-flow for disparity estimation. In which, there are $H \times W \times DR$ nodes with 6-connected node grid. The matching cost and the smoothness cost are well-defined on each edge, which can be regarded as pipes with different flow volumes due to

different costs. In this graph model, water from the source node would flow to the sink node through pipes. The min-cut means that a cut surface cross edges has the minimum flow, while the max-flow means that the allowed maximum flow from the source to the sink. The min-cut and the max-flow are equivalent problems. For the disparity estimation, the disparity map can be directly determined according to the resultant cut surface.

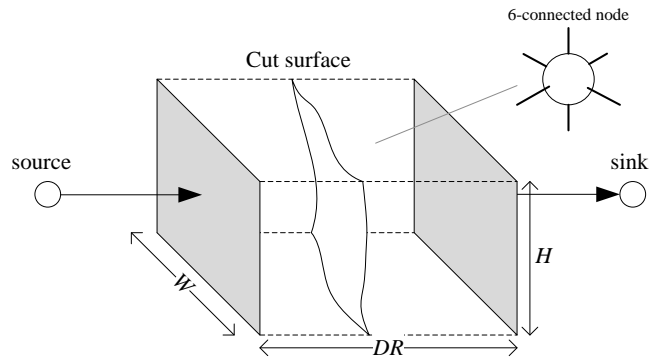


Figure II-10 Graph model of graph-cut algorithm

For the min-cut/max-flow problem, the common-used optimization techniques are the push-relabeling [16] and the augmenting path [17]. Their computational complexity is highly related to the number of label candidate (i.e. disparity range DR in disparity estimation). However, the optimization techniques suffer from extremely high computational complexity due to large disparity range.

To reduce the computational complexity, Boykov proposed the swap method [18] and an efficient augmenting path [19]. The swap method performs the optimization process disparity by disparity, and only one new disparity is considered in an iteration. Based on the swap method, Chou *et. al.* [20] proposed a fast algorithm to predict the disparities to skip the partial optimization process. On the other hand, for the push-relabeling approach, the computational speed depends on the processing order on nodes. Thus, Checkassky and Goldberg [21] proposed the highest-label order that is more efficient than the typical FIFO order. In addition, Delong and Boykov [22] proposed a block-based graph cut method to increase the parallelism of push-relabeling approach.

To sum up, the GC approach can perform accurate disparity results but is not suitable to be accelerated by GPU programming and VLSI design due to its irregular computation and low parallelism.

(3) Belief Propagation

Sun *et al.* [24] first applied the BP approach to solve the disparity estimation problem, and acquired accurate disparity maps. They perform the energy minimization on the graph model as shown in Figure II-11. In which, each node is corresponding to a pixel, and all nodes are connected by 4-connection grid. In the optimization process, the matching costs of each node are diffused through the *messages* to neighboring nodes iteration by iteration. This diffusion mechanism is called *message passing*. After several iterations, the matching costs and messages of a node are aggregated to determine the disparity result. Although the minimized energy could not definitely converge due to its loopy optimization process, the disparity maps could approach to a steady state.

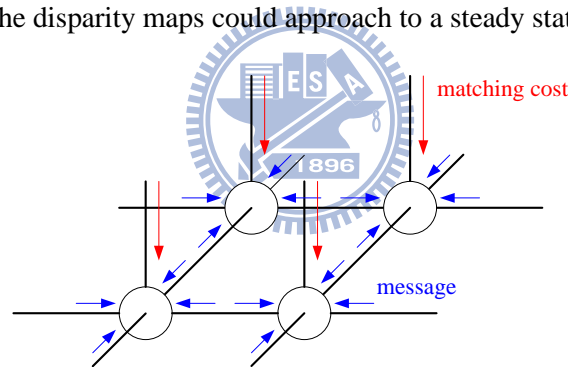


Figure II-11 Graph model of belief propagation approach

In the BP approach, the message passing suffers from the highest computational complexity, $O(H \times W \times DR^2 \times T)$, where T is the iteration count. The term of DR^2 results from the convolution, and the iteration count T should be more than 10. To reduce the computation of message passing, Felzenswalb and Huttenlocher [25] proposed the hierarchical BP (HBP) and the linear-time message passing. The former could accelerate the disparity convergent speed, and the latter could reduce the complexity of convolution from $O(DR^2)$ to $O(DR)$. In addition, Szeliski *et al.* [26] proposed the max-product loopy belief propagation, called BP-M, to reduce the iteration count by a scale. Because the computation of

BP approach is highly parallel, the BP approach is suitable to be accelerated by the GPU programming and VLSI design [27]-[33].

In addition, the BP approach also suffers from highly memory cost, $4HW \times DR$, for the matching costs and messages of whole frame. To address it, the bipartite graph [25] and the sliding approach [34] are proposed for the memory access, and the predictive coding scheme [35] could be applied for message compression.

To sum up, the above disparity optimization algorithms have different pros and cons. The DP approach could achieve real-time speed easier but has the streak artifacts. Its improvement methods would result in additional irregular computation. For the 2-D optimization approaches, the GC approach has high performance of disparity map, but its irregular computation limits the acceleration of GPU programming and VLSI design. On the other hand, the BP approach can also deliver accurate disparity maps and has highly parallelism. Therefore, this dissertation develops an efficient disparity estimation algorithm based on the BP approach.



4. Disparity Refinement

The final step refines the disparity maps by the post-processing methods: occlusion handling, object consistency enhancement, and temporal consistency enhancement. Their purpose and associated algorithms are reviewed as follows.

(1) Occlusion Handling

The occlusion problem results from that the object point is visible in one view and invisible in the other view. Thus, there is no correspondence pixel in the invisible view. Incorrect disparities would appear in the occlusion regions, and further induce artifacts in the view synthesis.

To handle the occlusion problem, the general approach is to detect the occlusion first, and then fill it by the background disparities. These two steps are called occlusion detection and occlusion filling. The basic methods for occlusion detection are surveyed in [45]. Various methods have different assumptions. The left-right check (LRC) assumes that a pair of correspondence should have identical

disparity, and the occlusion constraint (OCC) assumes that the disparity gap of two pixels would result in occlusion region in the other view. In addition, the order constraint (ORD) assumes that the order of two pixels should have the correspondences with the same order in the other view. In the above occlusion detection methods, the LRC is the most commonly applied for the disparity refinement [6], [40], and the OCC and the ORD are combined into the disparity optimization step [15], [24]. With the detected occlusion pixels, the occlusion filling step can directly replace them by the reliable background disparities.

(2) Object Consistency Enhancement

For an object, the disparities are usually identical or smooth changing. However, disparity maps often suffer from incorrect disparities, especially in the textureless regions. To remove the disparity noise, the plane fitting approach [46] is usually adopted by the high-performance disparity estimation algorithms [63], [39], [40]. In the plane fitting approach, the segment information is first computed by the watershed segmentation, mean-shift clustering, or K-mean clustering. According to the segment information, the disparities in a segment are used to compute a new 3-D plane by the linear regression method. Besides of the plane fitting method, the regional voting method [6] could also refine the disparity maps well. The regional vote method is simpler than the plane fitting method because the segment information is not required.

(3) Temporal Consistency Enhancement

Most of research develops their disparity estimation algorithms using the still image sequences [72]. However, they would miss the temporal consistency issue, which is important in the view synthesis application for video sequences. Without enhancing the temporal consistency, the disparity maps would suffer from flicker artifact, because each disparity frame is independently generated, and the disparities are unstable in the occlusion and textureless regions. This flicker artifact would further propagate to the view synthesis results, and is easily observed.

To address the temporal consistency, the neighboring frames should be considered in the disparity estimation. In the previous work [47]-[49], many disparity frames are buffered to construct a disparity

flow with the spatial and temporal dimensions, and different smooth approaches are performed in the disparity flow. On the other hand, with two adjacent frames, the temporal BP algorithm [41] preforms the BP optimization in a 6-connection grid graph, where the two additional connections link to the previous and next frames. In addition, the 3DVC's DERS algorithm [65]-[67] adds the temporal cost to matching cost according to previous disparity.

In summary, the disparity refinement step could fix the inconsistent disparities well, and improve the view synthesis quality for 3DTV applications.

2.2 View Synthesis

In 3DTV applications, view synthesis is one of the most important components to synthesize a single or multiple virtual view videos for the stereoscopic TV or the free-viewpoint TV [101]. A common approach for view synthesis is the depth-image-based rendering (DIBR) algorithm [51]-[57], which can warp a video to another view according to disparity maps.

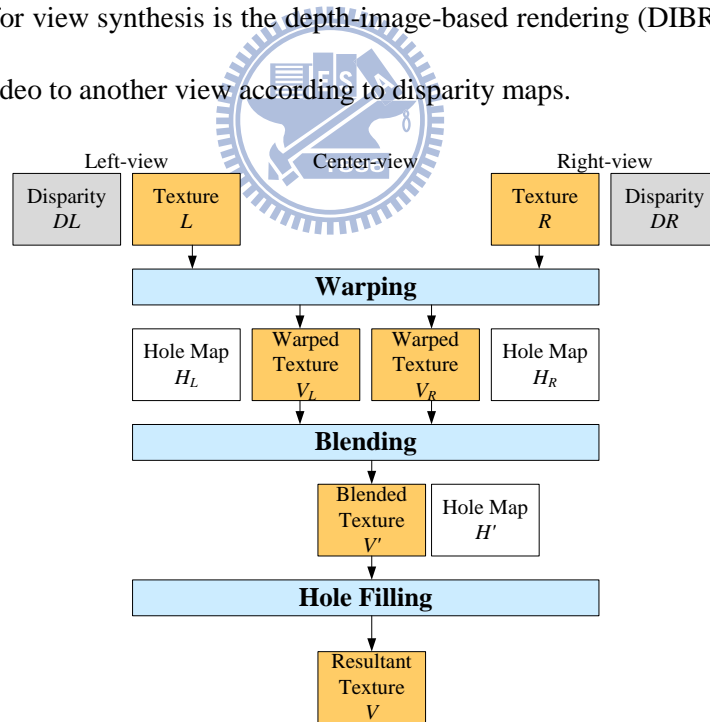


Figure II-12 General flow of view synthesis

A general DIBR algorithm could be divided into the three steps: warping, blending, and hole filling, as depicted in Figure II-12. For different number of input view, the DIBR algorithm has different challenges in its steps. With single-view input, the DIBR algorithm suffers from large

occlusion holes in the hole filling step, while with multiple-view inputs, it suffers from inconsistent warped pixels in the blending step. The concept and challenges of each step are presented in the following.

2.2.1 Warping

In Figure II-12, the warping step loads the textures and disparities of reference side-views generate the warped textures and hole maps of the target center-view. In the warping step, the reference textures are shifted to the target view according the reference disparity maps.

The methods of warping step can be classified into the one-step warping and the two-step warping as illustrated in Figure II-13. The one-step warping directly warps the reference textures to the target view according to the warping position of disparities, while the two-step warping first warps the target disparity and then uses it to synthesize the target texture. Rogmans *et al.* [58] and Morvan [59] show that the two-step warping could perform better because its sampling precision is higher.

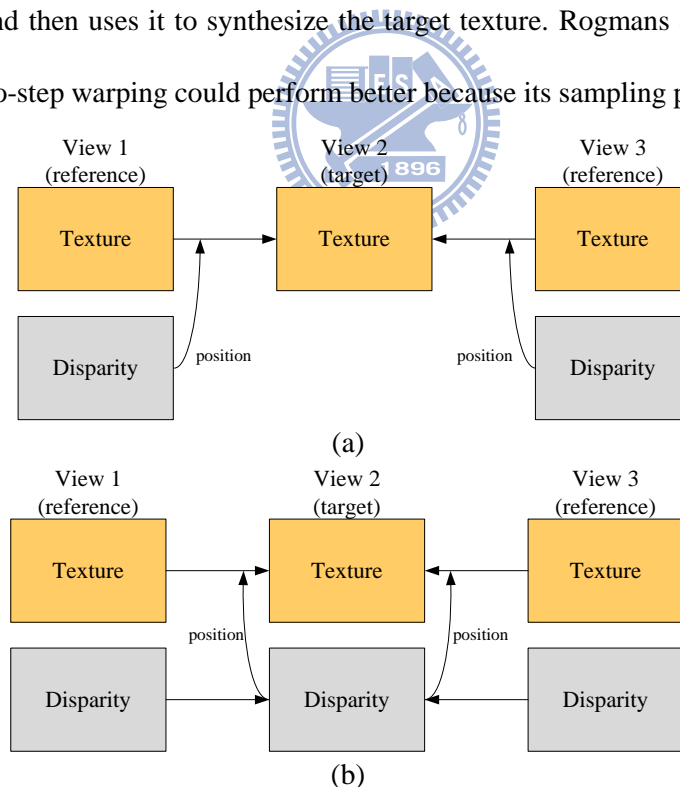


Figure II-13 Warping methods in view synthesis
(a) one-step warping, (b) two-step warping

2.2.2 Blending

With the multi-view inputs, the warping step will generate multiple textures for the target view as shown in Figure II-12. In other words, there are multiple warped pixels for a target position. However, the colors of these warped pixels are not consistent due to different radiometric gain and bias at different viewpoints. Therefore, the warped pixels should be blended by different methods for the three cases: visible pixel, occluded pixel, and disoccluded pixel, according to the hole maps. For the case of visible pixel, the pixel is labeled “non-hole” in hole maps, and could be seen at multiple viewpoints. Thus, its color can be computed by averaging the warped pixels. For the case of occluded pixel, the pixel is labeled “non-hole” in one hole map only, and could be seen at only one viewpoint. Thus, its color can refer to the only warped pixel. For the final case, the disoccluded pixel is labeled “hole” in all hole maps, and cannot be seen at any viewpoints. Thus, it should be handled in the next step. In addition, the hole regions can be dilated before blending to avoid the ghost artifact as shown in Figure II-14.



Figure II-14 Blending step in view synthesis
(a) without hole dilation, (b) with hole dilation

2.2.3 Hole Filling

With multiple-view inputs, most holes can be easily recovered by other views. For the remaining disoccluded holes, they can be filled by the advanced in-painting method [60]. On the other hand, with single-view input, the DIBR algorithm suffers from large occluded holes due to lack of other views.

The occluded holes can be handled by the disparity smoothing methods [52]-[55] to reduce hole sizes, and be filled by the interpolation method [53].

In summary, the 3DTV applications demand a view synthesis engine to generate virtual view videos, and the DIBR algorithm could satisfy this requirement through the above steps. However, the quality of view synthesis is highly dependent on the performance of disparity estimation. Therefore, it is necessary to develop a high-performance disparity estimation algorithm for the 3-D video production.

2.3 Review of DERS Algorithm from 3DVC

The 3D Video Coding (3DVC) team is organized in the Moving Picture Group Experts (MPEG) to support the associated techniques for 3DTV applications. The associated techniques include the disparity estimation, view synthesis, and multi-view video coding. The 3DVC team defines the configuration of input and output views for the 3DTV system, and delivers the reference software for disparity estimation [63] and view synthesis [64]. The algorithms in the reference software are respectively called DERS algorithm and VSRS algorithm. They also create a test bed and quality evaluation to assess the performance of 3-D videos. Furthermore, they combine the disparity estimation and view synthesis with the multi-view video coding (MVC) [107] for data compression and transmission. In this section, we introduce the 3DVC's DERS algorithm and point out its design challenges in the processing of high resolution videos. In addition, we present the 3DVC's I/O configuration and quality evaluation method, which are also adopted in this dissertation.

2.3.1 Input and Output View Configuration

The input and output setting is defined by the 3DVC [71] as shown in Figure II-15. In the 2-view configuration, the disparity estimation and view synthesis engines loads the original left-view and right-view videos to generate the virtual-view videos. Combining the synthesized video and one of the original videos can support the stereoscopic display. Figure II-15 (b) and (c) shows the 3-view

configuration. In which, two view videos are synthesized for the stereoscopic display. For the 9-view display, eight virtual-view videos need to be synthesized, and combined with the original center-view video. Based on the above configurations, the disparity estimation and view synthesis engines can be directly extended to support free viewpoint TV if more view videos are available.

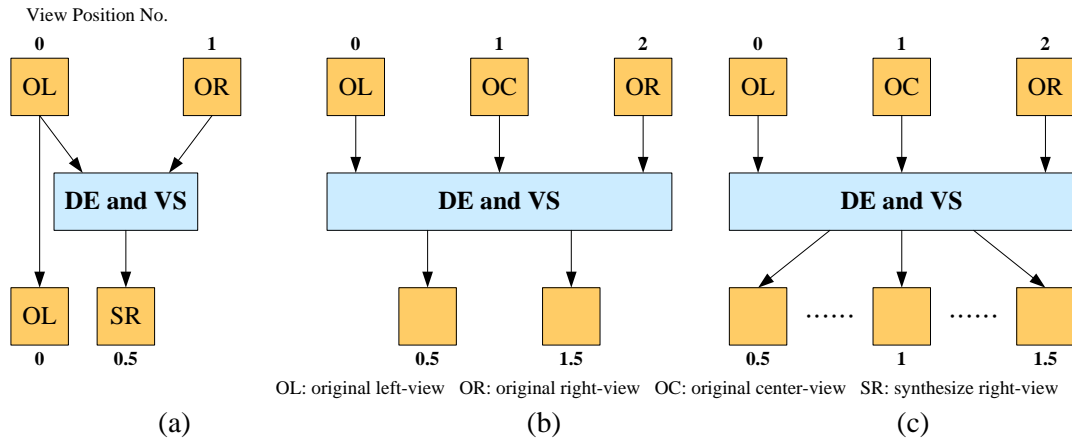
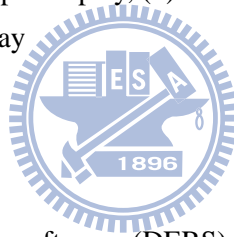


Figure II-15 Input and output view configuration defined by the 3DVC

(a) 2-view configuration for stereoscopic display, (b) 3-view configuration for stereoscopic display, (c) 3-view configuration for 9-view display

2.3.2 DERS Algorithm



The depth estimation reference software (DERS) algorithm [63] delivered by the 3DVC is illustrated in Figure II-16. The DERS algorithm uses the three view image frames to compute the center-view disparity map. In addition, the previous image frame and disparity map are also involved for the temporal consistency enhancement. Note that the DERS algorithm can support the input videos without rectification. The steps in the DERS algorithm are introduced in the following.

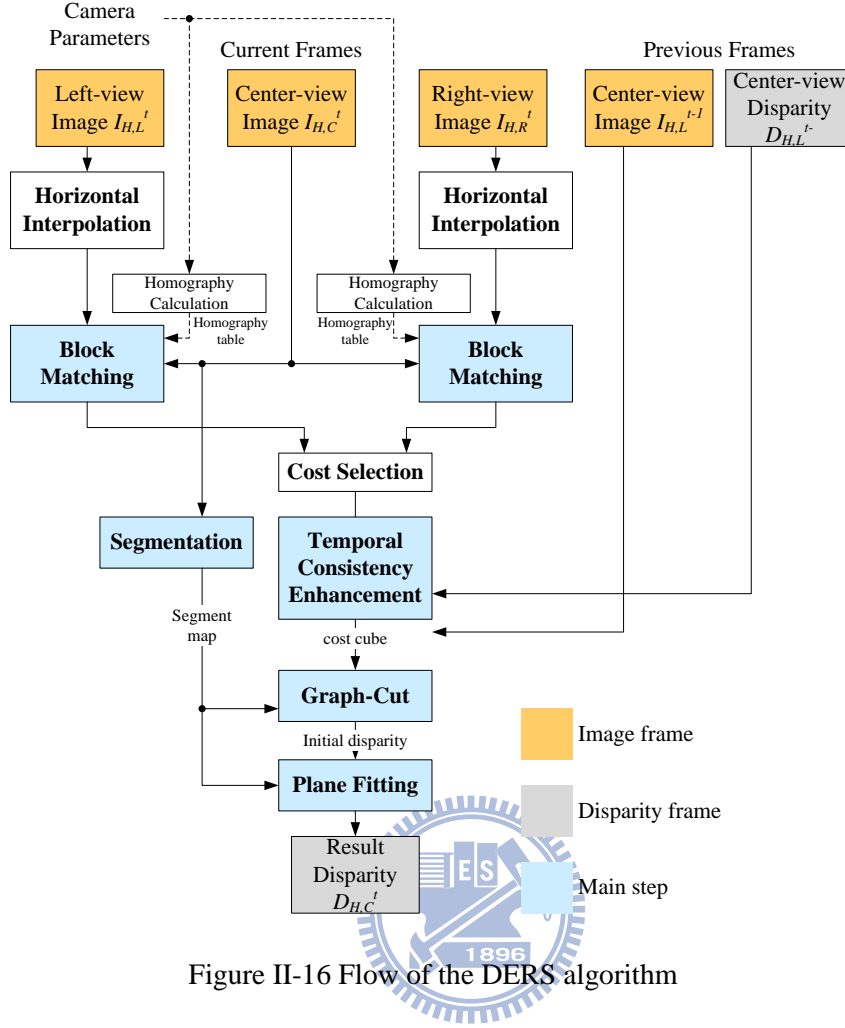


Figure II-16 Flow of the DERS algorithm

1. Initialization

Initially, the side-view images are scaled up by the horizontal interpolation method, which has the two options of half-pixel mode and quarter-pixel mode. The horizontal interpolation method includes the linear filter, cubic filter, and 6-tap interpolation filter in H.264/AVC standard. In addition, the homography matrix tables are calculated using the three-view camera parameters. With the homography matrix $\mathbf{H}(d)$, for a center-view pixel (u, v) , the positions of correspondence candidates (u', v') can be directly computed by

$$(u', v', 1)^T = \mathbf{H}(d)(u, v, 1)^T, \quad (\text{II-6})$$

where

$$\mathbf{H}(d) = \begin{bmatrix} h_{00}(d) & h_{01}(d) & h_{02}(d) \\ h_{10}(d) & h_{11}(d) & h_{12}(d) \\ h_{20}(d) & h_{21}(d) & 1 \end{bmatrix} . \quad (\text{II-7})$$

2. Block Matching

In the block matching, the left-view image and the right-view image are regarded as the reference views, and the center-view image is regarded as the target view. The block matching adopts the SAD match metric defined as

$$C_{SAD}(x, y, d) = \sum_{(u,v) \in \text{win}(x,y)} |I_{tar}(u, v) - I_{ref}(u', v')| , \quad (\text{II-8})$$

where the window size can be 1×1 or 3×3 , and the coordinates of the reference pixels can be computed by (II-6). In addition, the adaptive support-weight (ADSW) aggregation method [7] could be applied, and it is called soft-segmentation [68] in the DERS algorithm.

3. Temporal Consistency Enhancement

For the temporal consistency enhancement, the DERS algorithm [65], [66], [67] first detects the 16×16 motion block by the intensity difference of current and previous image frames. If the block color difference is high than a defined threshold, this block is regarded as a motion block. With the motion information, the temporal cost C_{temp} is computed by

$$C_{temp}(x, y, d) = \begin{cases} \tau_{temp} |d - D^{t-1}(x, y)| & , \text{no - motion block} \\ 0 & , \text{motion block} \end{cases} , \quad (\text{II-9})$$

where D^{t-1} is the previous disparity map, and τ_{temp} is a constant for scaling. The temporal cost C_{temp} is added to the block matching cost C_{SAD} by the equation,

$$C_{total}(x, y, d) = C_{SAD}(x, y, d) + C_{temp}(x, y, d) , \quad (\text{II-10})$$

With this temporal cost, the current disparity would be affected by the previous disparity.

4. Segmentation

The segmentation is performed only on the center-view image to assist the successive graph-cut and plane fitting. In the DERS algorithm, the segmentation method has the three options: mean-shift

segmentation [69], pyramid segmentation, and K-mean clustering, which apply the OpenCV library [70].

5. *Graph cut*

The DERS algorithm uses the fast GC approach [19], whose acceleration techniques include the swap method and the efficient augmenting path. In addition, the segment information calculated in previous step is also used to constrain the smoothness term in (II-5). In the DERS, the GC approach is performed for two iterations to obtain higher disparity quality.

6. *Plane Fitting*

Finally, by the segment information, the plane fitting mentioned in Section 2.1.2 is also adopted to refine the disparity map.

To sum up, by the general framework of disparity estimation, the DERS algorithm adopts the absolute difference (AD) for matching cost, the uniform weight for cost aggregation, and the GC approach for disparity optimization. Furthermore, it takes care of the temporal consistency and object consistency for the disparity refinement. In the DERS, the optional methods of all steps can be controlled by a configuration file. Note that the DERS algorithm can additionally support the semi-auto disparity estimation that needs a user-defined foreground map to increase the disparity quality. This approach is out of the dissertation scope.

2.3.3 Reference Software for 3-View Configuration

The functions of the DERS and VSRS algorithms are shown in Figure II-17 (a) where I_n is an image frame at viewpoint n , and D_n is a disparity map at viewpoint n . The DERS algorithm requires the three view image frames I_0, I_1, I_2 to calculate the disparity map D_1 , while the VSRS algorithm loads the two view image frames I_0, I_1 and disparity maps D_0, D_1 to synthesize the inter-view image frame $I_{0.5}$, which also can be another viewpoint between 0 and 1.

With the functions of DERS and VSRS, they have to be performed for several times for the 3-view configuration as shown in Figure II-17 (b). In which, five view image frames I_{-1} to I_3 are demanded for the DERS algorithm to compute the disparity maps D_0, D_1, D_2 . Then, the VSRS algorithm could use the three image frames I_0, I_1, I_2 , and disparity maps D_0, D_1, D_2 to generate the 9 view image frames. Compared to the original configuration in Figure II-15 (c), the DERS and VSRS algorithms additionally require two side-view image frames I_{-1} and I_3 .

Therefore, to meet the required function of 3-view configuration with minimum input views, our target disparity estimation engine would use only three view image frames to compute their corresponding disparity maps as shown in Figure II-17 (c).

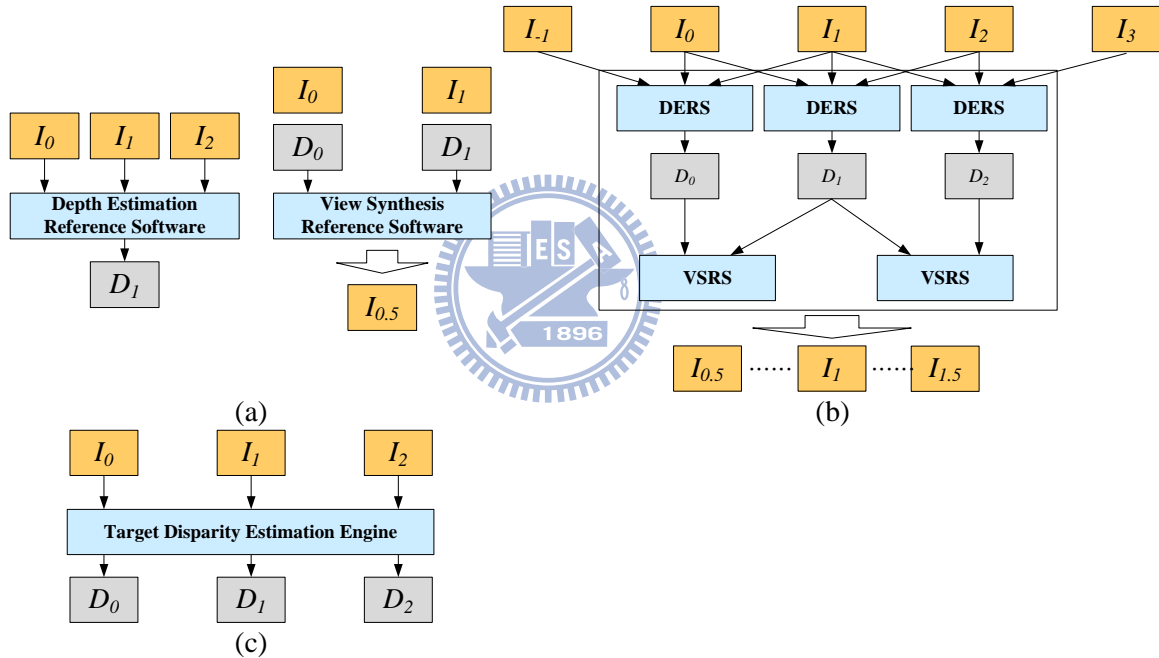


Figure II-17 Data flow for 3-view configuration

(a) functions of DERS and VSRS algorithms, (b) DERS and VSRS algorithms for 3-view configuration, (c) our target disparity estimation engine

2.3.4 Evaluation Method for Disparity Quality

To assess the disparity quality, the evaluation method in computer vision [72] is the disparity error rate that is computed by the difference of the estimated disparity map and a ground truth produced by the structured light method [73]. A disparity result would be considered as an error one if

the disparity difference is higher than a tolerance. However, different applications demand different disparity quality and the proper corresponding evaluation method.

For 3DTV applications, the disparity quality could be evaluated through the quality of view synthesis or the quality of multi-view video coding. This dissertation adopts the evaluation methods of view synthesis corresponding to our target application. The various evaluation methods for view synthesis analyze the frame difference between the synthesized and the really captured videos by different ways. The common-used evaluation methods are the peak signal-to-noise ratio (PSNR), the structural-similarity (SSIM) [74], and the peak signal-to-perceptible-noise ratio (PSPNR) [75], [76]. They are introduced as follows.

1. PSNR

The PSNR is commonly used in the video quality assessment, especially in video coding. The PSNR for the frame n is computed by

$$PSNR_n = 10 \log_{10} \left(\frac{255^2 HW}{\sum_{i=0}^{HW-1} |S_{i,n} - R_{i,n}|} \right), \quad (\text{II-11})$$

where S is the synthesized video, R is the reference video, and their subscripts (i, n) refers to the i th pixel in the n th frame. In the PSNR, the frame difference is analyzed by the mean square error (MSE).

2. SSIM

The SSIM considers the image structure information into the analysis of frame difference, because the human vision system is highly sensitive to the image structure. Thus, the SSIM analyze the frame difference within local region by comparing their pixels with the luminance and contrast normalization. The SSIM for the frame n is computed by

$$SSIM_n = \frac{1}{HW} \sum_{i=0}^{HW-1} \frac{(2\mu_{S_{i,n}}\mu_{R_{i,n}} + C_1) \times (2\sigma_{S_{i,n}R_{i,n}} + C_2)}{(\mu_{S_{i,n}}^2 + \mu_{R_{i,n}}^2 + C_1) \times (\sigma_{S_{i,n}}^2 + \sigma_{R_{i,n}}^2 + C_2)} \quad (\text{II-12})$$

where $\mu_{S_{i,n}}$, $\mu_{R_{i,n}}$ are the mean, $\sigma_{S_{i,n}}$, $\sigma_{R_{i,n}}$ are the standard deviation, and $\sigma_{S_{i,n}R_{i,n}}$ is the covariance. They are computed in an 11×11 window by

$$\mu_{S_{i,n}} = \sum_{j \in \text{window}(i)} w_j S_{j,n} , \quad (\text{II-13})$$

$$\mu_{R_{i,n}} = \sum_{j \in \text{window}(i)} w_j R_{j,n} , \quad (\text{II-14})$$

$$\sigma_{S_{i,n}} = \left(\sum_{j \in \text{window}(i)} w_j (S_{i,n} - \mu_{S_{i,n}})^2 \right)^{1/2} , \quad (\text{II-15})$$

$$\sigma_{R_{i,n}} = \left(\sum_{j \in \text{window}(i)} w_j (R_{i,n} - \mu_{R_{i,n}})^2 \right)^{1/2} , \quad (\text{II-16})$$

$$\sigma_{S_{i,n}R_{i,n}} = \sum_{j \in \text{window}(i)} w_j (S_{i,n} - \mu_{S_{i,n}})(R_{i,n} - \mu_{R_{i,n}}) , \quad (\text{II-17})$$

where the w_j is the weighting function. In the implementation of SSIM [77], the weight function adopts the Gaussian weight, and the constants C_1 , C_2 in (II-12) are 6.5025 and 58.5225, respectively.

3. PSPNR

The PSPNR focuses on not only the spatial quality in the above methods but also the temporal quality according to the human vision system. For the synthesized videos, the flicker artifact is the most noticeable noise, even if the flickering region is small. Thus, the PSPNR attempts to model the flicker artifact into the disparity quality evaluation. First, the spatial distortion (SD) is defined as

$$SD_{i,n} = S_{i,n} - R_{i,n} , \quad (\text{II-18})$$

which is the frame difference between the synthesized image frame S and the reference image frame R .

Then, by considering the spatial distortion visibility of human, the SD is converted to the perceptual spatial distortion (PSD) through the equation

$$PSD_{i,n} = \begin{cases} \text{Clip}(SD_{i,n} - VT, 0, ST - VT) & , \text{ if } SD_{i,n} \geq 0 \\ \text{Clip}(SD_{i,n} + VT, -(ST - VT), 0) & , \text{ if } SD_{i,n} < 0 \end{cases} . \quad (\text{II-19})$$

where VT is the visibility threshold as a lower bound of SD , ST is the saturation threshold as a upper bound of SD , and the function $Clip$ is for range truncation. The spatial distortion SD under VT is not perceptible in any background luminance, and the spatial distortion SD over ST is not distinguished by human.

With the perceptual spatial distortion PSD , the temporal noise TN is separately calculated for the motion regions and the static regions. For the static regions, there are four stages to describe a

temporal noise changing in successive frames as illustrated in Figure II-18. The temporal noise $TN_{i,n,S}$ is computed by the equation,

$$TN_{i,n,S} = \begin{cases} 0 & , \text{if } (i, n) \in \text{static stage} \\ PSN_{i,n-p} & , \text{if } (i, n) \in \text{maintaining stage} \\ PSN_{i,n} & , \text{if } (i, n) \in \text{increasing stage (type1)} \\ PSN_{i,n} + PSN_{i,n-1} & , \text{if } (i, n) \in \text{increasing stage (type2)} \\ PSN_{i,n-1} & , \text{if } (i, n) \in \text{declining stage} \end{cases} \quad (\text{II-20})$$

for the different stages.

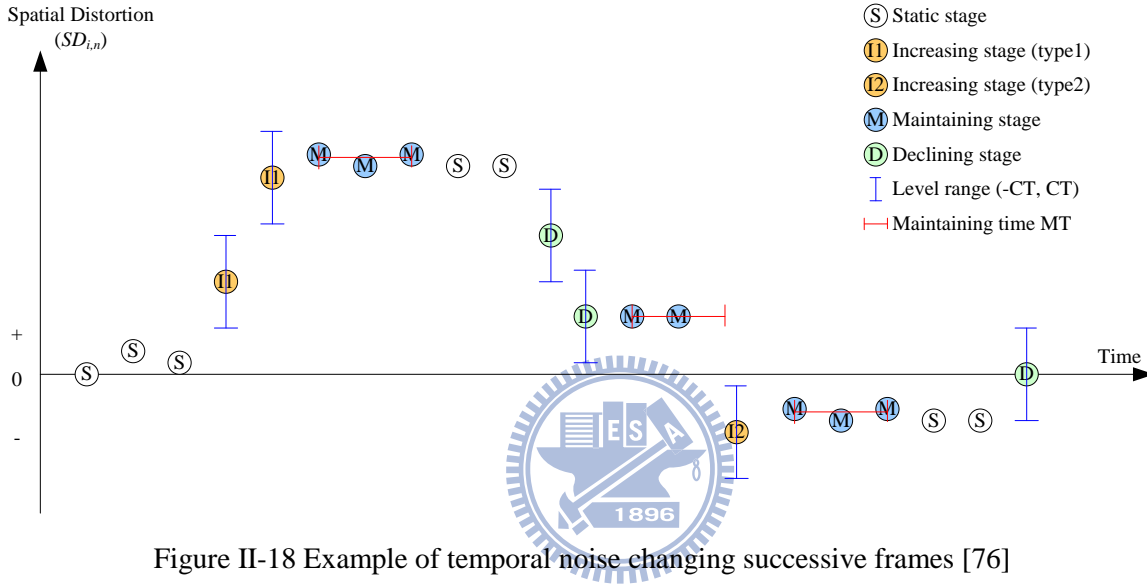


Figure II-18 Example of temporal noise changing successive frames [76]

In the static stage, a noticeable flicker artifact does not appear, and $SD_{i,n}$ does not change more than the specific level range CT , which has different values for different regions according to the distortion sensitivity of human. The level range CT is defined as

$$CT_{i,n} = \begin{cases} CT_{base} & \text{if } (i, n) \in \text{plain} \\ CT_{base}F_{edge} & \text{if } (i, n) \in \text{edge} \\ CT_{base}F_{texture} & \text{if } (i, n) \in \text{texture} \end{cases} . \quad (\text{II-21})$$

In the increasing stage, a noticeable flicker artifact appears, and $SD_{i,n}$ changes more than CT . If $SD_{i,n}$ has no polarization change, the temporal noise would increase with the increasing spatial distortion, and it is equal to $PSD_{i,n}$ for the type 1. On the other hand, if the $SD_{i,n}$ also has polarization change, the temporal noise should increase larger than spatial distortion, and also be larger than the other stages. Thus, the temporal noise for the type 2 is the sum of $PSD_{i,n}$ and $PSD_{i,n-1}$.

In the maintaining stage, the flicker artifact just disappears, and human still perceives the artifact for maintaining time (MT). Thus, the temporal noise should be $PSD_{i,n-p}$, which is propagated from the previous frame p in the increasing stage or declining stage. The MT is defined as 1/4 sec according to the study of human perception on flickers.

In the declining stage, the flicker artifact starts to disappear, but human suffers from the previous spatial distortion. Thus, the temporal noise is equal to the previous frame $PSD_{i,n-1}$.

On the other hand, for the motion regions, the temporal noise $TN_{i,n,M}$ is computed by

$$TN_{i,n,M} = \begin{cases} PSD_{i,n} & , \text{if } |PSD_{i,n}| \geq |PSD_{i,n-1}| - CT_{i,n} \text{ and } |PSD_{i,n}| > CT_{i,n} \\ 0 & , \text{otherwise} \end{cases} \quad (\text{II-22})$$

The motion region natively has change between two successive frames. If the change is higher than CT , it would be regarded as temporal noise. With the temporal noises $TN_{i,n,S}$ for static regions and $TN_{i,n,M}$ for motion regions, the sum of temporal noise for a frame is computed by

$$STN_n = \sum_{i=0}^{H \times W - 1} TN_{i,n} \quad (\text{II-23})$$

The final temporal peak signal-to-perceptible-noise ratio for whole synthesized videos is defined as

$$T_PSPNR = 10 \log_{10} \left(\frac{255^2 \times H \times W}{\sum_{f=1}^{F-1} STN_n / (F-1)} \right) \quad (\text{II-24})$$

The implementation of PSPNR is attached in the DERS [63].

2.3.5 Design Challenges

The DERS algorithm can deliver high quality disparity maps to support the view synthesis for 3DTV applications. However, it suffers from the following design challenges, especially for the requirement of high-definition videos.

1. Irregular Image Access in Block Matching

In the DERS algorithm, the block matching suffers from the irregular image access because the input videos are not rectified. Figure II-19 shows an example of block matching performed in the non-rectified images. In which, the epipolar lines in the target view are parallel, and they become

oblique ones in the reference view. The reference block would frequently result in memory row miss if the input videos are configured by one image rows in one memory row. Therefore, it is necessary to apply the rectification to the pre-processing, so that all the image accesses are regular by raster-scan order in the disparity estimation and the view synthesis.

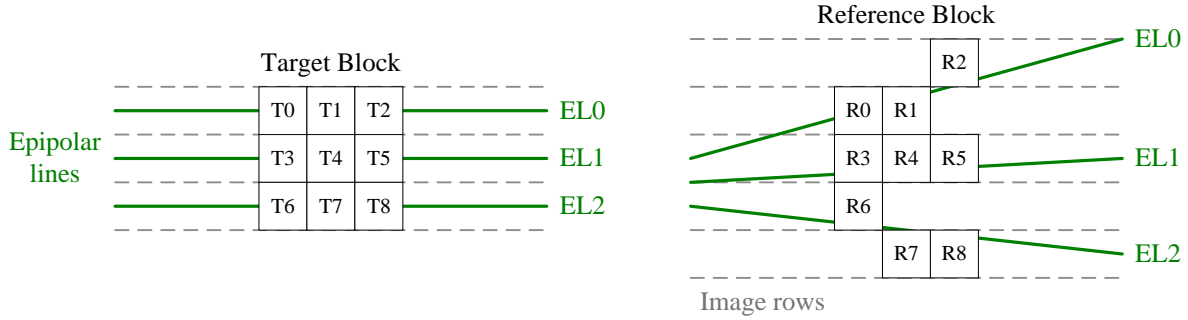


Figure II-19 Example of block matching in the DERS algorithm

2. Low Parallelism in Graph-Cut

The DERS algorithm adopts the fast GC approach [18], [19], which contains the swap method, and the efficient augmenting path method. However, the GC approach need to process on a tree-structural graph, and the connection of edges is frequently and irregularly changed. In addition, its computation has high data dependency because the graph is sequentially processed node by node. Therefore, the GC approach suffers from low parallelism in data access and computation

As mentioned in Section 2.1, the previous work [22] proposed the region-based push-relabeling approach that could increase the parallelism and the data locality for the GC approach. However, its computation and data access in a local region still suffers the same design challenges as the original GC approach. In addition, the real-time scalable GC engine [23] is proposed but it only supports a small graph with 16 nodes. To sum up, the GC approach is not suitable to be accelerated for our target performance. We will develop another new disparity estimation algorithm based on the DP or BP algorithm, which is natively high parallelism.

3. High Computational Complexity in Segmentation

In the DERS algorithm, the segmentation is used in the plane-fitting to enhance the object consistency, and in the GC approach to constrain the optimization process. In the previous work [50], the K-mean clustering method was implemented by VLSI design to achieve the throughput of 1 Mpixels/s by 440K gate counts. However, the hardware cost would be dramatically increased to support our required throughput of three view HD1080p videos in 60 frames/sec (i.e. 360 Mpixels/s).

2.4 Summary

In this chapter, we review the disparity estimation algorithms by a general framework, and introduce the associated view synthesis technique. In addition, we also present the state-of-the-art DERS algorithm delivered by the MPEG 3DVC. The DERS could produce high-quality disparity maps for the view synthesis application, but it suffers from the three design challenges: irregular image access in block matching, low parallelism in graph-cut, and high computational complexity in segmentation. Thus, the DERS algorithm could not be accelerated to achieve our target throughput by the VLSI design. Therefore, the goal of this dissertation is to develop a new high-quality and hardware-efficient disparity estimation algorithm, and implement its dedicated VLSI design to reach our target throughput, three view HD1080p disparity maps at 60 frames/sec.

III Baseline Disparity Estimation with Belief Propagation and Joint Bilateral Filter for High Definition 3DTV Applications

For the high definition 3DTV application, the disparity estimation natively suffers from high computational complexity due to large frame size. To conquer it, our strategy is to calculate the disparity map by a belief propagation-based algorithm in low resolution, and scale it to high resolution disparity map by an upsampling algorithm. For the two steps, we adopt the baseline belief propagation (BP) algorithm [24] and the joint bilateral upsampling (JBU) algorithm [79], [81]. The combination is called *baseline algorithm* in this dissertation.

The chapter is organized as follows. First, we introduce the BP and JBU algorithms. Then, we separately analyze their architecture, and design the key components to solve their major design challenges. Finally, the experimental result of the baseline algorithms is demonstrated by software implementation.

3.1 Introduction

3.1.1 Baseline Belief Propagation

The concept of the BP-based algorithm is illustrated in Figure III-1. In the BP-based algorithms, an energy function is generally formulated as

$$E(\mathbf{d}) = \sum_{i \in I} D(d_i) + \sum_{i \in I, j \in \text{Neighbor}(i)} V(d_i, d_j) \quad (\text{III-1})$$

for a 2-D graph in Figure III-1 (a). In this energy function, D is the data cost for each node corresponding to each pixel, V is the smoothness cost for each edge, and \mathbf{d} of the energy E is a selected disparity set for all nodes. The two costs can constrain selecting the disparity set \mathbf{d} . The data cost D

enforces that the correspondences are similar, and the smoothness cost V enforces that the neighboring nodes' disparities are consistent. To minimize the energy function and acquire an appropriate disparity set \mathbf{d} , the BP-based algorithms perform an iterative process called message passing. However, the shortage of the BP-based algorithms is that the energy function may not be convergent definitely. Nevertheless, the disparity map could approach to a steady state after sufficient iterations.

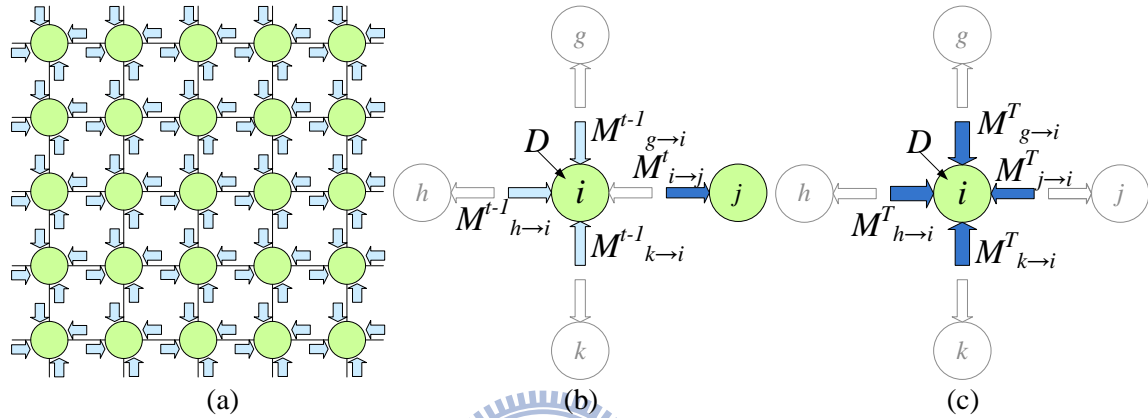


Figure III-1 Illustrations of BP
(a) node plane; (b) message passing; (c) belief calculation.

For the requirement of real-time processing, the direct hardware implementation of BP-based algorithms suffers from two design challenges: high computational complexity and storage in the message passing. For the example of $640 \times 480 @ 30\text{fps}$ and the disparity range of 32, the computational complexity is about 1,200 billion operations per second for the message passing, and the storage is about 157Mbytes for messages.

To address above problems, various approaches have been proposed. Felzenszwalb and Huttenlocher [25] proposed an efficient message passing to reduce computational complexity from $O(L^2)$ to $O(L)$, and the bipartite message approach to reduce 50% memory cost. Following their approach, Yang *et al.* [27] implemented it on a high performance GPU, and Park *et al.* [28] also designed an array processing architecture on two FPGA boards to achieve the performance of $320 \times 240 @ 30\text{fps}$ but with 880KB on-chip memory. Cheng *et al.* [29]-[33] proposed a tile-based BP and a fully parallel architecture for each message passing processing element (PE) to reach real-time

processing for the image size of 640x480. Nevertheless, all the implementation still suffers from high memory cost.

In summary, though previous work used parallel PEs to conquer the high complexity, the resulted logic still occupies too much area since each PE needs high area cost. In addition, all the work did not solve the memory cost well due to their fixed memory access approach.

To solve the mentioned problems, we propose a hardware efficient architecture for various BP-based algorithms through three techniques in Section 3.2. For the high memory cost, we propose a spinning-message approach which rearranges the message configuration in an internal memory to save 50% memory cost. In addition, we propose a sliding-bipartite node plane that combines the advantages of previous work to further reduce more memory cost. For the message passing PE, we propose a buffer-free PE architecture which removes all the large buffers and shares common operators to reduce logic cost without significant speed degradation. Both the proposed low memory access approaches and the buffer-free PE architecture could be applied to various BP-based algorithms together to significantly reduce their hardware cost as well as speed up to real-time processing without changing their disparity accuracies.

3.1.2 Joint Bilateral Upsampling

The JBU algorithm [79] is proposed to scale up the various results of image processing, such as tone mapping, colorization, photomontage, disparity map, and etc. The main idea of JBU is to apply a high resolution image to guide the upsampling process. For upsampling a disparity map, given the high resolution image I_H and the low resolution disparity map D_L , the high resolution disparity map D_H can be computed by

$$D_H(c) = \frac{1}{\kappa} \sum_{q_L \in S} D_L(q_L) \cdot f(\|c_L - q_L\|) \cdot g(\|I_H(c) - I_H(q)\|) , \quad (\text{III-2})$$

where f is the spatial kernel with the argument of spatial distance $\|c_L - q_L\|$ in low resolution, and g is the range kernel with the argument of color distance $\|I_H(c) - I_H(q)\|$ in high resolution. Note that the

positions c, q are in the high resolution frame, and the positions c_L, q_L are their corresponding positions in the low resolution frame. Both the two kernels are Gaussian weight function. In addition, κ is the sum of weights for normalization, and S is the window of spatial kernel.

Based on the original JBU algorithm, various modified JBU algorithms are proposed with different equation. Chan *et al.* [80] proposed the noise aware filter depth upsampling (NAFDU), which adds the range kernel h for low resolution image to reduce the texture copy artifact. The equation of NAFDU is defined as

$$D_H(c) = \frac{1}{\kappa} \sum_{q_L \in S} D_L(q_L) f(\|c_L - q_L\|) [\alpha g(\|I_H(c) - I_H(q)\|) + (1 - \alpha) h(\|I_L(c_L) - I_L(q_L)\|)], \quad (\text{III-3})$$

where α is blending value related to the disparity variance. Using the additional h , the JBU algorithm could resist the texture copy artifact due to its color distance in sampled frame. Thus, the effect of h is increased for the region with low disparity variance. In contrast, the effect of g is increased for the region with high disparity variance.

On the other hand, Riemens *et al.* [81] proposed the multi-step JBU algorithm that doubles the resolution of disparity map in each iteration. This approach can reduce the computational complexity by decreasing the window size of spatial kernel. In addition, the equation (III-2) is changed to

$$D_H(c) = \frac{1}{\kappa} \sum_{q_L \in S} D_L(q_L) \cdot f(\|c_L - q_L\|) \cdot g(\|I_L(c_L) - I_H(q)\|), \quad (\text{III-4})$$

where the high resolution pixel $I_H(q)$ of (III-2) is replaced with the low resolution pixel $I_L(c_L)$. The fast multi-step JBU algorithm was implemented by a programmable DSP platform to achieve the throughput of 720x576@50fps [82]. However, it is far from our target throughput due to the limited resource in DSP platform.

For the above different JBU algorithms, their computational characteristics are the same as the joint bilateral filtering (JBF), which is an extended version of bilateral filter (BF). The BF and the JBF are respectively defined as

$$I'(c) = \frac{1}{\kappa} \sum_{q \in S} I(q) f(\|c - q\|) g(\|I(c) - I(q)\|) \quad (\text{III-5})$$

and

$$J'(c) = \frac{1}{\kappa} \sum_{q \in S} J(q) f(\|c - q\|) g(\|I(c) - I(q)\|) . \quad (\text{III-6})$$

Therefore, the existing acceleration approaches for JBF and BF could be applied to the JBU algorithm. The state-of-the-art approaches proposed by Yang *et al.* [83] and Porikli [91] can achieve constant time complexity. But they suffer from extremely high memory cost. This dissertation focuses on the Porikli's approach for JBF because we could take advantage of its computational characteristic of single iterative raster-scan to reduce its memory cost.

The following two sections will analyze the computation of the baseline BP algorithm and the JBF algorithm, and propose an architecture design for the key components to solve their design challenges.

3.2 Analysis and Design of Baseline Belief Propagation

In this section, we first review the BP-based algorithms and points out their design challenges. Then, we present the proposed low memory access approaches, and elaborate the buffer-free PE architecture for message passing module, which is the most important component in BP-based algorithm. Finally, the implementation results and comparisons are demonstrated.

3.2.1 Analysis of Belief Propagation

In this sub-section, we review various BP-based algorithms and then indicate the general design problems in these algorithms.

1. Baseline BP

Sun *et al.* [24] first applied BP to disparity estimation. This baseline BP includes three steps: data cost calculation, message passing, and disparity selection, which are performed in the graph of Figure

III-1 (a). In this dissertation, the graph is called node plane whose size equals to an image in the baseline BP.

In the baseline BP, the first step is to calculate the data cost of each node, where the data cost is identical to the matching cost in local approaches. According to the data cost, local approaches can determine disparity maps using the winner-take-all scheme. In contrast, the baseline BP further propagates it to neighboring nodes.

In the second step, the messages, which are the arrows in Figure III-1 (a), are added around all nodes, and they propagate data cost to neighboring nodes. In the baseline BP, the propagating mechanism is called message passing. Figure III-1 (b) illustrates the message passing for calculating a new message, and its equation is as follows:

$$M_{i \rightarrow j}^t(d_j) = \frac{1}{\kappa} \min_{d_i} \left(V(d_j, d_i) + D(d_i) + \sum_{x \in \text{Neighbor}(i) \setminus j} M_{x \rightarrow i}^{t-1}(d_i) \right) \quad (\text{III-7})$$

where $M_{i \rightarrow j}^t$ is a new message of the node j at the iteration t from the node i , and $M_{x \rightarrow i}^{t-1}$ is an old message of the node i at the iteration $t-1$ from the nodes x which can be g , h , and k . In addition, V and D are smoothness cost and data cost in (III-1), and κ is a normalization term. Note that the indexes d_i and d_j are respectively for the nodes i and j . To calculate the new message $M_{i \rightarrow j}^t$, the three old messages $M_{g \rightarrow i}^{t-1}$, $M_{h \rightarrow i}^{t-1}$, and $M_{k \rightarrow i}^{t-1}$ are summed up with D by the index d_i . Then the result is convoluted with V by the cross indexes d_j and d_i . For the message passing in BP-based algorithms, the computation of (III-7) is performed on all four incoming messages of each node iteratively.

In the third step, the final incoming messages of each node are accumulated with its D to form a belief. The belief is used to determine a disparity by the following equation, and its illustration is shown in Figure III-1 (c).

$$d = \arg \min_i \left(D(d_i) + \sum_{x \in \text{Neighbor}(i)} M_{x \rightarrow i}^T(d_i) \right) \quad (\text{III-8})$$

In summary, the baseline BP alternates the initial data cost with the belief deriving from the message passing to deliver better disparity maps.

The major computational complexity of the baseline BP is in the message passing, and that is $O(4HWL^2T)$, where H and W are the height and width of the node plane, L is the disparity range, and T is the iteration count. The computation of the message passing can be undertaken by parallel PEs as shown in Figure III-2. These PEs use the nodes' data at the previous iteration to calculate new messages for the next iteration. With sufficient parallel PEs, the baseline BP could achieve real-time speed. However, that will result in high logic cost. In addition, high memory cost is also incurred since all the messages in the node plane have to be stored.

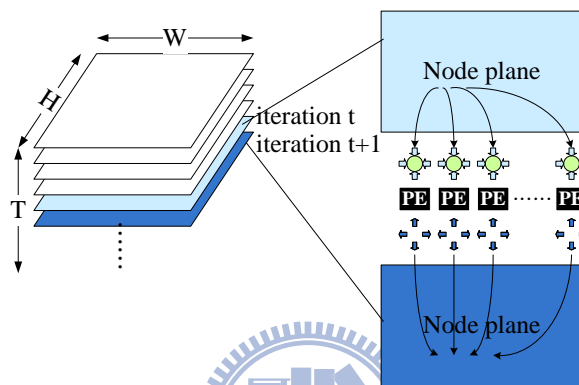


Figure III-2 Configuration of the message passing PEs

2. Various BP-based Algorithms

Based on the baseline BP, various BP-based algorithms have been developed recently to address the mentioned problems from the algorithm level.

To reduce the computational complexity, Felzenszwalb and Huttenlocher [25] proposed the hierarchical BP that downsamples the node plane to multiple resolutions and then performs the message passing from coarser levels to finer levels. Because the messages in the coarser levels could propagate data cost to farther nodes and become initial messages for the next level, the disparity maps could converge faster than the baseline BP. Therefore, the hierarchical BP could take less time and deliver better disparity maps than the baseline BP.

To reduce the memory cost, our previously proposed block-based BP [36] partitions the node plane into independent blocks. The memory cost is significantly reduced from image-scale to block-scale, so that all data in the message passing can be placed in an internal memory, instead of an

external memory. However, its disparity maps would suffer from blocky artifact. Furthermore, Cheng *et al.* [29] proposed the tile-based BP to improve the blocky artifact. In contrast with the independent blocks, the tile-based BP preserves the boundary messages of each tile in an external memory to link blocks.

For all the above algorithms, their computation shares the same feature: the message passing performed in a rectangular node plane. For example, the node plane is image-scale in the baseline BP and the hierarchical BP, and block-scale in the block-based BP and the tile-based BP. Therefore, in the following we will show how to develop techniques for a rectangular node plane that can be applied to various BP-based algorithms.

3.2.2 Proposed Low Memory Cost Access Approach

In a rectangular node plane, the memory cost is constituted of the messages and the data cost. In this dissertation, we focus on the messages, which occupy the most of the cost. A straightforward memory access approach for the messages is the ping-pong buffer approach, which needs a pair of node planes and requires $8HWL$ memory. Unfortunately, this cost is too large to be on-chip. Even if the messages are stored in an external memory, its required bandwidth is still impractical, especially for the image-scale node planes.

1. Previous Work

To reduce the memory cost of messages, Yu *et al.* [35] compressed the messages by the envelope point transform method that can achieve eight times compression without significant degradation of disparity maps. However, this compression method needs the overheads of compression and decompression.

On the other hand, much previous work focuses on the computing order of message passing on the node plane to resize the node plane for memory cost reduction. Park *et al.* [34] proposed the fast BP structure approach which resizes the pair of node planes from HW to TW , where T is usually

smaller than H . In our previous work [38], we proposed the in-place message update approach that resizes one of the pair node planes from HW to $3W$ for buffering partial new messages temporally. Felzenszwalb and Huttenlocher [25] delivered the bipartite scan which only needs one node plane, and can also reduce computation to half. Different from above computing orders, Szeliski *et al.* [26] proposed the BP-M scan which updates messages direction by direction for whole node plane to accelerate convergence speed, and only needs one node plane. Although the BP-M scan can converge faster than others, the memory cost of BP-M scan is still too high and could not be further reduced because of its iterative directional process and overlapping data lifetime in all messages. Thus, the BP-M scan is not discussed in this dissertation.

Excluding the BP-M scan, the memory access in the previous work belong to the fixed memory access approach which binds messages at fixed memory positions, and thus would limit the possibility to reduce memory cost. Figure III-3 shows the data dependency of the traditional fixed memory access approach between successive iterations in a simplified 1-D node line, where each square represents a memory position, the arrow inside the square represents a stored message, and the cross line linking two messages (e.g. m_3 at t_1 to m_2 at t_2) represents that they have data dependency. In the traditional approach, each node's messages are stored at fixed memory positions. For example, the node n_3 's messages m_3 are always located at the same memory position pos_3 in all iterations. These messages m_3 are used to calculate the neighboring nodes n_2 's and n_4 's new messages m_2 and m_4 for next iterations. However, the new messages cannot overwrite their old ones at the memory position pos_2 and pos_4 since their old ones are still needed for new messages computation at other nodes. Thus, an access conflict would occur between the old and new messages of the neighboring nodes. To solve the access conflict, a straightforward method is to allocate an additional memory to buffer the new messages, but it will increase extra cost.

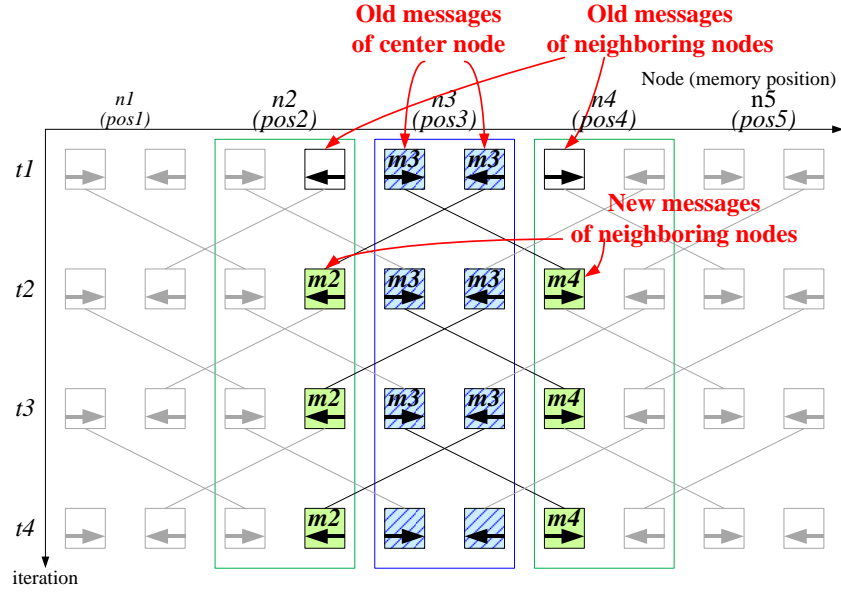


Figure III-3 Traditional fixed memory access approach in a 1-D node line for node $n3$ computation

2. Spinning-Message Approach

To address the access conflict and reduce memory cost, we propose the spinning-message approach that frees the bind between the messages and the memory positions, and eliminates the extra memory. In addition, the proposed approach could be applied to the reduction techniques mentioned in previous sub-section to further save 50% memory cost.

Figure III-4 (a) shows the main idea of the proposed approach. The old messages of the center node are used to calculate the new messages of the neighboring nodes, and their data life time is ended. Therefore, the new messages of the neighboring nodes can overwrite the outdated messages without access conflict, and are stored at the center memory positions instead of the neighboring memory positions.

Based on the main idea, Figure III-4 (b) shows the details of the proposed spinning-message approach by a 1-D node line for the node $n3$ as an example. Other nodes follow the same procedure. At the iteration $t1$, the messages $m3$ are stored at the center memory position $pos3$ that is the centralized mode. For the transition to the iteration $t2$, the messages $m3$ are used to calculate the new messages $m2$ and $m4$ of the neighboring nodes $n2$ and $n4$. The old messages $m3$ can be replaced by the new messages at the center memory position $pos3$ without the access conflict. After the calculation

and replacement, the centralized mode changes to the distributed mode since every node's messages are distributed at its neighboring memory positions (e.g. $m3$ at $pos2$ and $pos4$) at the iteration $t2$. Then, the distributed messages $m3$ are used to calculate the new messages $m2$ and $m4$, and the distributed messages $m3$ can also be replaced by the new messages without the access conflict. With another calculation and replacement, every node's messages are returned to the centralized mode at the iteration $t3$.

In summary, the messages are centralized at their own memory positions for odd iterations and distributed at their neighboring memory positions for even iterations. With this approach, we can save the memory while avoid the access conflict. Figure III-5 shows the proposed approach extended to a 2-D node plane.

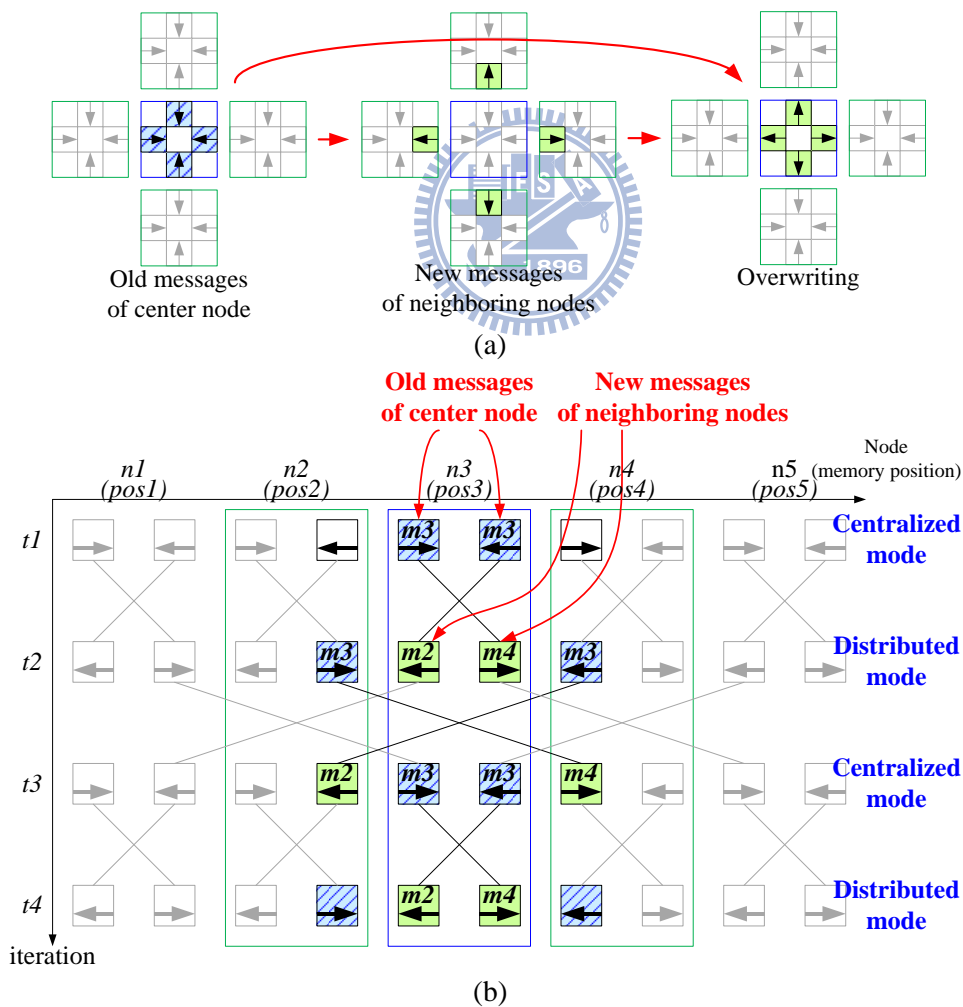


Figure III-4 Proposed spinning-message approach

(a) main idea; (b) memory access in a 1-D node line for node $n3$ computation.

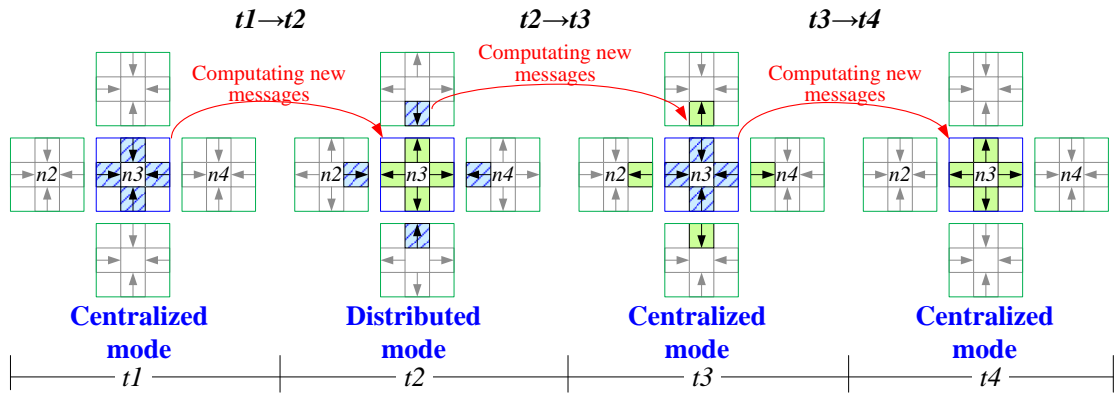


Figure III-5 Proposed spinning-message approach in a 2-D node plane for node n_3 computation

3. Applications

The proposed spinning-message approach can be applied to different types of node plane to further reduce their memory cost.

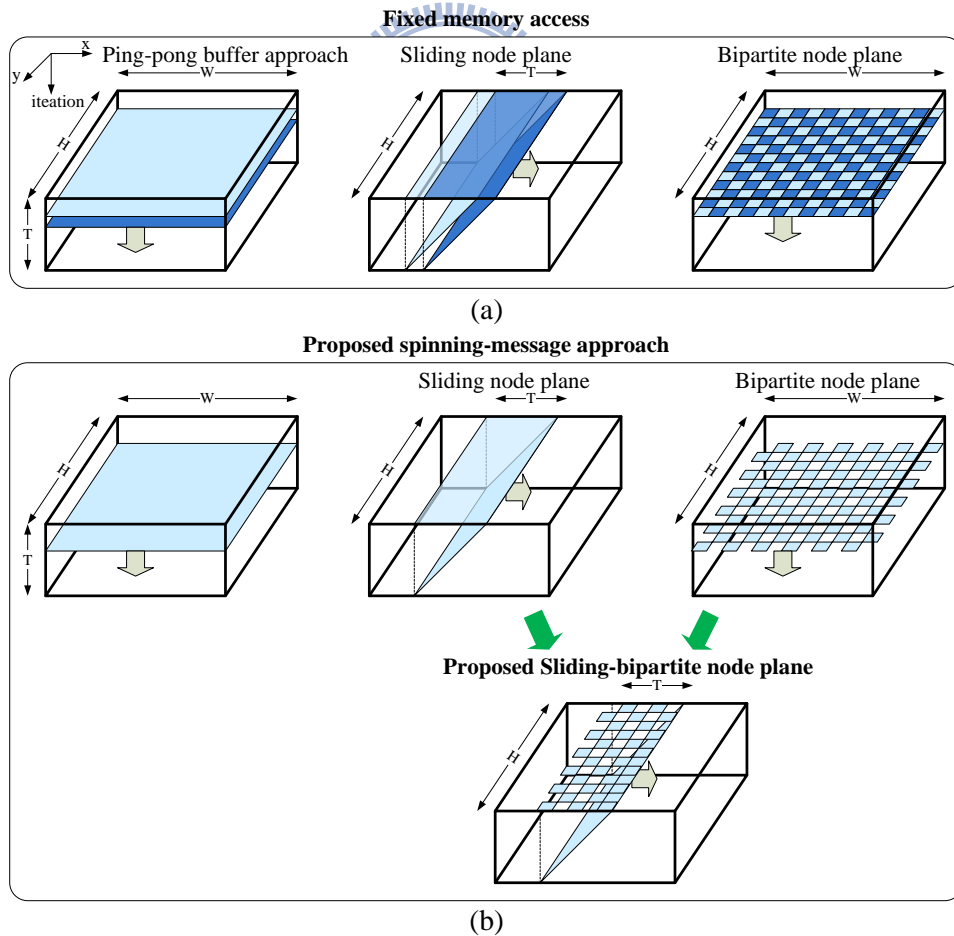


Figure III-6 Comparison of memory access approaches in different node planes
 (a) proposed spinning-message approach, (b) traditional fixed memory access approach

(1) Sliding Node Plane

In the original BP, the messages in a node plane are iteratively updated by the space-first (x-y plane) computing order, and the node plane moves along the iteration axis as shown in the ping-pong buffer approach of Figure III-6 (a). In contrast, the sliding node plane moves orthogonal to the iteration axis, and their messages are updated by the iteration-first computing order. The size of sliding node plane is its projective area on the x-y plane, which is smaller than the original node plane.

Figure III-7 shows three sliding directions. In which, the sizes of node planes are WT for the vertical sliding and HT for the horizontal sliding, and the diagonal sliding. The vertical sliding node plane was proposed by the fast BP structure approach in [28]. However, its size is larger than the other two because W is usually larger than H . Therefore, we recommend the horizontal sliding node plane, which totally requires $8HTL$ memory for messages.

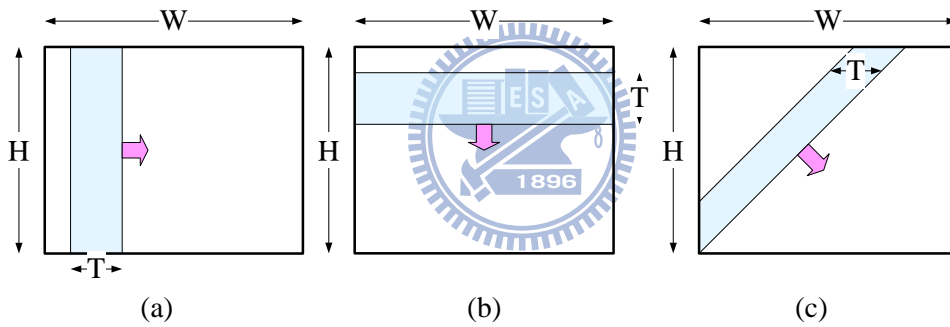
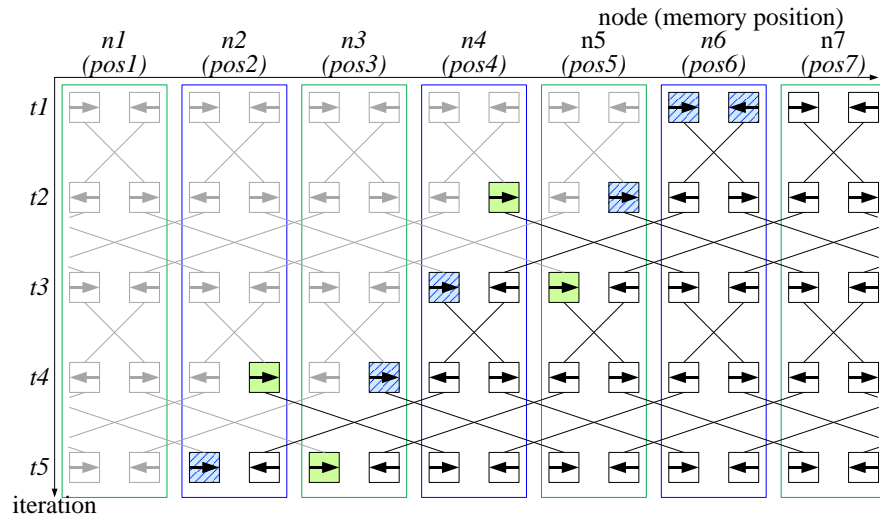


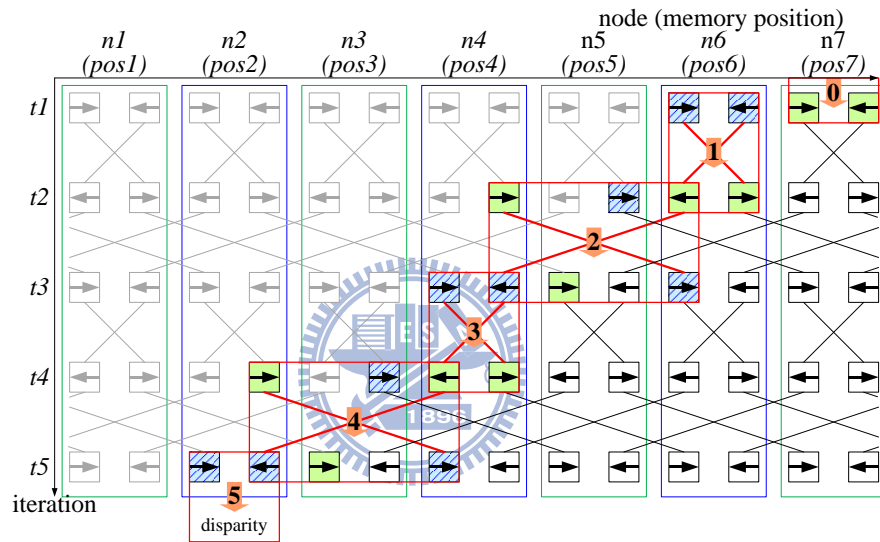
Figure III-7 Sliding node plane in different directions

(a) vertical sliding; (b) horizontal sliding; (c) diagonal sliding.

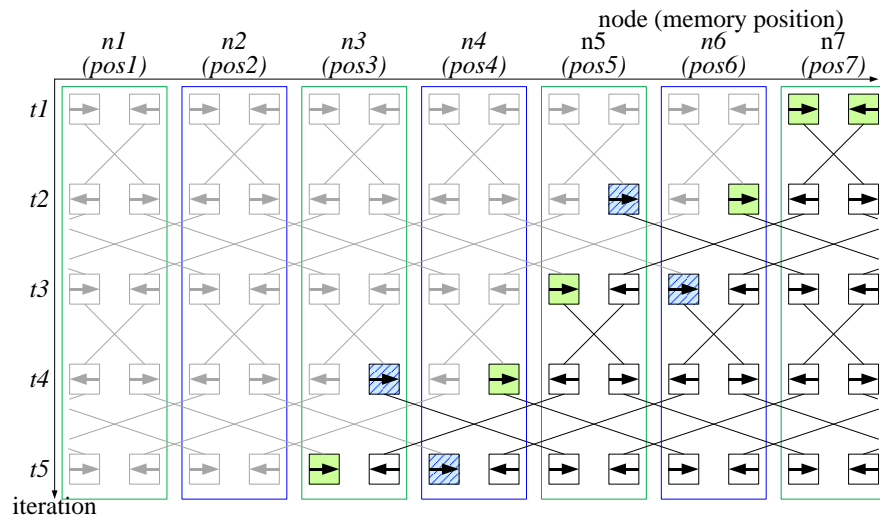
The memory cost can be further reduced to $4HTL$ by the proposed spinning-message approach as shown in Figure III-6 (b). Figure III-8 shows the details of the spinning-message approach performing on the horizontal sliding node plane. The initial state of the messages is shown in Figure III-8 (a), where the front of the node plane arrives at the node $n6$. Then, in Figure III-8 (b), the new messages in the node plane are computed from the node $n7$ to $n2$ step by step. With the spinning-message approach, the new messages can overwrite the old messages at the same memory positions. After that, in Figure III-8 (c), the front of node plane will slide to the node $n7$. According to the above flow, the spinning-message approach could cooperate with the sliding node plane well to further save 50% memory cost.



(a)



(b)



(c)

Figure III-8 Sliding node plane with the spinning-message approach

(a) the node plane slides to the node n_6 ; (b) the computing order of the message passing; (c) the node plane slides to the node n_7 .

(2) Bipartite Node Plane

The bipartite node plane was proposed in [25] that divides nodes into two parts, like a chessboard as shown in Figure III-6 (a). In which, one part is computed at odd iterations, and the other is computed at even iterations. Its memory cost is reduced from a pair of node planes in ping-pong buffer approach to only one node plane of $4HWL$.

Above memory cost can be further reduced to $2HWL$ by the proposed spinning-message as shown in Figure III-6 (b). Figure III-9 shows the spinning-message approach performs on the bipartite node plane at odd iterations and even iterations. At the odd iteration in Figure III-9 (a), the messages of the white nodes are used to calculate the new messages of the black nodes, and these messages of the black nodes can overwrite those of the white nodes. Then the state of node plane is transformed to Figure III-9 (b). Similarly, the messages at the even iteration can be returned to the next odd iteration. Thus by the spinning-message approach, only the white nodes need memory, and 50% memory cost can be saved.

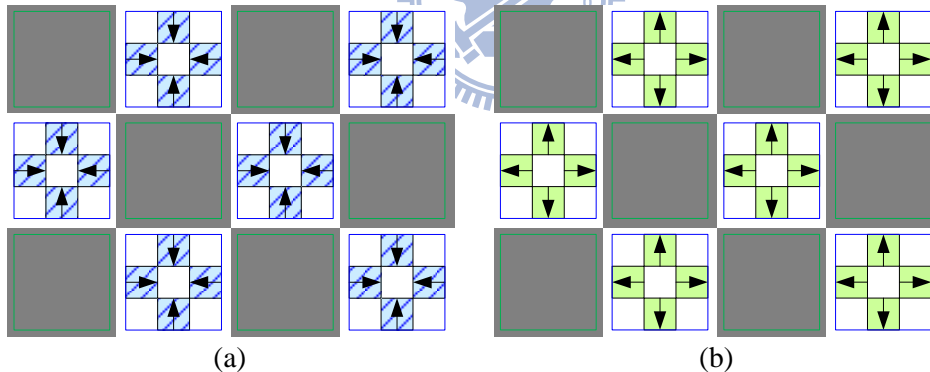


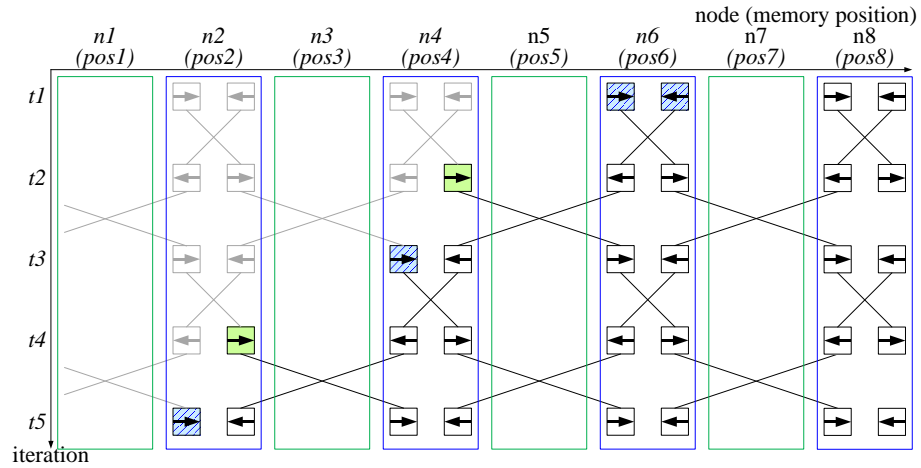
Figure III-9 Bipartite node plane with the spinning-message approach

(a) message passing for white nodes at odd iterations; (b) message passing for black nodes at even iterations.

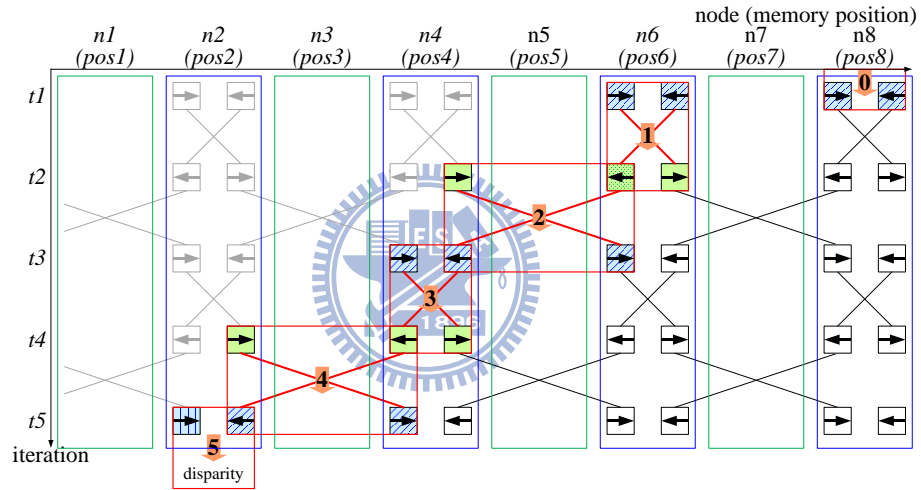
(3) Proposed Sliding-Bipartite Node Plane

By combining the above sliding node plane and bipartite node plane, the memory cost can be reduced to $4HTL$. Furthermore, applying the proposed spinning-message approach, the memory cost can be reduced to $2HTL$ as shown in Figure III-6 (b). Figure III-10 shows the spinning-message approach performs on the sliding-bipartite node plane. In a similar way as the sliding node plane, the front of the sliding-bipartite node plane can slide from the node n_6 to n_8 by the computing order in

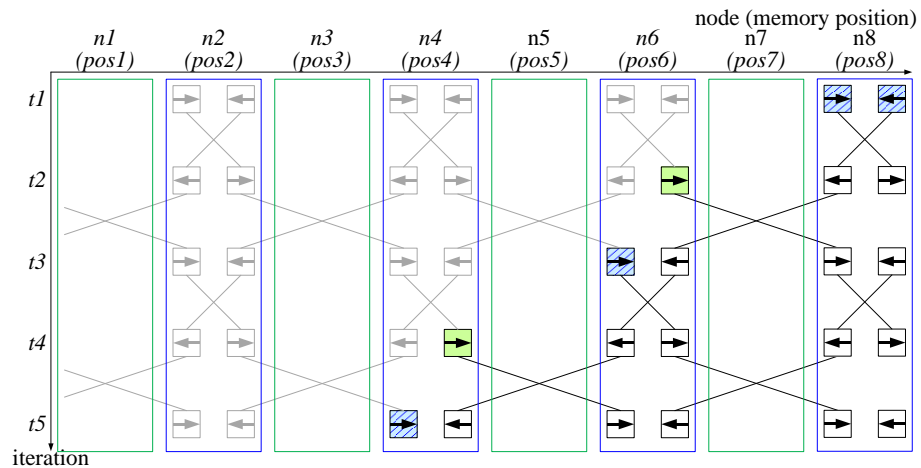
Figure III-10 (b). Therefore, the proposed sliding-bipartite node plane takes advantages of the sliding node plane and the bipartite node plane to reduce memory cost.



(a)



(b)



(c)

Figure III-10 Proposed sliding-bipartite node plane

(a) the node plane slides to the node $n6$; (b) the computing order of the message passing; (c) the node plane slides to the node $n8$.

3.2.3 Proposed Efficient PE

Following above proposed approaches for memory access, the message passing could be performed by parallel PEs with the configuration in Figure III-13 (a). However, the logic of each PE costs too much due to the high computational complexity of message passing. To conquer the high logic cost, we propose the buffer-free PE architecture in this section.

1. Previous Work

In the message passing, both the computational complexity and logic cost are significantly affected by the model of smoothness cost V . Kumar and Torr [37] took advantage of a truncated model to propose a low-memory generalized BP. This reduction is effective if the convolution of (III-7) is fully unrolled. On the other hand, Felzenszwalb and Huttenlocher [25] reduced the message passing from $O(L^2)$ to $O(L)$ by the benefit of a linear model. Figure III-11 presents the pseudo code of their proposed message passing to calculate one new message. This code includes three loops: aggregation and forward process, backward process, and normalization process. The latency of each loop is L iterations.

Based on the above flow, Park *et al.* [34] directly designed a PE architecture as shown in Figure III-12. In this architecture, the node plane additionally stores the data cost. By sequential computation, four old incoming messages and data cost of a node are fetched, and four new messages of neighboring nodes are produced. This architecture uses three pipeline stages corresponding to three loops in Figure III-11. They are divided by the two large message buffers mf and mb with L message entries, which dominate the hardware cost of this PE.

Aggregation and forward process	
1	$mf_0(-1) = MAX$
2	$mini_0 = MAX$
3	Loop1:
4	for $d=0$ to $L-1$ {
5	$Ag_0(d) = D(d) + M_0^{t-1}(d) + M_2^{t-1}(d) + M_3^{t-1}(d)$
6	$mini_0 = \min\{Ag_0(d), mini_0\} + Kv$
7	$mf_0(d) = \min\{Ag_0(d), mf_0(d-1)\} + Cv$
8	}
Backward process	
9	$mb_0(-1) = -MAX$
10	$norm_0 = 0$
11	Loop2:
12	for $d=L-1$ to 0 {
13	$temp = \min\{mf_0(d), temp + Cv\}$
14	$mb_0(d) = \min\{temp, mini_0\}$
15	$norm_0 = norm_0 + mb_0(d)$
16	}
Normalization process	
17	$norm_0 = norm_0 / L$
18	Loop3:
19	for $d=0$ to $L-1$ {
20	$M_0^t(d) = mb_0(d) - norm_0$
21	}

Figure III-11 Pseudo code of the message passing for calculating a new message

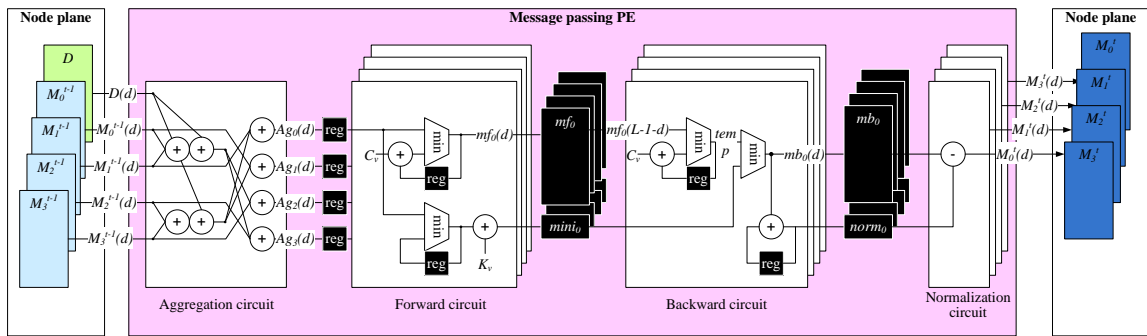


Figure III-12 Architecture of Park's PE

2. Buffer-free PE Architecture

Because the message buffers are the major logic cost of the previous PE, the strategy in our architecture is to remove all the message buffers of the previous PE. Figure III-13 (a) shows the overall configuration of the parallel buffer-free PEs. The parallel PEs fetch and store data by the proposed low memory access approaches, and each buffer-free PE can calculate four messages at the same time.

Figure III-13 (b) shows the detailed architecture of the buffer-free PE. Based on the pseudo code in Figure III-11, we first propose the post-normalization approach that merges the computation of the normalization on line 20 with the aggregation on line 5. The benefit of this merging is that the message buffer mb could be eliminated, but the $norm$ storing the normalization term should be changed to node plane. It causes that the memory of each message in node plane is increased by one memory entry. Second, we propose the convolution circuit that combines the forward process on lines 6 and 7 with the backward process on lines 13 to 15. These two have identical computations, two adders and two comparators, so that these computations can share the operators with additional multiplexers for selecting data path. Thus we can remove the message buffer mf . Finally, we also add the pipelining registers $z0$ and $z1$ to cut the critical path in this architecture.

The schedule of data access and computation in the proposed PE architecture are presented in Figure III-13 (c). In the step (1), the normalization terms, the old messages and the data cost are read to calculate the forward messages. In the step (2), the forward messages are stored in the node plane sequentially. In the step (3), the forward messages are read to calculate the backward messages. Finally, in the step (4), the backward messages and new normalization terms are stored in the node plane. The memories of the node plane are implemented by two-port register files because the proposed PE read and write them at the same time. Although the proposed PE takes about double latency of the Park's PE, the logic cost has been significantly reduced because all the message buffers are removed.

The proposed buffer-free PE can compute four messages of one node at the same time. It can also compute one message of multiple nodes for different scan schemes by the following simple modification. First, the post-normalized and aggregation circuit is modified to receive 3 messages. Then, the convolution circuit is modified to be only one module. Finally, the accessed node plane should be properly modified according to the specific scan scheme. This modification can make the proposed PE work well for one message, but will slightly degrade the hardware efficiency due to no sharing operators in the post-normalized and aggregation circuit.

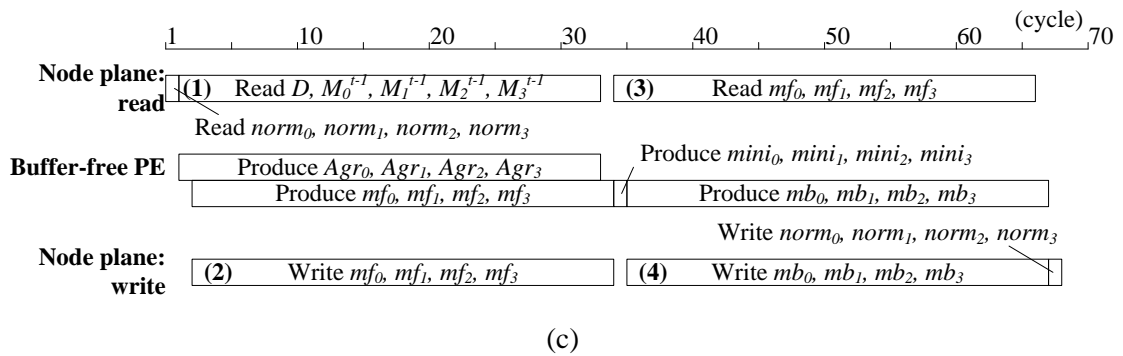
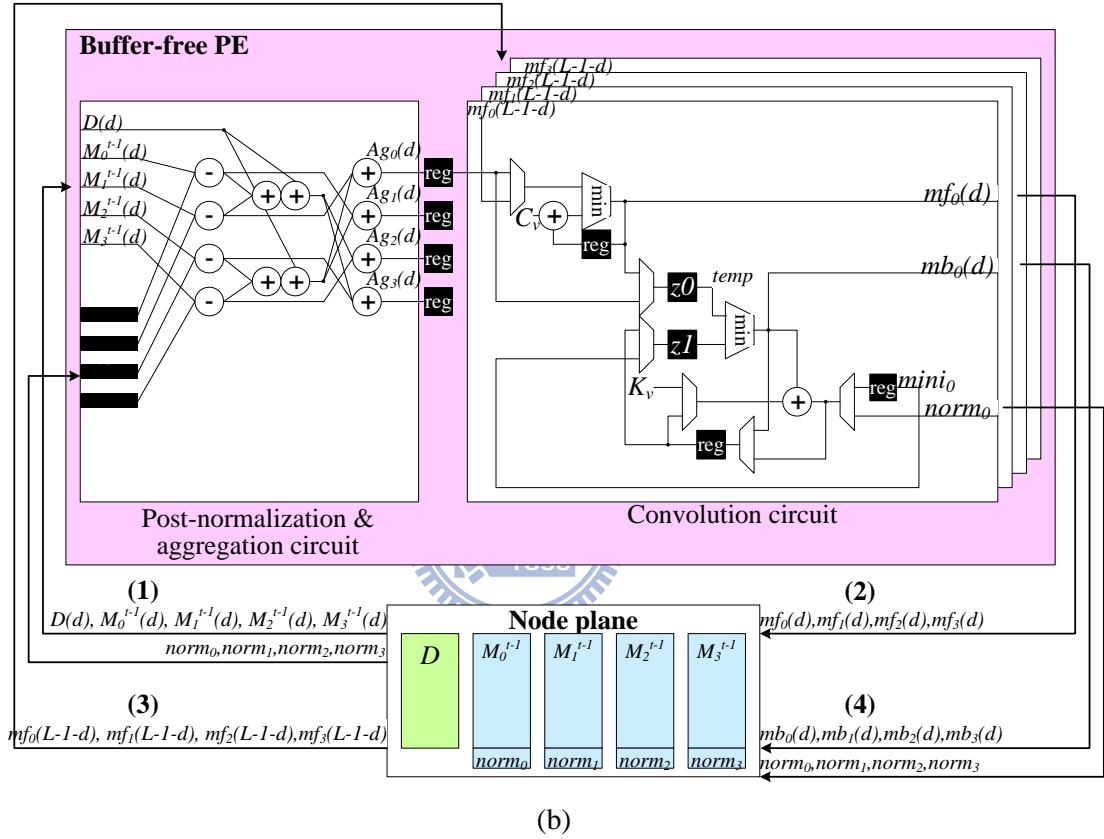
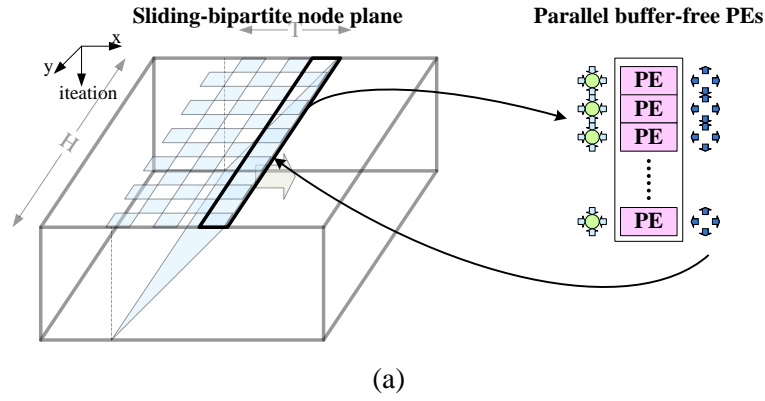


Figure III-13 Proposed architecture

(a) configuration of parallel PEs on the sliding-bipartite node plane; (b) architecture of the buffer-free PE; (c) schedule of the buffer-free PE.

3.2.4 Implementation Result

1. Memory Cost Comparison

The memory cost is affected by the type of node plane and memory access approach. As mentioned in Section III, the type of node plane would affect the computing order of PEs, and the memory access approach would provide a data access order for node planes. Both of the type of node plane and the memory access approach do not change the computational efficiency of the message passing but the memory cost.

Table III-1 compares the memory cost used by various types of node plane adopting the traditional fixed memory access approach and the proposed spinning-message approach. The size of node plane is substituted with block-scale and image-scale magnitudes, and each entry of messages is 16-bit. Compared to the traditional approach, most types of the node planes can save 50% memory cost in both the scales with the proposed spinning-message approach. The only exception is our previous in-place message node plane that has less saving with our approach since its original memory cost has been reduced to near 50%. In the comparison of overall hardware efficiency, the proposed spinning-message approach is better than the traditional approach. The reasons are that the proposed approach needs the same cycle counts as the traditional one while saving much memory cost. The only overhead of the proposed approach is a simple address generator, which has similar complexity as that in the traditional one.

Table III-1 Comparison of memory cost in memory access approaches for the iteration count of 30

Type of Node Plane	Memory Access Approach	Memory Cost of Message (16-bit)	Block-scale		Image-scale	
			W=16, H=16 (KB)	W=32, H=32 (KB)	W=320, H=240 (KB)	W=640, H=480 (KB)
Ping-pong buffer	Fixed	$8HWL$	131	524	39,321	157,286
	Spinning-message	$4HWL$	65	262	19,660	78,643
In-place Message [38]	Fixed	$4(HW+3W)L$	77	286	19,906	79,134
	Spinning-message	$4HWL$	65	262	19,660	78,643
Vertical Sliding [34]	Fixed	$8TWL$	131	491	4,915	9,830
	Spinning-message	$4TWL$	65	245	2,457	4,915
Horizontal Sliding	Fixed	$8HTL$	131	491	3,686	7,372
	Spinning-message	$4HTL$	65	245	1,843	3,686
Bipartite [25]	Fixed	$4HWL$	65	262	19,660	78,643
	Spinning-message	$2HWL$	32	131	9,830	39,321
Sliding-Bipartite	Fixed	$4HTL$	65	245	1,843	3,686
	Spinning-message	$2HTL$	32	122	921	1,843

Figure III-14 compares the memory cost among different types of node plane using the same proposed spinning-message approach for different sizes of node plane. In this figure, all the memory cost ratios are relative to the ping-pong buffer with the traditional fixed memory access approach. Compared to the sliding node planes, the bipartite node plane can save more memory cost in the block-scale. On the contrary, the sliding node planes can reduce more in the image-scale. The proposed sliding-bipartite node plane combines their benefits to reduce more memory cost in the block-scale and image-scale. Its memory cost reduction can achieve 1.2% in the image-scale of 640x480 and 23.4% in the block-scale of 32x32. Note that the sliding-based node planes would decrease its memory cost reduction when the iteration count T is larger than H or W .

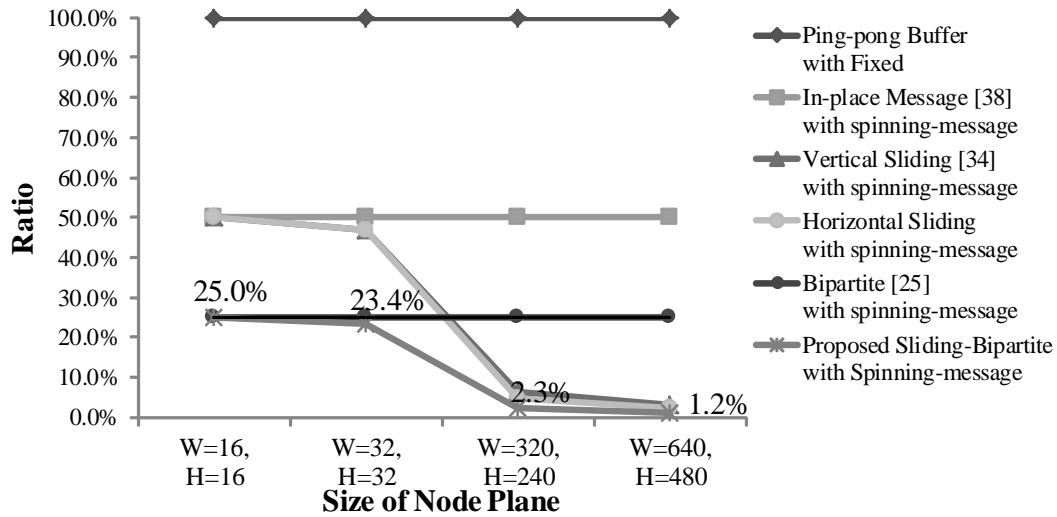


Figure III-14 Ratio of memory cost in different node planes with spinning-message approach

2. Implementation

The proposed buffer-free PE architecture has been implemented by Verilog and synthesized by the 90-nm CMOS technology process. To compare with the Park's PE [34], we also implemented their PE design since their original implementation is on two FPGA boards. In addition, the Cheng's PE [31] is implemented in the same design condition for a fair comparison since some details are not disclosed in the dissertation. All the data widths are 16-bit in each implementation. Table III-2 compares the logic cost of the proposed buffer-free PE with the other PEs. In these PEs, the Cheng's PE takes the least latency to calculate a new message because of its fully parallel architecture. The Park's PE and the proposed buffer-free PE belong to sequential architecture that causes higher latency. Although the proposed PE requires the most latency, its hardware efficiency is 3.6 times of the Park's PE and 1.4 times of the Cheng's PE. That is because we remove all the message buffers and common circuits to reduce logic cost, as well as add a pipeline stage on its critical path in the proposed buffer-free PE.

Note that the hardware efficiency in our PE excludes the memory overhead by the post-normalization approach, which is highly related to the size of node plan, instead of the number of PE. Thus, our hardware efficiency will be still higher than Cheng's design when the size of node plan

is smaller than 35 for one PE case. For the 32 PEs case as in Table 3, the proposed approach will still have better hardware efficiency for node planes up to 35x32 (1,120) nodes. With this size, our proposed PE is suitable for the block-scale BP algorithms, such as block-based BP and tile-based BP, whose overall cost will be more practical than that in the image-scale BP.

Table III-2 Logic cost comparison of PE architectures

	Cheng's PE [31]	Park's PE [34]	Proposed buffer-free PE	Proposed buffer-free PE (32 PEs)
Operating Frequency (MHz)	100	222	285	285
Disparity Range (L)	32	32	32	32
CMOS Tech. process	UMC 90-nm	UMC 90-nm	UMC 90-nm	UMC 90-nm
Gate Count	69.6K	50K	8.3K	256.6K
Latency (Cycle)	1	32	68	68
	(1 msg)	(4 msg)	(4 msg)	(128 msg)
Throughput (Node/Second)	25,000K	6,938K	4,191K	134,117K
Hardware Efficiency (Throughput/Gate count)	359	139	505	505

The proposed low memory access approach and buffer-free PE architecture could be generally applied to the various BP-based algorithms together. Table III-3 shows the implementation results of four typical BP-based algorithms for the real-time constraint of 640x480 and the disparity range of 32.

In these BP-based algorithms, their algorithm flows and iteration counts affect the required throughput. The message passing is performed for the baseline BP on a whole image, and for the hierarchical BP on multiple resolution images with different iteration counts. Thus, their required throughput is proportional to the image size and corresponding iteration count. For the block-based and tile-based BP, the message passing is performed on each block (tile) in an image. In addition to the iteration count for each block, the tile-based BP has the outer iteration count for re-processing the image. Their required throughput is proportional to the total iteration count as well as the block's count and size. To satisfy the required throughput of these BP-based algorithms, we should use sufficient parallel PEs. Note that the maximal number of PE is equal to H due to the configuration of parallel

PEs in the sliding-bipartite node plane. As a result, the block-based BP and tile-based BP designs just approach to real-time speed. With the buffer-free PE architecture, the logic cost of all the BP-based algorithms are less than the gate counts of 300K.

The memory cost of this table contains the messages and the data costs, which is proportional to the size of sliding-bipartite node plane according to Table III-1. The total memory cost of the baseline BP and hierarchical BP is larger than others because they allocate image-scale node planes. In contrast, the block-based BP and tile-based BP are more suitable to be integrated into stereoscopic video systems.

Table III-3 Implementation results of various BP-based algorithms

	Baseline BP [24]	Hierarchical BP [25]	Block-based BP [36]	Tile-based BP [29]
Iteration T	30	5, 5, 10, 5	30	inner=8, outer=2
Required Throughput (Node/Frame)	4,608,000	1,212,000	4,608,000	4,915,200
Operating Frequency (MHz)	285	285	285	285
Number of PE	33	32	32	32
Gate Count (K)	273.9	74.7	265.6	265.6
Size of Sliding-Bipartite Node Plane	30x480 (image-scale)	5x480 (image-scale)	30x32 (block-scale)	8x32 (block-scale)
Memory Cost of Messages and Data costs (KB)	2,793	465	186	49
FPS	30.01	31.12	29.11	27.29

frame size=640x480, disparity range=32

3.3 Analysis and Design of Joint Bilateral Filtering

In this section, we first review the previous acceleration approaches for BF and JBF. Then we focus on the integral histogram approach and point out its design challenges. To solve it design challenges, we propose three memory reduction methods, and a real-time architecture design for the JBF, which is the most important component in JBU for disparity estimation.

3.3.1 Related Acceleration Approaches

Various acceleration approaches for BF have been proposed, and can be classified into two categories: target-pixel-first approach and support-pixel-first approach, according to their

computational characteristics, as illustrated in Figure III-15. Most of the acceleration approaches could be applied to the JBF.

The target-pixel-first approach is an aggregation process that focuses on a target pixel c and accumulates its support pixels q . On the other hand, the support-pixel-first approach is a diffusion process that regards a support pixel q as a center to diffuse for its target pixels c . With the classification, the previous approaches are reviewed in this Section, and their computational complexity and memory cost are compared in Table III-4. In which, R is the range domain from 0 to 255 for gray-level

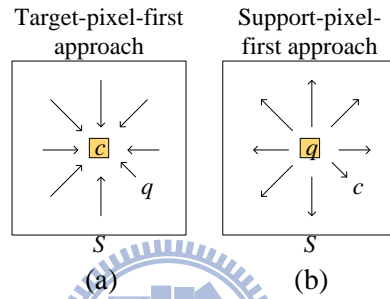


Figure III-15 Classification of BF acceleration approaches

Table III-4 Comparison of BF acceleration approach in computational complexity and memory cost

Approach		Computational Complexity (per pixel)		Memory Cost (per frame)	
Support Pixel First	Brute-Force	All	$O(S ^2)$	0	
	Basic	LUT Construction	$O(R)$	$4MN$	
		2-D Conv. by FFT	$O(S /\log S)$		
	Durand and Dorsey [84]	Piecewise-linear Subsampling	LUT Construction	$O(R /s_r)$	$4MN/s_s^2$
			2-D Conv. by FFT	$O(S /s_s^2 \log(S /s_s^2))$	
	Yang <i>et al.</i> [83]	Piecewise-linear	LUT Construction	$O(R /s_r)$	$4MN$
2-D Conv. by Approx. Gaussian	$O(1)$				
Paris and Durand [85]	Bilateral Grid	LUT Construction	$O(R /s_r)$	$MN R /(s_r s_s^2)$	
3-D Conv. by FFT	$O(S R /(s_r s_s^2) \log(S R /(s_r s_s^2)))$				
Target Pixel First	Pham and Vliet [89]	Separable	1-D Aggre. for Col.	$O(S)$	0
			1-D Aggre. for Row	$O(S)$	
	Basic	Histogram	Histogram Calculation	$O(R S ^2)$	0
			1-D Conv.	$O(R)$	
	Huang [90]	Extended Histogram	Histogram Calculation	$O(R S)$	$ S+E ^2 R $
			1-D Conv.	$O(R)$	
Weiss [92]	Distributed Histogram	Histogram Calculation	$O(R /\log S)$	$ S+E ^2 R $	
		1-D Conv.	$O(R)$		
Porikli [91]	Integral Histogram	Histogram Calculation	$O(R /s_r)$	$MN R /s_r$	
		1-D Conv.	$O(R /s_r)$		

M : frame height, N : frame width, $|S|$: filter window width, $|R|$: intensity range s_s : quantization factor for S , s_r : quantization factor for R , E : extension pixel count

1. Support-Pixel-First Approaches

The main idea of the support-pixel-first approaches is to convert the original nonlinear convolution to linear convolution, so that the linear convolution can be accelerated by existing algorithms, such as the Fast Fourier transform (FFT). To convert (III-5) to linear convolution, the terms $g(\|I(c)-I(q)\|)I(q)$ and $g(\|I(c)-I(q)\|)$ are pre-calculated and stored in memories as look-up tables (LUT). Hence, the approaches consist of two steps, LUT construction and linear convolution. For the implementation issues, the former needs a large storage and the later needs an efficient computation.

Durand and Dorsey [84] first proposed the support-pixel-first approach that contains the piecewise-linear scheme and the subsampling scheme to respectively quantize the range domain R and spatial domain S . Both the memory cost and computational complexity can be reduced by the quantization factors s_r, s_s^2 . Based on the piecewise-linear scheme, Yang *et al.* [83] adopted a constant-time approximate Gaussian filtering for the convolution to achieve real-time processing by the GPU programming.

Paris [85], [86] indicates that the piecewise-linear scheme would suffer from poor approximation on texture's discontinuity since it cannot exactly interpolate dense results. To address that, the bilateral grid scheme was proposed to perform a 3-D convolution on $S \times R$, instead of the typical 2-D convolution only on S . However, its memory cost and computational complexity are scaled on the dimension R . Following the bilateral grid scheme, Chen [87] implemented it by the GPU programming to achieve real-time processing. In addition, Adams [88] adopts the Gaussian KD-tree to improve its speed.

To sum up, the support-pixel-first approaches can convert BF and JBF to linear convolution but suffer from high memory cost for LUTs. Unfortunately, the size of LUTs should be frame-scale-magnitude since their algorithms iteratively performs on whole frame.

2. Target-Pixel-First Approaches

The main idea of the target-pixel-first approaches is to aggregate the support pixels with kernels, which needs the computational complexity of $O(|S|^2)$. To accelerate it, Pham and Vliet [89] proposed the separable BF that directly changes the original 2-D aggregation to two-step 1-D aggregation for columns and a row. Thus it can reduce the computational complexity to $O(|S|)$ but suffers from the axis-aligned artifact.

On the other hand, the histogram-based approaches could reduce computation without significant quality degradation. In the approaches, the space kernel f is simplified to a box filter with constant coefficient, so that (III-5) is rewritten as

$$I'(c) = \frac{\sum_{q \in S} g(\|I(c) - I(q)\|)I(q)}{\sum_{q \in S} g(\|I(c) - I(q)\|)} = \frac{\sum_{b \in R} g(\|I(c) - b\|)hc_c(b)b}{\sum_{b \in R} g(\|I(c) - b\|)hc_c(b)}, \quad (\text{III-9})$$

Before convoluting each support pixel $I(q)$ with g , the support pixels in the filter window S are classified into the pixel count histogram hc_c , whose subscript refers to the target pixel c . Figure III-16 shows the concept of the classification. According the support pixel $I(q)$, the corresponding bin b is accumulated. For the exact result of gray-level, the number of bin N_b is set as 256. After classifying all support pixels, the histogram bin value $hc_c(b)$ can refer to the number of support pixels with the intensity b in S . Then, (III-9) can be finally calculated by 1-D convolution in the range domain R , instead of the original space domain S . In summary, the histogram-based approaches include two parts: histogram calculation and 1-D convolution. The key point of the histogram-based approaches is that the convolution can be decreased from the larger $|S|^2$ to $|R|$. However, the major computational complexity is $O(|R||S|^2)$ in the histogram calculation that demands other acceleration techniques.

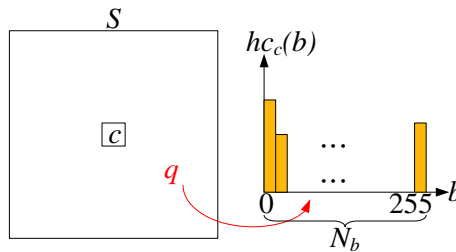


Figure III-16 Concept of histogram-based approaches

To speed up the histogram calculation, Huang [90] proposed the extended histogram approach that calculates multiple target pixels' histograms and shares their partial histograms in run time. Its computational complexity can be reduced to $O(|R|/S)$, but it spends extra memory cost. Based on the extended histogram approach, Weiss [92] proposed the distributed histogram approach that reassembles the histogram calculation of each row, and reduces computational complexity to $O(|R|/\log|S|)$. Furthermore, Porikli [91] proposed the integral histogram approach to decrease computational complexity to $O(|R|/s_r)$, which is independent of the filter window size. In addition, the factor s_r quantizes the support pixel's intensity. The integral histogram approach can be faster than the brute-force approach when $|R|/s_r$ is smaller than $|S|^2$. That implies this approach is suitable to be applied when BF has large filter window size. Based on the integral histogram approach, Ju [93] modified (III-9) to

$$J'(c) = \frac{\sum_{q \in S} g(\|I(c) - I(q)\|)J(q)}{\sum_{q \in S} g(\|I(c) - I(q)\|)} = \frac{\sum_{b \in R} g(\|I(c) - b\|)hi_c(b)}{\sum_{b \in R} g(\|I(c) - b\|)hc_c(b)}, \quad (\text{III-10})$$

to further support JBF. Different from (III-9), the histogram in the numerator is the pixel intensity histogram hi_c that accumulates the pixel intensity for each bin, instead of the pixel count in hc_c .

In summary, the integral histogram approach is the state-of-the-art in target-pixel-first approaches, but its memory cost is frame-scale-magnitude, like the support-pixel-first approaches. However, as mentioned above, the memory cost of the support-pixel-first approach is hard to be reduced due to its iterative computing, instead of progressive computing in the integral histogram approach. Thus, this dissertation focuses on the integral histogram approach.

3.3.2 Analysis of Integral Histogram Approach

In this Section, we introduce the integral histogram approach in details, and then analyze the design challenges of JBF, which can be applied to BF as well.

1. Integral Histogram Approach

Table III-5 presents the computational flow and computational analysis of the integral histogram approach for JBF to calculate 1-pixel result, which consists of the integration, extraction, kernel calculation, and convolution processes. In which, the former two are for the histogram calculation step, and the latter two are for the 1-D convolution step.

For ease of explanation, we use the area view to show how this approach operates and the memory view to show the memory usage, as illustrated in Figure III-17 (a). In the area view, IH_O^X is a histogram of the rectangular area stretched from the pixel O to X . Thus, the addition and subtraction of IH can be regarded as area merging and cutting, respectively. In the memory view, the data of IH_O^X are stored at X , and the gray region represents occupied memory usage. With these representations, Figure III-17 (b) and (c) illustrate the integration and extraction processes.

Table III-5 Computational flow and analysis for a pixel in the integral histogram approach

Process	Complexity (operation)	BW for IH (data)	BW for pixel (data)
Integration process:			
<u>Pixel count histogram hc_c</u>			
Loop $b=0$ to N_b-1			
$IHc_o^S(b) = IHc_o^Q(b) + IHc_o^R(b) - IHc_o^P(b)$	ADD: $3N_b$	$4N_b$	
$IHc_o^S(I_S) += 1$	ADD: 1		
<u>Pixel intensity histogram hi_c</u>			
Loop $b=0$ to N_b-1			
$IHi_o^S(b) = IHi_o^Q(b) + IHi_o^R(b) - IHi_o^P(b)$	ADD: $3N_b$	$4N_b$	
$IHi_o^S(I_S) += J_s$	ADD: 1		2 pixels
Extraction process:			
<u>Pixel count histogram hc_c</u>			
Loop $b=0$ to N_b-1			
$hc_c(b) = IHc_o^D(b) + IHc_o^A(b) - IHc_o^B(b) - IHc_o^C(b)$	ADD: $3N_b$	$4N_b$	
<u>Pixel intensity histogram hi_c</u>			
Loop $b=0$ to N_b-1			
$hi_c(b) = IHi_o^D(b) + IHi_o^A(b) - IHi_o^B(b) - IHi_o^C(b)$	ADD: $3N_b$	$4N_b$	
Kernel calculation process:			
Loop $b=0$ to N_b-1			
$G(b) = g(I_c - b)$	ADD, LUT: N_b		1 pixel
Convolution process:			
Nu=0, De=0			
Loop $b=0$ to N_b-1			
De += $G(b) \times hc_c(b)$	MUL, ADD: N_b		
Nu += $G(b) \times hi_c(b)$	MUL, ADD: N_b		
Result = Nu / De	DIV: 1		1 pixel
Total	$17N_b+3$	$16N_b$	4 pixels

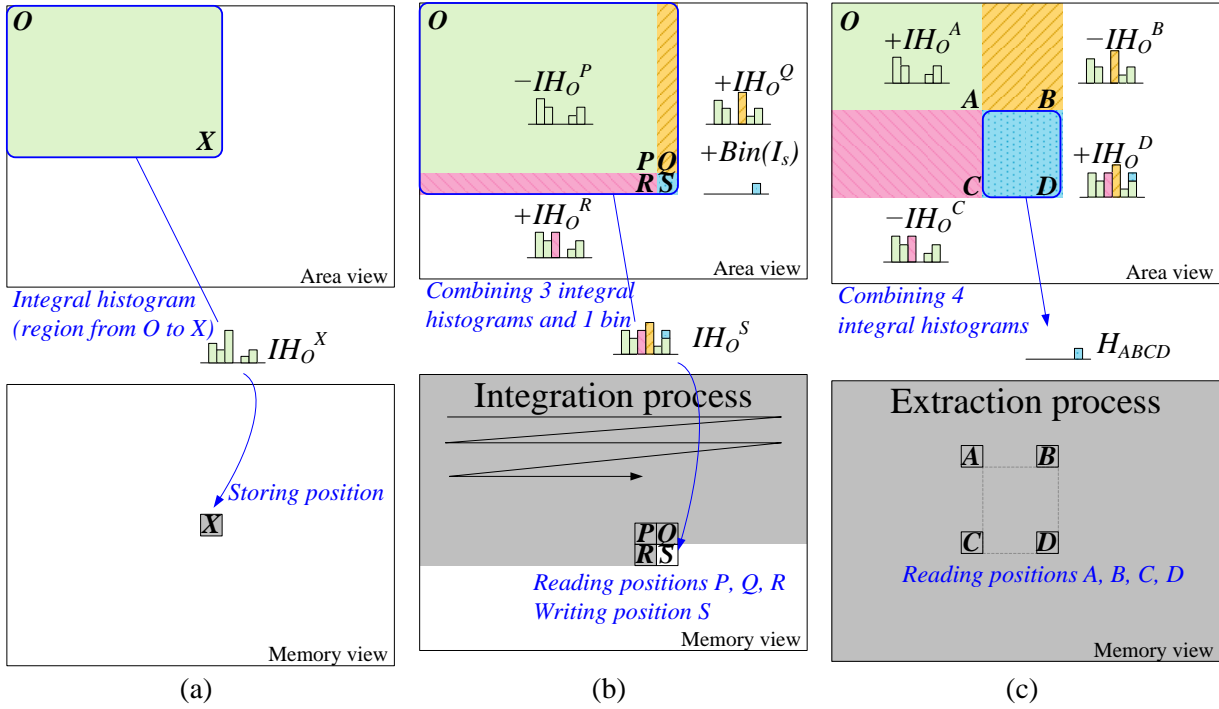


Figure III-17 Concept of integral histogram approach

(a) representation of an integral histogram IH_O^X for the region from O to X in area view and memory view, (b) integration process performed by raster-scan order to compute the integral histogram IH_O^S , (c) extraction process performed to extract the histogram H_{ABCD} of the rectangle $ABCD$.

First, the integration process progressively calculates the IH of each pixel using the equation,

$$IH_O^S = IH_O^Q + IH_O^R - IH_O^P + Bin(I_S) \quad (III-11)$$

For the pixel count histogram hc_c and the pixel intensity histogram hi_c , their IHs (i.e. IHc and IHi) are computed separately as shown in Table III-5. For hc_c , $Bin(I_S)$ is 1 for the corresponding bin and 0 for others. On the other hand, for hi_c , this term is J_s for the corresponding bin, and also 0 for others. After this process, the IH of each pixel is produced and stored into memory.

Second, given the IHs, the extraction process can extract the histogram hc_c or hi_c of the filter window $ABCD$ centered by the target pixel c using the equation,

$$H_{ABCD} = IH_O^D + IH_O^A - IH_O^B - IH_O^C \quad (III-12)$$

As shown in Figure III-17 (c), a histogram with arbitrary filter window size can be obtained by using the IHs of four corners. With this property, the integral histogram approach can reduce computational complexity to independent of filter window size.

Third, the kernel calculation process computes the range kernel by a range table, which includes 256 items for the 256 possible values of $\|I_c - b\|$. Finally, given the range kernel g and the histograms hc_c and hi_c , the convolution process calculates the result of target pixel c by (III-10).

2. Design Challenges

Since the complexities listed in Table III-5 are pixel wise as well as bin number dependent, they will grow quickly as resolution and bin number grow. The detailed design challenges are described below.

(1) High Memory Cost for Integral Histogram

During the integration process, all the IHs of whole image are stored in memory. BF needs a frame-scale-magnitude memory for hc_c , and JBF additionally needs another one for hi_c . Therefore, the total memory cost of JBF is

$$MN \cdot N_b w_b + MN \cdot N_b (w_b + 8) , \quad (\text{III-13})$$

where the former term is for hc_c , and the later term is for hi_c . M and N is the frame height and width, N_b is the number of bin, and w_b is the bit width of a bin. Note that w_b is related to the maximal area of integration, and its value equals $\log_2(MN)$. In addition, the bit width of hi_c is more than hc_c by 8 bits because the intensity of a pixel requires 8 bits.

Above memory cost would be 829.4 Mbytes for the HD1080p resolution (i.e. $N=1920$, $M=1080$, $w_b=21$, $N_b=64$). For a VLSI design, these massive data could be configured into off-chip DRAM or on-chip SRAM. However, the off-chip memory suffers from longer access latency and limited bandwidth usage in a system. Hence, our strategy for the design challenge is to reduce the memory requirement and enable data be stored in on-chip memory.

(2) High Computational Complexity in All Processes

According to the complexity in Table III-5, generating 1-pixel result needs $15N_b+2$ additions, $2N_b$ multiplications, and 1 division. If N_b is 64, the total complexity will be 2,262.3 million operations for

an HD1080p image. To meet above demands, a VLSI design with sufficient parallel operators is necessary.

(3) High Bandwidth in Integration and Extraction

In Table III-5, the bandwidth for IH requires $16N_b$ for 1-pixel result, and that will reach 106.168 Gbits for an HD1080p image as shown in Table III-8. That is because the IHs are accessed frequently. With the strategy for the memory cost problem, the IHs are stored in on-chip memory, and its data bus should be increased to address the high bandwidth problem. However, it results in over-partitioned memory and increased area. Thus, a method to reduce the bandwidth is needed.

(4) Large Range Table in Kernel Calculation

In the kernel calculation process, a range table with 256 items is needed. However, with the parallel operations for the computational complexity problem, this table should be duplicated. Thus, both the size and number of the range table results in large area.

In summary, the integral histogram approach can speed up JBF and BF well but suffers from above design challenges. To address them, a VLSI design with suitable memory reduction and architecture design techniques is necessary.

3.3.3 Proposed Memory Reduction Methods

To solve the high memory cost problem, we can take advantage of the raster-scan computing order to reduce the memory cost from a frame to a multiple rows by the runtime updating method (RUM). The memory cost could be further reduced by the stripe-based method (SBM) to slice frame into stripes. Finally, we propose the sliding origin method (SOM) that moves the origin of each IH stripe progressively with the computing and can reduce the multiple row buffers to single row buffer. With these memory methods, the memory cost can be reduced to 0.003%-0.020%. The details of the proposed methods are described below.

1. Runtime Updating Method (RUM)

The concept of the RUM is to perform the integration process and the extraction process at the same time, instead of two separate iterations in the original flow. Figure III-18 illustrates its memory configuration in the memory view. In Figure III-18 (a), the integration process is from the pixel O to D , and the extraction process can extract the histogram H_{ABCD} . From the data lifetime analysis, all the IHs before the pixel A are unnecessary. Thus, only the IHs from A to D require memory space, so that the memory cost is

$$|S|N \cdot N_b w_b + |S|N \cdot N_b (w_b + 8) , \quad (\text{III-14})$$

where M in (III-13) is replaced by the filter window width $|S|$.

Figure III-18 (b) and (c) illustrate that the memory is updated when the two processes moves to the next pixel S . In Figure III-18 (b), the integration process calculates the new IH_O^S using IH_O^D , $IH_O^{D'}$, $IH_O^{S'}$, and then the new IH_O^S can overwrite the memory position of the discarded IH_O^A . In Figure III-18 (c), the extraction process can extract H_{PQRS} . With the proposed RUM, the memory cost could be reduced from a full frame to a partial frame. This method can gain considerable reduction since $|S|$ is usually much smaller than M .

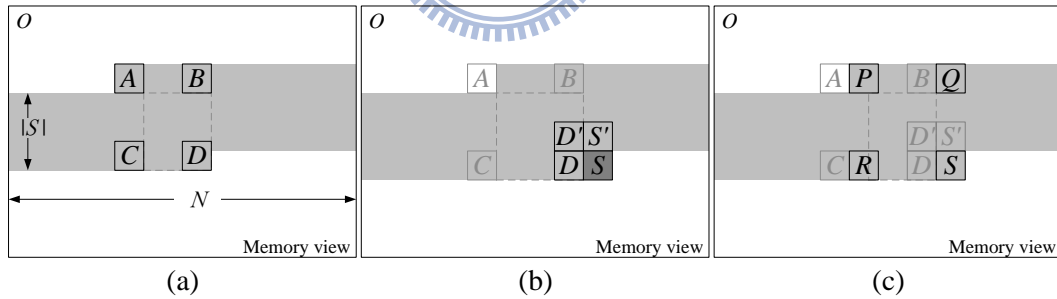


Figure III-18 Runtime updating method (RUM)

(a) extraction for H_{ABCD} , (b), integration to S , (c) extraction for H_{PQRS}

2. Stripe-Based Method (SBM)

The main idea of the SBM is to slice the whole frame into many vertical stripes, and the integration and extraction processes are performed stripe by stripe. Figure III-19 illustrates a whole frame partitioned into stripes. Note that the integration process should additionally be carried out on the extended region, which contains the surrounding support pixels for the target pixels on the stripe boundary. Thus, the total memory cost of the SBM is

$$M(|S| + w_s - 1) \cdot N_b w_b + M(|S| + w_s - 1) \cdot N_b (w_b + 8) , \quad (\text{III-15})$$

where w_s is the stripe width, and w_b equals $\log_2[M/(S+w_s-1)]$. Compared to the original cost in (III-13), the SBM could reduce significant memory if $(|S|+w_s-1)$ is much smaller than N . The overhead of the SBM is that the extended regions result in extra computation and bandwidth in the integration process due to repeated performing on these regions.

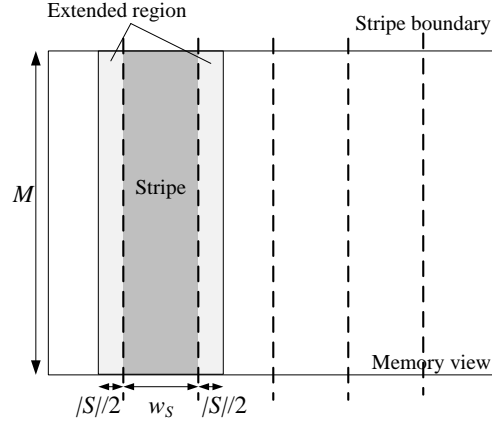


Figure III-19 Stripe-based method (SBM)

3. Sliding Origin Method (SOM)

The concept of the SOM is to vertically slide the origin pixel O with the integration and extraction processes to reduce memory cost from a plane to a line as shown in Figure III-20. With the sliding origin pixel, the two processes can be simplified as described below.

For the extraction process in the area view of Figure III-20 (a), the original $I_{H_O^A}$ and $I_{H_O^B}$ are zero because O is under A and B , and they cannot form meaningful histogram rectangles. Hence, the equation (III-12) can be simplified to

$$H_{ABCD} = I_{H_O^D} - I_{H_O^C}. \quad (\text{III-16})$$

For the integration process in Figure III-20 (b), the new $I_{H_O^S}$ is computed by

$$I_{H_O^S} = I_{H_O^D} + I_{H_{O'}^{S'}} - I_{H_{O'}^{D'}} + \text{Bin}(I_S) . \quad (\text{III-17})$$

However, the S' and D' are on the previous row of S and D , and their corresponding origin should be O' , instead of O . Therefore, the $I_{H_O^{S'}}$ and $I_{H_O^{D'}}$ in (III-17) should be changed to $I_{H_{O'}^{S'}}$ and $I_{H_{O'}^{D'}}$ by

$$I_{H_O^S} = I_{H_O^D} + I_{H_{O'}^{S'}} - \text{Bin}(I_{O'}) - I_{H_{O'}^{D'}} + \text{Bin}(I_S) . \quad (\text{III-18})$$

With above simplification, the required memory space could be reduced to

$$N \cdot N_b w_b + N \cdot N_b (w_b + 8) , \quad (\text{III-19})$$

where w_b equals $\log_2(\lceil S/N \rceil)$ since the maximal area of integration is $\lceil S/N \rceil$. Compared to the original cost in (III-13), the height dimension M is eliminated, and w_b is much smaller.

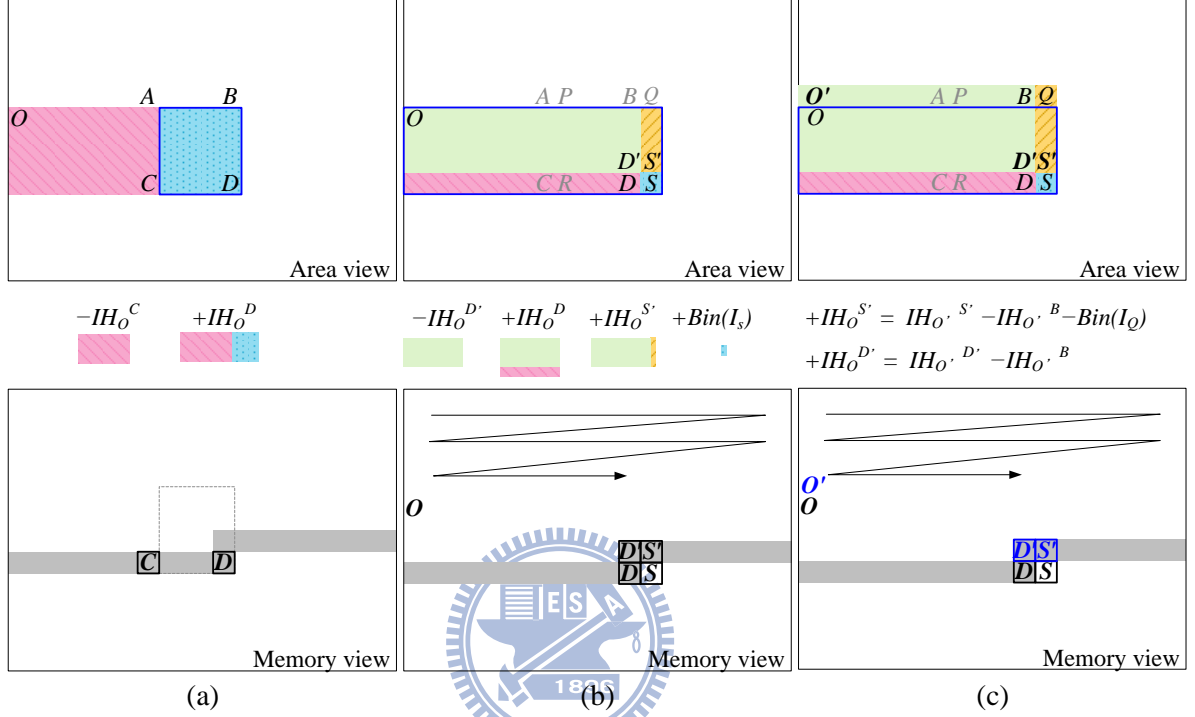


Figure III-20 Sliding origin method (SOM)

(a) extraction process with sliding origin O , (b) integration process to next pixel S , (c) modified integration process to next pixel S .

4. Combination

The proposed memory reduction methods could be simply combined as follows. First, the SBM partitions a whole frame into stripes. Then, in each stripe, the RUM and SOM are performed row by row. This combination can reduce the memory cost to

$$(|S| + w_s - 1) \cdot N_b w_b + (|S| + w_s - 1) \cdot N_b (w_b + 8) , \quad (\text{III-20})$$

where w_b equals $\log_2[\lceil S/(\lceil S/w_s \rceil + w_s - 1) \rceil]$. Compared to the original cost in (III-13), M is decreased to 1 due to the RUM and SOM, and N is decreased to $(|S| + w_s - 1)$ due to SBM.

3.3.4 Proposed Architecture

With above memory reduction methods, the computational flow of JBF in Table III-5 is changed to that in Table III-6. The details of these design techniques are presented below.

Table III-6 Modified computational flow and analysis for a pixel in the integral histogram approach

Process	Complexity (operation)	BW for IH (data)	BW for pixel (data)
Integration process:			
<u>Pixel count histogram hc_c</u>			
Loop $b=0$ to N_b-1			
$IHC_o^S(b)=IHC_o^D(b)+IHC_o^{S'}(b)-IHC_o^{D'}(b)$	ADD: $2N_b$	$4N_b$	
$IHC_o^S(I_S) += 1, IHC_o^S(I_Q) -= 1$	ADD: 2		
<u>Pixel intensity histogram hi_c</u>			
Loop $b=0$ to N_b-1			
$IHi_o^S(b)=IHi_o^D(b)+IHi_o^{S'}(b)-IHi_o^{D'}(b)$	ADD: $2N_b$	$4N_b$	
$IHi_o^S(I_S) += J_S, IHi_o^S(I_Q) -= J_Q$	ADD: 2		4 pixels
Extraction process:			
<u>Pixel count histogram hc_c</u>			
Loop $b=0$ to N_b-1			
$hc_c(b) = IHC_o^S(b) - IHC_o^R(b)$	ADD: N_b	N_b	
<u>Pixel intensity histogram hi_c</u>			
Loop $b=0$ to N_b-1			
$hi_c(b) = IHi_o^S(b) - IHi_o^R(b)$	ADD: N_b	N_b	
Kernel calculation process:			
Loop $b=0$ to N_b-1			
$G(b) = g(I_c - b)$	ADD, LUT: N_b		1 pixel
Convolution process:			
Nu=0, De=0			
Loop $b=0$ to N_b-1			
De += $G(b) \times hc_c(b)$	MUL, ADD: N_b		
Nu += $G(b) \times hi_c(b)$	MUL, ADD: N_b		
Result = Nu / De	DIV: 1		1 pixel
Total	$11N_b+5$	$10N_b$	6 pixels

1. Overall Architecture

Figure III-21 shows the overall architecture that contains two parts, interface and core. In this architecture, the image pixels and the IHs are stored at the off-chip and on-chip memory, respectively. The interface accesses pixels from the off-chip memory through a 64-bit bus, and the core performs the computation of JBF.

In the interface, the access controller allocates the bus priority to the input and output first-in-first-out (FIFO) buffers by round-robin policy. The size of each buffer is associated with off-chip bandwidth. Large buffers can support data reuse schemes to reduce the off-chip bandwidth. Because of sufficient bandwidth in this architecture, we do not apply any data reuse schemes here, and

set its size as 16-pixel to meet the bus width and support ping-pong mechanism for simultaneous reading and writing.

The operations of the architecture are described below with the schedule in Figure III-22, which is hierarchically sliced from a frame to pipeline tiles. The computation of one stripe row requires 90 cycles for the stripe width w_s of 60 and the filter window width $|S|$ of 31. Note that this architecture takes 96 cycles for one stripe row, and the last 6-cycles are the bubble cycles for simplifying controlling logic. For the process in a pipeline tile, the access controller in the interface fetches pixels from the off-chip memory into the FIFO buffers. Then the two histogram calculation engines in the core begin to compute hi_c and hc_c , and the convolution engine consecutively produces 8 pixels to the output FIFO buffer. Finally, the interface moves results from the buffer to the off-chip memory.

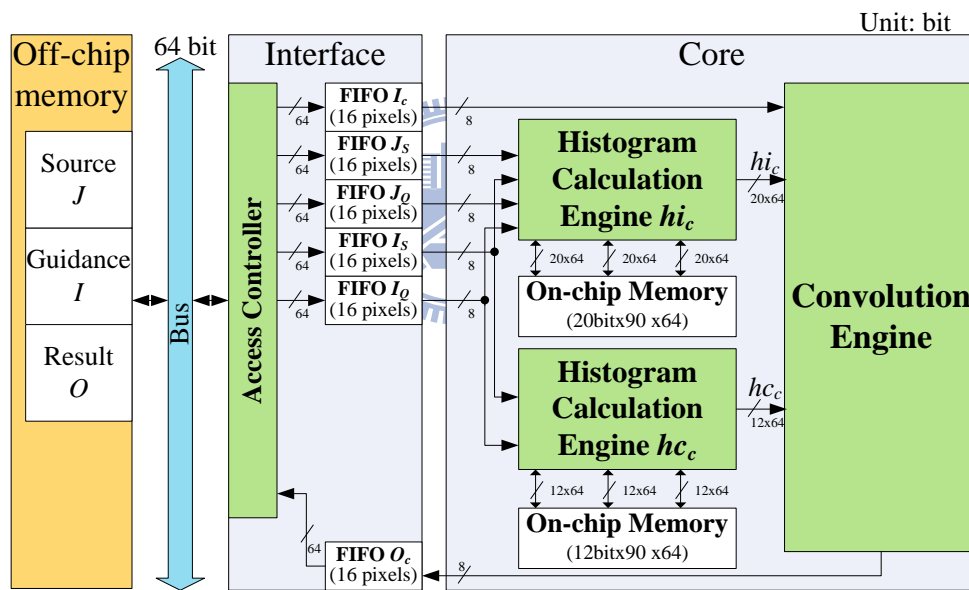


Figure III-21 Proposed architecture of JBF.

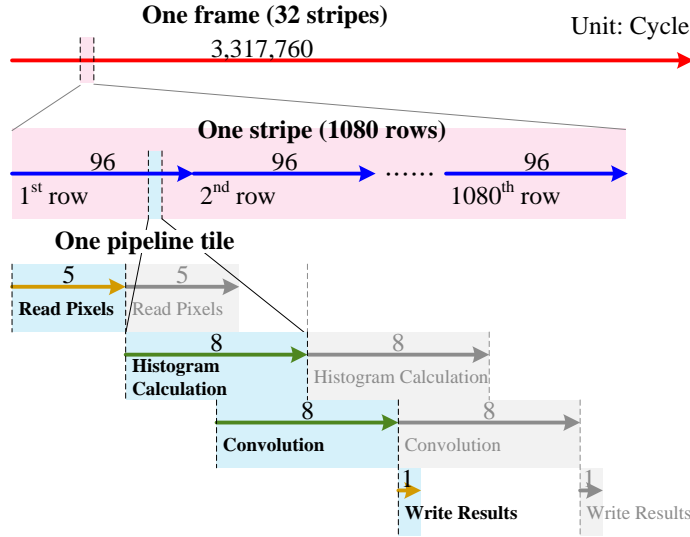


Figure III-22 Schedule of the proposed architecture

2. Architecture Components

In the core, the main components are two histogram calculation engines and one convolution engine for the computation in Table III-6, which have high computational complexity as mentioned above. Thus, the proposed R -parallelism method unrolls all computational loops in the range domain R . The details of this method are described in each engine as follows.

(1) Histogram Calculation Engine

The histogram calculation engines perform the integration and extraction processes for hc_c and hi_c as shown in Table III-6. With the R -parallelism method, we design their architectures as shown in Figure III-24, where the selected-bin adder (SBA) is depicted in Figure III-23. These two engines can achieve the throughput of 1 histogram/cycle. Note that the difference of the two engines is that the integral value of SBAs is the source pixel J in the engine hi_c , instead of the constant 1 in the engine hc_c . In addition, all bit widths of data in the engine hi_c are more than those in hc_c by 8 bits.

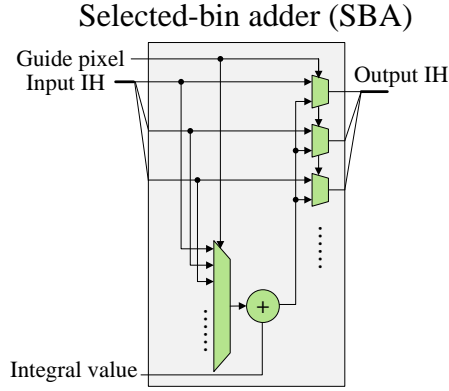


Figure III-23 Selected-bin adder in the histogram calculation engines

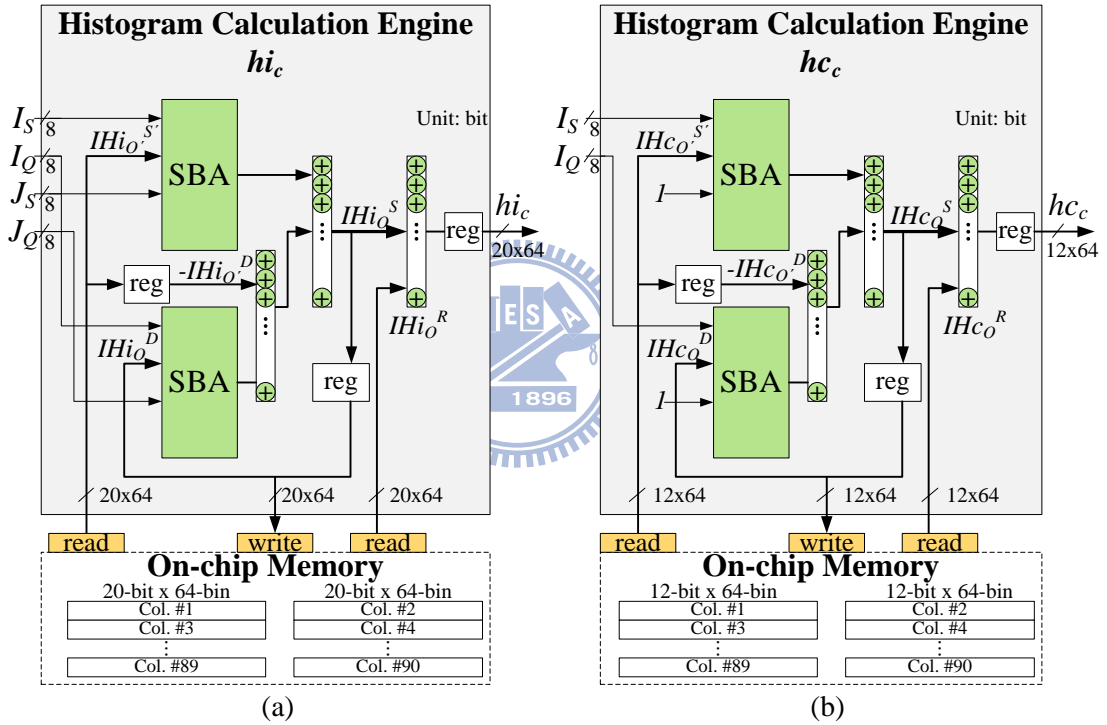


Figure III-24 Proposed architectures of histogram calculation engines hi_c and hc_c

In above architectures, each engine needs to access the five IHs: $IH_O^{S'}$, $IH_O^{D'}$, IH_O^S , IH_O^D , and IH_O^R , from on-chip memory in one cycle. To reduce the bandwidth problem, we propose the delay-buffer method, which is presented as follows by data dependency of the associated IHs in two successive cycles. Assume that the pixels S , S' , D , and D' shown in Figure III-20 (d) are located (x,y) , $(x,y-I)$, $(x-I,y)$, and $(x-I,y-I)$ in the cycle t , respectively. Hence, their IHs can be notated by

$$S^{(t)}: IH_0^{(x,y)}, S'^{(t)}: IH_0^{(x,y-1)}, D^{(t)}: IH_0^{(x-1,y)}, D'^{(t)}: IH_0^{(x-1,y-1)} . \quad (III-21)$$

For the next cycle $t+1$, their x -coordinates are increased by 1 as follows,

$$S^{(t+1)}: IH_0^{(x+1,y)}, S'^{(t+1)}: IH_0^{(x+1,y-1)}, D^{(t+1)}: IH_0^{(x,y)}, D'^{(t+1)}: IH_0^{(x,y-1)} . \quad (III-22)$$

From the (III-21) and (III-22), we can find that $D^{(t+1)}$ equals $S^{(t)}$, and $D'^{(t+1)}$ equals $S'^{(t)}$. That means $IH_0^{D'}$ and IH_0^D can be obtained by delaying $IH_0^{S'}$ and IH_0^S for one cycle, respectively. Therefore, we can use two delay-buffers to avoid accessing $IH_0^{D'}$ and IH_0^D from the on-chip memory, and reduce bandwidth from five IHs to three IHs.

(2) Convolution Engine

The convolution engine uses the histograms hc_c and hi_c to further compute the result pixel by the kernel calculation and convolution processes in Table III-6. Its architecture is shown in Figure III-25 (a). With the proposed R -parallelism method, the convolution process can achieve the throughput of 1 pixel/cycle. Higher throughput can be further attained by adding the registers at the available cut-lines for pipelining in the figure, which can enable operating frequency be higher.

The R -parallelism method brings high throughput but suffers from large size and large number of range table. For the large size, we take advantages of the symmetry and truncation property of Gaussian function to decrease its size from 256 to 32. In addition, to avoid the large number of range table, we share one table by the table selection module as shown in Figure III-25 (b), which reduces the number of table to one. Note that the result of divisor would directly be in the range of 8-bit because it is used to normalize the sum of pixels with weight (III-10).

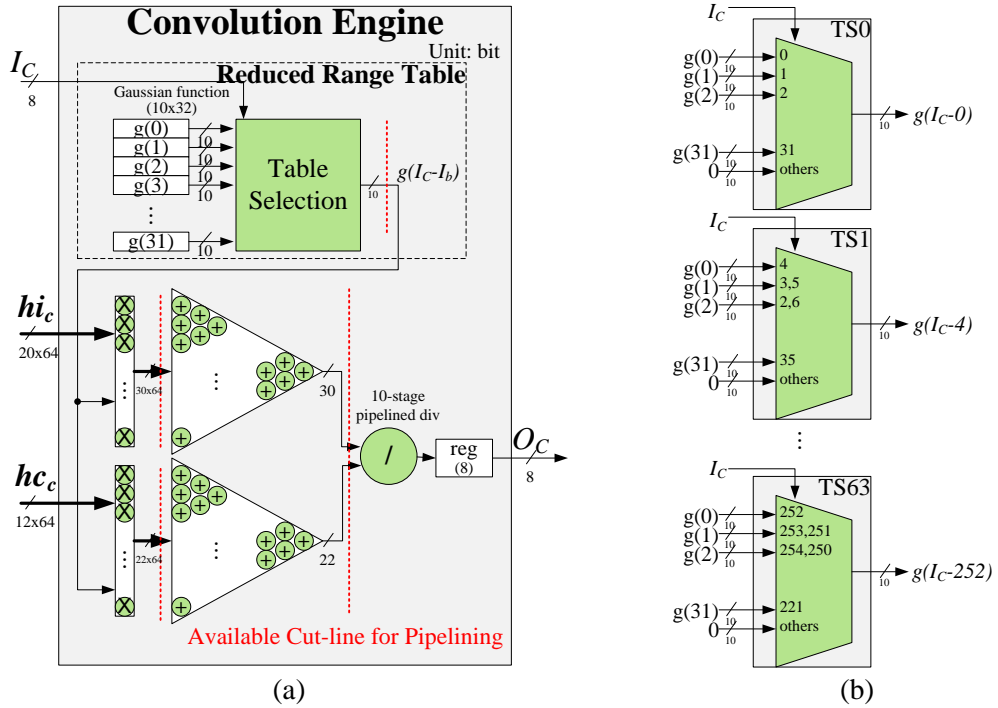


Figure III-25 Proposed architecture of (a) convolution engine and (b) its table selection modules

Furthermore, the histogram calculation engines and the convolution engine can be serially connected to achieve the throughput of 1 pixel/cycle. More engines can be used to process multiple cascaded pixels simultaneously for higher throughput. The proposed memory reduction methods could be directly extended to support the processing of multiple pixels.

3.3.5 Implementation Result

Referring to the quality analysis in [91], we select 31 for $|S|$ and 64 for N_b in our implementation. The proposed architecture of JBF has been implemented by Verilog and synthesized under the 90-nm CMOS technology process. Table III-7 lists the implementation result of the proposed architecture. The hardware design could achieve the throughput of HD1080p 60 frames/s that is 124 Mpixels/s by 23K-byte memory cost and 356K gate counts.

Table III-7 Example implementation result of the proposed architecture

Technology Process		UMC 90nm	
Image Size $M \times N$		1920x1080	
Number of Bin N_b		64	
Filter Window Size $ S ^2$		31x31	
Stripe Width w_s		60	
Clock Rate (Hz)		100M	200M
Frame Rate (Frame/s)		30	60
Logic Cost Excluding Memories (Equivalent Gate-Count)	Interface	9,578	9,917
	Histogram Cal.	97,766	148,649
	Convolution	168,333	197,351
	Total	276,178	355,917
On-chip Memory (Byte)		23K	23K

Table III-8 compares the hardware costs between the proposed methods and the original integral histogram in different resolutions. With the proposed memory reduction and architecture design techniques, the complexity can be reduced to 0.15%, and the memory requirement can be reduced to 0.003%-0.02%. In addition, the bandwidth for IH (i.e. on-chip bandwidth) can be reduced to 32%-36%, but the bandwidth for pixel (i.e. off-chip bandwidth) is increased to 20.3-132.7 Mbits. Nevertheless, the off-chip bandwidth is affordable by the 64-bit bus processing at 200 MHz.

Table III-8 Comparison of hardware cost per frame

	Resolution	Complexity (million operation)	Memory Requirement (Kbyte)	Bandwidth for IH (Mbit)	Bandwidth for pixel (Mbit)
Original	VGA	335.1 (100%)	113,050 (100%)	14,470 (100%)	9.8 (100%)
	HD720p	1,005.5 (100%)	353,894 (100%)	45,299 (100%)	29.5 (100%)
	HD1080p	2,262.3 (100%)	829,440 (100%)	106,108 (100%)	66.4 (100%)
Mem. Reduction	VGA	197.0 (59%)	23 (0.020%)	9,083 (63%)	20.3 (206%)
	HD720p	591.1 (59%)	23 (0.007%)	27,250 (60%)	60.8 (206%)
	HD1080p	1,289.7 (57%)	23 (0.003%)	59,454 (56%)	132.7 (200%)
Mem. Reduction +	VGA	5.1 (0.15%)	23 (0.020%)	5,191 (36%)	20.3 (206%)
	HD720p	1.5 (0.15%)	23 (0.007%)	15,571 (34%)	60.8 (206%)
Archi. Design Tech.	HD1080p	3.3 (0.15%)	23 (0.003%)	33,974 (32%)	132.7 (200%)

Table III-9 compares our proposed hardware design with the previous VLSI implementations. The previous implementations [94], [97] could support large filtering window but low throughput, while the implementations [95], [96] could reach high throughput for small filtering window only. Our design can not only achieve high throughput but also support large filtering window. Table III-10 compares our design with the other previous GPU and CPU implementations. Comparing to other

design, the proposed architecture could efficiently utilize the hardware cost to achieve high throughput.

Table III-9 Previous VLSI implementations of bilateral filtering

	[94]	[95]	[96]	[97]	Our Design
Supported Window Size	15x15 BF	3x3 BF-like	5x5 BF	11x11 BF	31x31 BF/JBF
Implementation Method	Xilinx Spartan-3 FPGA	Altera Cyclone-II FPGA	Xilinx Vertex-5 FPGA	TSMC 0.18um Tech. Proc.	UMC 90nm Tech. Proc.
Throughput (pixel/s)	4.8M	124M	41.9M	11M	124M

Table III-10 Comparison of different implementations

	Support-Pixel-First				Target-Pixel-First	
	Durand and Dorsey [84]	Chen <i>et al.</i> [87]	Yang <i>et al.</i> [83]	Adams <i>et al.</i> [88]	Porikli [91]	Proposed
Approach	Piecewise-linear Subsampling ($s_s=24, s_r=19$)	Bilateral Grid ($s_s=16, s_r=10$)	Piecewise-linear ($s_r=32$)	Gaussian KD-tree	Integral Histogram ($s_r=4$)	Integral Histogram ($s_r=4$)
Implementa tion	CPU P4 2GHz	GPU Geforce 8800GTX	GPU Geforce 8800GTX	GPU GeForce GTX260	CPU P4 3.2GHz	ASIC
Transistor count (Tech. Process)	55M (130nm)	681M (90nm) [98]	681M (90nm) [98][98]	1,400M (TSMC 65nm) [99]	55M (130nm)	2.5M (UMC 90nm)
Image Size (Pixel)	10.4M	1.0M	1.0M	10M	1.0M	2.07M
Frame Rate (Frame/sec)	0.16 (high dynamic range)	222	66	0.01-1	3.22	60
Throughput (Pixel/sec)	1.6 M	222M	66M	0.1M-10M	3.22M	124M
Memory (Byte)	-	625K	4M	1G-100M	96M	23K

3.4 Baseline Disparity Estimation Algorithm

In this section, we first present the baseline disparity estimation algorithm, which applies the baseline BP and JBU algorithms. Then, we demonstrate the disparity quality comparison between the proposed baseline algorithm and the DERS algorithm.

3.4.1 Baseline Algorithm

Figure III-26 shows the baseline disparity estimation algorithm that combines the baseline BP and JBU algorithms. In the baseline algorithm, the sampling factor is set as 1/2, 1/4 for horizontal and vertical direction. Note that the horizontal sampling factor could not further decrease since the detailed disparity would be lost. In addition, all the steps in the algorithm flow are performed for three times for calculating the three view disparity maps by the software implementation.

In the first step, the 3×3 SAD match metric is adopted to calculate the initial cost cube C_0 using the high resolution images $I_{H,L}$, $I_{H,C}$, $I_{H,R}$. Note that the matching costs are computed only for the sampled pixels but with the full disparity range. Thus, the size of C_0 is $(H/4) \times (W/2) \times DR$. Then, the 5×7 ADSW cost aggregation method [3] and the baseline BP algorithm [24] are performed to compute the low resolution disparity maps $D_{L,L}$, $D_{L,C}$, $D_{L,R}$. In the baseline BP, we employ the Potts model to the smoothness term and data term, and execute the baseline BP for 15 iterations. Finally, the low resolution disparity maps are scaled up to the high resolution ones $D_{H,L}$, $D_{H,C}$, $D_{H,R}$ by the JBU algorithm.

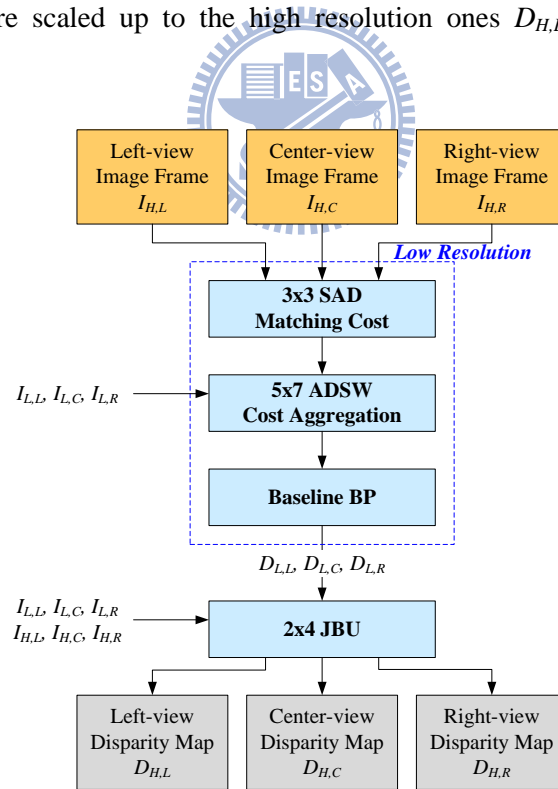


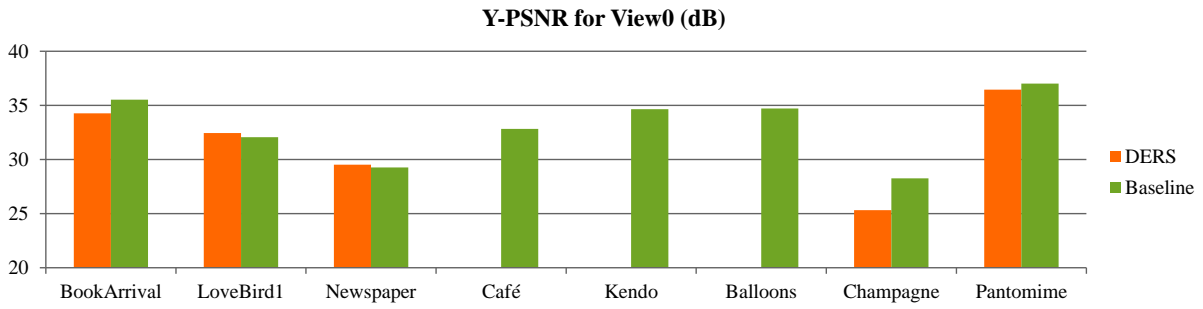
Figure III-26 Flow of the proposed baseline disparity estimation algorithm

3.4.2 Comparison

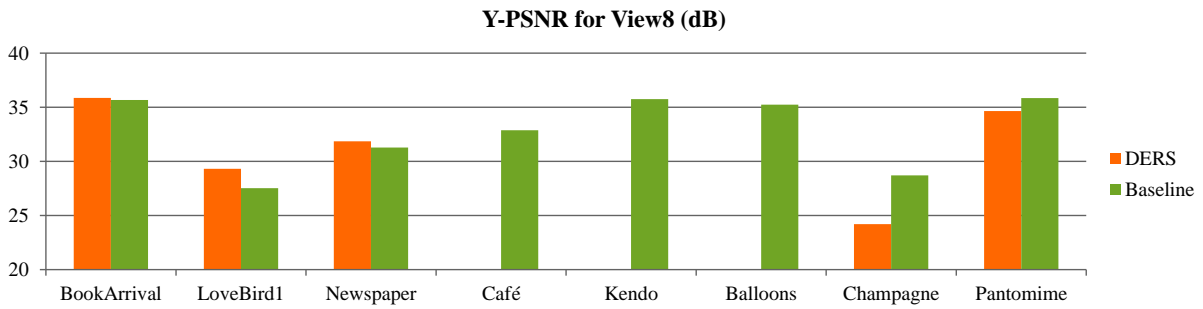
In the experiment, the disparity maps and the synthesized videos are computed by the baseline algorithm and the DERS algorithm with the configuration of Figure II-17. Furthermore, the synthesized videos are evaluated by the PSNR and PSPNR methods mentioned in the Section 2.3.4.

Figure III-27 shows the evaluation results of them. In the evaluation results, the “View0” and “View8” refer to the most-left-view and the most-right-view videos in the output of 3-view configuration for 9-view display. The results of the DERS algorithm are not available for the test sequences Café, Kendo, and Balloons due to insufficient input views. The more details of the test sequences are presented in Chapter V. For the Y-PSNR results, the baseline algorithm has the quality changes from -1.78 dB to 4.51 dB, compared to the DERS algorithm. On the other hand, the T_PSPNR results have large variance in the test sequences. The worst case has the large drop of 4.49 dB because of no temporal consistency enhancement method adopted in the baseline algorithm.

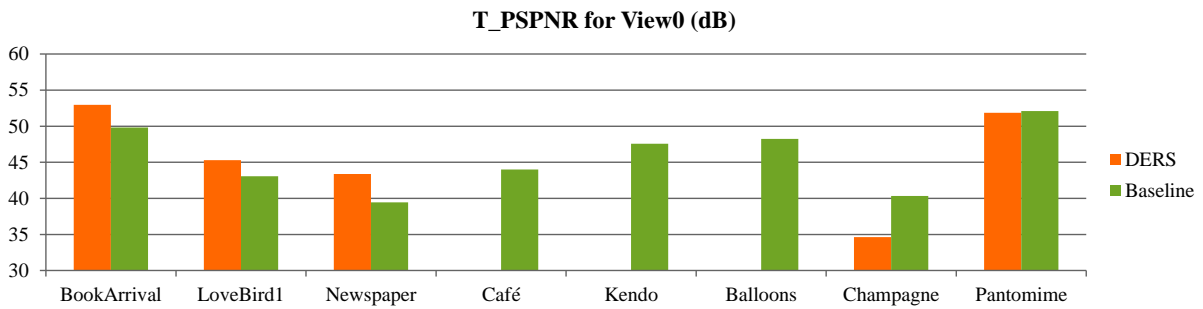
Figure III-28 and Figure III-29 shows the disparity maps and the synthesized images. Compared to the DERS algorithm, the baseline algorithm additionally suffers from the incorrect disparities in the textureless regions. Nevertheless, those incorrect disparities do not impact on the synthesized image of baseline algorithm. In addition, the disparities at the object boundary are over blurred in the baseline algorithm. That would result in the background distortion if the background has texture.



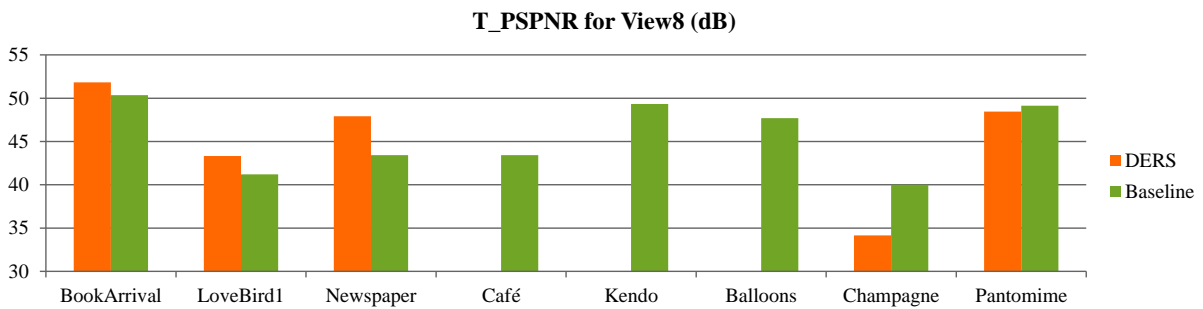
(a)



(b)



(c)



(d)

Figure III-27 Experimental results of the baseline algorithm and the DERS algorithm



(a)



(b)



(c)



Figure III-28 Center disparity maps and synthesized View8 of baseline algorithm at the 100th frame
 (a) BookArrival, (b) LoveBird1, (c) Newspaper, (d) Champagne, (e) Pantomime



(a)



(b)



(c)

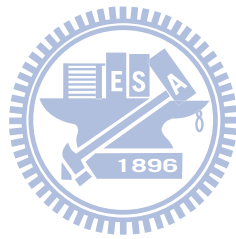


Figure III-29 Center disparity maps and synthesized View8 of DERS algorithm at the 100th frame
 (a) BookArrival, (b) LoveBird1, (c) Newspaper, (d) Champagne, (e) Pantomime

3.5 Summary

For the high definition 3DTV applications, our strategy is to increase the computational parallelism by the baseline BP algorithm, and reduce the processed frame size by the JBU algorithm. The computational characteristics and design challenges of the two main algorithms are analyzed in this chapter. To conquer their design challenges, we propose the low-memory-cost memory access approaches, and the parallel computing architectures for their kernel components. In the experimental results, the baseline algorithm could deliver comparable disparity quality to the DERS algorithm. However, it still suffers from high computational complexity because of high iteration count in BP. In

addition, the disparity quality should be further improved, especially for the temporal consistency problem.



IV Advanced Disparity Estimation Algorithms for High Definition 3DTV Applications

Based on the previous baseline algorithm, we propose three new advanced disparity estimation algorithms in this chapter. The first high-quality algorithm focuses on the disparity quality improvement, including the temporal consistency enhancement and the occlusion handling. The second sparse-computation algorithm could reduce the computation of high-quality algorithm by the sparse-computation strategy, and it could be accelerated by the implementation of software programming. The third hardware-efficient algorithm simplifies the massive computation in high-quality algorithm, and reduces the high memory cost of BP optimization. The experimental results and evaluation will be compared with the DERS algorithm in Chapter V, and the last hardware-efficient algorithm will be further implemented by VLSI design in Chapter VI.

4.1 High-Quality Disparity Estimation Algorithm

The proposed high-quality disparity estimation (HQ-DE) algorithm is presented in this section. This section first reviews the state-of-the-art disparity estimation algorithms, and then describes the details of the proposed HQ-DE algorithm.

4.1.1 Related Work

The state-of-the-art disparity estimation algorithms are the 3DVC's DERS algorithm [63] and the top algorithms in the Middlebury rank [72]. The details of DERS algorithm has been described in Section 2.3. For the algorithms in the Middlebury rank, we review the high-quality BP-based algorithms including the adaptive-BP [39] and the double-BP [40]. In addition, we also introduce the enhanced-BP [41], since it additionally takes the temporal consistency into consideration.

1. Adaptive-BP

Figure IV-1 shows the algorithm flow of adaptive-BP, whose main idea is to apply the BP optimization to a segment-based graph, instead of the conventional pixel-based graph. In this algorithm, the mean-shift segmentation [69] is first performed to obtain over-segment information. Then the SAD match metric is applied to the pixel matching cost using for pixel intensity and gradient. In the segment cost calculation, the plane fitting [46] is used to determine a disparity plane for each segment, and the pixel costs of a disparity plane are summed up as the segment cost. According to the disparity planes and segment costs, this step iteratively merges disparity planes and segments. Finally, the segment-based BP is performed in a segment-based graph. The adaptive-BP could produce high quality disparity results, but it suffers from irregular computation due to its complex connected segment-based graph.

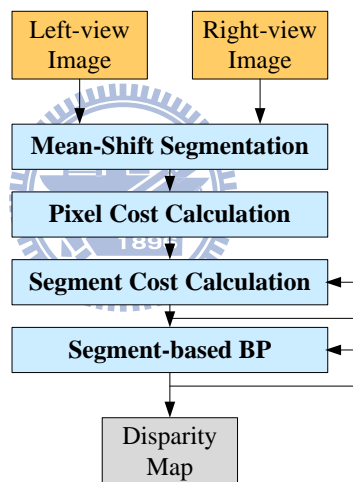


Figure IV-1 Flow of the adaptive-BP algorithm [39]

2. Double-BP

Figure IV-2 shows the algorithm flow of double-BP, which consists of the three main steps: initial stereo, pixel classification, and iterative refinement. The initial stereo step computes the initial cost cubes by the ADSW approach [7], and performs the hierarchical BP (HBP) [25] to obtain the initial disparity maps for two views. With the initial disparity map and cost cubes, the pixel classification step categorizes the pixels into occluded, stable, and unstable ones using the mutual consistency and correlation confidence checks. Then the iterative refinement step performs the plane fitting and the

HBP for five iterations. In the iterative process, the disparity map and the cost cube are updated in each iteration. The double-BP could deliver better disparity maps than the adaptive-BP because it has different approaches to deal with the classified pixels.

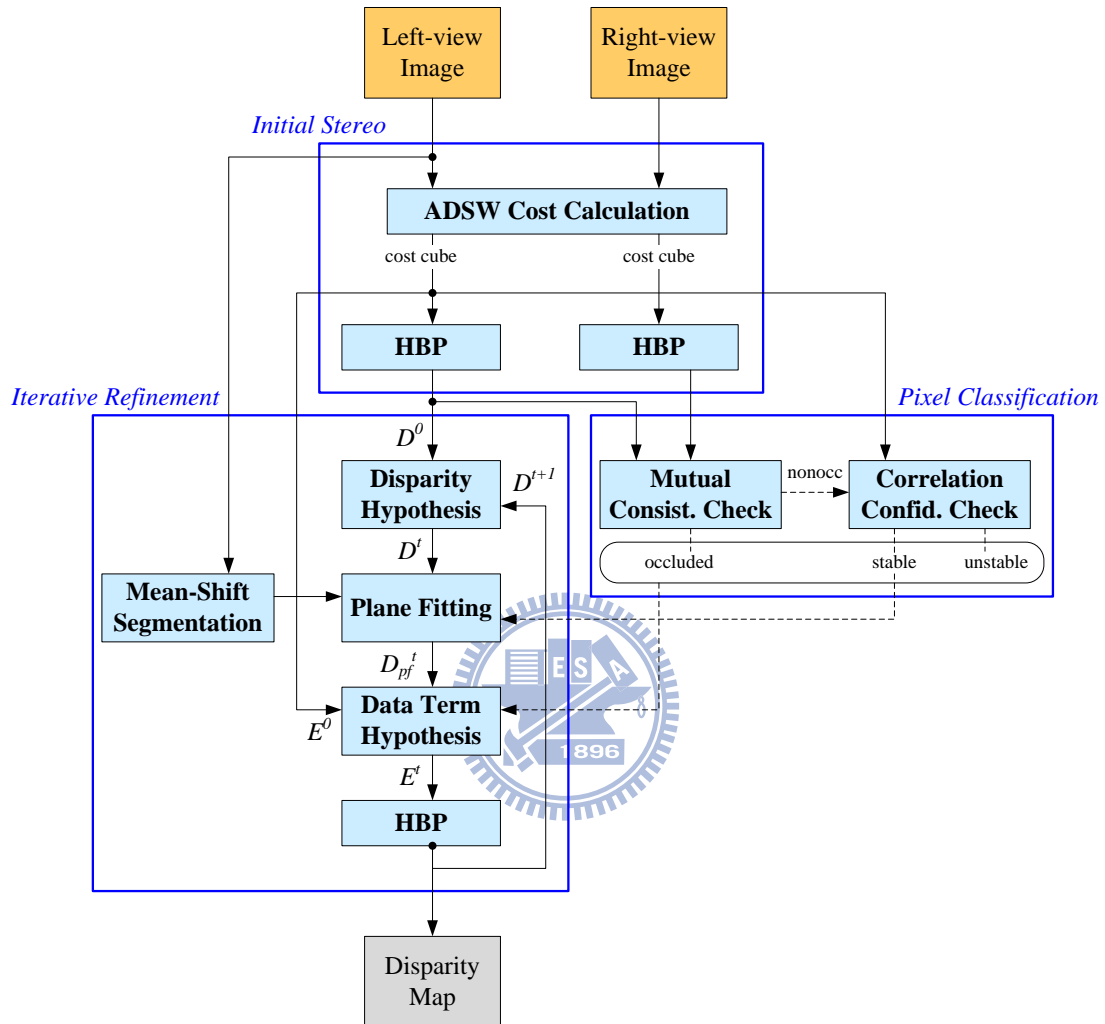


Figure IV-2 Flow of the double-BP algorithm [40]

3. Enhanced-BP

The enhanced-BP [41] proposed three techniques to improve the conventional BP algorithm. The first technique defines a new graph with 6-connected nodes, which have 4 original spatial neighbors and 2 temporal neighbors, to enhance the temporal consistency. In BP optimization, the connection to neighbors would be broken at the boundaries of color segment and motion regions. The second technique is to deal with the occlusion problem by the plane fitting methods or the background clustering method. The last technique is to accelerate the optimization process by inputting the

matching costs to the initial messages, and removing the matching costs from the computation of message passing.

In summary, as the above mentioned state-of-the-art BP-based algorithms and the 3DVC's DERS algorithm, they could produce high quality disparity maps by the common steps: color-constrained cost aggregation, disparity optimization, and segment-based refinement. The color-constrained cost aggregation could be the ADSW method or the segment cost method, the disparity optimization could be GC or BP approach, and the segment-based refinement is the common-used plane-fitting. Therefore, our developed HQ-DE algorithm should include the above common steps.

4.1.2 Observation in DERS and Baseline Algorithms

Based on the baseline algorithm in Section 3.4.1, the HQ-DE algorithm adopts the BP for disparity estimation, and the joint bilateral upsampling (JBU) algorithm to reduce the native computation in high resolution frame. For the disparity quality improvement, we focus on the temporal consistency and the occlusion problems.

At first, we observe the disparity results of the DERS and the baseline algorithms as follows. Figure IV-3 shows the flicker artifact of the baseline algorithm. The stand of poster behind the chair has a little change on its boundary in the continuous frames due to no temporal consistency enhancement in the baseline algorithm. The slight change would result in the noticeable flicker artifact for human. On the other hand, the DERS algorithm has the temporal consistency enhancement but suffers from the foreground copy artifact as shown in Figure IV-4. In which, the door pivot is changed after the man passed because the disparity of the man remains on the door pivot.

Figure IV-5 shows the occlusion problem in the DERS and the baseline algorithms. Compared to the reference golden image in Figure IV-5 (a), the DERS and the baseline algorithms suffer from the distortion of red sketch because its background disparities are incorrect.



Figure IV-3 An example of flicker artifact of the baseline algorithm in BookArrival
Synthesized videos from left to right are the 9th to 12th frames.

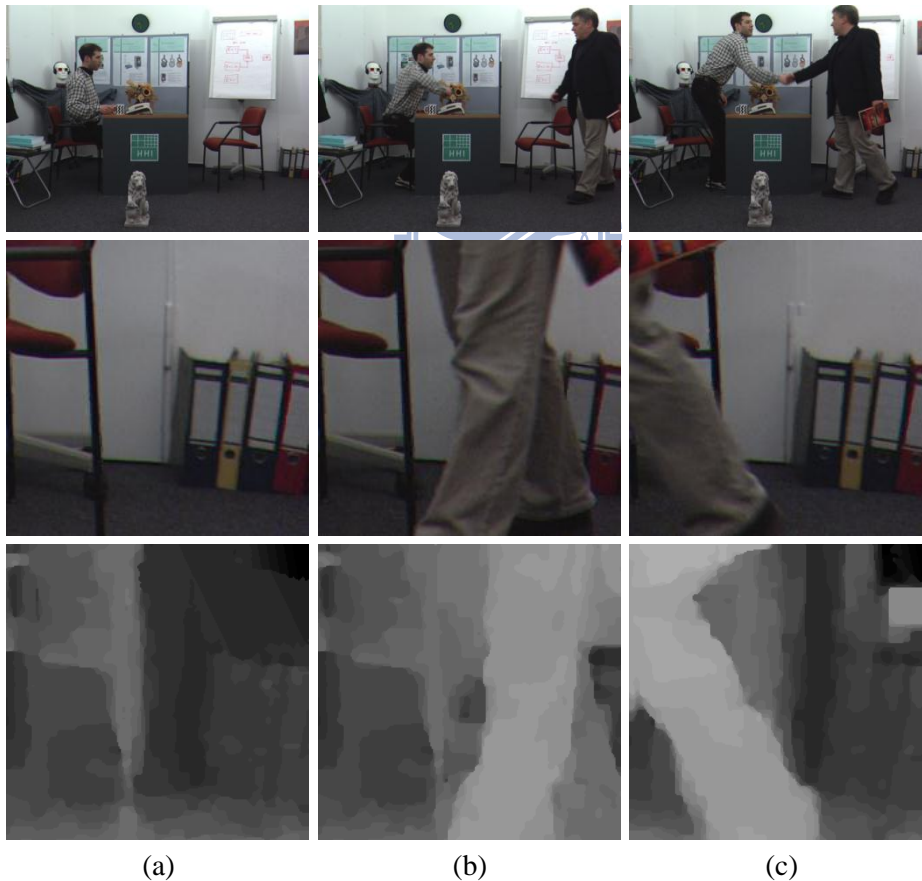


Figure IV-4 An example of foreground copy artifact of the DERS algorithm in BookArrival
Top to bottom are the synthesized frame, interested region of synthesized image and disparity map.
Left to right are (a) the 1st frame, (b) the 25th frame, (c) the 40th frame.

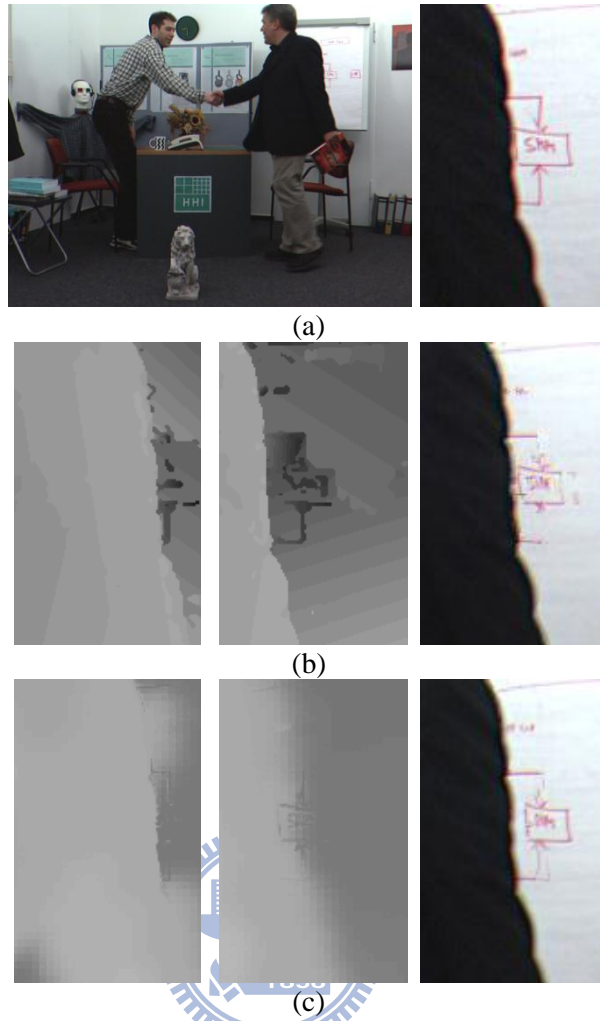


Figure IV-5 An example of occlusion problem at the 44th frame of BookArrival
 (a) reference golden image, (b) the disparity maps of DERS algorithm and the synthesized image, (c) disparity maps of baseline algorithm and synthesized image.

4.1.3 Proposed Algorithm Flow

With the above observation, the temporal consistency and the occlusion problems need to be solved in the HQ-DE algorithm. The main flow of the proposed HQ-DE algorithm is shown in Figure IV-6 for the center view and Figure IV-7 for the left and right views. In which, I and D refer to the image frame and disparity map, respectively, and the superscript t and $t-1$ refer to the current frame and the previous frame. Besides, the first subscript, L or H , means the low resolution or the high-resolution frame, and the second subscript means the view point.

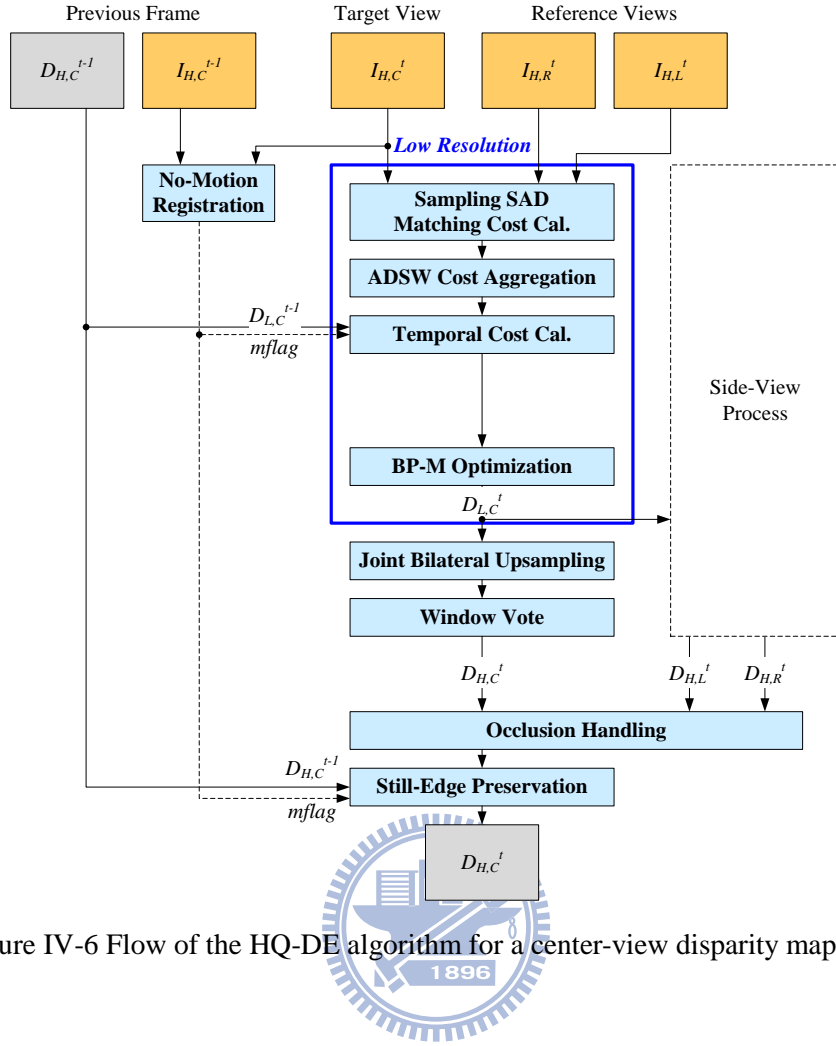


Figure IV-6 Flow of the HQ-DE algorithm for a center-view disparity map

In the main flow, the high-resolution image frames at target-view and reference-view are fetched to compute an initial low-resolution disparity map D_L^t by the steps of matching cost calculation, cost aggregation, and BP optimization. Then, the low-resolution disparity map D_L^t is scaled up to the high-resolution disparity map D_H^t by the JBU algorithm, and refined by the window vote method.

For the occlusion problems, the three view disparity maps are cross handled in the occlusion handling step. On the other hand, for the temporal consistency problems, the proposed no-motion registration (NMR) method and the still-edge preservation (SEP) method are attached into the main flow to respectively deal with the foreground copy artifact and the flicker artifact. In addition, the side-view algorithm flow additionally has the inter-view cost calculation step, which could constrain the side-view disparity estimation using the more reliable center disparity map $D_{L,C}^t$. The details of each step are described in the following sections.

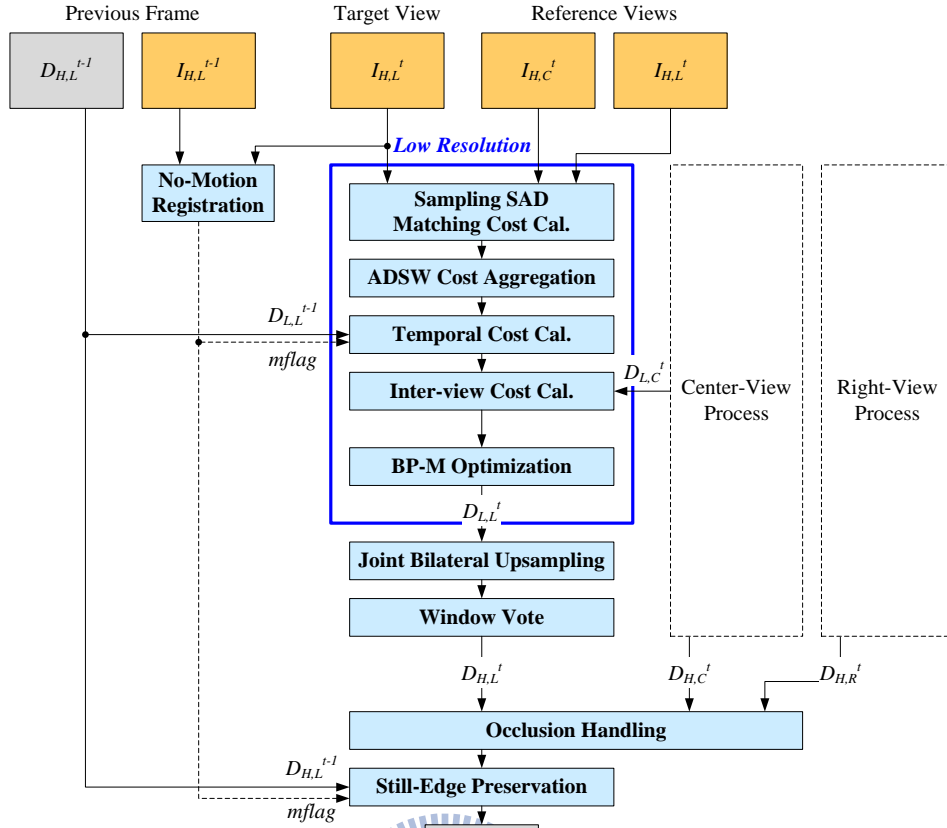


Figure IV-7 Flow of the HQ-DE algorithm for a side view disparity map

4.1.4 Downsampled Disparity Estimation for Full Range Disparity

The downsampled disparity estimation process consists of the matching cost, cost aggregation, and the BP optimization steps, which are performed only for the downsampled pixels at the positions as shown in Figure IV-10 (a). In which, the sampling factor is 1/2 for horizontal direction and 1/4 for vertical direction according to the simulation results as listed in Table IV-1 and depicted in Figure IV-8. The details of test sequences and evaluation method are elaborated in Chapter V. The selected sampling factors could keep the view synthesis quality for all resolutions, especially for the smaller size of 1024x768. Figure IV-9 compares the results of the sampling factors 1/2x1/4 and 1/4x1/4. The latter one would suffer from more serious artifacts in the disparity map and synthesized image.

Table IV-1 Simulation results with different sampling factors in Y-PSNR (dB)

Hori. Sampling Factor	Vert. Sampling Factor	Book Arrival	Love Bird1	Newspaper	Café	Kendo	Balloons	Champagne	Pantomime	Avg.
1/2	1/2	36.40	30.72	30.77	N.A.	36.17	34.10	30.73	38.47	33.91
1/2	1/4	36.34	30.89	30.79	33.96	36.00	33.97	30.46	38.38	33.85
1/2	1/8	36.07	30.85	30.62	33.78	35.89	33.67	29.38	37.56	33.48
1/2	1/16	35.57	30.85	30.29	33.05	35.24	32.77	28.84	37.54	33.02
1/4	1/2	36.00	30.77	30.73	33.82	35.75	33.81	29.83	38.48	33.65
1/4	1/4	35.90	30.92	30.64	33.96	36.04	33.64	29.79	38.46	33.67
1/4	1/8	35.68	30.87	30.57	33.71	35.73	33.18	29.93	38.46	33.52
1/4	1/16	35.27	30.90	30.12	32.45	35.10	32.66	29.12	38.40	33.00
1/8	1/2	35.66	30.77	30.24	33.21	35.50	33.00	29.04	38.48	33.24
1/8	1/4	35.49	30.77	30.15	33.16	35.37	32.89	29.40	38.49	33.21
1/8	1/8	35.23	30.73	30.18	32.16	35.08	32.43	29.01	38.47	32.91
1/8	1/16	34.66	30.68	29.76	30.84	34.53	32.32	28.82	38.44	32.50
1/16	1/2	34.56	30.38	29.43	32.04	34.42	31.92	29.30	38.42	32.56
1/16	1/4	34.62	30.51	28.95	31.66	34.55	32.15	29.07	38.45	32.50
1/16	1/8	34.46	30.46	29.11	31.11	34.29	31.89	34.13	38.43	32.99
1/16	1/16	34.17	30.67	28.08	30.60	33.88	31.58	27.85	38.49	31.92

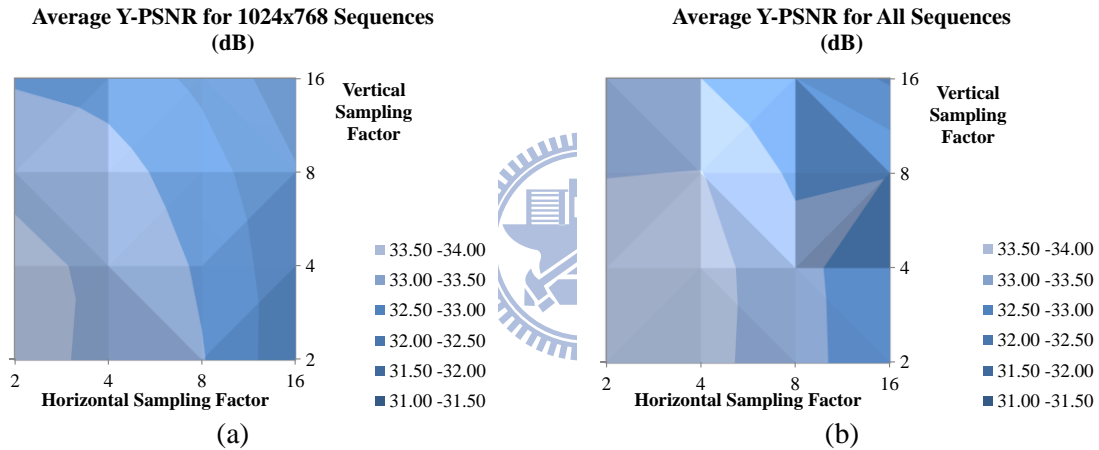


Figure IV-8 Comparison of different sampling factors in the average Y-PSNR of two frames

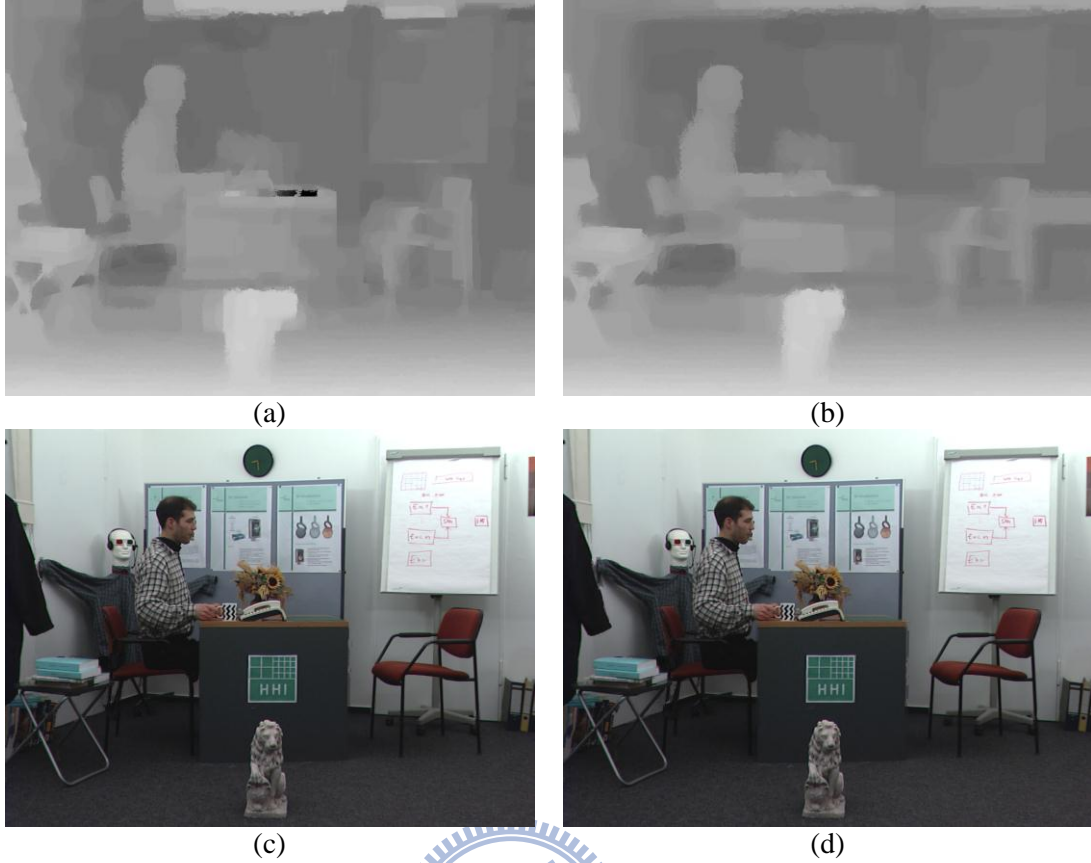


Figure IV-9 Simulation results using the sampling factors of $1/2 \times 1/4$ and $1/4 \times 1/4$

(a) and (b) are the center-view disparity map and the most left synthesized image with $1/2 \times 1/4$. (c) and (d) are the center-view disparity map and the most left synthesized image with $1/4 \times 1/4$.

In the matching cost calculation step, we propose the sampling sum of absolute difference (SSAD) match metric, which calculates the matching costs only for the downsampled pixel, and considers the full disparity range to avoid the loss of disparity precision. Figure IV-10 (b) illustrates the SSAD match metric. For the target downsampled pixel, the reference pixels in the full disparity range are used to calculate matching costs by the 2×4 SAD match metric. Thus, the SSAD matching cost is defined as

$$SSAD_{tar-ref}(x, y, d) = \sum_{\substack{2x \leq u < 2(x+1) \\ 4y \leq v < 4(y+1)}} |I_{H,tar}(u, v) - I_{H,ref}(u + d, v)|, \quad (IV-1)$$

where $I_{H,tar}$ is the high-resolution target-view image, $I_{H,ref}$ is the high-resolution reference-view image, and $SSAD_{tar-ref}$ is the low-resolution matching cost. By the SSAD match metric, the initial cost cubes $C_{0,C}$, $C_{0,L}$, $C_{0,R}$ for the three input views are calculated by

$$C_{0,C}(x, y, d) = \min\{SSAD_{C-L}(x, y, d), SSAD_{C-R}(x, y, -d)\} , \quad (IV-2)$$

$$C_{0,L}(x, y, d) = \min\{SSAD_{L-C}(x, y, -d), SSAD_{L-R}(x, y, -2d)\} , \quad (IV-3)$$

$$C_{0,R}(x, y, d) = \min\{SSAD_{R-C}(x, y, d), SSAD_{R-L}(x, y, 2d)\} , \quad (IV-4)$$

where the minimal *SSAD* from two reference views is selected for the initial cost cube. Note that the disparity index in the *SSAD* match metric is associated with the relative position of target view and reference view.

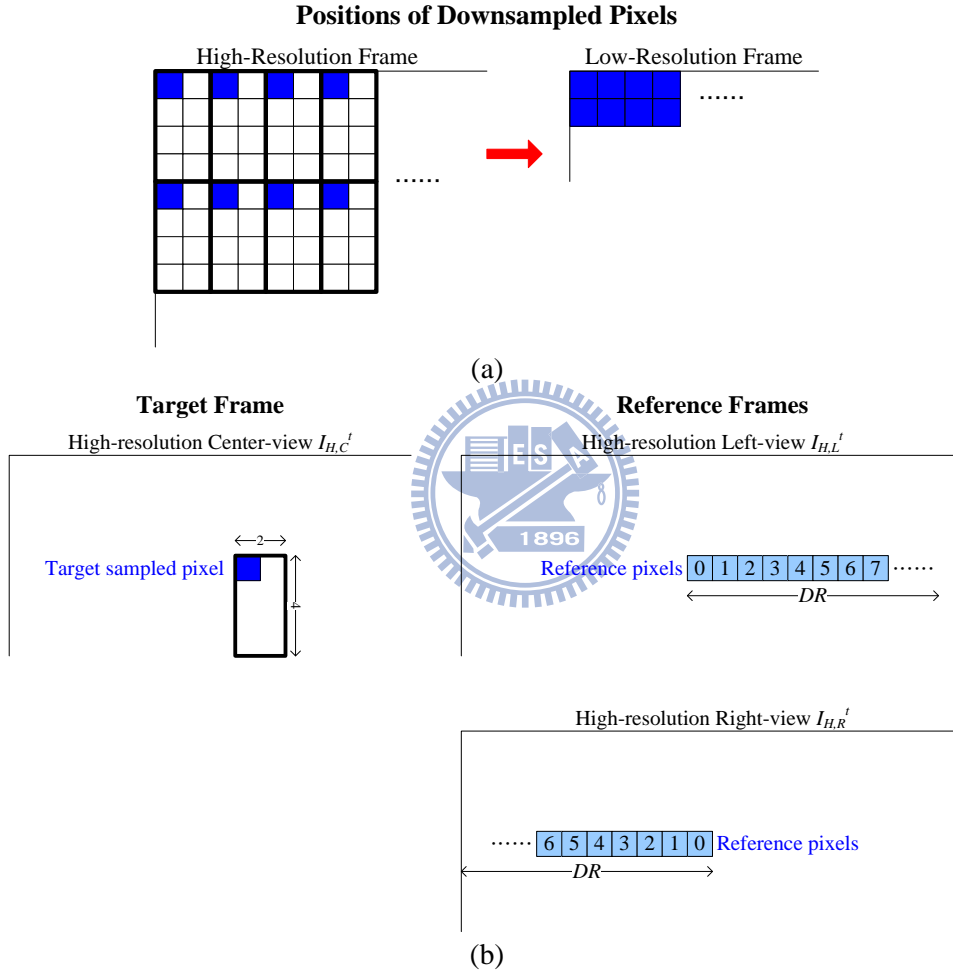


Figure IV-10 Illustration of downsampled disparity estimation for full disparity range
(a) positions of downsampled pixels in high-resolution frame, (b) example of the matching cost for the center view

In the cost aggregation step, we adopt the simplified adaptive support-weight (ADSW) [10], which is defined as

$$C_{aggr,C}(x, y, d) = \frac{1}{\kappa} \sum_{(u,v) \in S} C_{0,C}(u, v, d) f(\|(x, y) - (u, v)\|) g(\|I_{L,C}(x, y) - I_{L,C}(u, v)\|) \quad (IV-5)$$

for center view disparity estimation. In this equation, the initial matching costs $C_{0,C}$ in the aggregation window S are accumulated with the Gaussian weights of spatial kernel f and range kernel g , and κ is the normalized term. The size of the aggregation window $|S|$ is set as 5×7 in the HQ-DE algorithm. Compared to the original ADSW in [7], the simplified ADSW contains two simplification techniques. First, only one support-weight referring to the target view image is used. Second, the computation of spatial distance and color distance is simplified to the Manhattan color distance. The same computation in (IV-5) could be applied to compute the left-view and right-view costs $C_{aggr,L}$, $C_{aggr,R}$.

With the three view aggregated cost cubes, the BP optimization is separately performed to calculate the low resolution disparity maps $D_{L,C}$, $D_{L,L}$, $D_{L,R}$. For the BP optimization, the baseline algorithm adopts the baseline BP [24] but suffers from slow convergence due to one-pixel-distance message passing in each iteration. In the BP-based algorithms, the hierarchical BP (HBP) [22] and the max-product loopy BP (BP-M) [26] could address the slow convergence problem. The former performs the message passing by the coarse-to-fine manner, while the later on performs the message passing separately in four directions. In the proposed HQ-DE algorithm, we adopt the BP-M with single iteration. In addition, the Potts model is applied to the data and smoothness term. They are defined as

$$D(d_i) = \min\{C_{aggr}(x, y, d), \tau_D\} \cdot \lambda_D \quad (IV-6)$$

$$V(d_i, d_j) = \min\{|d_i - d_j|, \tau_V\} \cdot \lambda_V \quad (IV-7)$$

where τ_D , τ_V are for truncation, and λ_D , λ_V are for scaling.

In addition to the above three steps, the downsampled disparity estimation has the inter-view cost calculation step for the side views as shown in Figure IV-7. The concept of inter-view cost C_{view} is that the center-view disparity map is more reliable than other two because of its less occluded regions, and it can be used to constrain the side-view disparity estimation. The inter-view costs for two side view are computed by

$$C_{view,L}(x, y, d) = \lambda_{view} |d - D_{L,C \rightarrow L}(x, y)|, \quad (IV-8)$$

$$C_{view,R}(x, y, d) = \lambda_{view} |d - D_{L,C \rightarrow R}(x, y)|, \quad (IV-9)$$

where $D_{L,C \rightarrow L}$ and $D_{L,C \rightarrow R}$ are the left-view and right-view disparity maps warped from the center-view one in low resolution, and λ_{view} is a constant for scaling. To compute the inter-view costs, the center-view disparity map needs to be first computed by the BP optimization, and it is warped to the side views by the method in Section 2.2.1. Then, we assume the side-view disparity maps will be approximate to the warped one, and give a penalty for the inconsistency through the inter-view cost C_{view} . Besides of the inter-view cost, the temporal cost C_{temp} is also added to the cost cube by

$$C_{total}(x, y, d) = C_{aggr}(x, y, d) + C_{temp}(x, y, d) + C_{view}(x, y, d). \quad (IV-10)$$

Thus, the original cost cube C_{aggr} is replaced by the cost cube C_{total} and is substituted into (IV-6) as the data term for the BP-M optimization. The more details of the temporal cost are presented in Section 4.1.7.

4.1.5 Joint Bilateral Upsampling



The associated disparity upsampling techniques have been introduced in Section 3.1.2. In the proposed HQ-DE algorithm, we adopt the same joint bilateral upsampling (JBU) algorithm [81] as that in the baseline algorithm. Note that the JBU is performed by the single-step process, instead of the original multi-step process. Thus, the single-step JBU is defined as

$$D_H^t(i) = \frac{1}{\kappa} \sum_{j_L \in S} D_L^t(j_L) \cdot f(\|i_L - j_L\|) \cdot g(\|I_L^t(i_L) - I_H^t(j)\|) \quad (IV-11)$$

where the window S is in the low-resolution frame, and its size is set as 7×7 for the HQ-DE algorithm.

In addition, the upsampled disparity map D_H^f is further refined by the proposed window vote method that is modified from the regional vote method in [6]. The original regional vote method could remove the disparity noise by taking the disparity with the most votes in a local region. The regional vote method could approximate the purpose of plane fitting method, which is usually applied in the state-of-the-art disparity estimation algorithms. However, the regional vote method does not perform well for the highly textured region due to its continuous grown region and limited shape as shown in

Figure IV-11 (a). To address it, our proposed window vote method considers all the support disparities in a window, and gives votes to the support pixels $I(u, v)$ if their colors are consistent to the center pixel $I(x, y)$ for all color channels. The proposed window vote method is calculated by

$$D'_H(x, y) = \arg \max_d \{vote(x, y, d)\} \quad (\text{IV-12})$$

where

$$vote(x, y, d) = \sum_{(u,v) \in S} (d = D_H^t(u, v)) \wedge (|I_H^t(x, y) - I_H^t(u, v)| < \tau_{vote}). \quad (\text{IV-13})$$

Figure IV-11 (b) shows that the proposed window vote could obtain the correct result for the case of highly texture. In the HQ-DE algorithm, the window size $|S|$ is 15×15 .

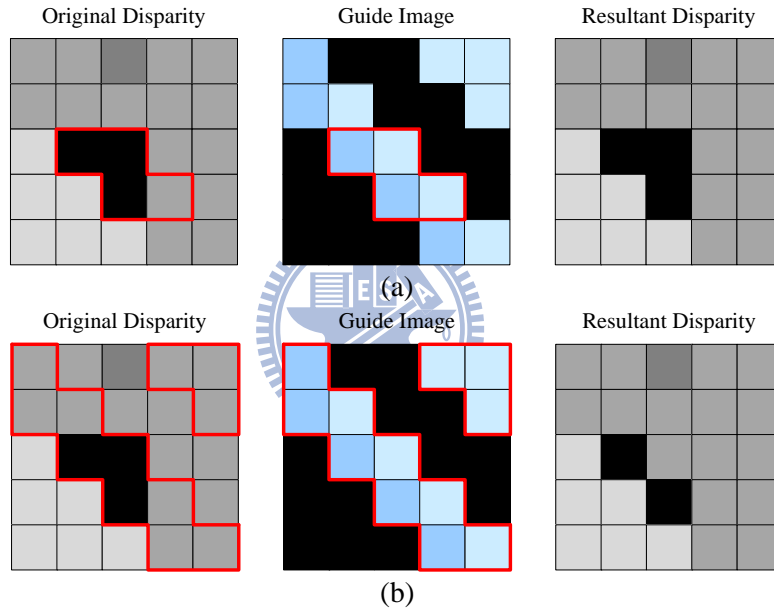


Figure IV-11 Comparison between the original regional vote [6] and the proposed window vote

4.1.6 Occlusion Handling

As the concept of occlusion handling in Section 2.1.2, we proposed a new method for the occlusion handling problem based the left-right check (LRC) method. The proposed occlusion handling method consists of the occlusion detection and the occlusion filling steps. They are described as follows.

1. Occlusion Detection

In the LRC method, the three view disparity maps $D_{H,C}^t$, $D_{H,L}^t$, $D_{H,R}^t$ are cross considered. For an example of center view as the target, the disparities

$$D_{H,C}^t(x,y), \quad D_{H,L}^t(x + D_{H,C}^t(x,y),y), \quad D_{H,R}^t(x - D_{H,C}^t(x,y),y) \quad (IV-14)$$

are compared, and the occlusion map $O_{H,C}$ is determined by

$$O_{H,C}(x,y) = \begin{cases} true & |D_{H,C}^t(x,y) - D_{H,L}^t(x + D_{H,C}^t(x,y),y)| > \tau_{OCC} \\ true & |D_{H,C}^t(x,y) - D_{H,R}^t(x - D_{H,C}^t(x,y),y)| > \tau_{OCC} \\ false & otherwise \end{cases} \quad (IV-15)$$

where τ_{OCC} is the threshold for disparity gap. If the disparity gap in inter views is more than τ_{OCC} , the position would be regarded as an occlusion pixel.

With the occlusion map, we further refine the occlusion region to fit the object boundary by the proposed occlusion extension method. In general, the foreground disparity has stronger confidence, and would affect the neighboring weak background disparity. Thus, the proposed occlusion extension method extends the foreground side of occlusion region according the image information, and dilates the background side by one pixel as illustrated in Figure IV-12. Note that for the original occlusion region detected by the LRC method, its foreground and background sides are determined by the disparities on the boundary for the center view. In which, the boundary with larger disparity is foreground side, and the other boundary with smaller disparity is background side. On the other hand, for the side view, the foreground and background side is fixed. The left-hand-side boundary is background side for the left-view occlusion map, and foreground side for the right-view occlusion map.

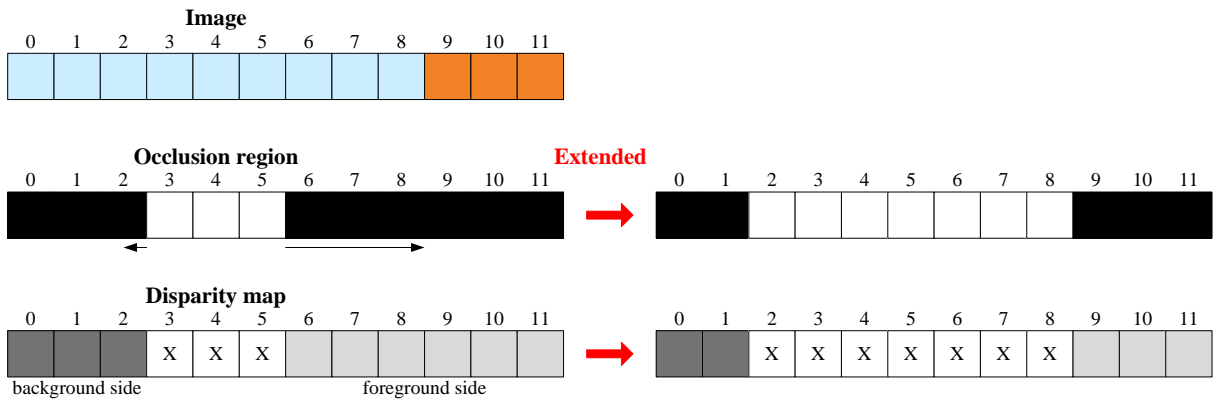


Figure IV-12 Illustration of the proposed occlusion detection method

2. Occlusion Filling

In this step, the occlusion regions are filled with the disparities of visible pixels. The visible pixels could be obtained from the intra frame, inter frame, and inter-view frame. In the intra frame, we could refer to the disparities of background pixels surrounding the occlusion region for occlusion filling. In the inter frames, the occluded position would be seen at other time if the camera or the foreground object moves. Thus, we could refer to the disparity at the non-occluded frame for occlusion filling. In the inter-view frames, the occlusion regions might be visible in other two views. Thus, we could refer to the disparity in other view for occlusion filling.

In the HQ-DE algorithm, we adopt the intra-frame approach and apply the modified window vote method to fill occlusion regions. In addition to the color consistency constraint, the modified window vote method gives votes to the non-occluded support pixels only. The window size for the occlusion filling is 11×11 . Figure IV-13 shows that the proposed occlusion handling methods could improve the occluded regions in the disparity map and the synthesized image. Moreover, compared to the previous results of DERS and baseline algorithms in Figure IV-5, the red sketch is clearer, and the HQ-DE algorithm could perform better.

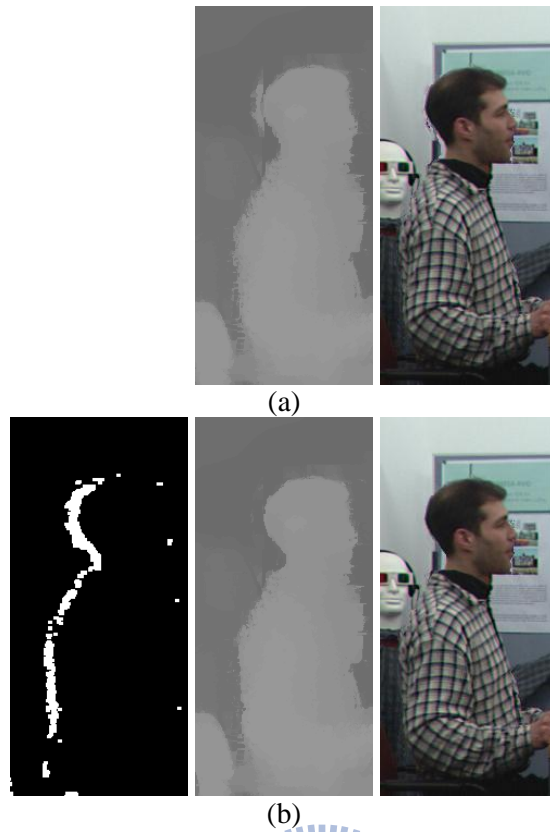


Figure IV-13 Results with and without the proposed occlusion handling method in BookArrival (a) disparity map and synthesized image without occlusion handling, (b) occlusion map, disparity map, and synthesized image with occlusion handling.

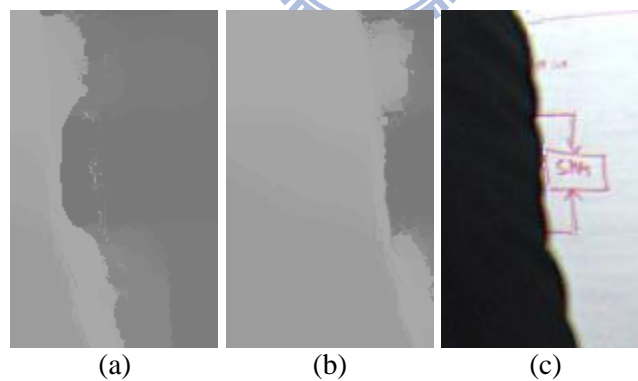


Figure IV-14 Results of the HQ-DE algorithm in BookArrival compared to Figure IV-5 (a) center-view disparity map, (b) right-view disparity map, (c) synthesized image

4.1.7 Temporal Consistency Enhancement

In previous work and the baseline algorithm, the temporal consistency problems include the flicker artifact and foreground copy artifact due to no enhancement method and over enhancement

method, respectively. To address it, we propose the NMR method and the SEP method based on the conventional method in the 3DVC's DERS algorithm.

1. Conventional Method

First, we introduce the conventional temporal consistency enhancement method in the 3DTV's DERS algorithm, and point out its drawback. In the conventional method, the main idea is to propagate previous disparity map to current one for no-motion regions by adding the temporal cost C_{temp} to cost cube. The conventional method first applies the bilateral filter to smooth the previous frame and the current frame, and then partition the frames into 16×16 macroblocks for calculating the motion absolute difference (MAD) by

$$MAD = \frac{1}{16 \times 16} \sum_{(u,v) \in \text{macroblock}} |I_H^t(u, v) - I_H^{t-1}(u, v)| . \quad (\text{IV-16})$$

If MAD is less than a defined threshold γ_{temp} , the block would be regarded as a no-motion block. Thus, the temporal cost can be computed by

$$C_{temp}(x, y, d) = \begin{cases} \lambda_{temp} |d - D_H^{t-1}(x, y)| & \text{if } MAD < \gamma_{temp}, \\ 0 & \text{else} \end{cases} \quad (\text{IV-17})$$

where λ_{temp} is a scaling term. In this equation, the no-motion block will suffer from the penalty if its disparity is inconsistent to previous frame.

The conventional method can solve the flicker artifact, but incurs the foreground copy artifact because the background object does not have enough time to update its disparity. On the hand, the previous disparity upsampling step would result in the flicker artifact even if the conventional temporal consistency enhancement is adopted in the HQ-DE algorithm. That is because the object boundary suffers from mixed color of background and foreground, so that the disparity at the boundary would be unstable after the disparity upsampling. To sum up, the HQ-DE algorithm has the foreground copy artifact and the flicker artifact if the convention method is adopted. The following proposed two methods could solve them.

2. No-Motion Registration Method

Figure IV-15 illustrates the concept of the proposed no-motion registration (NMR) method by a common case. In which, the pixel is changed from a foreground pixel to a background pixel when a foreground object passes. The conventional method in DERS algorithm propagates the previous disparity to current one when the pixel is no-motion pixel, and takes short frame time to update the disparity from foreground to background while the foreground object is moving out. That would result in foreground copy artifact because of insufficient updating time. To address it, the proposed NMR method extends the motion interval by τ_{NMR} frames to provide sufficient updating time. In other word, the no-motion frame count NMC begins to be accumulated while the pixel becomes no-motion one. If NMC is more than τ_{NMR} , the temporal cost C_{temp} would be computed to propagate previous disparity to current frame.

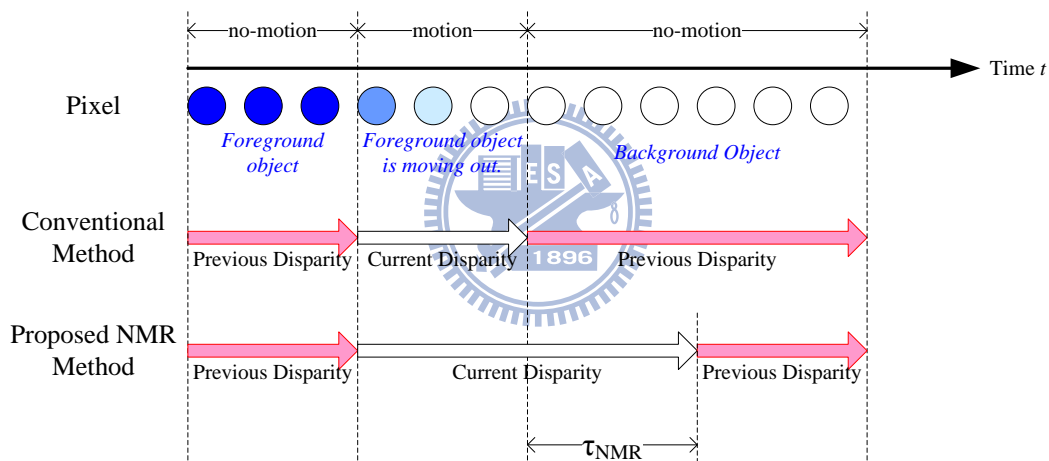


Figure IV-15 Concept of the proposed no-motion registration (NMR) method

Figure IV-16 shows the resultant disparity maps and the synthesized images of the proposed NMR method. Compared to the conventional method in Figure IV-4, the door pivot could be recovered well using the proposed method. In addition, Figure IV-17 shows the change of disparity maps and synthesized images in the successive frames while the man is passing away. In Figure IV-17 (c), the door pivot is temporarily distorted as the same as that in Figure IV-4, because the background disparity is still updating. Nevertheless, the distortion could disappear in Figure IV-17 (d).

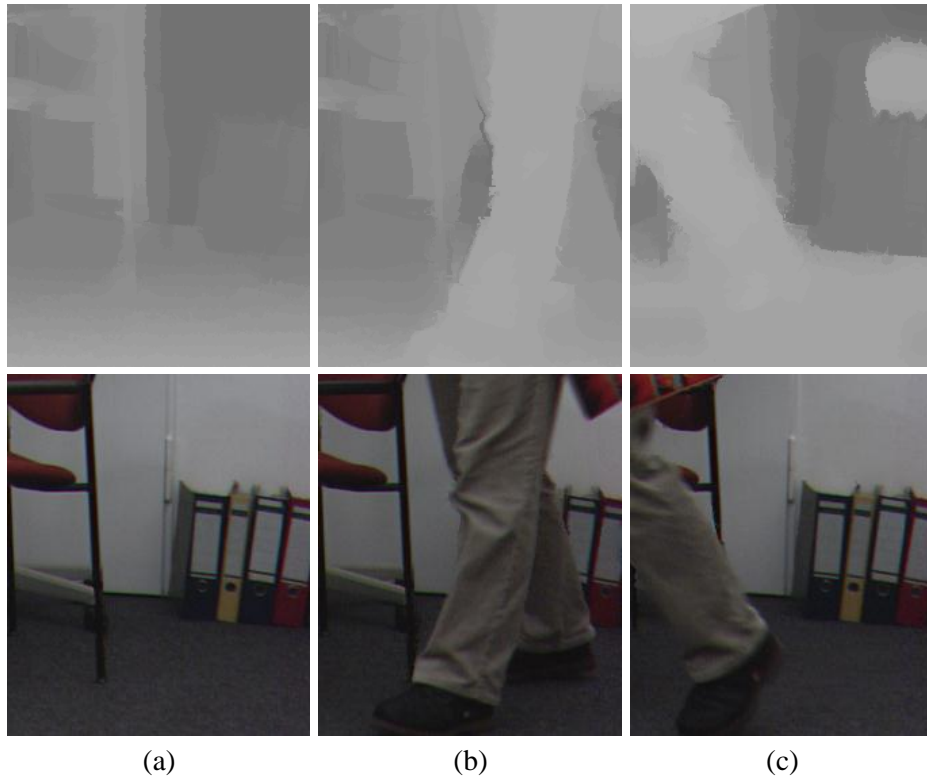


Figure IV-16 Results of the proposed NMR method in BookArrival
 (a) the 1st frame, (b) the 25th frame, and (c) the 40th frame

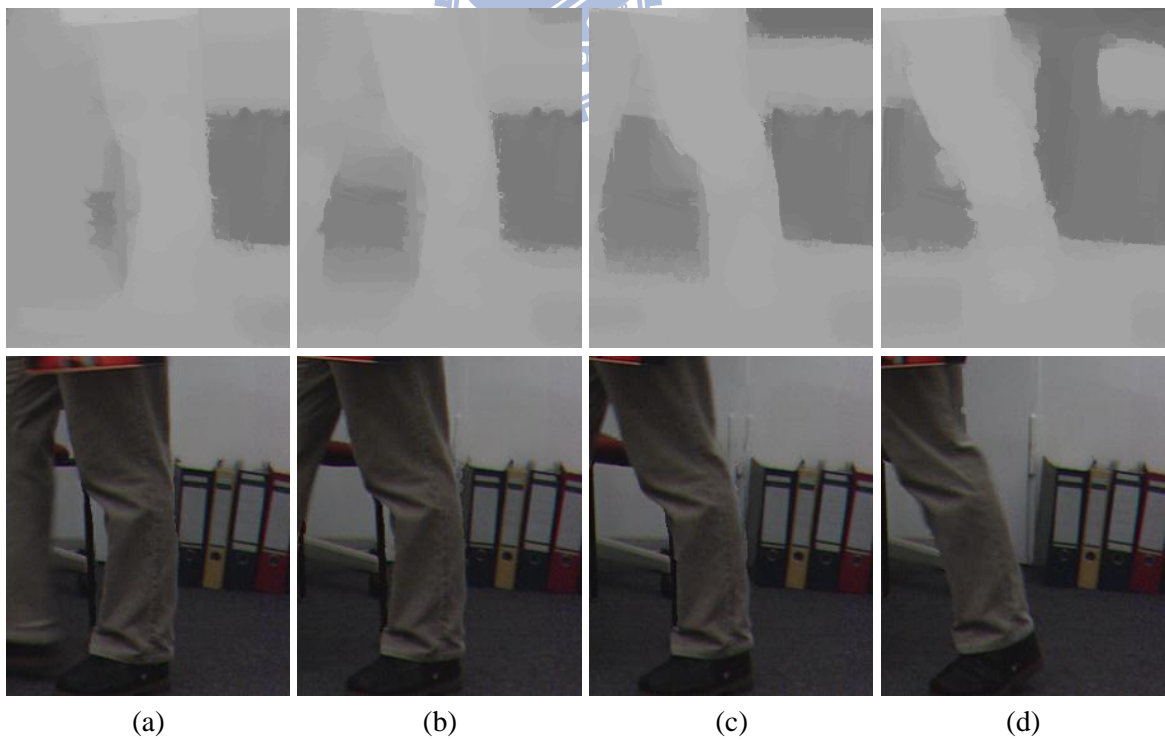


Figure IV-17 Results of the proposed NMR method in the 32th, 34th, 36th, 38th frames

3. Still-Edge Preservation Method

The main idea of the proposed SEP method is to preserve the previous disparity for the still-edge. In the SEP method, we use the bilateral filter to de-noise image, and apply the Sobel filter with a gradient threshold to detect edges. Combining with the above motion and no-motion information, we could find the still edge, which is no-motion pixel and edge pixel. For the still edges, the current disparity is directly propagated from the previous frame.

Figure IV-18 shows the synthesized result using the disparity maps of SEP method. Compared to the results of baseline algorithm in Figure IV-3, the SEP method could address the flicker artifact on the object boundary.

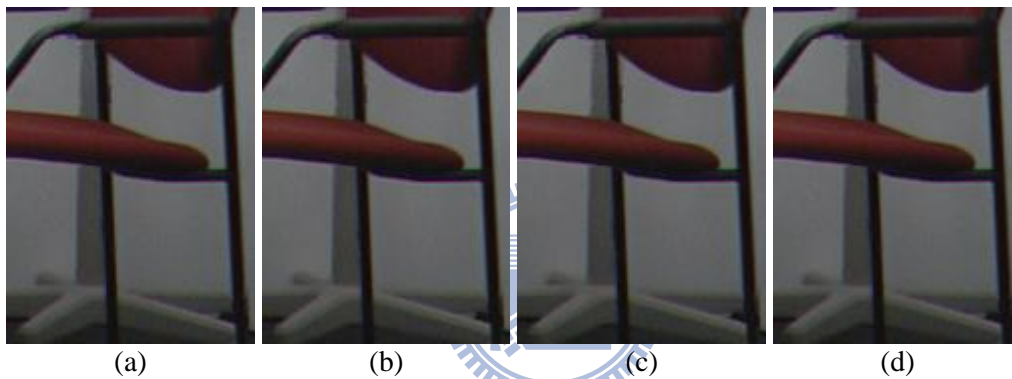


Figure IV-18 Results of the proposed SEP method in BookArrival
(a) the 9th frame, (b) the 10th frame, (c) the 11th frame, (d) the 12th frame

To sum up, the proposed HQ-DE algorithm could address the temporal consistency and occlusion problems to deliver better disparity maps than the previous work. Taking advantage of the disparity upsampling technique and the fast convergent BP-M approach, the HQ-DE algorithm could also save computation time for high definition disparity estimation. Note that the window sizes in the joint bilateral upsampling and window vote methods are selected from several sampled sizes, and they could be finely tuned to attain the higher quality. The associated objective quality evaluation of HQ-DE algorithm will be presented in Chapter V.

4.2 Sparse-Computation Disparity Estimation Algorithm

This section proposes the sparse-computation disparity estimation (SC-DE) algorithm that accelerates the HQ-DE algorithm by the strategy of sparse computation. In this section, we review the related fast BP-based algorithms and summarize their reduction strategies. Then, we present our proposed algorithm in details.

4.2.1 Related Work

In the previous disparity estimation, the hierarchical BP (HBP) [25] is commonly used to accelerate the baseline BP by the coarse-to-fine order in the spatial domain. Based on the HBP, the approximate BP [42] merges the outgoing messages between hierarchical blocks into one to reduce the number of messages. In addition, the constant-space BP [43] additionally performs the hierarchical computation in the disparity domain by the fine-to-coarse order, and can keep the memory usage constant. Unlike the above acceleration approach with regular computation, the sparse BP [44] first applies the adaptive mesh technique to select essential pixels, and then computes the disparities for the sparse pixels by the baseline BP. Finally, a dense disparity map is recovered.

The acceleration strategies in above work are to perform BP optimization for the sparse points in the spatial domain and disparity domain. Their selected sparse points are at the hierarchically regular positions or the selected irregular positions. In the video processing, besides of the spatial and disparity domains, the temporal domain can also be considered into the computational reduction.

4.2.2 Proposed Algorithm Flow

1. Profiling of HQ-DE Algorithm

First, we analyze the profiling of HQ-DE algorithm on PC platform by the Visual Studio 2010 Profiler Tool. Figure IV-19 shows the profiling result in the XGA sequence “BookArrival” and the HD1080p sequence “Hall1”. The distributions of the two sequences are similar. The first part is occupied by the BP-M optimization. The second part is the window vote and the no-motion registration because they apply the median filter and bilateral filter to de-noise image frames.

In the computation of HQ-DE algorithm, the SSAD, ADSW steps are proportional to DR , and the BP-M step is proportional to DR^2 . Nevertheless, the SSAD and ADSW do not suffer from heavy computation, because they are performed in low resolution, and have small window sizes. On the other hand, the window vote and no-motion registration use large window process, and are performed in high resolution. Thus, they suffer from high computational complexity.

As the above mentioned analysis, in the SC-DE algorithm, we mainly focus on the acceleration of BP-M optimization, and further try to introduce the idea of sparse computation into other steps.

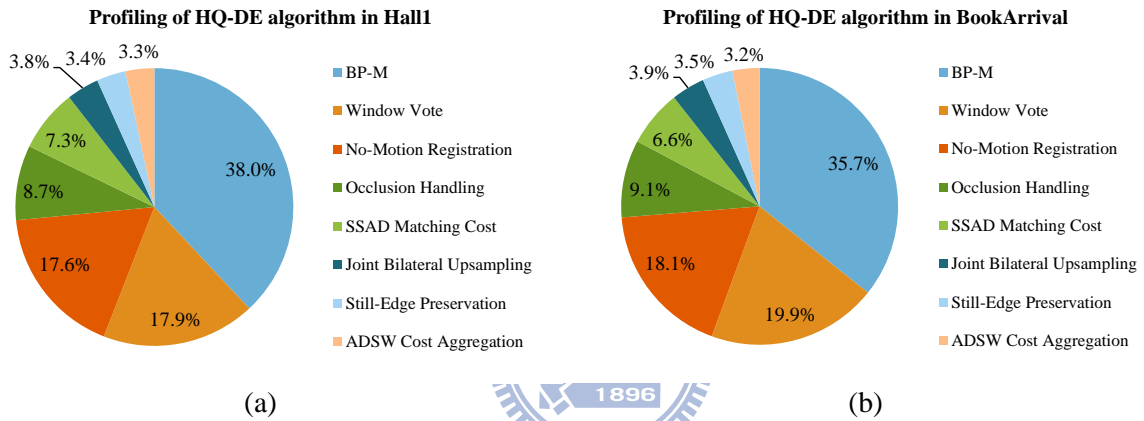


Figure IV-19 Profiling of the HQ-DE algorithm on PC

(a) BookArrival with 100 frames (1024×768), (b) Hall1 with 200 frames (1920×1088)

2. Proposed Sparse-Computation Disparity Estimation Algorithm

To reduce the computational complexity of HQ-DE algorithm, our strategy is to propagate the disparity map and the cost cube of previous frame to current frame, and update partial of them to compute the current disparity map. In the SC-DE algorithm, we perform the same processes in HQ-DE algorithm for the first frame, and store both the computed disparity maps and the cost cubes for the next frame. In the following frames, the SC-DE algorithm updates the cost cubes and applies the new sparse BP-M method for the selected regions to calculate the disparity maps. The selected regions are differently defined for center view and side views. In the video sequences, only the disparities in motion regions are changed, and should be updated. Thus, for the center view, the disparities are

re-computed in the motion regions. On the other hand, for the side views, most of disparities could be warped from the center-view disparity map, and only the occlusion regions have different disparities. Thus, only the disparities in the occlusion regions have to be recomputed for the side view.

With the above strategy, we proposed the flow of SC-DE algorithm in Figure IV-20 for center view and Figure IV-21 for side views. In the SC-DE algorithm, the center-view disparity map should be computed first, and then refers it to the other side-view disparity maps. In the proposed algorithm flow, the motion map M_H , edge map E_H , and occlusion map O_H are initially computed to determine the sparse pixels. The details of their computation are described in the next sub-section.

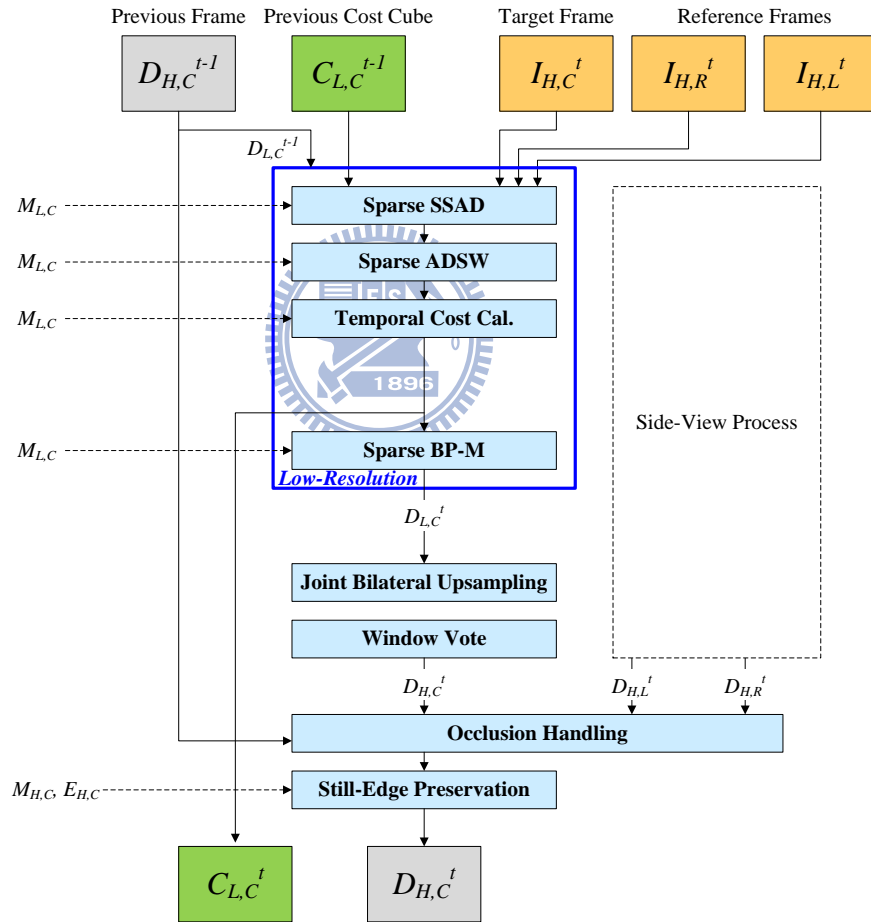


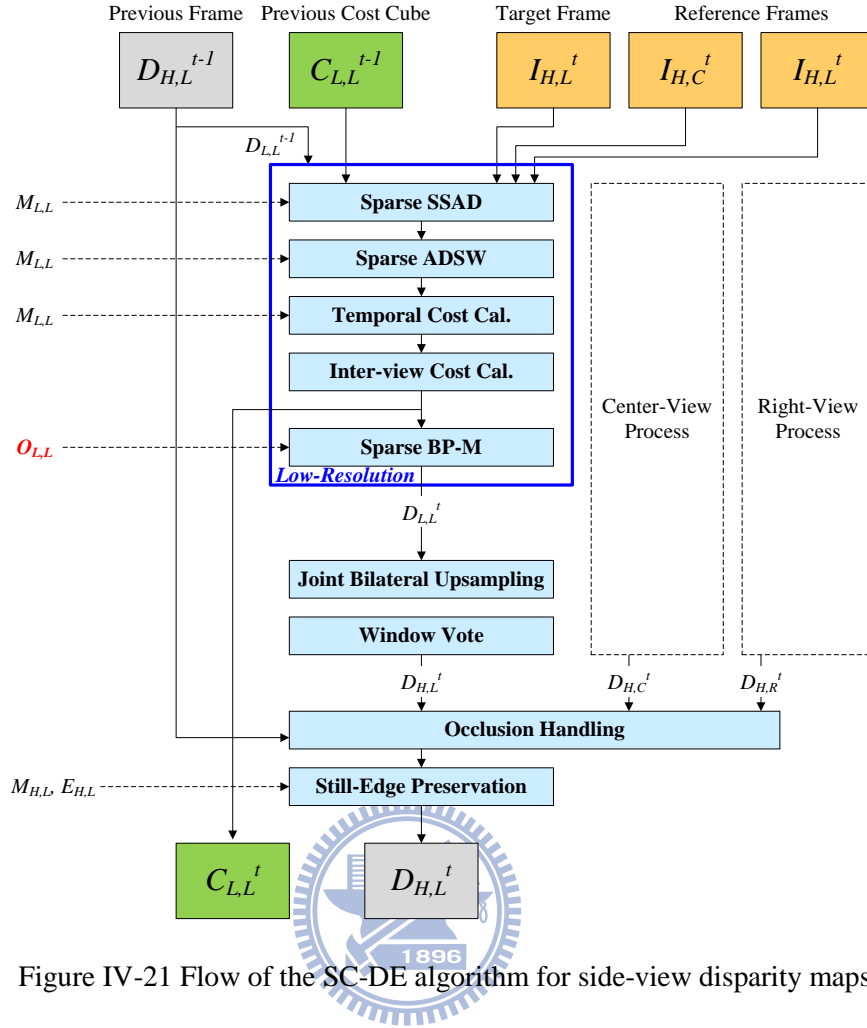
Figure IV-20 Flow of the SC-DE algorithm for center-view disparity map

In Figure IV-20 for the center-view disparity estimation, the cost cube $C_{L,C}^{t-1}$ and the disparity map $D_{H,C}^{t-1}$ in previous frame are updated by the sparse-computation steps: sparse SSAD, the sparse

ADSW, temporal cost calculation, sparse BP-M. The sparse-computation steps are guided by the sampled motion map M_{LC} , and they only process on the motion regions. With these four steps, the new cost cube C_{LC}^{t-1} and the new low-resolution disparity map D_{LC}^t are produced.

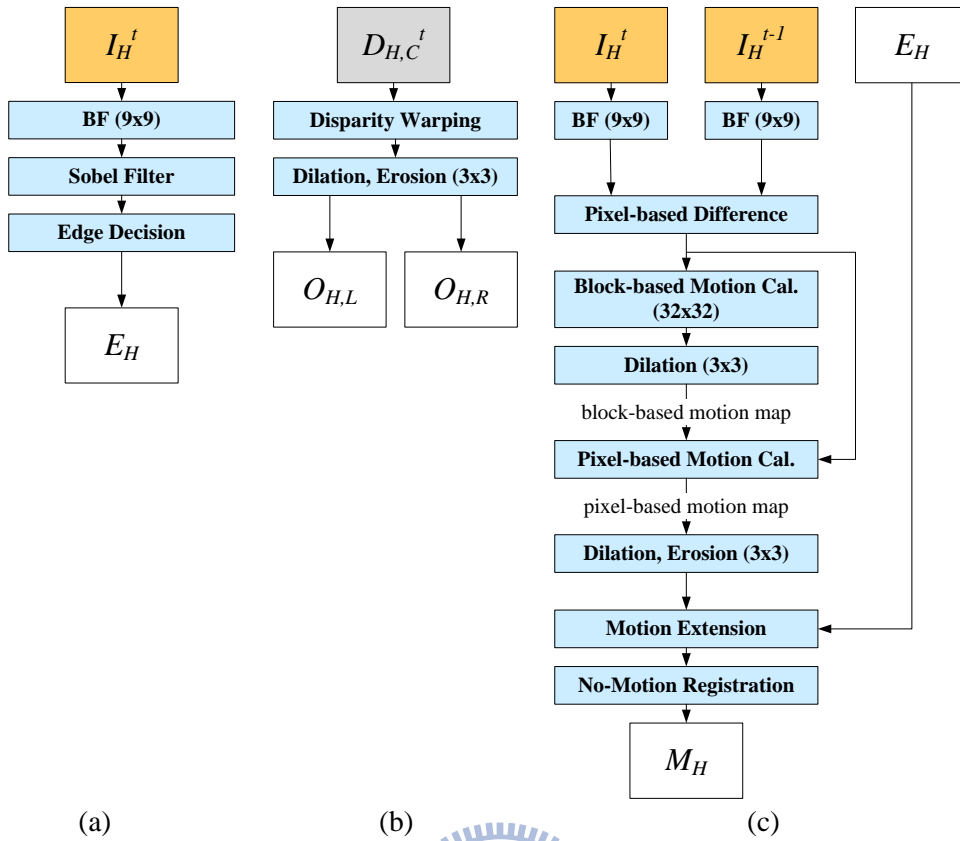
With the low-resolution disparity map D_{LC}^t , the JUB algorithm and the window vote methods are adopted to scale up and refine the high-resolution disparity map D_{HC}^t . The sparse-computation approach could not be applied to the JUB and window vote steps because the new updated disparities in motion regions are not consistent with those in the no-motion regions. The consistency would be expended in following frames, and result in serious quality drop. Thus, the JBU and window vote steps are still performed by dense-computation approach. Finally, the occlusion handling and the still-edge preservation (SEP) steps are performed to deal with the occlusion and temporal consistency problems.

In Figure IV-21 for the side-view disparity estimation, the sparse-computation approach is also applied to the SSAD, ADSW, temporal cost calculation, and BP-M steps. Although the side-view disparity estimation needs to update the disparities only in occlusion regions, the cost cube still requires to be updated for the motion regions. Thus, the former three steps, sparse SSAD, sparse ADSW, and the temporal cost calculation, are guided by the motion map M_{LL} , while the later sparse BP-M is guided by the occlusion map O_{LL} . The rest steps are the same as the center-view disparity estimation.



4.2.3 Sparse Pixel Selection

The sparse pixel selection is to determine the sparse pixels which should be processed by the sparse-computation steps as mentioned above. To find the sparse pixels, the edge detection, the occlusion detection, and the motion detection are required. Their algorithm flow is shown in Figure IV-22.



(a) (b) (c)
 Figure IV-22 Flow of region detection for sparse pixel selection
 (a) edge detection, (b) occlusion detection, (c) motion detection

1. Edge Detection

Figure IV-22 (a) shows the flow of edge detection. First, the bilateral filter with the window size of 9×9 is applied to de-noise the input frame. Then the Sobel filter is used to compute the gradients for the horizontal and vertical directions. Finally, the edge decision step determines the edge pixel if the gradient magnitude is higher than a threshold. The produced edge map E_H is used in the motion detection and the still-edge preservation steps in the SC-DE algorithm. Figure IV-23 shows an example of edge maps in the sequence BookArrival.

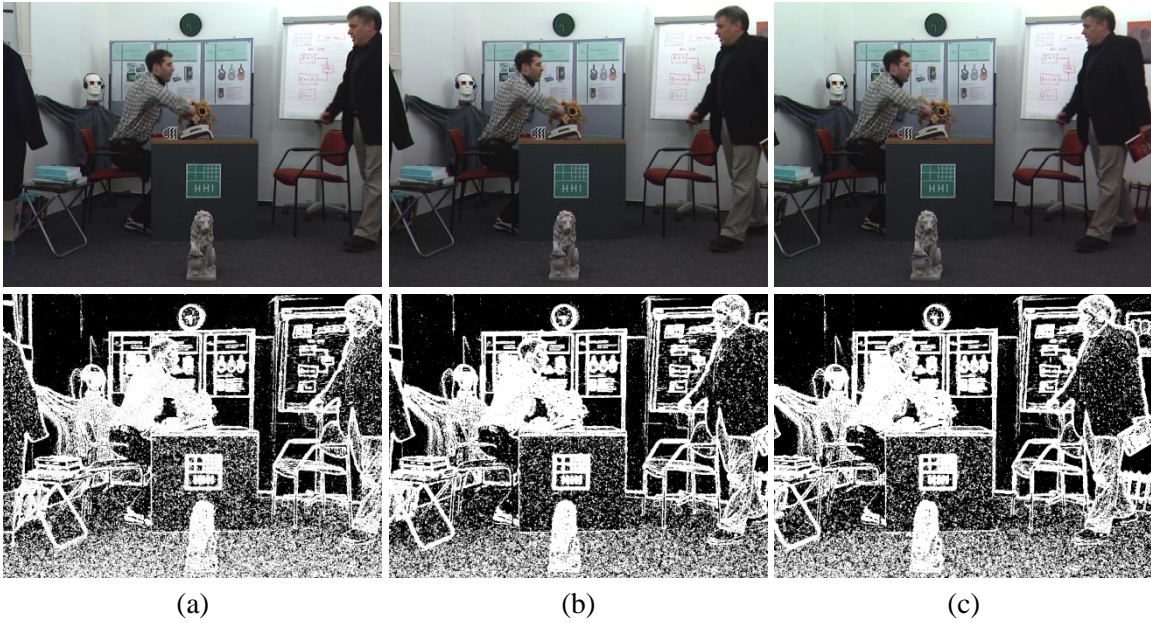


Figure IV-23 Example of edge maps in BookArrival

(a) left-view frame $I_{H,L}$, $D_{H,L}$, (b) center-view frame $I_{H,C}$, $D_{H,C}$, (c) right-view frame $I_{H,R}$, $D_{H,R}$

2. Occlusion Detection

The occlusion region is detected using the center-view disparity map. In Figure IV-22 (b), the center-view disparity map is warped to left view and right view. In the warped disparity maps, the position without any disparity value is regarded as an occlusion pixel. Then, the occlusion map is further processed by the dilation and erosion filter to remove the small occlusion regions, which are considered as noise. Figure IV-24 shows an example of occlusion maps $O_{H,L}$, $O_{H,R}$ generated using the center-view disparity map $D_{H,C}^t$.

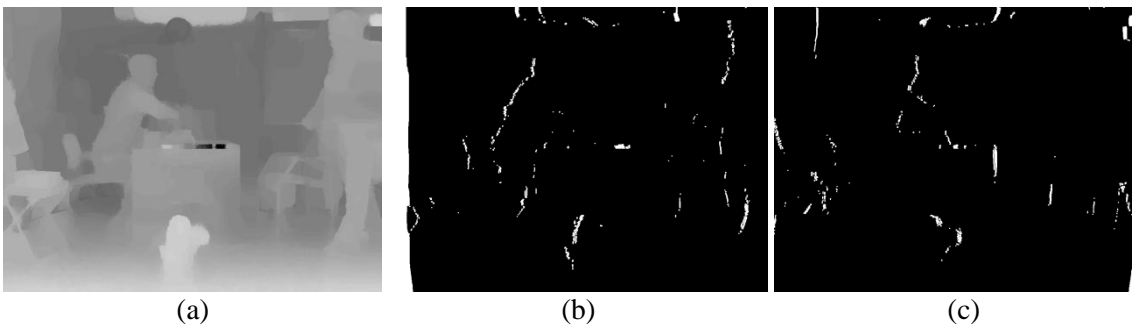


Figure IV-24 Example of occlusion maps in BookArrival

(a) center-view disparity map $D_{H,C}$, (b) left-view occlusion map $O_{H,L}$, (c) right-view disparity map $O_{H,R}$

3. Motion Detection

The motion information in the SC-DE algorithm is used to not only the temporal consistency enhancement, but also the sparse-computation guidance. The motion map should be more precise in the SC-DE algorithm, because the quality of SC-DE algorithm depends on the selected sparse regions. Thus, we modify the original motion detection method in HQ-DE algorithm to the new one as described in Figure IV-22 (c). The new motion detection method is to first compute the block-based motion map, and then refine it to pixel-based motion map.

In the motion detection method, the bilateral filter is first applied to de-noise the previous and the current image frames, and the difference of the two frames are computed by the Manhattan distance for each pixel. Then, the block-based motion map is calculated according the sum of frame difference in a 32×32 block. If the sum of frame difference is high than a threshold, this block would be regarded as a motion block. Note that the block-based motion map should be dilated by a 3×3 filter, because the no-motion block neighboring motion one maybe contains a few motion pixels.

Finally, the block-based motion map is refined to be a pixel-based motion map. For the pixels in motion blocks, the pixel would be considered as a motion pixel if its frame difference is high than a threshold. Since there are some noising motion pixels, the dilation and erosion filter is adopted to remove them.

Furthermore, the pixel-based motion map is processed by the motion extension step, which extends the motion regions along the edge pixels. Finally, the no-motion registration (NMR) is performed to handle the mentioned foreground copy artifact.

Figure IV-25 shows the example results of the motion map in the sequence BookArrival. Some motion regions are over extended by the motion extension step, and it will result in more computational complexity in the sparse-computation steps. Nevertheless, the over-extended motion map could avoid missing the motion pixel, whose disparity is necessary to be updated.

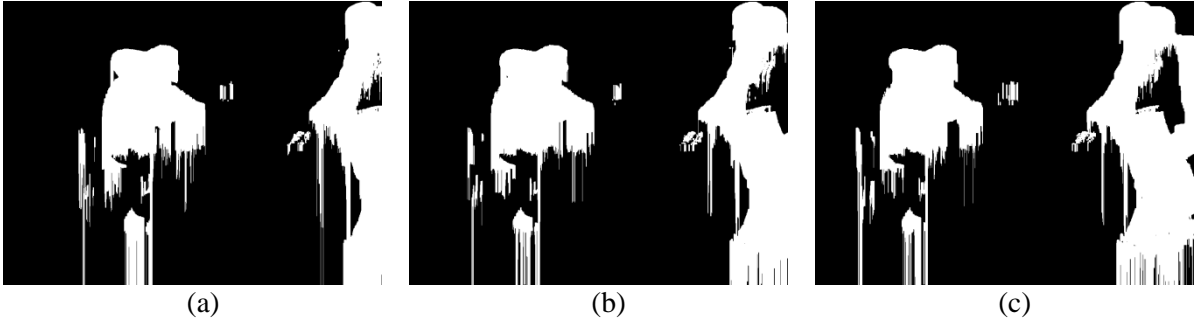


Figure IV-25 Example of motion maps in BookArrival

(a) left-view motion map $M_{H,L}$, (b) center-view motion map $M_{H,C}$, (c) right-view motion map $M_{H,R}$

4.2.4 Sparse-Computation Steps

The sparse-computation steps include the sparse SSAD, the sparse ADSW, and the sparse BP-M. Their detailed flow is described as follows.

1. Sparse SSAD and ADSW for Cost Cube

The sparse SSAD and sparse ADSW steps are to update the previous cost cube C_L^{t-1} and generate the new cost cube C_L^t for the current frame. Figure IV-26 illustrates the concept of the two sparse-computation steps. The updated target pixels are the motion pixels. To compute the new costs for these motion pixels, the matching costs of all the associated support pixels should be calculated by the SSAD match metric. Using the matching costs of support pixels, the sparse ADSW then aggregates them for the motion pixels. The sparse SSAD and the sparse ADSW could further reduce the computational complexity of the original dense method in the HQ-DE algorithm.

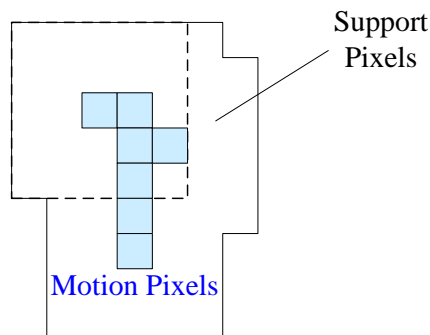


Figure IV-26 Concept of sparse SSAD and sparse ADSW methods

2. Sparse BP-M Optimization

For the sparse BP-M optimization, the guide for sparse-computation is the motion map $M_{L,C}$ for center view, and the occlusion maps $O_{L,L}$, $O_{L,R}$ for the side views. Figure IV-27 illustrates the concept of the sparse BP-M method. In which, the “updated region” is processed by the original BP-M algorithm. To connect the “updated region” and “other regions”, the sparse BP-M method also passes the message from the “other regions”. This connection could decrease the disparity incoherency between the updated and other regions. The sparse BP-M method could significantly reduce the original dense BP-M because the motion regions in center view and the occlusion regions in side views are very small.

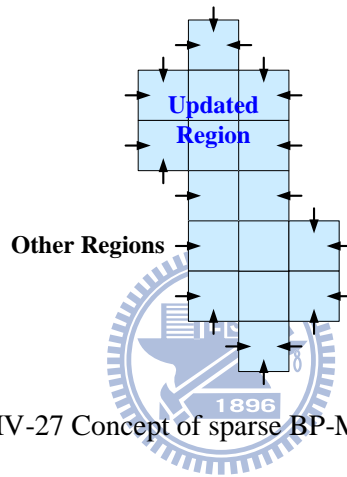


Figure IV-27 Concept of sparse BP-M method

4.2.5 Computational Reduction

This subsection compares the computational distribution of main steps between the HQ-DE algorithm and the SC-DE algorithm. For the execution time and disparity quality, the associated analysis is presented in Chapter V. Figure IV-28 shows the profiling of SC-DE algorithm in the sequences BookArrival and Hall1. In which, the motion detection is added into the execution time of no-motion registration step, and the edge detection is added into the still-edge detection step.

The percentage of BP-M is significantly decreased from 35.7% to 8.0% in BookArrival and from 38.0% to 9.0% in Hall1. Table IV-2 further lists the execution time of each step in the HQ-DE and SC-DE algorithms. The computation of all the steps with sparse-computation approach is significantly reduced. Compared to the HQ-DE algorithm, the major computation of BP-M is decreased to 13.4% in

the SC-DE algorithm. In addition, the total execution time could be reduced to less than 60%. Note that the execution time of still-edge preservation increases near twice because it replaces the original median filter with the bilateral filter for better de-noising but incurs heavy computation.

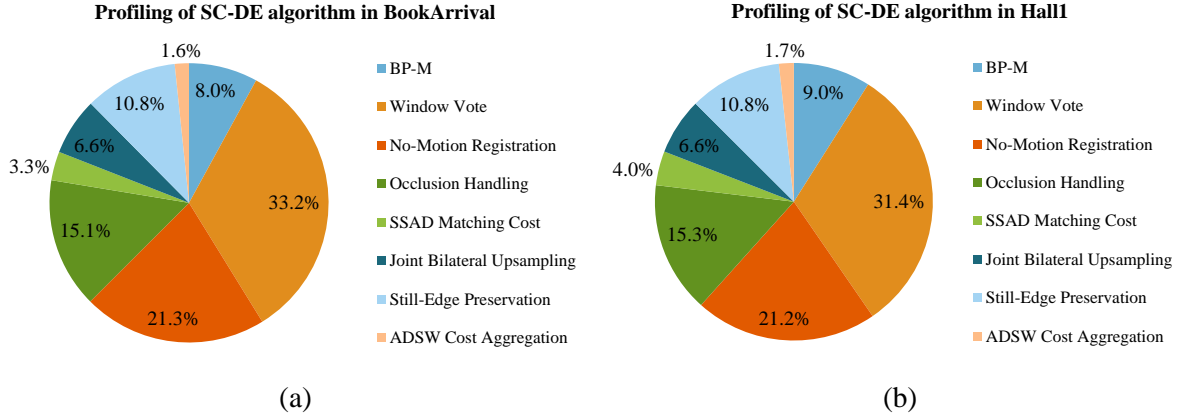


Figure IV-28 Profiling of the SC-DE algorithm on PC

(a) BookArrival with 100 frames (1024×768), (b) Hall1 with 200 frames (1920×1088)

Table IV-2 Comparison of execution time of HQ-DE and SC-DE algorithms

	HQ-DE		SC-DE	
	BookArrival	Hall1	BookArrival	Hall1
BP-M	358,361	2,127,407	47,833	285,861
Window Vote	199,819	1,000,995	199,042	1,001,075
No-Motion Registration	181,615	982,290	127,767	676,116
Occlusion Handling	90,877	487,167	90,346	486,272
SSAD Matching Cost	66,365	408,361	20,036	126,827
Joint Bilateral Upsampling	39,175	210,521	39,756	210,177
Still-Edge Preservation	35,558	189,802	64,614	343,325
ADSW Cost Aggregation	31,740	187,311	9,845	55,041
Total	1,003,510	5,593,854	599,239	3,184,694

Unit: Sampled time on PC

4.3 Hardware-Efficient Disparity Estimation Algorithm

The proposed SC-DE algorithm could significantly reduce the computation complexity of HQ-DE algorithm, but is not suitable to be further accelerated by VLSI design due to its regular computation and large storage for the information of previous frame. In this section, we proposed the hardware-efficient disparity estimation (HE-DE) algorithm that could significantly reduce the computation and memory cost of HQ-DE algorithm. In this section, we first point out the design

challenges in the HQ-DE algorithm. Then, we present the main algorithm flow of our proposed HE-DE algorithm.

4.3.1 Design Challenges in High-Quality Algorithm

In the HQ-DE algorithm, the main design challenge consists of the high memory cost and the high computational complexity. They are explained as follows.

1. High Memory Cost in Belief Propagation

The problem of high memory is the fatal disadvantage of BP-based algorithm. The requirement in BP-based algorithm includes the cost cube and the messages. Our low memory-cost approach in Section 3.2.2 could significantly reduce the memory cost, but the memory cost is still proportional to the disparity range DR , even if the block-based [36] or tile-based [29] method is adopted. For example, if the block size is 32×32 , DR is 128, and each data is 1-byte, the memory requirement would be 131Kbytes for the cost cube and 524Kbytes for the messages. The extremely high memory space could not be affordable in the internal memory. If the massive data are configured in the external memory, it would incur high bandwidth. Thus, to directly conquer the high memory cost problem, we need to develop another new optimization algorithm that could not only have memory requirement independent to disparity range, but also acquire approximate results to BP-M's.

2. Large Image Buffers

Figure IV-29 Image buffer required by the SSAD and ADSW steps (a) shows that the required pixels for computing a target aggregated cost. For the target aggregated cost, the ADSW cost aggregation step aggregates the 5×7 matching costs in low resolution. These 5×7 matching costs is computed by the SSAD matching cost step using the 10×28 pixels in high resolution. Therefore, computing a target cost needs 1280 pixels in the target view image, and these pixels are cross 28 image rows.

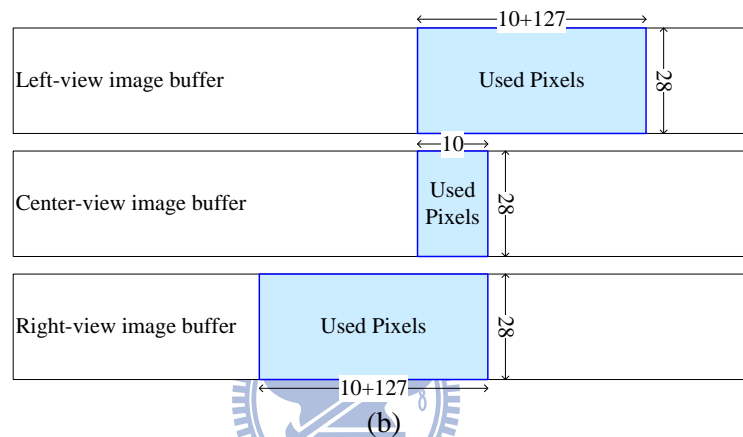
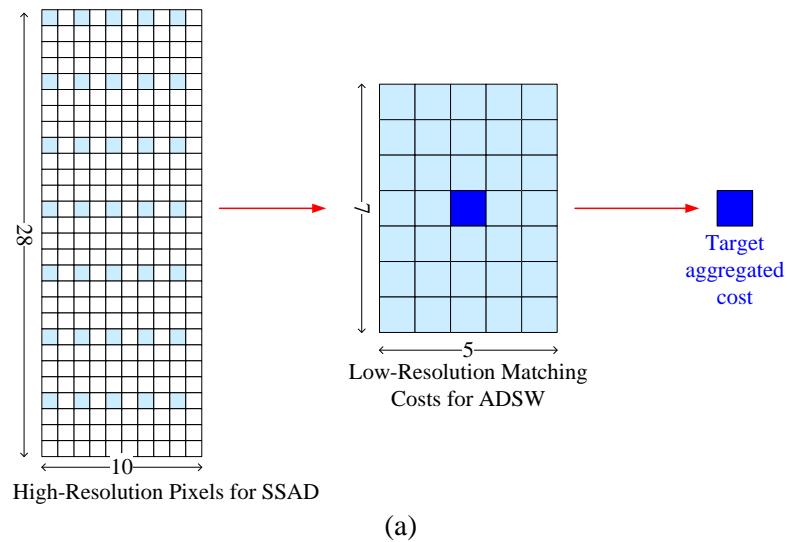


Figure IV-29 Image buffer required by the SSAD and ADSW steps

(a) required pixels for computing a target aggregated cost, (b) image buffers for one matching cost row

For the above data dependency, all the 28 rows of three view images should be buffered into the internal memory, so that the external bandwidth be minimized. However, such the multiple-row image buffers are too large. For example of 1920×1080 sequences, the memory requirement for the image buffers would be $1920 \times 28 \times 3$ pixels (i.e. 483Kbytes for YUV444 format). On the other hand, if the SSAD matching costs are stored for data reuse technique, the memory requirement is proportional to disparity range DR , and would be $960 \times 7 \times 128$ (i.e. 860Kbytes) for the DR of 128. In addition, if the image pixels are accessed from external memory in run time, the image buffer could be reduced to the “used pixels” region but the external bandwidth would be $1920 \times 1080 \times 7 \times 3$ pixels/frame (i.e. 130.6Gbytes/frame for YUV444 format).

To sum up, no matter what the data configuration method is applied, the required image data in the SSAD and ADSW steps would incur the large image buffer or high external bandwidth. Thus, we should simplify the SSAD and ADSW steps in the HE-DE algorithm to reduce the image buffer.

3. High Computational Complexity in Filtering

In the HQ-DE algorithm, there are many filter-based processes, such as bilateral filter, joint bilateral upsampling, window vote, and the ADSW cost aggregation. These filter-based processes suffer from high computational complexity due to their larger window size. Table IV-3 lists all the filter-based processes in the HQ-DE algorithm. In which, the bilateral filter (BF) computation suffers from 11×11 for the high resolution in the NMR step. In addition, the WVote step requires the largest window size of 15×15 . Because of their large window sizes, they occupy the high percentage of computation as analyzed in Figure IV-19. Thus, we decrease the window size of the filter-based processes in the HE-DE algorithm under the condition of preserving the disparity quality.

Table IV-3 Window sizes of filter-based processes in HQ-DE algorithm

Step	Computation	Frame Resolution	Window Size
No-Motion Registration (NMR)	BF	High	11×11
Adaptive Support-Weight Cost Aggregation (ADSW)	BF	Low	7×5
Occlusion Handling (OCC)	Vote	High	9×9
Joint Bilateral Upsampling (JBU)	JBF	High	7×7
Window Vote (WVote)	Vote	High	15×15
Still-Edge Preservation (SEP)	Median	High	3×3

4. Irregular Computation in Occlusion Handling

The final design challenge is the irregular computation in the occlusion handling step. This step first detects the occlusion region by left-right check (LRC) method, and then extends the occlusion region for background and foreground. Finally it fills the occlusion regions by the modified window vote method. The irregular computation is in the occlusion extension process, which needs to extend the occlusion region until the foreground is touched. This irregular computation is not compatible to all the other raster-scan computations, and is not suitable to be implemented by high-throughput

pipelining architecture. Thus, we develop another new regular occlusion handling in the HE-DE algorithm.

In summary, for the high memory cost, the BP-M needs a frame-scale-magnitude memory space to store the cost cube and messages for whole frame, and cost cube calculation requires a large image buffer in run-time. On the other hand, for the high computational complexity, the filter-based computation is performed using too large window size, and the computation of proposed occlusion handling is not regular for extending occlusion region. Therefore, the proposed HE-DE algorithm focuses on these design challenges and conquers them.

4.3.2 Proposed Algorithm Flow

Figure IV-30 shows the main flow of the proposed HE-DE algorithm for center view. This algorithm flow also could be applied to the process of side views. In this algorithm, for the cost cube calculation, we propose the new window-based SSAD method to replace the block-based SSAD and ADSW steps in the HQ-DE algorithm. The new method could reduce the image buffers from 28 image rows to 5 image rows. For the temporal cost calculation, the same method in the HQ-DE algorithm is adopted.

Note that this algorithm removes the inter-view cost calculation step in the HQ-DE algorithm for high parallelism, because the step would result in the data dependency between the center view and side views. In other words, with the inter-view cost calculation step, the center-view disparity map should be computed first, and the side-view disparity maps are computed latter. Moreover, to support the computing order, the three-view input data would be loaded for three times for matching cost calculation. Therefore, we remove the inter-view cost calculation from our algorithm flow, and take care of the inter-view consistency in the occlusion handling step.

With the computed cost cube, we propose the cost diffusion method to compute the low-resolution disparity maps. The proposed cost diffusion method could replace the BP-M to reduce the memory requirement to be independent to the disparity range.

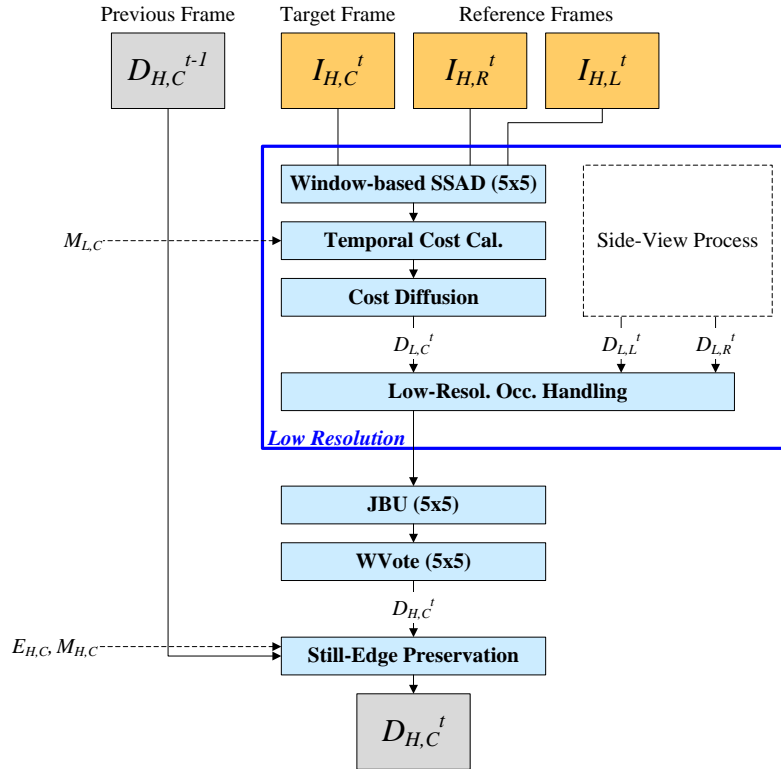


Figure IV-30 Flow of the HE-DE algorithm for center view

For the occlusion handling step, the new regular method is performed in the low resolution, and it also considers the inter-view consistency at the same time. Finally, the low-resolution disparity maps are scaled and refined by the JBU, window vote, and still-edge preservation steps. To reduce the computational complexity in filter-based processes, we decrease all the window size of filters to 5×5 under the condition of no observable quality degradation.

The mentioned design challenges in the HQ-DE algorithm are solved by the following method in the proposed HE-DE algorithm.

4.3.3 Cost Diffusion Algorithm

In this subsection, we first discuss about the memory requirement of BP-M, and then propose the low memory-cost cost diffusion method to replace the BP-M.

1. Memory Requirement in BP-M

The original BP-M updates the messages in four directions as illustrated in Figure IV-31 (a), where the message passing is performed direction by direction. Figure IV-31 (b) shows the data dependency of message passing in the node level. In which, the new message is computed for the “updated message” using the “used messages”. First, the right message passing updates the left incoming message of each node in the order of left-to-right and row-by-row. Then, the left message passing is performed in inverse direction to update the right incoming message of each node. With the same manner, the down message passing and the up message passing is performed column by column. Note that the “used messages” in the right message passing and left message passing could be removed, because their values are initially 0 and the horizontal message passing are performed for one time in the single iterative BP-M.

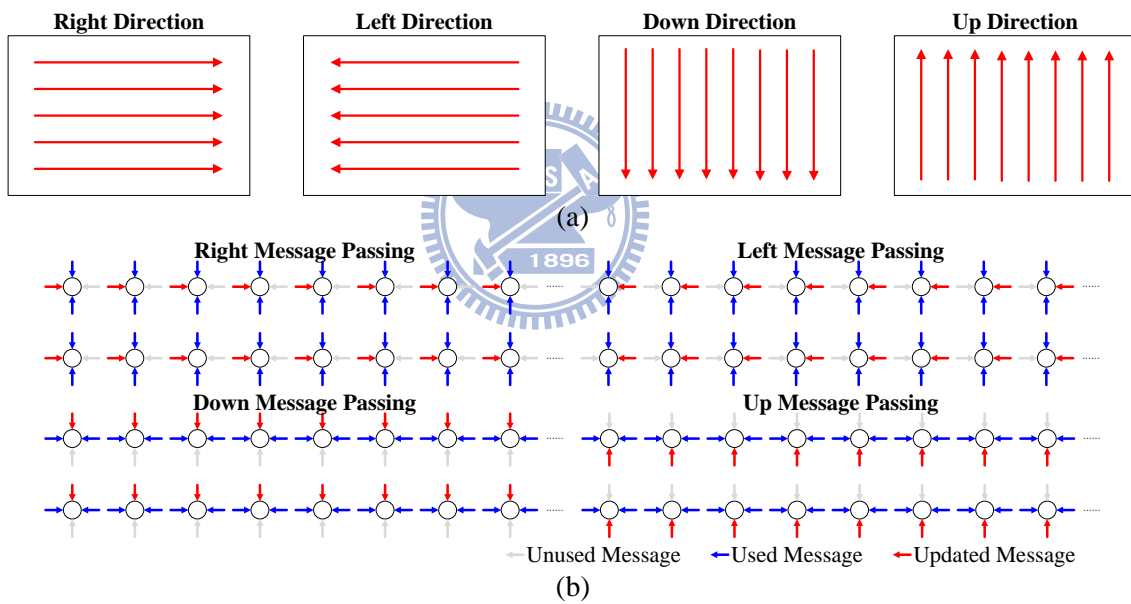


Figure IV-31 Concept of BP-M computation

(a) Message passing in four directions, (b) data dependency of messages in four directions

With the data dependency of BP-M, all the messages of whole frame have to be stored in memory until the up message passing and the final disparity selection is performed. Thus, the memory requirement is $4H \times W \times DR$ for messages in the HQ-DE algorithm as listed in Table IV-4.

To reduce the memory cost in BP-M, we first propose the horizontal-only BP-M that only performs the left message passing and the right message passing steps. The horizontal-only BP-M

could reduce the memory cost from frame-scale-magnitude to row-scale-magnitude as shown in Table IV-4. However, its memory cost still has the factor of disparity range DR . Thus, we further propose the cost diffusion method to completely address the high memory cost problem.

Table IV-4 Comparison of memory requirement between BP-M and cost diffusion methods

Method	Memory Requirement	Operation Times of Message Passing
Single iterative BP-M (HQ-DE)	$4H \times W \times DR$ (Message) $H \times W \times DR$ (Matching Cost)	$4H \times W$
Horizontal-Only BP-M	$W \times DR$ (Message) $W \times DR$ (Matching Cost)	$2H \times W$
Cost Diffusion (HE-DE)	W (Matching Cost) W (Disparity Map)	$2H \times W$

2. Proposed Cost Diffusion Method

The main idea of the proposed cost diffusion method is to diffuse the matching cost of current pixel to its neighbor by the message passing mechanism, and immediately determine the best disparity for the current pixel. The cost diffusion method includes the strong horizontal diffusion and the weak vertical diffusion. That is because the human eyes weakly percept the vertical disparity and are sensitive to the horizontal disparity [108], [109]. It implies the demand of vertical disparity is lower than that of horizontal disparity. Therefore, the horizontal diffusion applies a complicated mechanism and the vertical diffusion applies a simple one.

The horizontal diffusion method consists of the right cost diffusion process and the left cost diffusion process. The two processes can generate two disparity rows, which will be merged into one by our specific constraint. In the two processes, the disparities are computed immediately and the diffused costs could be thrown at the same time. In the proposed cost diffusion method, only the best disparity row and the corresponding minimal matching costs need to be stored. Its memory requirement is listed in Table IV-4.

The details of the proposed cost diffusion method are described using the right cost diffusion process as an example. In the right cost diffusion process, the final cost RC_{final} is computed from left to

right, and it is combined with the original cost C_{total} to determine the disparity row. The final cost RC_{final} is computed by

$$RC_{final}(x, y, d) = C_{total}(x, y, d) + RDC(x - 1, y, d) \quad . \quad (IV-18)$$

where the diffused cost $RDC(x-1, y, d)$ at the left neighbor is defined as

$$RDC(x - 1, y, d) = \min_{d_s} \left(V(d, d_s) + RC_{final}(x - 1, y, d_s) \right) - \kappa \quad , \quad (IV-19)$$

where V is the smoothness term in (IV-7), and κ is the average of RDC for normalization. This equation is similar to the calculation of message passing in (III-7) but all the messages are removed.

Then, the previous diffused cost RDC is combined with the matching cost of current pixel by for the current pixel (x, y) . With the final cost RC_{final} , the temporary best disparity RD_{best} could be immediately calculated by the winner-take-all (WTA) manner, and the minimal cost RC_{min} is also computed for the final disparity decision. They are calculated by

$$RC_{min}(x, y) = \min_d RC_{final}(x, y, d) \quad (IV-20)$$

$$RD_{best}(x, y) = \arg \min_d RC_{final}(x, y, d) \quad . \quad (IV-21)$$

By the above right cost diffusion process, we could obtain the temporary best disparity RD_{best} and the minimal cost RC_{min} of one frame row. We also can acquire the LD_{best} and LC_{min} by the left cost diffusion process. Finally, the two disparity row RD_{best} and LD_{best} are merged into one by the equation

$$D_L(x, y) = \begin{cases} RD_{best}(x, y) & \text{if } RC_{min}(x, y) < LC_{min}(x, y) \\ LD_{best}(x, y) & \text{else} \end{cases} \quad . \quad (IV-22)$$

according to the minimal costs RC_{min} and LC_{min} . In which, we take the disparity with minimal cost as the final disparity.

On the other hand, the concept of vertical cost diffusion is to propagate the disparities of previous row into the current row. Thus, we define another new vertical cost C_{vert} as

$$C_{vert}(x, y, d) = \lambda_{vert} |d - D_L^t(x, y - 1)| \quad (IV-23)$$

where $D_L^t(x, y-1)$ is the disparity in previous row, λ_{vert} is a scaling term. Note that this cost is constrained by the color consistency between the current pixel $I_L^t(x, y)$ and the previous row pixel $I_L^t(x,$

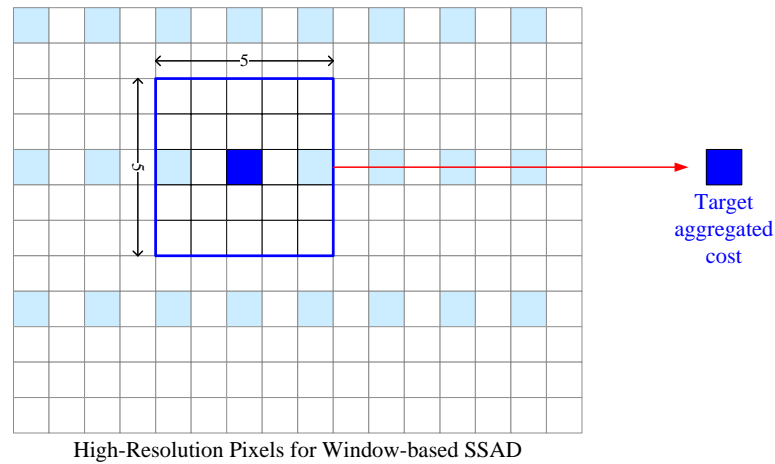
$y-1$). If the two pixels are inconsistent, the vertical cost C_{vert} would be 0. Thus, the total cost cube in the HE-DE algorithm is defined as

$$C_{total}(x, y, d) = C_{aggr}(x, y, d) + C_{temp}(x, y, d) + C_{vert}(x, y, d) . \quad (IV-24)$$

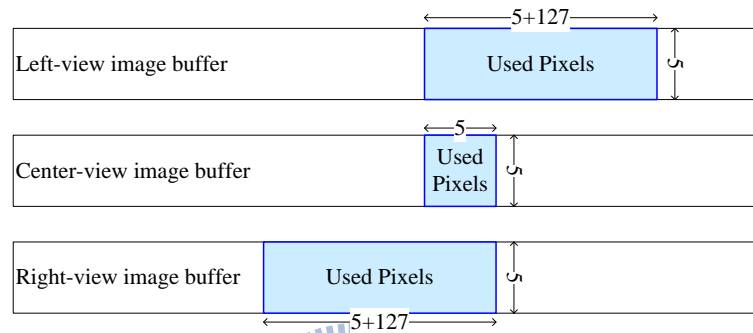
To minimize the memory requirement of cost diffusion method, we could perform the right cost diffusion process first and the left cost diffusion process latter. Thus, only one disparity row and one cost row required to be stored. Compared to the BP-M, the proposed cost diffusion method could reduce the memory cost to 0.00029% for the HD1080p resolution and 128 disparity levels.

4.3.4 Image Buffer Reduction Methods

As mentioned in Section 4.3.1, the SSAD matching cost calculation and the ADSW cost aggregation steps requires 28 image rows to minimize the external bandwidth. However, the memory cost of such the image buffer is too high. Thus, in the HE-DE algorithm, we modify the cost cube calculation method, and propose the window-based SSAD, which can reduce the requirement of image buffer to 5 image rows. Figure IV-32 illustrates the concept of the proposed window-based SSAD. In which, the pixels in a 5×5 window are fetched to compute a target aggregated cost by the 5×5 SAD metric. Without the ADSW cost aggregation step, the disparity would have slight degradation, compared to the HQ-DE algorithm. The comparison results are demonstrated in Chapter V. To compute the matching costs for full disparity range, the “used pixels” are needed for the center-view disparity estimation as shown in Figure IV-32 (b). We could use the image buffers with the size of five image rows to increase the data reuse and minimize the external bandwidth usage. The memory requirement of this configuration would be $1920 \times 5 \times 3$ pixels (i.e. 86Kbytes for YUV444 format). Compared to the original method in HQ-DE algorithm, the memory requirement is saved by 82%.



(a)



(b)

Figure IV-32 Concept of the proposed window-based SSAD method

(a) required pixels for computing a target aggregated cost, (b) image buffers for one matching cost row

4.3.5 Small Filter Window Size

The filter-based processes suffer from high computational complexity due to its large window size as listed in Table IV-3. To reduce their computation, we decrease the window size of filter-based processes while keeping the disparity quality without significant drop. Table IV-5 lists the window sizes of filter-based processes in the HE-DE algorithm. In which, most of the processes are changed to use 5×5 window. In addition, the ADSW cost aggregation step is removed in the HE-DE algorithm, and the median filter in the SEP step is replaced by the 3×3 bilateral filter.

Table IV-5 Window sizes of filter-based processes in HE-DE algorithm

Step	Computation	Resolution	Window Size
No-Motion Registration (NMR)	BF	High	5×5
Adaptive Support-Weight Cost Aggregation (ADSW)	-	-	-
Occlusion Handling (OCC)	Vote	High	3×3
Joint Bilateral Upsampling (JBU)	JBF	High	5×5
Window Vote (WVote)	Vote	High	5×5
Still-Edge Preservation (SEP)	BF	High	3×3

4.3.6 Regular Occlusion Handling

For the irregularity problem in the original occlusion handling, we propose a new occlusion handling method that could be performed by raster-scan order, and take care of the inter-view consistency at the same time. Figure IV-33 shows the flow of proposed new occlusion handling method, which consists of the left-right check (LRC) to detect occlusion regions, and the inter-view and the intra-view reference steps to fill the occlusion regions.

The inter-view reference step fills the target-view disparity map using the other two view disparity maps. For example, the two view disparity maps D_{LL}^t, D_{LR}^t are warped to the center view. Only the non-occluded disparity pixels could be warped. If there are many disparities warped to the same position, the highest disparity is selected. Then, the occlusion regions could be filled by the warped disparity map. The inter-view reference step can not only recover most of the occlusion regions, but also enhance the inter-view consistency because of its cross warping.

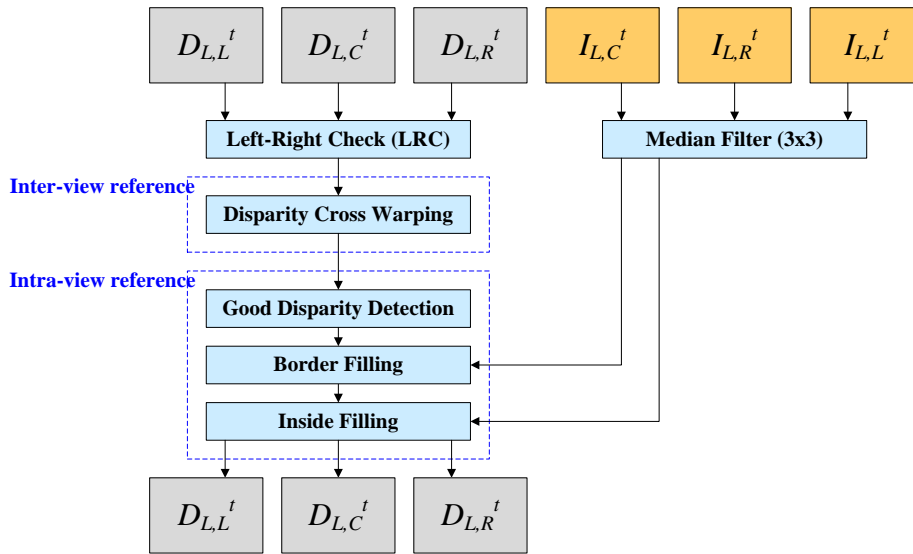


Figure IV-33 Flow of proposed occlusion handling method in HE-DE algorithm

Then, the rest of occlusion regions are filled by the intra-view reference step, which consists of the good disparity detection, the border filling, and the inside filling. The main idea of the intra-view reference step is to fill the occlusion regions by the neighboring non-occlusion disparity pixel in intra frame. To find the reliable non-occlusion disparity pixels, the good disparity detection applies the double-LRC method to find the “good disparity”. The double-LRC method checks the disparity consistency by referring to the other two views, instead one view in the original LRC method. The “good disparity” passing the examination of double-LRC can be used to fill the rest of occlusion regions. Finally, the filling process contains the border part and the inside part of frame. They also adopt the modified window vote method proposed in Section 4.1.6, and the center pixel of vote window should be a “good disparity”.

The computation of the proposed new occlusion handling method does not have the occlusion extension process, which would result in irregular computation. All the computation in this method is performed in raster-scan order. In addition, the inter-view consistency could be enhanced by the inter-view reference step.

4.3.7 Simple Region Detection

In addition to the above methods to deal with the design challenges in the HQ-DE algorithm, the edge detection and the motion detection are also simplified in the HE-DE algorithm. Figure IV-34 shows the flow of the simplified edge detection and motion detection. In which, the bilateral filter uses the window size of 5×5 , and the block size of the block-based motion calculation is reduced from 32×32 to 4×2 that is equal to the downsampling factor.

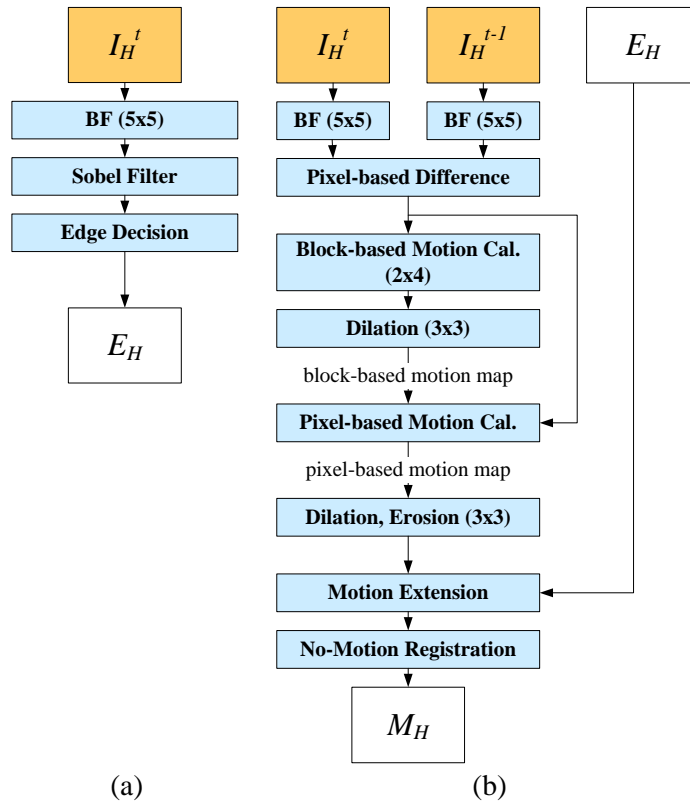


Figure IV-34 Flow of edge detection and motion detection in HE-DE algorithm

(a) edge detection, (b) motion detection

To sum up, the proposed HE-DE algorithm could solve the high memory cost and high computational complexity in HQ-DE algorithm by our simplification. For the high memory cost problems, the proposed cost diffusion method could replace the BP-M optimization to reduce the memory requirement to only one data row whose size is independent to the disparity range. In addition, the proposed window-based SSAD could decrease the image buffers to the size of five image rows. On the other hand, for the high computational complexity problems, the window size of filter-based processes are decreased, and the original irregular occlusion handling method is improved. With these simplifications, the HE-DE algorithm is suitable to be implemented by VLSI design. The architecture design of HE-DE algorithm is presented in Chapter VI.

4.4 Summary

In this chapter, we propose the HQ-DE algorithm to improve the temporal consistency and the occlusion problems in the baseline algorithm. In addition, the BP-M approach is also applied to accelerate the BP optimization.

Based on the HQ-DE algorithm, we further propose two new fast disparity estimation algorithms for different implementation methods. For the software-based implementation, we deliver the SC-DE algorithm, which performs the matching cost calculation, cost aggregation, and BP-M on the sparse pixels, and updates partial disparity map in successive frames. The sparse pixels are no-motion ones for center-view disparity estimation, and occlusion ones for side-view disparity estimation. Compared to the HQ-DE algorithm, the major computation in BP-M could be reduced to 13.4% in the SC-DE algorithm. The SC-DE algorithm is suitable to be executed on the software-based platform because of its sparse computation.

On the other hand, for the VLSI implementation, we propose the HE-DE algorithm, which improves the design challenges of high memory cost and computational complexity in the HQ-DE algorithm. For the high memory cost problem, we propose the cost diffusion method and the window-based SSAD to replace the original methods. The major memory cost in BP-M could be reduced to 0.00029% by the proposed cost diffusion method. For the high computational complexity problem, we decrease the filter size and propose a new occlusion handling method with regular computation.

The above advanced disparity estimation algorithms are evaluated on the disparity quality and execution time in the next chapter.

V Experimental Results

The previous chapter presents the proposed baseline disparity estimation (baseline) algorithm, high-quality disparity estimation (HQ-DE) algorithm, sparse-computation disparity estimation (SC-DE) algorithm, and hardware-efficient disparity estimation (HE-DE) algorithm using different strategies. They have different improvement in the disparity quality or the computational speed. In this chapter, we first introduce the experiment setting about the test sequences and the input/output configuration. Then, we compare those algorithms by the execution time on PC and the objective quality evaluation through the view synthesis results.

5.1 Experiment Setting

5.1.1 Test Sequences



Figure V-1 shows the test sequences adopted in the experiment, and Table V-1 lists their detailed information. The test sequences are provided by different research institutes. The frame size includes 1024×768 (XGA), 1920×1080 (HD1080p), and 1280×960 . In these sequences, the Kendo, Balloons, Hall1, and Hall2 are captured by the moving cameras, and others are captured by fixed cameras. In addition, all the sequences are rectified by the similar processes as described in [78]. In the processes, the brightness, contrast, and gamma among views are adjusted to be consistent. Then, the lens distortion and chromatic aberration are rectified in the normalization process. Finally, all the view images are re-projected to the position with parallel optic axis. Because of the rectification processes, the source videos could be directly used to disparity estimation without any pre-processing, and the disparity range can be limited in 1-D space.



Figure V-1 Clips of test sequences in center view

(a) BookArrival, (b) LoveBird1, (c) Newspaper, (d) Café, (e) Kendo, (f) Balloons, (g) Champagne, (h) Pantomime, (i) Hall1, (j) Hall2, (k) Street, (l) CarPark

Table V-1 Test sequences

Sequence Name	Provider	Frame Size	Frame Rate (frame/s)	Number of Frame	Number of View	Camera Spacing (cm)	Is Moving Camera
BookArrival	HHI	1024×768	16.67	300	16	6.5	No
LoveBird1	ETRI	1024×768	30	300	12	3.5	No
Newspaper	GIST	1024×768	30	300	9	6.5	No
Café	GIST	1920×1080	30	200	5	5	No
Kendo	Nagoya	1024×768	30	300	7	5	Yes
Balloons	Nagoya	1024×768	30	300	7	5	Yes
Champagne	Nagoya	1280×960	30	300	80	5	No
Pantomime	Nagoya	1280×960	30	300	80	5	No
Hall1	Poznan	1920×1088	25	200	9	13.75	Yes
Hall2	Poznan	1920×1088	25	200	9	13.75	Yes
Street	Poznan	1920×1088	25	250	9	13.75	No
CarPark	Poznan	1920×1088	25	250	9	13.75	No

HHI: Fraunhofer Heinrich Hertz Institute, Germany

ETRI: Electronics and Telecommunications Research Institute, Korea

GIST: Gwangju Institute of Science and Technology, Korea

Nagaya: Nagoya University, Japan

Poznan: Poznan University of Technology, Poznan

5.1.2 Input and Output Configuration

As mentioned in Section 2.3.1, the MPEG 3DVC defines the 2-view configuration and the 3-view configuration for different displays. Table V-2 lists the selected views of all the test sequences for 2-view configuration. The frame ranges of test sequences are also defined for the disparity quality evaluation and the coding performance evaluation. This table only lists the frame range for disparity estimation. On the other hand, Table V-3 shows the selected input and output views for the 3-view configuration. In which, the output views for the stereoscopic display are located near the center-view input I_C . For the 9-view displays, the most left output is located at the middle of center-view input I_C and left-view input I_L , while the most right output is located at the middle of center-view input I_C and right-view input I_R . The wider spacing between the most left and right output, the higher performance required in the view synthesis and disparity estimation algorithms. In this dissertation, we focus on the 3-view configuration for 9-view displays.

Table V-2 Input and output views for 2-view configuration [71]

Sequence Name	Input View No. (I_L-I_R)	Synthesized Pair (I_L-V_C or V_C-I_R)	Frame Range for Disparity Estimation
BookArrival	10-8	10-9	0-99
LoveBird1	6-8	7-8	0-299
Newspaper	4-6	5-6	0-299
Café	1-3	2-3	0-299
Kendo	2-4	3-4	0-299
Balloons	2-4	3-4	0-299
Champagne	39-41	40-41	0-499
Pantomime	39-41	40-41	0-499
Hall1	2-1	2-1.5	0-199
Hall2	7-6	7-6.5	0-199
Street	4-3	4-3.5	150-349
CarPark	4-3	4-3.5	200-399

Table V-3 Input and out views for 3-view configuration [71]

Sequence Name	Input View No. ($I_L-I_C-I_R$)	Output for Stereoscopic Display	Output for 9-view Display
BookArrival	10-8-6	8.25-7.75	9 to 7
LoveBird1	4-6-8	5.75-6.25	5 to 7
Newspaper	2-4-6	3.75-4.25	3 to 5
Café	1-3-5	2.75-3.25	2 to 4
Kendo	1-3-5	2.75-3.25	2 to 4
Balloons	1-3-5	2.75-3.25	2 to 4
Champagne	37-39-41	37.75-39.25	38 to 40
Pantomime	37-39-41	37.75-39.25	38 to 40
Hall1	3-2-1	2.125-1.875	2.5 to 1.5
Hall2	7-6-5	6.125-5.875	6.5 to 5.5
Street	5-4-3	4.125-3.875	4.5 to 3.5
CarPark	5-4-3	4.125-3.875	4.5 to 3.5

Table V-4 summarizes the our experiment setting for the DERS algorithm and our proposed algorithms. The target outputted views are the most left and the most right ones in the 3-view configuration for 9-view display. The disparity ranges are dependent on the sequence content, and the frame ranges are the same as those in Table V-2. For the inputted views, our proposed algorithm only requires three views, and meets the defined configuration in Table V-3. However, the DERS algorithm requires five views because of its functionality described in Section 2.3.1. It would result in that the DERS algorithm could not produce complete three view disparity maps in the sequences, Kendo, Balloons and Café, for view synthesis due to insufficient inputted views. For the three sequences, the column “Avail.” is marked by “No” in Table V-4.

In addition, the sequences Hall1, Hall2, Street, and CarPark, cannot be evaluated by objective method, because the common evaluation methods need the real captured videos to compare with the synthesized videos. However, these sequences target outputs are at the fractional positions, that means there are no real captured videos. Thus, the four sequences could not be used in the objective evaluation, and their column “Eval.” is marked by “No”.

To sum up, the DERS algorithm could provide the results of sequences BookArrival, Pantomime, Champagne, LoveBird1, and Newspaper for the objective evaluation. On the other hand, our algorithms could not provide only the results of sequences Hall1, Hall2, Street, and CarPark for the objective evaluation.

Table V-4 Experiment setting in our evaluation

Sequence Name	Output No.	Frame Size	Disparity Range	Frame Range	DERS			Our algorithms		
					Input No.	Avail.	Eval.	Input No.	Avail.	Eval.
BookArrival	9, 7	1024×768	70	0-99	12-10-8-6-4	Yes	Yes	10-8-6	Yes	Yes
LoveBird1	5, 7	1024×768	70	0-299	2-4-6-8-10	Yes	Yes	4-6-8	Yes	Yes
Newspaper	3, 5	1024×768	88	0-299	0-2-4-6-8	Yes	Yes	2-4-6	Yes	Yes
Café	2, 4	1920×1080	160	0-299	-	No	No	1-3-5	Yes	Yes
Kendo	2, 4	1024×768	64	0-299	-	No	No	1-3-5	Yes	Yes
Balloons	2, 4	1024×768	64	0-299	-	No	No	1-3-5	Yes	Yes
Champagne	38, 40	1280×960	110	0-499	35-37-39-41-43	Yes	Yes	37-39-41	Yes	Yes
Pantomime	38, 40	1280×960	40	0-499	35-37-39-41-43	Yes	Yes	37-39-41	Yes	Yes
Hall1	2.5, 1.5	1920×1088	80	0-199	4-3-2-1-0	Yes	No	3-2-1	Yes	No
Hall2	6.5, 5.5	1920×1088	64	0-199	8-7-6-5-4	Yes	No	7-6-5	Yes	No
Street	4.5, 3.5	1920×1088	64	150-349	6-5-4-3-2	Yes	No	5-4-3	Yes	No
CarPark	4.5, 3.5	1920×1088	64	200-399	6-5-4-3-2	Yes	No	5-4-3	Yes	No

5.2 Comparison

5.2.1 Execution Time

For the comparison of execution time, all the algorithms are compiled by the Microsoft Visual Studio 2010 with the optimization option of O2. The compiled programs are executed on the same PC that has the 2.83-GHz Intel Core2 Quad CPU and the 4-Gbyte RAM with the operation system of 32-bit Windows 7. Table V-5 compares the average execution time of the proposed algorithms for one frame. The execution time is measured in the calculation of 3 view disparity maps for the defined

frame range listed in Table V-4. Moreover, we scale the average execution time to the same resolution of 1920×1080 and disparity range of 128 for our target specification. Table V-6 lists the scaled average execution time. In which, the HQ-DE algorithm could take advantage of the single iterative BP-M to speed up the baseline algorithm by 2.7 times in average. In addition, the HQ-DE algorithm is 9.3 times faster than the DERS algorithm.

Compared to the HQ-DE algorithm, the SC-DE algorithm could reduce the execution time to 62.9% by the sparse computation method. On the other hand, the HE-DE algorithm employs the proposed efficient cost diffusion method and the filter computation with decreased window size to reduce the execution time to 57.2%.

Table V-5 Average execution time of proposed algorithms on PC for one frame

Sequence Name	DERS	Baseline	HQ-DE	SC-DE	HE-DE
BookArrival	161,182	100,327	36,907	21,448	20,534
LoveBird1	248,399	73,460	31,990	20,020	18,620
Newspaper	281,858	138,565	42,376	22,627	25,121
Café	N.A	1,011,112	206,917	N.A	131,773
Kendo	N.A	73,755	31,854	21,999	17,890
Balloons	N.A	72,604	31,640	22,531	17,880
Champagne	652,850	306,348	77,766	35,707	47,589
Pantomime	498,999	58,091	39,762	30,672	19,858
Hall1	286,225	297,946	100,916	57,361	59,135
Hall2	800,368	220,713	88,216	55,355	49,354
Street	1,187,748	184,441	83,457	53,340	47,269
CarPark	1,377,180	195,976	84,309	54,188	47,902

Unit: ms

Table V-6 Average execution time scaled to HD1080p resolution and disparity range of 128

Sequence Name	DERS	Baseline	HQ-DE	SC-DE	HE-DE
BookArrival	777,129	483,719	177,944	103,410	99,003
LoveBird1	1,197,638	354,182	154,238	96,525	89,775
Newspaper	1,080,990	531,428	162,522	86,780	96,345
Café	N.A	808,890	165,534	N.A	105,418
Kendo	N.A	388,942	167,980	116,010	94,342
Balloons	N.A	382,873	166,852	118,816	94,289
Champagne	1,281,961	601,556	152,704	70,116	93,447
Pantomime	2,694,595	313,691	214,715	165,629	107,233
Hall1	454,592	473,208	160,278	91,103	93,920
Hall2	1,588,966	438,180	175,135	109,896	97,982
Street	2,358,028	366,170	165,687	105,896	93,843
CarPark	2,734,108	389,070	167,378	107,579	95,100
Average	1,574,223	460,993	169,247	106,524	96,725
Compared to HQ-DE	930.1%	272.4%	100.0%	62.9%	57.2%

Unit: ms

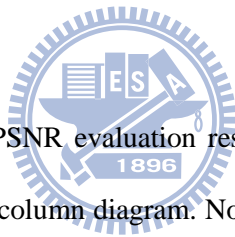
5.2.2 Objective Quality Evaluation

The experiment setting follows the description in previous section. As mentioned in Section 2.3.4, the common-used objective quality evaluation methods are PSNR, SSIM, and T_PSPNR. Their main idea is to evaluate disparity quality by view synthesis results. Thus, they compare the difference between the real captured videos and the synthesized videos, and then analyze the frame difference by different methods. The PSNR and SSIM could be used to evaluate the spatial distortion, and the T_PSPNR could be used to evaluate the temporal distortion. The associated software tools can be obtained from [63], [77]. Note that the view synthesis algorithms are different for the DERS algorithm and our proposed algorithms. The DERS algorithm cooperates with the VSRS algorithm [64], while our proposed algorithms cooperates with the simplified VSRS algorithm [62] that adopts the Gaussian filter for the hole filling and has approximate quality to the original VSRS algorithm.

1. PSNR Evaluation Results

Table V-7 and Table V-8 shows the PSNR evaluation results for luminance channel only, and Figure V-2 plots the corresponding data by column diagram. Note that the “View0” and “View8 mean the left most and the right most views for the 9-view displays. Note that the results of Café, Kendo, and Balloons are not available in the DERS algorithm due to the reason described in previous section. In addition, the proposed SC-DE algorithm could not generate disparity maps for the sequence Café because of insufficient memory space on PC to support the extremely high resolution and large disparity range. In this table, Δ PSNR is the PSNR difference of our algorithm and the DERS algorithm. The positive Δ PSNR refers to our algorithm performs better than the DERS algorithm, and vice versa.

Compared to the DERS algorithm, the baseline algorithm could not perform better in most sequences because the baseline algorithm only focuses on the computational reduction, instead of the disparity quality improvement. With the temporal consistency and occlusion improvement methods, the HQ-DE algorithm could has higher PSNR than the DERS algorithm in average.



The SC-DE algorithm is accelerated version of HQ-DE algorithm, and suffers from slight PSNR drops. On the other hand, the HE-DE algorithm, the other accelerated version of HQ-DE algorithm, has the slight quality drops in all sequences except the sequence Champagne, and the average PSNR is higher than other algorithms. That implies the proposed cost diffusion method and the new irregular occlusion handling method could deliver better disparity maps than the other proposed algorithms.

Table V-7 Evaluation results of Y-PSNR for View0

	DERS	Baseline		HQ-DE		SC-DE		HE-DE	
	PSNR	PSNR	ΔPSNR	PSNR	ΔPSNR	PSNR	ΔPSNR	PSNR	ΔPSNR
BookArrival	34.28	35.54	1.26	35.98	1.70	35.85	1.58	35.80	1.53
LoveBird1	32.45	32.07	-0.38	32.63	0.18	32.58	0.13	31.53	-0.92
Newspaper	29.53	29.27	-0.27	29.90	0.37	29.84	0.31	30.03	0.49
Café	N.A.	32.83	-	33.30	-	N.A.	-	33.22	-
Kendo	N.A.	34.66	-	34.84	-	34.82	-	34.88	-
Balloons	N.A.	34.72	-	35.07	-	34.79	-	34.91	-
Champagne	25.32	28.27	2.95	27.63	2.31	24.99	-0.32	31.07	5.75
Pantomime	36.46	37.01	0.55	35.94	-0.52	35.58	-0.88	34.66	-1.80
Average	31.61	33.04	0.82	33.16	0.81	32.64	0.16	33.26	1.01

Unit: dB

Table V-8 Evaluation results of Y-PSNR for View8

	DERS	Baseline		HQ-DE		SC-DE		HE-DE	
	PSNR	PSNR	ΔPSNR	PSNR	ΔPSNR	PSNR	ΔPSNR	PSNR	ΔPSNR
BookArrival	35.87	35.68	-0.19	35.89	0.02	36.08	0.21	36.02	0.15
LoveBird1	29.31	27.53	-1.78	28.23	-1.08	28.22	-1.09	27.98	-1.33
Newspaper	31.86	31.29	-0.57	31.76	-0.10	31.65	-0.20	31.92	0.06
Café	N.A.	32.87	-	33.01	-	N.A.	-	33.04	-
Kendo	N.A.	35.75	-	36.24	-	36.12	-	36.36	-
Balloons	N.A.	35.24	-	35.63	-	35.40	-	35.58	-
Champagne	24.20	28.72	4.52	28.11	3.91	27.46	3.26	29.73	5.53
Pantomime	34.65	35.85	1.20	36.00	1.35	36.13	1.48	35.61	0.96
Average	31.18	32.87	0.64	33.11	0.82	33.01	0.73	33.28	1.08

Unit: dB

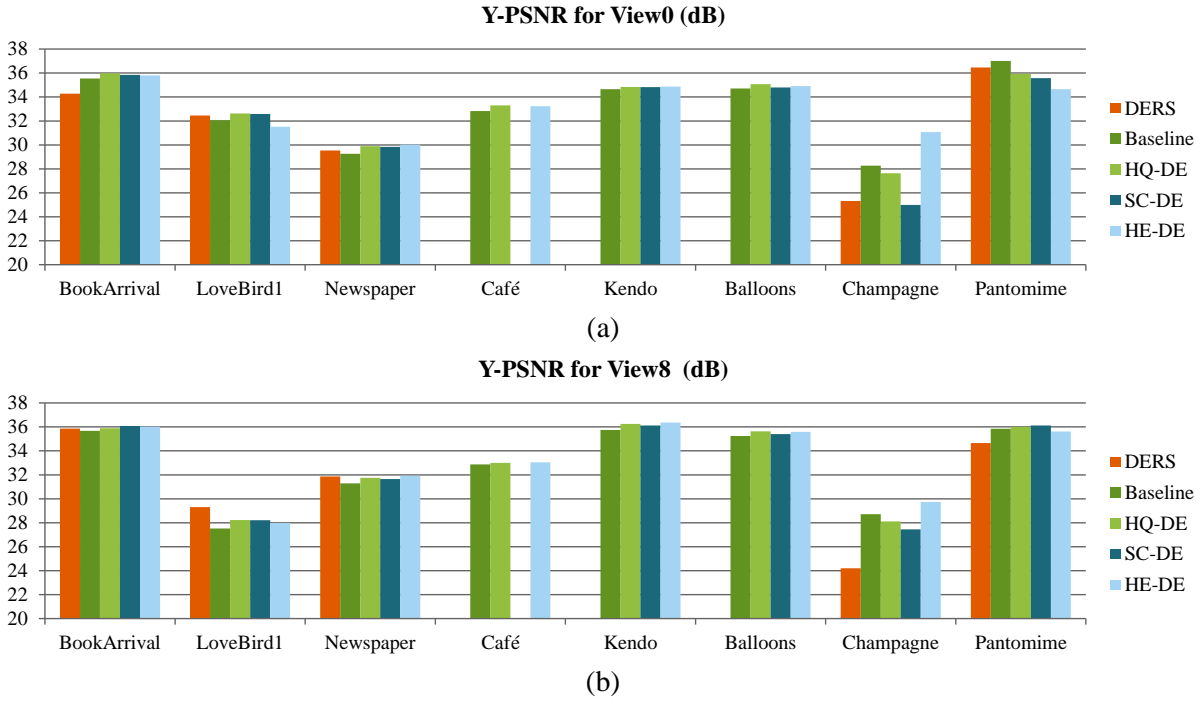


Figure V-2 Evaluation results of Y-PNSR

2. SSIM Evaluation Results

In the SSIM evaluation, we calculate the average of the SSIMs in the three channels, R, G, and B for each sequence. Table V-9 and Table V-10 list the SSIM evaluation results for the View0 and View8, and Figure V-3 shows the corresponding column diagrams. With the SSIM evaluation results, all the proposed algorithms could have the approximate quality to the DERS algorithm but suffer from slight drops less than 0.02.

Table V-9 Evaluation results of SSIM for View0

	DERS	Baseline		HQ-DE		SC-DE		HE-DE	
	SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM
BookArrival	0.98	0.95	-0.02	0.95	-0.02	0.95	-0.02	0.95	-0.02
LoveBird1	0.95	0.95	0.00	0.96	0.00	0.96	0.00	0.95	0.00
Newspaper	0.99	0.99	0.00	0.99	0.00	0.99	0.00	0.99	0.00
Café	N.A.	0.99	-	0.99	-	N.A.	-	0.99	-
Kendo	N.A.	0.98	-	0.98	-	0.98	-	0.98	-
Balloons	N.A.	0.97	-	0.98	-	0.98	-	0.97	-
Champagne	0.97	0.97	0.00	0.96	-0.01	0.95	-0.02	0.97	0.00
Pantomime	0.98	0.98	0.00	0.98	0.00	0.97	0.00	0.97	0.00
Average	0.97	0.97	-0.01	0.97	-0.01	0.85	-0.01	0.97	0.00

Table V-10 Evaluation results of SSIM for View8

	DERS	Baseline		HQ-DE		SC-DE		HE-DE	
	SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM
BookArrival	0.97	0.95	-0.02	0.95	-0.02	0.95	-0.02	0.95	-0.02
LoveBird1	0.93	0.92	-0.01	0.92	-0.01	0.92	-0.01	0.92	-0.01
Newspaper	0.99	0.98	-0.01	0.99	0.00	0.99	0.00	0.99	0.00
Café	N.A.	0.99	-	0.99	-	N.A.	-	0.99	-
Kendo	N.A.	0.98	-	0.99	-	0.99	-	0.99	-
Balloons	N.A.	0.98	-	0.98	-	0.99	-	0.98	-
Champagne	0.97	0.97	0.00	0.96	0.00	0.96	-0.01	0.97	0.00
Pantomime	0.97	0.97	0.00	0.97	0.00	0.97	0.00	0.97	0.00
Average	0.97	0.97	-0.01	0.97	-0.01	0.97	-0.01	0.97	-0.01

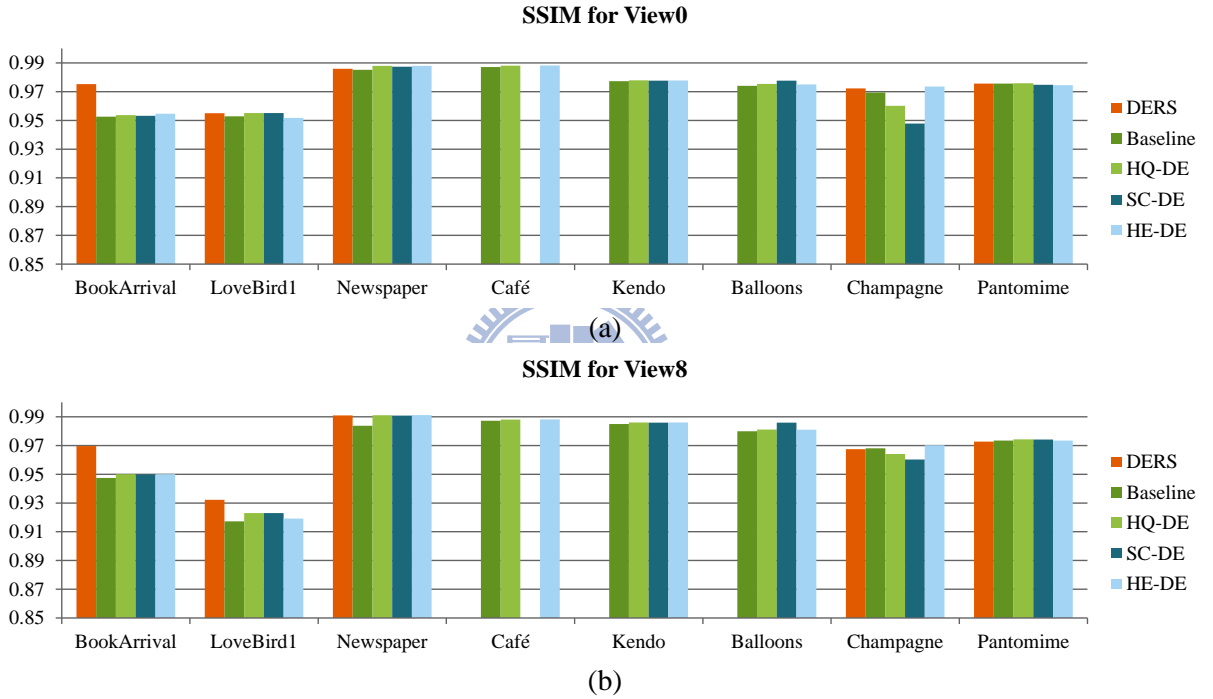


Figure V-3 Evaluation results of SSIM

3. PSPNR Evaluation Results

The PSPNR evaluation method [76] consists of the S_PSPNR for spatial distortion and the T_PSPNR for temporal distortion. In this dissertation, we adopt the T_PSPNR to evaluate the temporal consistency of disparity maps. Table V-11 and Table V-12 list the T_PSPNR evaluation results, and Figure V-4 plots the corresponding column diagrams. Compared to the DERS algorithm, the baseline algorithm has serious quality degradation due to no temporal consistency enhancement applied. Taking advantage of the proposed temporal consistency enhancement methods, the HQ-DE

algorithm could have higher performance than the DERS algorithm. Such the high performance is slightly decreased in the SC-DE and HE-DE algorithms in most sequences because of their acceleration methods. Nevertheless, the two fast algorithms still perform better than the DERS in most of the sequences.

Table V-11 Evaluation results of T_PSPNR (dB) for View0

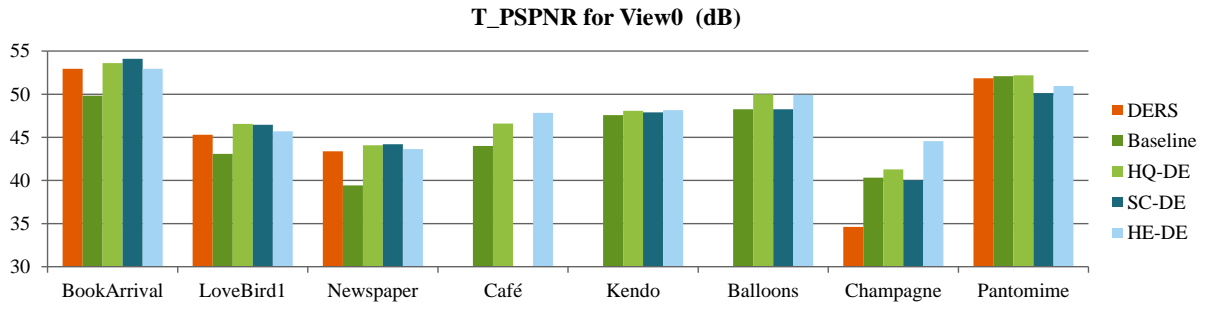
	DERS	Baseline		HQ-DE		SC-DE		HE-DE	
	T_PSPNR	T_PSPNR	ΔT_PSPNR	T_PSPNR	ΔT_PSPNR	T_PSPNR	ΔT_PSPNR	T_PSPNR	ΔT_PSPNR
BookArrival	52.96	49.83	-3.13	53.60	0.64	54.10	1.14	52.94	-0.02
LoveBird1	45.30	43.08	-2.23	46.57	1.26	46.46	1.16	45.70	0.39
Newspaper	43.38	39.44	-3.94	44.09	0.71	44.19	0.82	43.65	0.27
Café	N.A.	44.00	-	46.59	-	N.A.	-	47.83	-
Kendo	N.A.	47.57	-	48.08	-	47.90	-	48.15	-
Balloons	N.A.	48.25	-	49.99	-	48.25	-	49.93	-
Champagne	34.62	40.34	5.72	41.28	6.66	40.03	5.41	44.56	9.94
Pantomime	51.85	52.10	0.25	52.19	0.35	50.12	-1.72	50.95	-0.90
Average	45.62	45.57	-0.67	47.80	1.92	41.38	1.36	47.96	1.94

Unit dB

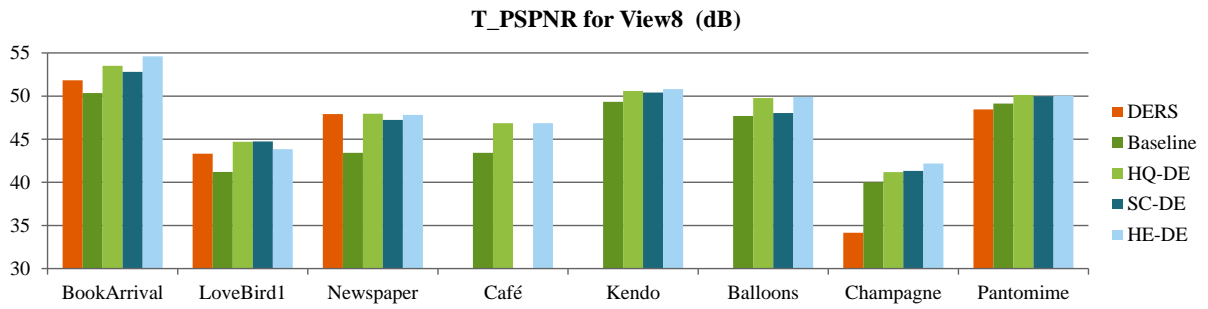
Table V-12 Evaluation results of T_PSPNR for View8

	DERS	Baseline		HQ-DE		SC-DE		HE-DE	
	T_PSPNR	T_PSPNR	ΔT_PSPNR	T_PSPNR	ΔT_PSPNR	T_PSPNR	ΔT_PSPNR	T_PSPNR	ΔT_PSPNR
BookArrival	51.82	50.34	-1.48	53.52	1.70	52.81	0.99	54.62	2.79
LoveBird1	43.33	41.21	-2.11	44.70	1.37	44.75	1.42	43.84	0.51
Newspaper	47.92	43.43	-4.49	47.96	0.04	47.24	-0.67	47.82	-0.09
Café	N.A.	43.42	-	46.86	-	N.A.	-	46.85	-
Kendo	N.A.	49.34	-	50.58	-	50.41	-	50.81	-
Balloons	N.A.	47.69	-	49.76	-	48.03	-	49.90	-
Champagne	34.16	40.00	5.84	41.18	7.02	41.32	7.16	42.19	8.03
Pantomime	48.45	49.13	0.68	50.12	1.67	49.98	1.53	50.06	1.61
Average	45.14	45.57	-0.31	48.09	2.36	47.79	2.09	48.26	2.57

Unit dB



(a)



(b)

Figure V-4 Evaluation results of T_PSPNR

4. Disparity Maps and Synthesized Images

Finally, the disparity maps and view synthesis results are demonstrated in Figure V-5 to Figure V-16. The HQ-DE algorithm could improve the disparity maps and synthesized images better than the baseline algorithm, and has comparable results to the DERS algorithm. Compared to the HQ-DE algorithm, the SC-DE and the HE-DE algorithms has disparity noising at the object boundaries due to their simplified methods.





Figure V-5 Disparity maps and view synthesized images in the 50th frame of BookArrival Results from top to down are the produced by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.

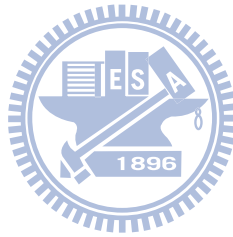






Figure V-6 Disparity maps and view synthesized images in the 50th frame of LoveBird1
Results from top to down are the produced by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.

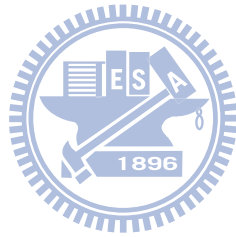






Figure V-7 Disparity maps and view synthesized images in the 100th frame of Newspaper
 Results from top to down are the produced by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.



Figure V-8 Disparity maps and view synthesized images in the 50th frame of Café
 Results from top to down are the produced by the baseline, HQ-DE, HE-DE algorithms.

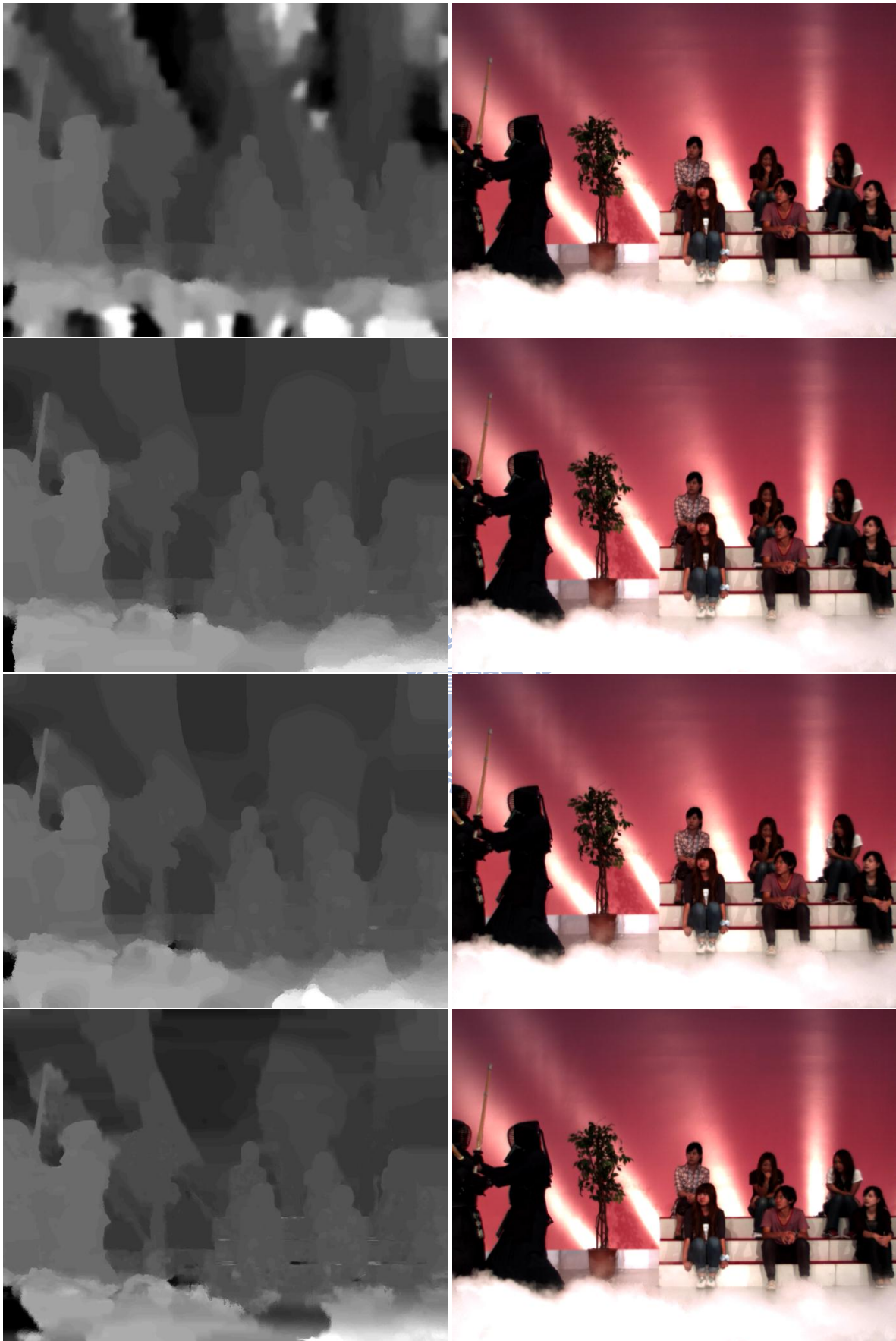


Figure V-9 Disparity maps and view synthesized images in the 50th frame of Kendo
 Results from top to down are the produced by the baseline, HQ-DE, SC-DE, HE-DE algorithms.

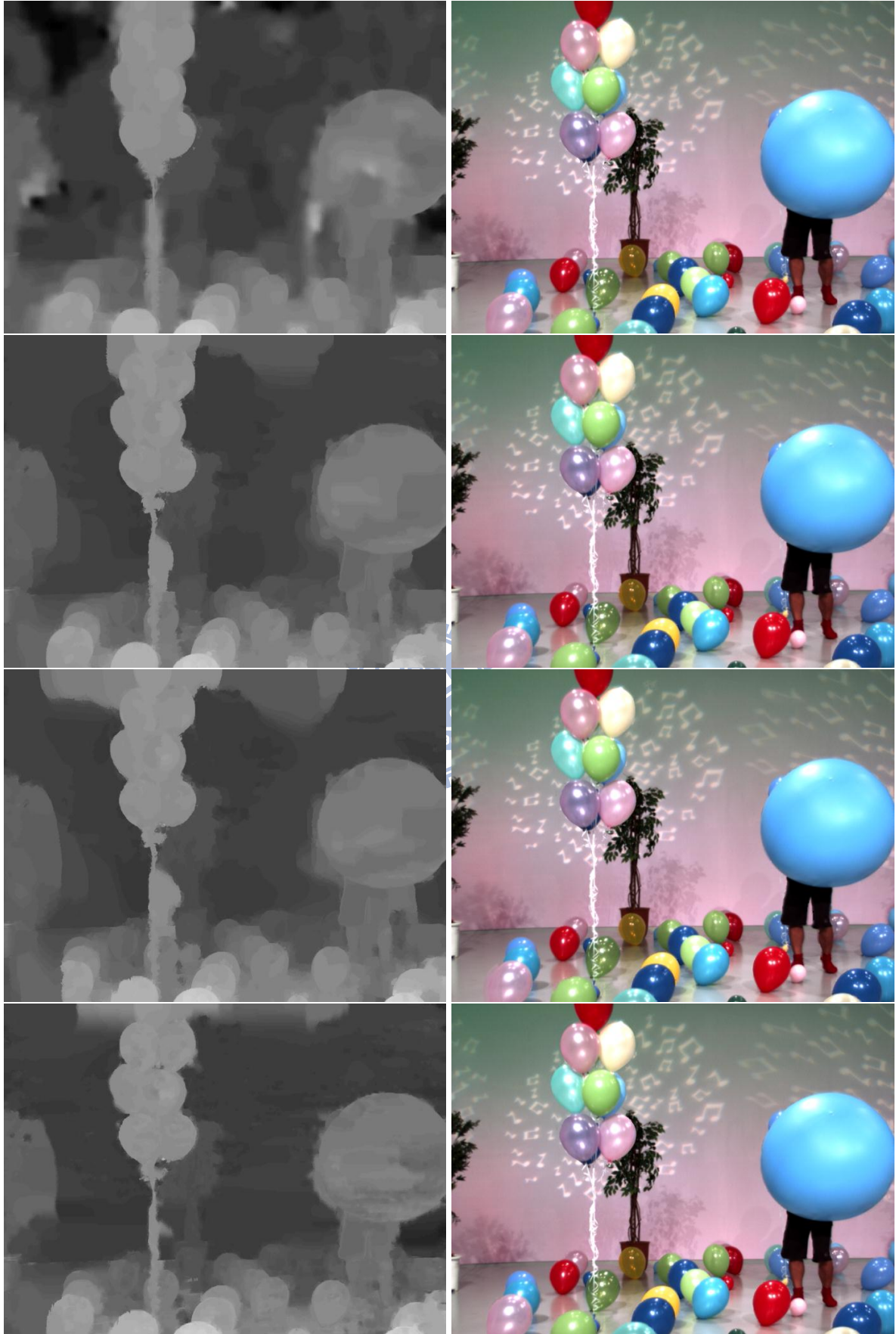
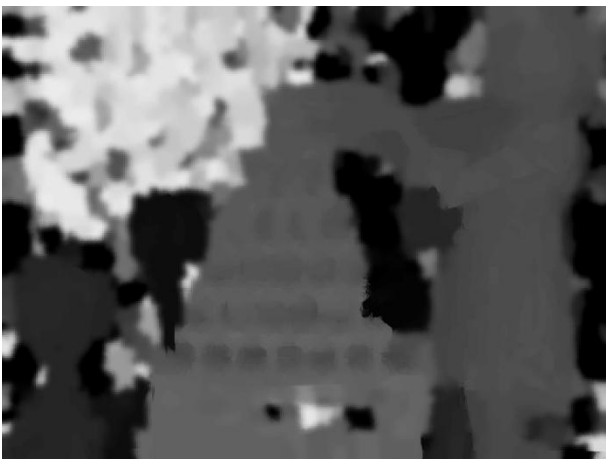


Figure V-10 Disparity maps and view synthesized images in the 100th frame of Balloons Results from top to down are the produced by the baseline, HQ-DE, SC-DE, HE-DE algorithms.



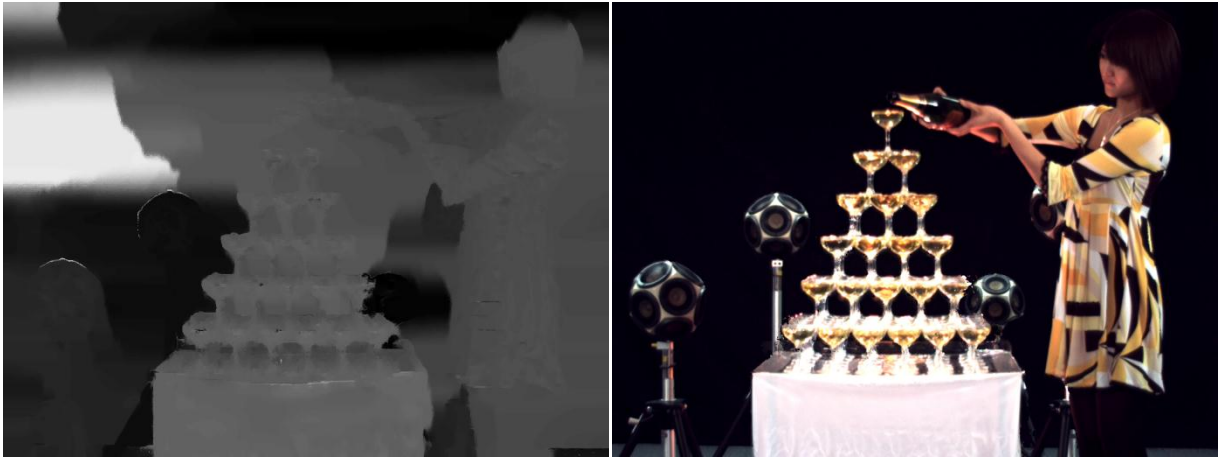
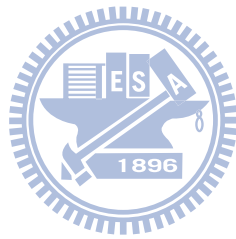


Figure V-11 Disparity maps and view synthesized images in the 50th frame of Champagne
Results from top to down are the produced by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.





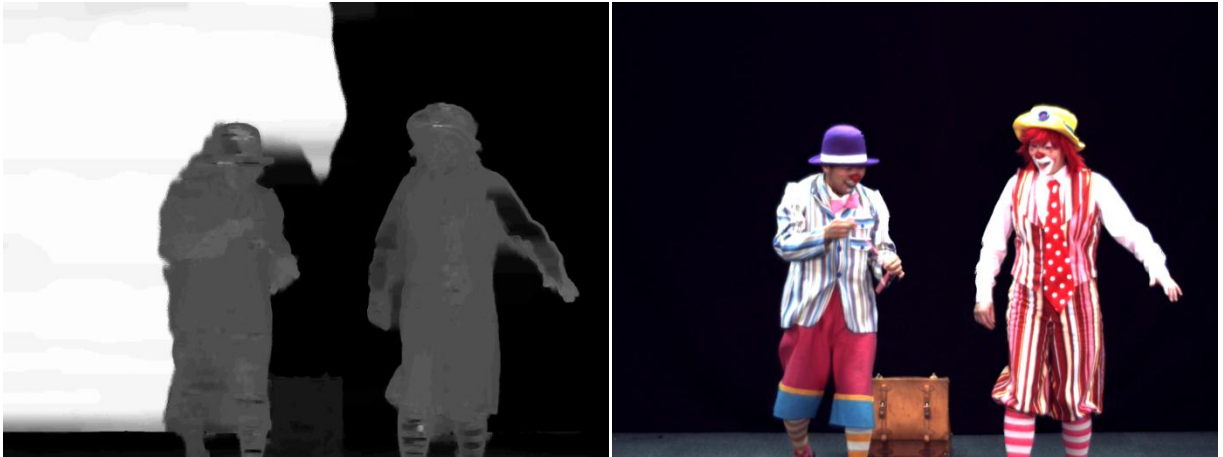


Figure V-12 Disparity maps and view synthesized images in the 50th frame of Pantomime. Results from top to down are the produced by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.

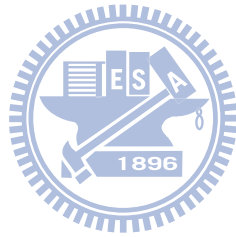




Figure V-13 Disparity maps and view synthesized images in the 50th frame of Hall1
 Results from top to down are by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.

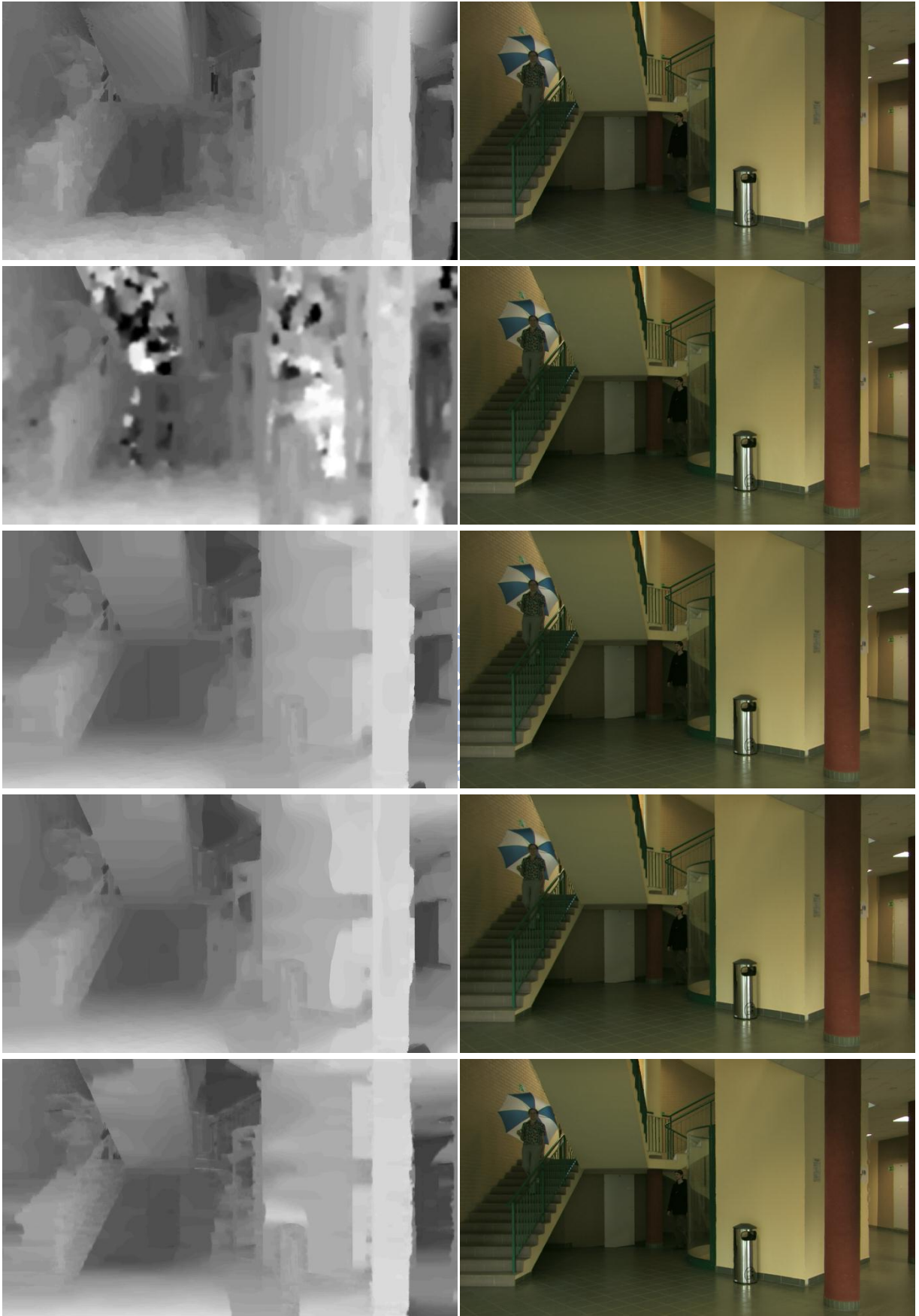


Figure V-14 Disparity maps and view synthesized images in the 50th frame of Hall2
 Results from top to down are by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.



Figure V-15 Disparity maps and view synthesized images in the 167th frame of CarPark Results from top to down are by the DERS, baseline, HQ-DE, SC-DE, HE-DE algorithms.

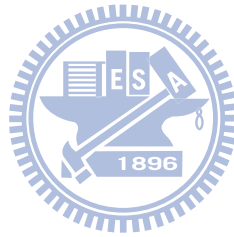


Figure V-16 Disparity maps and view synthesized images in the 50th frame of CarPark
 Results from top to down are the produced by the baseline, HQ-DE, SC-DE, HE-DE algorithms.

5.3 Summary

The disparity quality and execution time of the proposed algorithms are examined using the test bench for view synthesis application. Compared to the DERS algorithm, the proposed HQ-DE

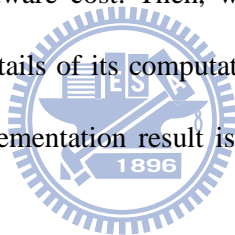
algorithm has high disparity quality in the temporal PSPNR evaluation, and approximate disparity quality in the spatial PSNR evaluation. For the computational comparison, our proposed HQ-DE algorithm is more efficient than the DERS algorithm because our processing resolution is decreased by the disparity upsampling technique. The computation of HQ-DE algorithm could be significantly reduced by the proposed SC-DE and HE-DE algorithms with slight disparity quality change. Moreover, according to their computational characteristics, the SC-DE algorithm could be further accelerated by processor-based platforms, and the HE-DE algorithm could be accelerated by VLSI design. In the next chapter, the HE-DE algorithm is implemented by VLSI design to achieve the required throughput of high resolution 3DTV applications.



VI Design of Disparity Estimation Engine for High Definition 3DTV Applications

The main target of this dissertation is to deliver a disparity estimation engine that can generate three view HD1080p disparity maps in the throughput of 60 frames/s. To achieve this target, we simplify the hardware-efficient disparity estimation (HE-DE) algorithm for lower hardware cost, and propose a corresponding hardware design. The implementation result shows that the proposed disparity estimation engine could achieve the target throughput, and outperform the previous implementation.

This chapter is organized as follows. First, we analyze the data dependency of HE-DE algorithm, and simplify it to reduce more hardware cost. Then, we present the proposed architecture for the simplified HE-DE algorithm. The details of its computational modules and memory access schedule are also described. Finally, the implementation result is demonstrated and compared with previous work.



6.1 Architectural Analysis

6.1.1 Analysis of Hardware-Efficient Disparity Estimation Algorithm

The HE-DE algorithm could significantly reduce the memory cost and computational complexity of HQ-DE algorithm by the proposed methods. However, the HE-DE algorithm still suffers from high hardware cost while considering its detailed architecture. In Figure VI-1, we analyze the data dependency of the HQ-DE algorithm that consists of the main process and the branch process in the whole flow. The main process is from the window-based SSAD to the still-edge preservation steps for the computation of cost cube, low-resolution disparity map, and high-resolution disparity map, while the branch process includes the motion detection and the edge detection steps for the assistant

information of temporal consistency enhancement. Because of no feedback data path in this flow, we could adopt the pipelining architecture to increase the throughput of HE-DE algorithm. With the pipelining architecture, the hardware design of HE-DE algorithm has the high memory cost problem due to the following reasons.

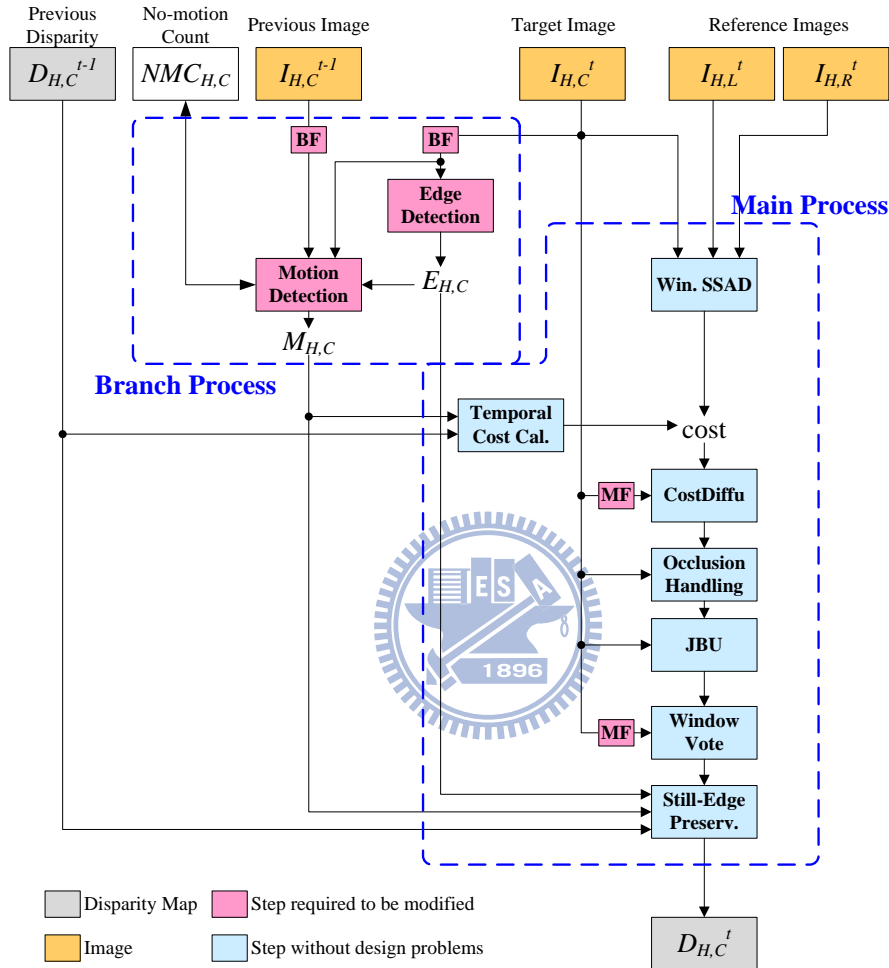


Figure VI-1 Data dependency of the HE-DE algorithm

1. Long Pipelining Stall for Branch Process

In the pipelining architecture, the size of pipelining buffer is related to the computational characteristics of two steps and the stall cycles. The main process has the critical latency but no pipelining stall since the produced data between each two steps can be immediately used. However, the branch process suffers from long pipelining stall because the motion and edge maps are fetched by

the start and the end of main process. Therefore, the branch process requires large memory space to store the data.

2. Filter-based Process

The HE-DE algorithm employs many filter-based processes, such as bilateral filter, median filter, dilation, erosion, and etc. By decreasing their filter size, the computational complexity is significantly reduced in the HE-DE algorithm. However, the filter-based processes still result in high memory cost even if the filter size is minimized to 3×3 . Figure VI-2 shows the required buffers for two continuous filter processes. The step 1 performs a 3×3 filter, and its filter center has moved to the position (x, y) . With the calculated result of Step 1, the step 2 could perform the 5×5 filter for the center position $(x-2, y-2)$. The two steps in the pipelining architecture demand a 2-row buffer and a 4-row buffer, whose total memory size is 1920×6 pixels (i.e. 34.5Kbytes for 3-channel pixel) for the HD1080p resolution. To sum up, the filter-based process is expensive on the memory cost, and a filter with radius r needs a buffer with $2r$ frame rows at least. Therefore, we should try to remove the filter-based processes in the HE-DE algorithm under the condition of no observable impact on disparity quality.

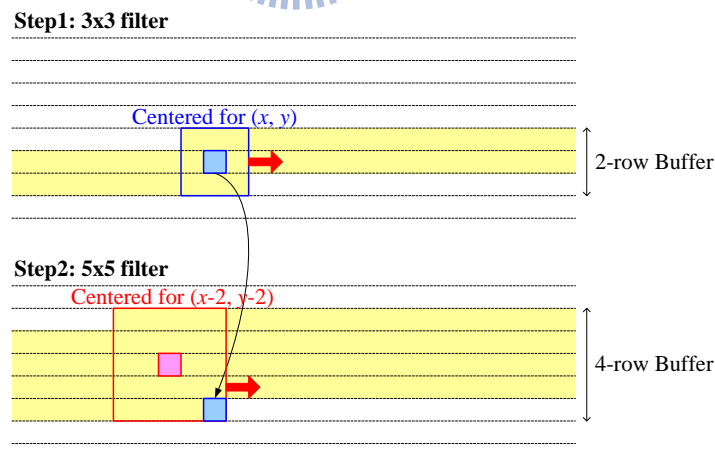


Figure VI-2 Required row buffers in filter-based processes for pipelining architecture

3. Motion Detection with Sequential Steps

The motion detection in the HE-DE algorithm is to find the motion map for the temporal cost calculation and the still-edge preservation. It mainly includes the frame difference computation and the

motion map extension as shown in Figure IV-34 (a). The motion detection method is simple but needs many pipelining buffers between each two steps as shown in Figure VI-5. It results from that each step in the motion detection is sequentially performed, and their required data are cross multiple rows. In addition, the motion detection also suffers from the problems of filter-based processes due to the dilation and erosion. Therefore, we should further simplify the motion detection with the consideration of memory cost.

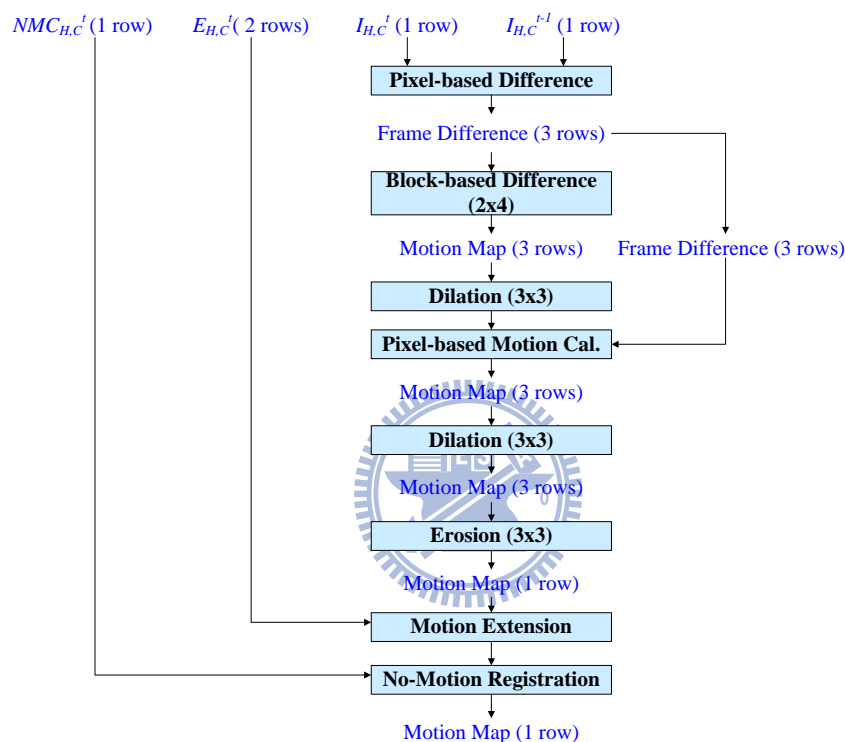


Figure VI-3 Memory buffers in the motion detection

To solve the above problems, we modify partial blocks of the HE-DE algorithm in Figure VI-1. In which, we would merge the edge detection into the sill-edge preservation, remove all the bilateral filters (BF) and the median filters (MF), and simplify the motion detection. The improved HE-DE algorithm could be implemented by VLSI design, and is called hardware-based disparity estimation (HW-DE) algorithm in this dissertation.

6.1.2 Proposed Hardware-Based Algorithm

Figure VI-4 shows the flow of proposed HW-DE algorithm. In which, the window sizes are minimized to 3×3 for the window-based SSAD and the window vote. In addition, all the bilateral and median filters for de-noising images are removed in the HW-DE algorithm. But it would result in that the image noise affects the disparity quality. Therefore, the cost diffusion, temporal cost and motion detection are also improved in the HW-DE algorithm for keeping disparity quality and lower hardware cost.

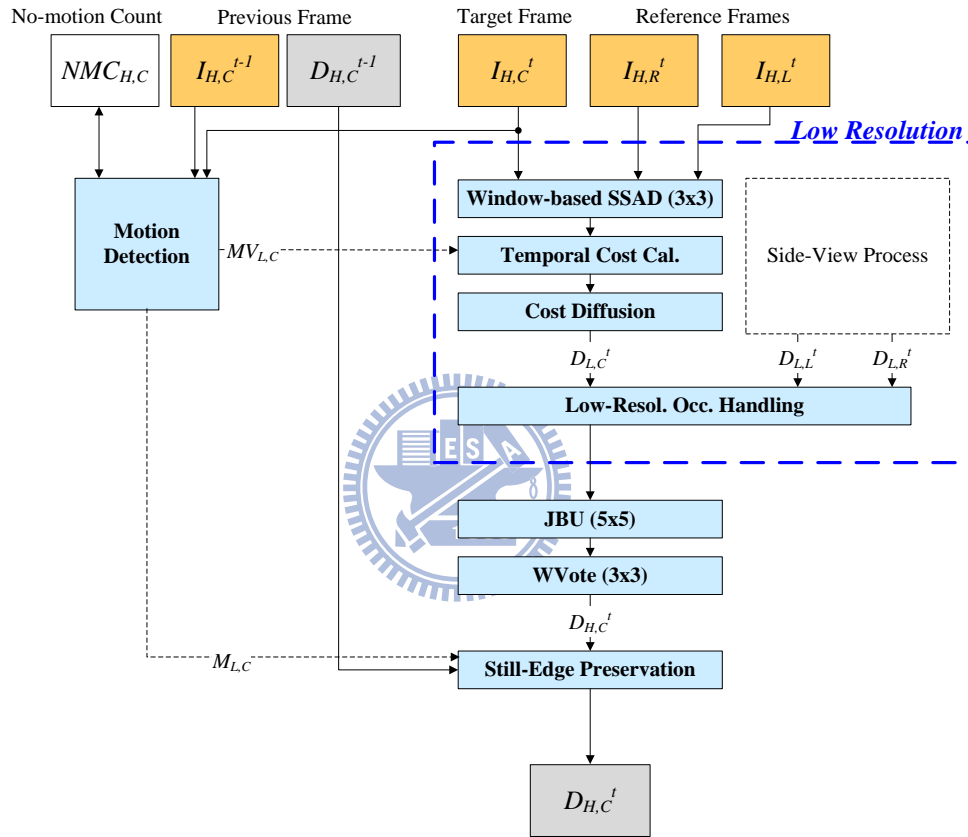


Figure VI-4 Flow of the proposed HW-DE algorithm

1. Improved Cost Diffusion

The original vertical cost in (IV-23) strongly propagates previous row disparity to current costs if their corresponding pixels are consistent. However, the pixel consistency would be not accurate because of the image noise. Therefore, we should modify the scaling term λ_{vert} and disparity difference term for the vertical cost to decrease the dependency on the pixel consistency. The main idea of new

defined vertical cost is to introduce the Potts model into the scaling term and disparity difference, so that the new vertical cost is defined as

$$C_{vert}(x, y, d) = \lambda_{vert}(\Delta I_L^t) \times \min\{|d - D_L^t(x, y - 1)|, \tau_{vert}\} , \quad (VI-1)$$

where τ_{vert} is for truncating the disparity difference, λ_{vert} is for scaling the cost value according to the color distance of $I_L^t(x, y)$ and $I_L^t(x, y-1)$. The value of scaling function λ_{vert} should be increased while the color distance is decreased. Thus, the scaling function λ_{vert} is defined as

$$\lambda_{vert}(\Delta I_L^t) = \lambda_{vert,max} - \lambda_{vert,slope} \min\{\Delta I_L^t, \gamma_{vert}\} . \quad (VI-2)$$

where γ_{vert} and $\lambda_{vert,slope}$ are for truncation and scale in the Potts model. This new vertical cost could tolerate the inaccurate pixel consistency, because its value is adaptive with the color consistency ΔI_L^t . In addition, the disparity candidates far from previous row would not suffer from too much penalty by the truncation term τ_{vert} .

2. Improved Motion Detection

Figure VI-5 shows the simplified motion detection for the HW-DE algorithm. Compared to the original motion detection in Figure IV-34, the motion calculation steps are replaced by the motion value calculation, and the dilation and erosion steps are removed. Without the separate pixel-based and block-based motion calculation, the simplified motion detection directly computes the motion value to decide the motion map, and passes the motion value to the temporal cost calculation. The motion value for the low-resolution pixel at (x, y) is defined as

$$MV_L^t(x, y) = \frac{1}{3 \times 3} \sum_{(u,v) \in S} |I_H^t(u, v) - I_H^{t-1}(u, v)| , \quad (VI-3)$$

where S is a 3×3 window centered for $(2x, 4y)$ in the high-resolution frames. With the motion value MV_L^t and the old no-motion count NMC_L^{t-1} , the motion decision step determines the motion flag M_L^t by

$$M_L^t(x, y) = \begin{cases} 1 & \text{if } MV_L^t(x, y) > \tau_M \text{ or } NMC_L^{t-1}(x, y) < \tau_{NMC} \\ 0 & \text{else} \end{cases} . \quad (VI-4)$$

In addition, the no-motion registration step also updates the old no-motion count NMC_L^{t-1} to the new one NMC_L^t by

$$NMC_L^t(x, y) = \begin{cases} NMC_L^{t-1}(x, y) + 1 & \text{if } MV_L^t(x, y) \leq \tau_M. \\ 0 & \text{else} \end{cases} \quad (\text{VI-5})$$

Note that the edge information is not necessary in the improved motion detection. Therefore, the edge map would not result in the high memory cost for long pipelining stall.

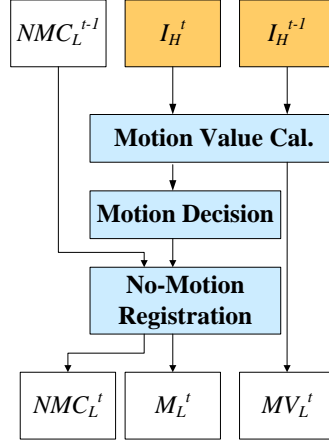


Figure VI-5 Proposed motion detection in the HW-DE algorithm

3. Improved Temporal Cost

With the simplified motion detection, the outputted binary motion map M_L^t could well support the still-edge preservation but the temporal cost calculation due to the performance of motion map is affected by the image noise. Thus, using the same method in the modification of vertical cost, we modify the original temporal cost calculation from (IV-17) to

$$C_{temp}(x, y, d) = \lambda_{temp}(MV_L^t(x, y)) \times \min\{|d - D_L^{t-1}(x, y)|, \tau_{temp}\} \quad , \quad (\text{VI-6})$$

where λ_{temp} is changed from a constant to the function of motion value MV_L^t . The function is λ_{temp} defined as

$$\lambda_{temp}(MV) = \lambda_{temp,max} - \lambda_{temp,slope} \min\{MV, \gamma_{temp}\}. \quad (\text{VI-7})$$

With the improvement, the temporal cost is adapted according to the motion value, instead of only the binary motion map contaminated by image noise. The lower motion value the more impact from previous disparity.

The proposed HW-DE algorithm could reduce most of the memory cost that results from the long pipelining stall in the edge detection, the filter-based processes, and the motion detection. In addition,

the proposed improved vertical cost and temporal cost could tolerate the image noise, and keep the disparity quality without significant degradation. The proposed algorithm is implemented by VLSI design in the following sections, and its disparity quality is evaluated in Section 6.5.

6.2 Overview of Disparity Estimation Engine

The major design challenges of the disparity estimation engine have been addressed in the algorithm level. In the architectural design level, how to meet the target throughput with less hardware cost is the main task. To achieve the task, this section presents the proposed high-throughput architecture and the initial schedule for the computational circuits that could meet the target throughput.

6.2.1 Proposed Three-Stage Pipelining Architecture

Figure VI-6 shows the proposed architecture of disparity estimation engine and the associated peripheral resource. The proposed architecture consists of the main core and the I/O interface. The I/O interface accessed the required and resultant data from the external memory through a 128-bit bus, and the main core uses the fetched data to calculate the disparity maps.

According to the computational characteristics in the proposed disparity estimation algorithm, we propose the three-stage pipelining architecture for the main core. The first low-resolution disparity estimation stage processes in low resolution frame, and produces the initial disparity maps and motion information for the following stages. Then, the second stage deals with the occlusion problem in different processing directions for the three views. Finally, the high-resolution disparity estimation stage upsamples and refines the disparity maps in high resolution frame. Note that the pipelining stages are row-based buffers, and the buffers between the second and third stages are the external memory to decrease the internal memory cost.

On the other hand, in the I/O interface, the memory access controller serves all the requests from main core to access the data in the external memory. To decrease the idle time of main core, we

propose an efficient memory access schedule for the memory access controller, and the corresponding data configuration for the external memory in Section 6.4.

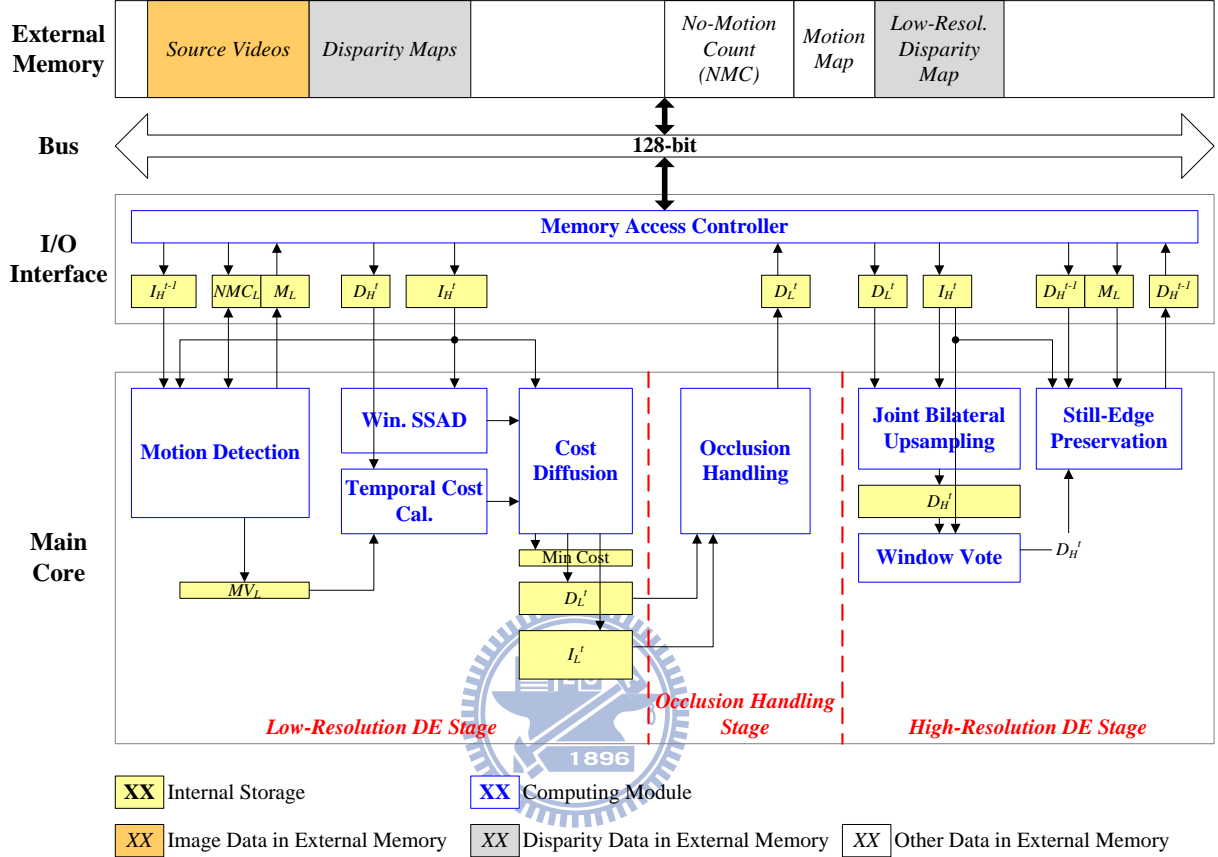


Figure VI-6 Overview architecture of the proposed disparity estimation engine

6.2.2 Schedule of Main Core

We assume the proposed disparity estimation engine could calculate one disparity for three views in one cycle. This engine can achieve the throughput of 60 frames/s for three view HD0180p disparity maps if the main core can work at the higher frequency than 125MHz. With this assumed throughput, we propose the computational schedule of main core for calculating one disparity frame in Figure VI-7. Note that the three view disparity maps are simultaneously produced. In this schedule, the computation of one disparity frame requires 1920×1080 cycles, and a schedule tile has 1920×4 cycles for four disparity rows. In a schedule tile, the former two stages produce one low-resolution disparity row, and last stage uses it to further produce the corresponding four high-resolution disparity rows.

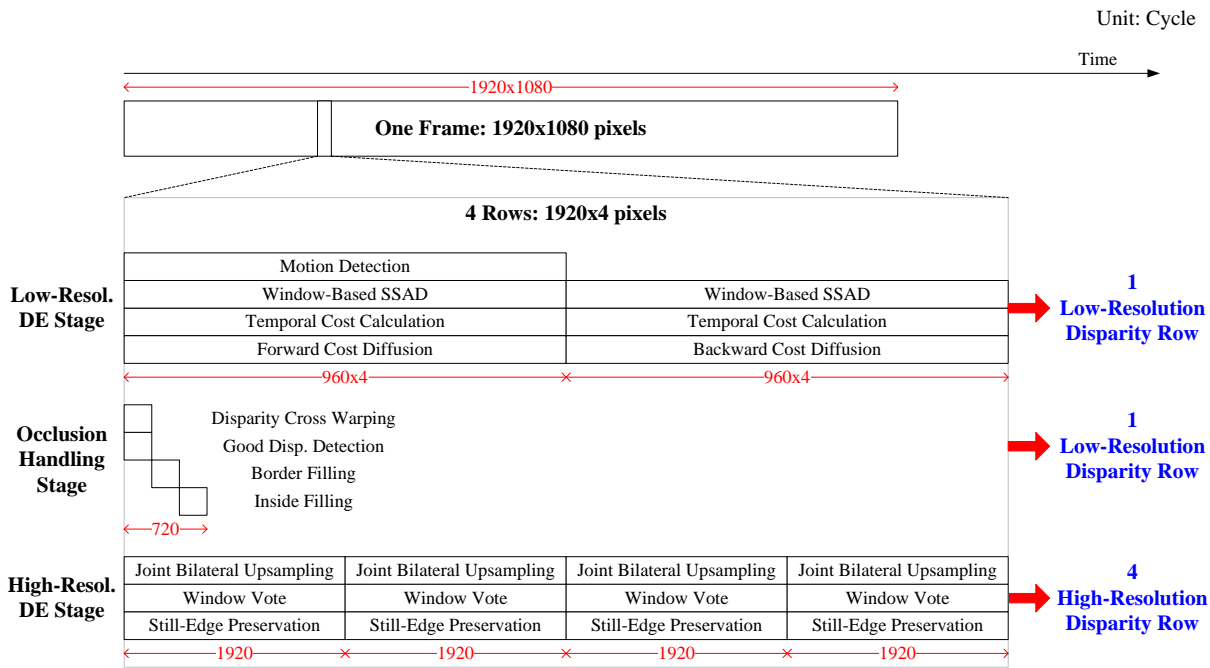


Figure VI-7 Proposed computational schedule for main core

In the low-resolution disparity estimation stage, the schedule is dominated by the forward and backward cost diffusion. To cooperate with them, the required data should be calculated twice or once with a data reuse technique. Considering the memory cost into the other computation, the motion detection applies the once calculation with data reuse technique, and the window-based SSAD and temporal cost calculation applies the twice calculation. Therefore, for the matching cost-related calculation, we have 4 cycles to compute the costs of a pixel with full disparity range, and the throughput of this stage would be 1/8 pixels/cycle. For the occlusion handling stage, we do not spread their calculation to the whole slot because of no heavy computation. For the final high-resolution disparity estimation stage, the required throughput is 1 pixel/cycle to meet the target performance.

Based on the computational schedule of main core, we could further design the architecture of each computational module according to the above mentioned throughput. Note that the computational schedule will be modified by considering the external memory access in Section 6.4.

6.3 Detailed Architectural Design

In this section, we describe the details of computational modules in each pipelining stage by the pipelining stage scope and the module scope. The pipelining stage scope focuses on the data flow among modules and the internal memory configuration, while the module scope focuses on the computational logic.

6.3.1 Low-Resolution Disparity Estimation Stage

Figure VI-8 shows the architecture of low-resolution disparity estimation stage. In which, all the computational modules has three parallel PEs for three target views. The data from the external memory are buffered in groups of registers to support the wide data access of main core. By the schedule in Figure VI-7, the computation of this stage consists of the forward process and the backward process. In the forward process, the motion detection module finishes all tasks, and stores the motion value into the internal memory `lo_mval` for data reuse in the next process. In addition, the updated no-motion count and motion flag are written to the external memory for the last pipelining stage. At the same time, the modules from window-based SSAD to the horizontal diffusion are performed in one frame row from left to right in the forward process. Their temporary minimal cost and disparity rows are stored in the memory `lo_min_cost` and `lo_cur_disp`. Then, in the backward process, they are performed in the opposite direction using the temporary data, and reuse the motion values of the internal memory `lo_mval`. The produced disparity and downsampled image rows are placed in the internal memory `lo_cur_disp` and `lo_cur_img` for the next pipelining stage.

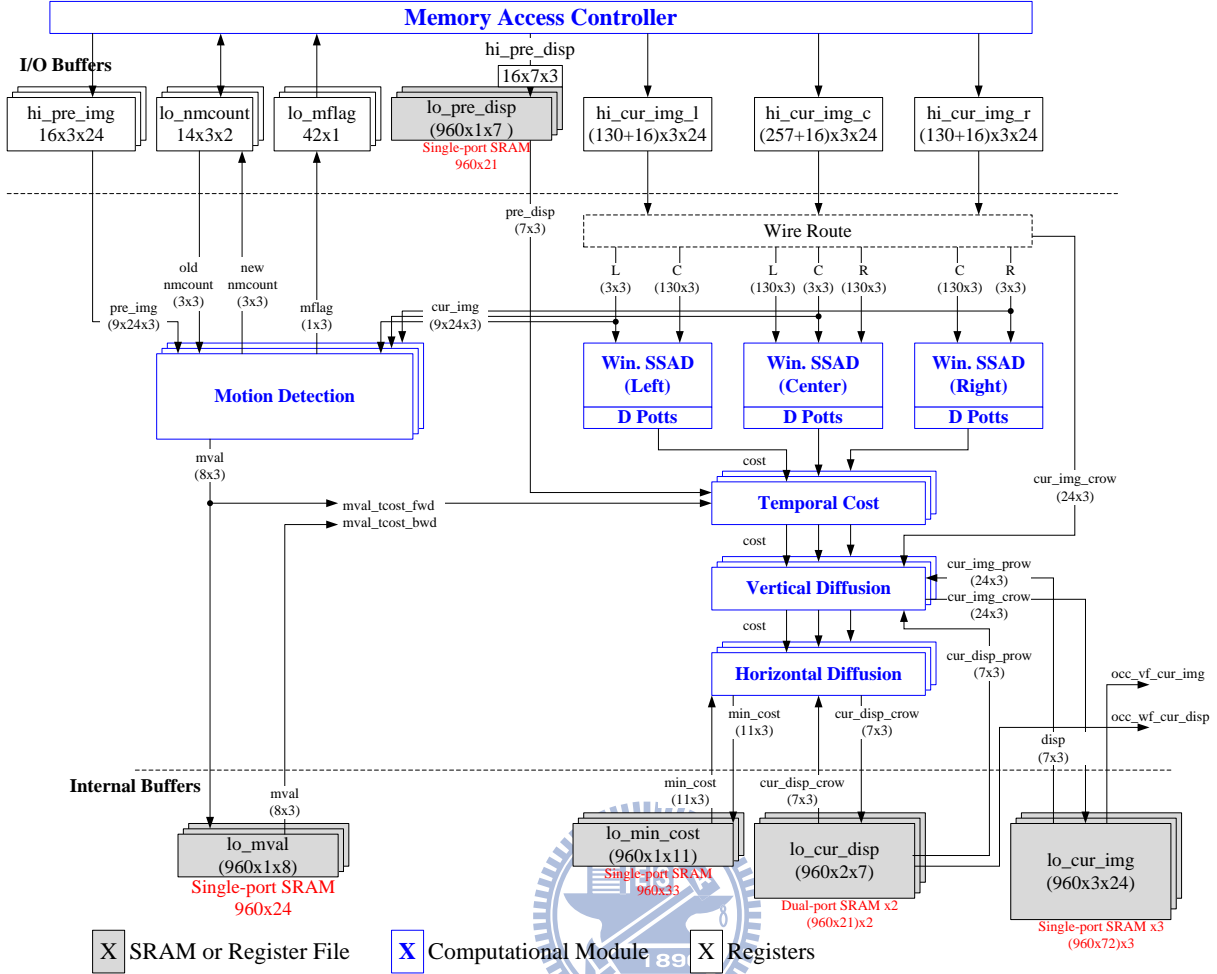


Figure VI-8 Architecture of the low-resolution disparity estimation stage

The architecture of the computational modules is presented as follows.

1. Motion Detection

Figure VI-9 illustrates the input and output data in the frame coordinate system for the motion detection module, and Figure VI-10 shows the architecture of motion detection module. In Figure VI-9, the motion detection module uses the two 3×3 windows from the current frame I_H^t and previous frame I_H^{t-1} to compute the motion value MV_L^t and motion flag M_L^t . In addition, the old no-motion count NMC_L^{t-1} is updated and used to extend motion map for the foreground copy artifact. The architecture of motion detection module is directly implemented according to (VI-3), (VI-4), and (VI-5). However, a divider is required for normalization in (VI-3). To remove the divider, all the associated values are

multiplied by 9. In addition, the pixel difference in (VI-3) adopts the Manhattan color distance for low hardware cost. Note that the truncation of MV_L^t in the temporal cost calculation is pre-performed here to reduce the memory cost of motion value.

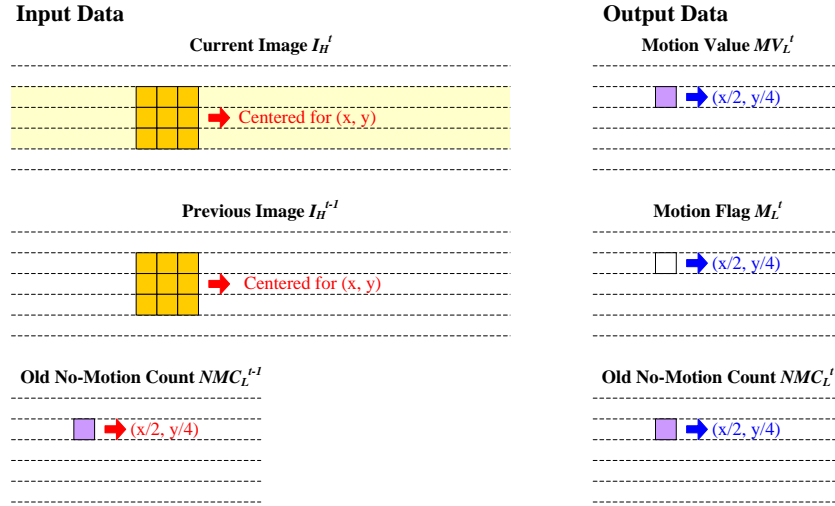


Figure VI-9 Data access of the motion detection module in the frame coordinate system

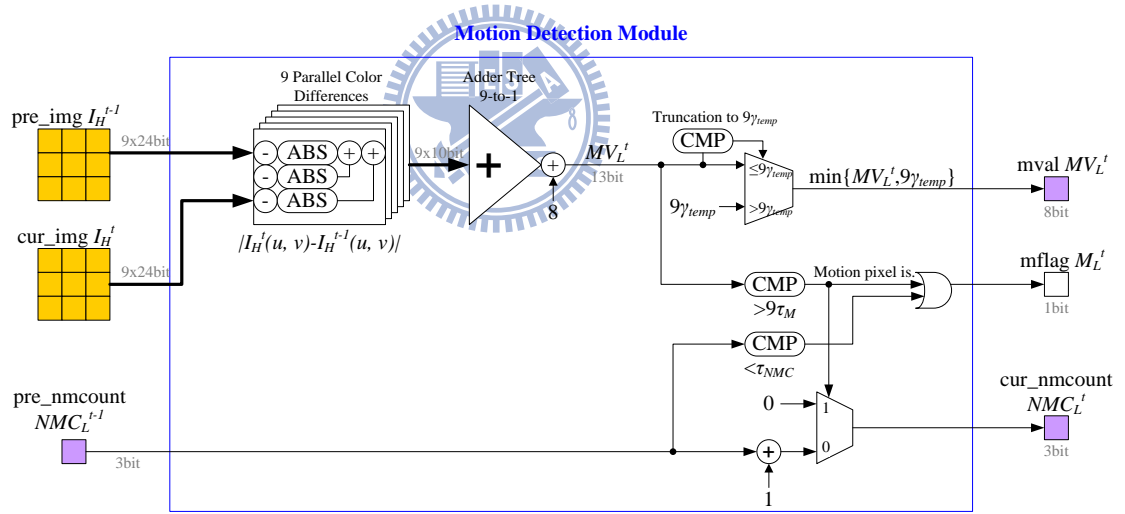


Figure VI-10 Architecture of the motion detection module

2. Matching Cost Calculation

For each target view, the original matching cost calculation uses the other two views as the reference frames. In the disparity estimation engine, we simplify the side-view matching cost calculation only using the center-view as reference frame to reduce the hardware cost. Figure VI-11 shows the input data and required data for computing the full matching costs of one pixel. For example of the left-view matching cost calculation, the required data contains the 3×3 block of left-view input

data (i.e. block No. 2), and the $(DR+2) \times 3$ block of center-view input data (i.e. block No. 4 and 5). The required data for the center-view and right-view matching cost calculation are also illustrated.

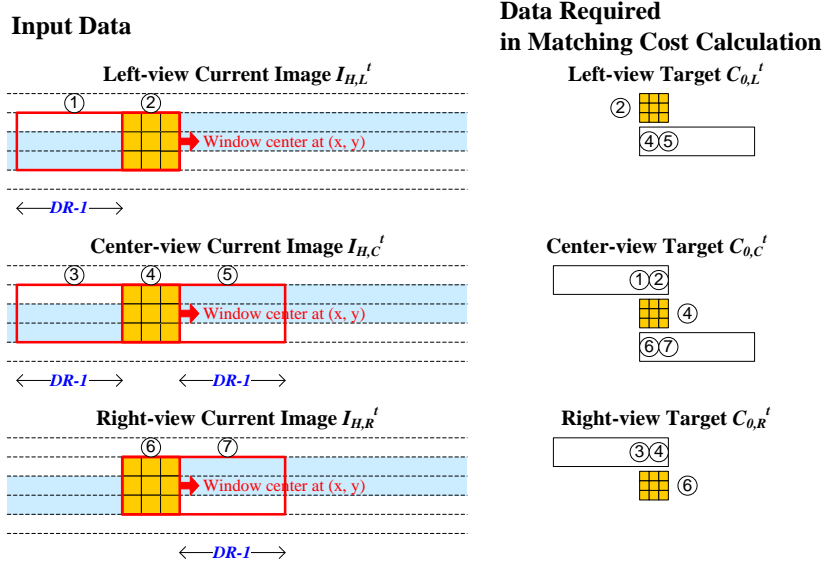


Figure VI-11 Input and required data in matching cost calculation for three target views

As mentioned in Section 6.2.2, the throughput of this module should be $DR/4$ costs/cycle, and it is designed with the parallelism factor of 32 for the disparity range of 128. Figure VI-12 shows the proposed architecture of this module for three views. Note that the center-view has double 32-parallel-SAD PEs because of two reference views. For each disparity, the minimum matching cost is selected. Thus, the three-view matching costs are defined as

$$C_{0,C}^t(x, y, d) = \min\{SSAD_{C-L}(x, y, +d), SSAD_{C-R}(x, y, -d)\} \quad (VI-8)$$

$$C_{0,L}^t(x, y, d) = SSAD_{L-C}(x, y, -d) \quad (VI-9)$$

$$C_{0,R}^t(x, y, d) = SSAD_{R-C}(x, y, +d) \quad (VI-10)$$

where the window-based SSAD is calculated by

$$SSAD_{tar-ref}(x, y, d) = \sum_{(u,v) \in S} \|I_{H,tar}^t(u, v) - I_{H,ref}^t(u + d, v)\| \quad (VI-11)$$

In which, S is a 3×3 window centered for (x, y) , and the Manhattan difference is adopted for the color difference. The initial matching costs are substituted into the DPotts model by

$$C_D^t(x, y, d) = \lambda_D \min\{C_0^t(x, y, d), \tau_D\} \quad (VI-12)$$

Finally, the produced C_D^t is added with the temporal cost C_{temp}^t and the vertical cost C_{vert}^t for the horizontal cost diffusion to calculate the disparity maps. They are summed up by

$$C_{total}^t(x, y, d) = C_D^t(x, y, d) + C_{temp}^t(x, y, d) + C_{vert}^t(x, y, d) \quad (VI-13)$$

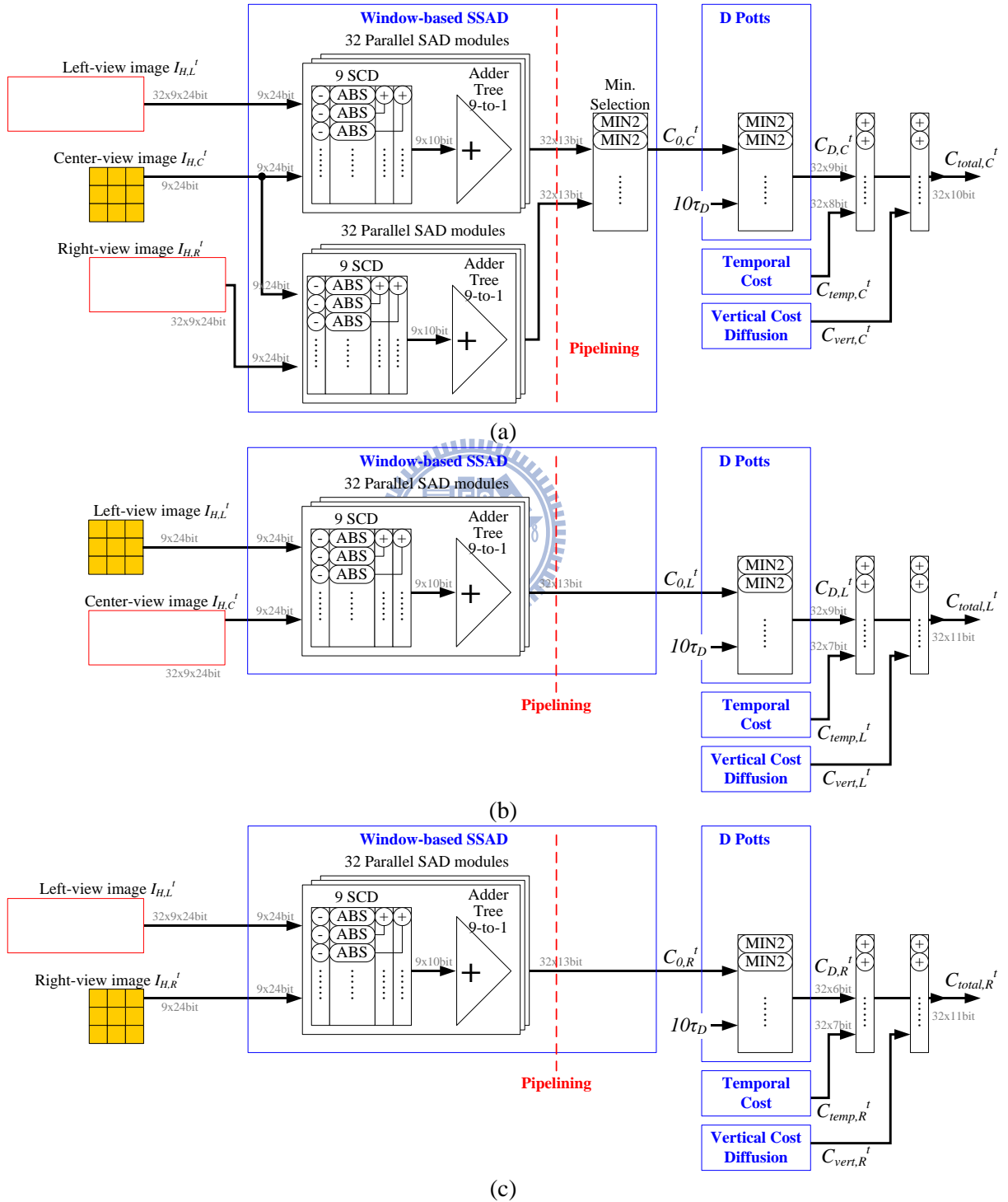


Figure VI-12 Architecture of the window-based SSAD and DPotts modules

(a) center view, (b) left view, (c) right view

With the above equations, the window-based SSAD and the DPotts modules could be directly implemented. The two modules calculate the matching costs pixel by pixel from left to right in the forward cost diffusion, and the opposite direction in the backward cost diffusion. Note that the required images are loaded twice for the two direction steps, and the sliding image buffers are applied to reduce the external memory access.

3. Temporal Cost Calculation and Vertical Cost Diffusion

The temporal cost calculation is to generate the full temporal costs C_{temp}^t using the previous disparity $D_H^{t-1}(x, y)$ and the motion value $MV_L^t(x, y)$ for four iterations. Figure VI-13 shows the architecture of temporal cost calculation module that is implemented according to (VI-6) and (VI-7). The 32 parallel disparity differences of $|d - D_H^{t-1}(x, y)|$ is implemented by look-up table to reduce 32 subtractors, and the function λ_{temp} is also implemented by look-up table to fit its curve. The truncation and the multiplication in (VI-6) can be simplified to adders and shifters as shown in Figure VI-13 (b).

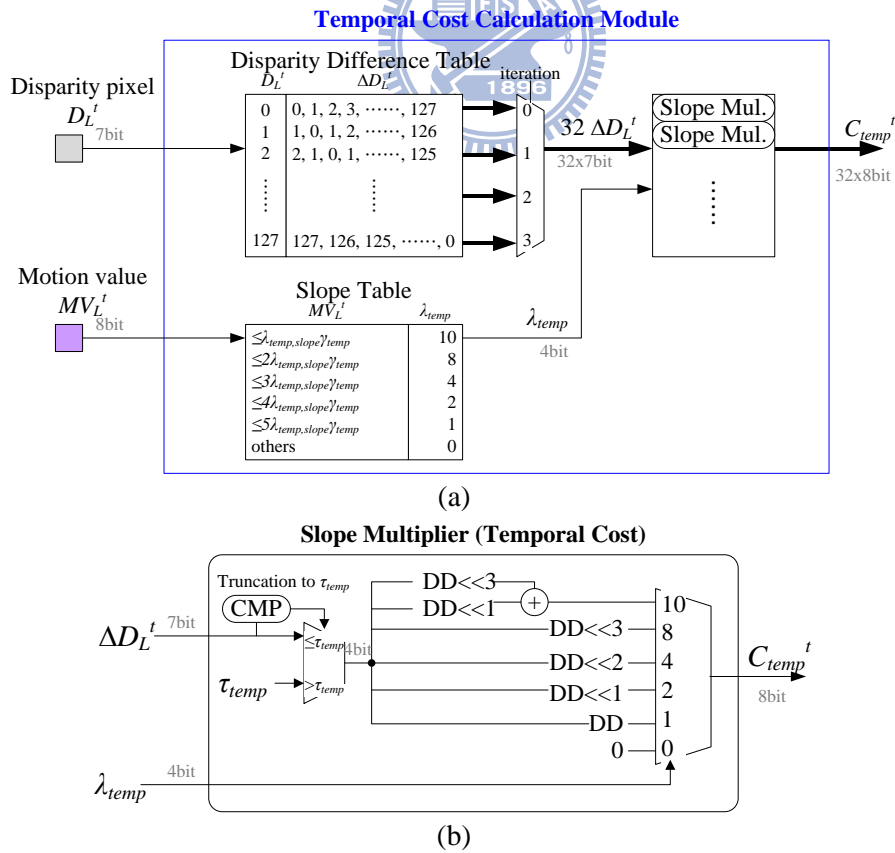


Figure VI-13 Architecture of the temporal cost calculation module

With the same simplification architecture, Figure VI-14 shows the architecture of vertical cost diffusion module that calculates the full vertical costs C_{vert} for four iterations. Except for the similar architecture of disparity difference table, slope table, and slope multipliers, the vertical cost diffusion module needs to compute the pixel difference ΔL^t by the Manhattan color distance. The computation of pixel difference would result in critical path in this architecture. Thus, we install a pipelining stage as shown in Figure VI-14 (a).

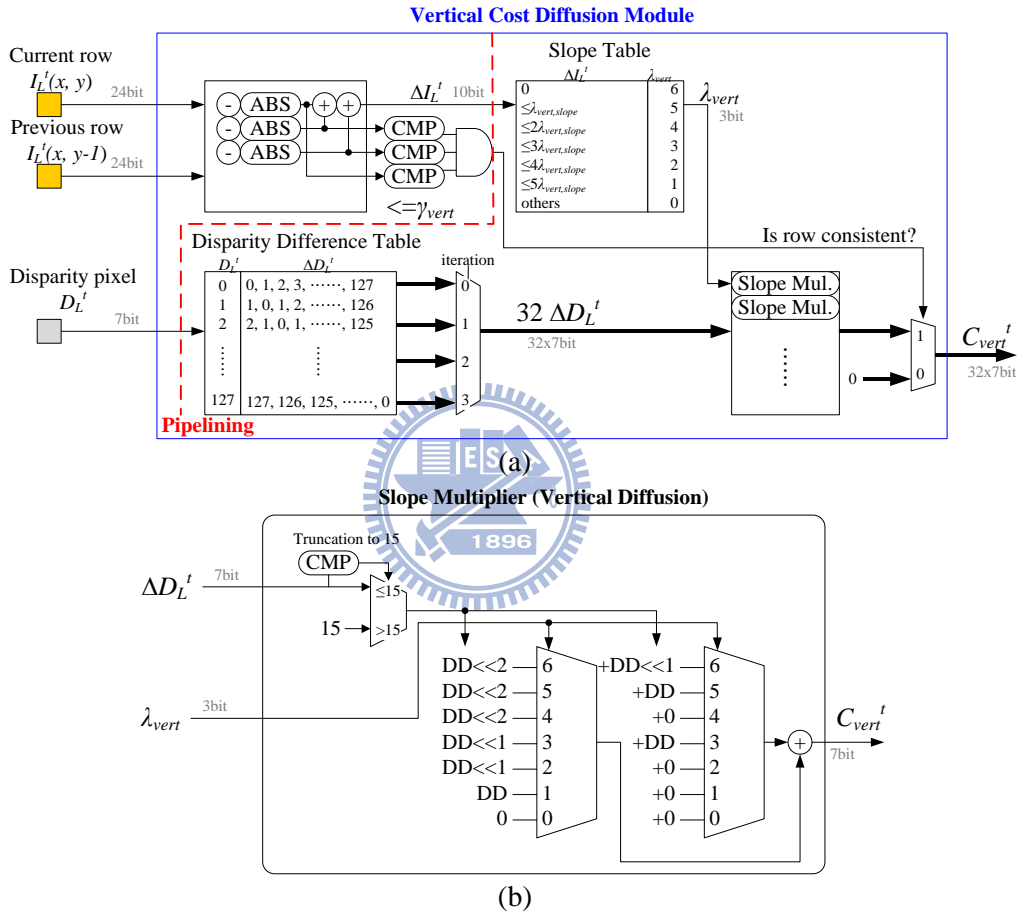


Figure VI-14 Architecture of vertical cost diffusion module

4. Horizontal Cost Diffusion

For the horizontal cost diffusion, we first propose a fully parallel architecture with the highest throughput to analyze its computational characteristics, and then simplify the initial architecture to just meet our target throughput with less hardware cost. Figure VI-15 shows the fully parallel architecture that contains the convolution, normalization, addition, and winner-take-all (WTA) corresponding to (IV-19) to (IV-21). The straightforward architecture can achieve the highest throughput of one best

disparity RD_{best} and one minimum cost RC_{min} per cycle. Note that the labels of data are for the right cost diffusion as an example. The left cost diffusion can also apply this module directly.

In this fully parallel architecture, we adopt the parallel architecture proposed in [33] to the convolution PE. The hardware cost in the convolution PE depends on the truncation term τ_V in the smoothness term V in (IV-19). To reduce the hardware cost of this PE, we change τ_V from the original 15 to 5 that could reduce the number of parallel adders from 3,728 to 1,338 but suffers from slight disparity quality change as demonstrated in Section 6.5. On the other hand, in the normalization PE, we change the normalization term κ in (IV-19) from the average of diffusion costs to the minimum to avoid a high data-width adder tree for the average computation. Finally the addition and WTA are directly implemented according to (IV-18) and (IV-20), (IV-21), respectively.

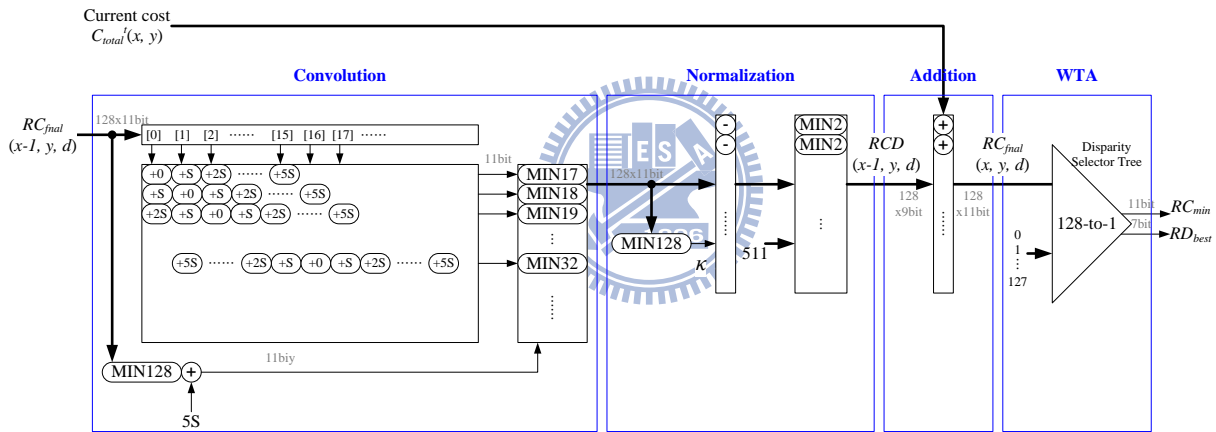


Figure VI-15 Fully parallel architecture of the horizontal cost diffusion module

However, the fully parallel architecture suffers from the two design problems: long critical path and feedback data path, as shown in Figure VI-16 (a). The pipelining approach could only solve the long critical path problems but it would violate the original functionality due to the feedback data path. Thus, we propose the sequential architecture in Figure VI-16 (b) to simplify the fully parallel architecture and meet our target throughput. In the sequential architecture, the four steps in the horizontal cost diffusion are sequentially performed. The advantage of this architecture is that the four steps could share the same registers and PE to further reduce the hardware cost.

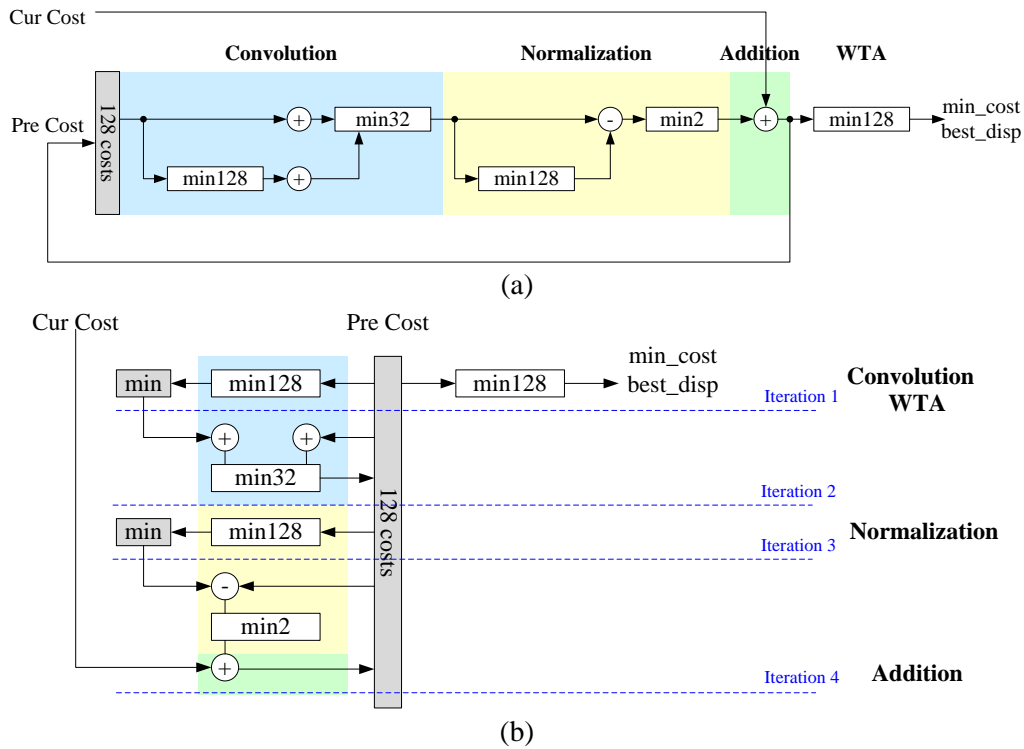


Figure VI-16 Architecture of the horizontal cost diffusion module
 (a) fully parallel architecture, (b) proposed sequential architecture

6.3.2 Occlusion Handling Stage

Figure VI-17 shows the proposed architecture for the occlusion handling pipelining stage. In which, the disparity cross warping and the good disparity detection modules are performed in the first iteration, and then the border filling and the inside filling modules are performed in the second and third iterations. By the schedule in Figure VI-7, the disparity cross warping module progressively fetches a disparity row from the previous pipelining stage. Then, it detects the occlusion map and initially fills the partial occlusion pixels. At the same time, the good disparity detection module receives the results of disparity cross warping module to find the good disparity map. The produced disparity, occlusion, and good disparity rows are stored in the internal memories *lo_warp_disp*, *lo_occ*, *lo_good* for the following two iterations.

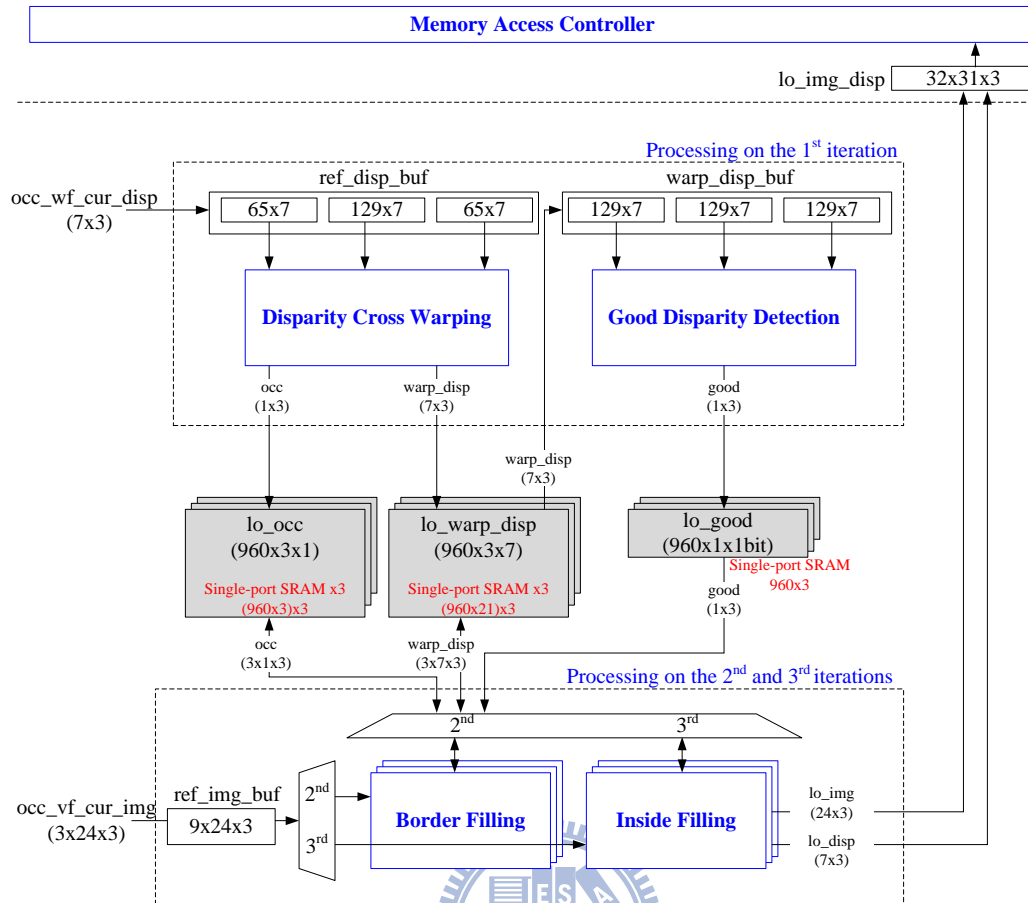


Figure VI-17 Architecture of the occlusion handling stage

In the second and third iterations, the border filling module and the inside filling module fix the rest of occlusion by the window vote method. The required image data in the window vote are loaded from the shared image buffers in the previous pipelining stage. With the required image data and the other data in internal memories, the recovered disparity row is produced in the raster-scan order and written to the external memory through the memory access controller. The detailed architecture of the computational modules is presented as follows.

1. Disparity Cross Warping

Figure VI-18 shows the proposed architecture for the disparity cross warping. The computation of this module is corresponding to the two steps of left-right check (LRC) and disparity cross warping in Figure IV-33. It produces the disparity and occlusion rows pixel by pixel in the direction from left to right. In this module, the input disparity from previous stage is pushed into the FIFO registers. With

the buffered disparities, the occlusion detection PEs perform the LRC process to generate the occlusion label O_L , O_C , O_R for three views. Figure VI-19 (a) shows the architecture of the occlusion detection PE that compares the target disparity and the corresponding disparity in reference view to determine occlusion label. Both the target disparity and its correspondence are in the FIFO registers.

Furthermore, the produced occlusion labels are shifted into the occlusion buffers for the warp filling PEs. According to the occlusion information, two warp filling PEs in each view warps the non-occluded disparities from the other two views to fill its occlusion pixels. Figure VI-19 (b) shows the architecture of the warp filling PE that uses the warped non-occlusion disparity to replace one of the original disparities to form the new disparities.

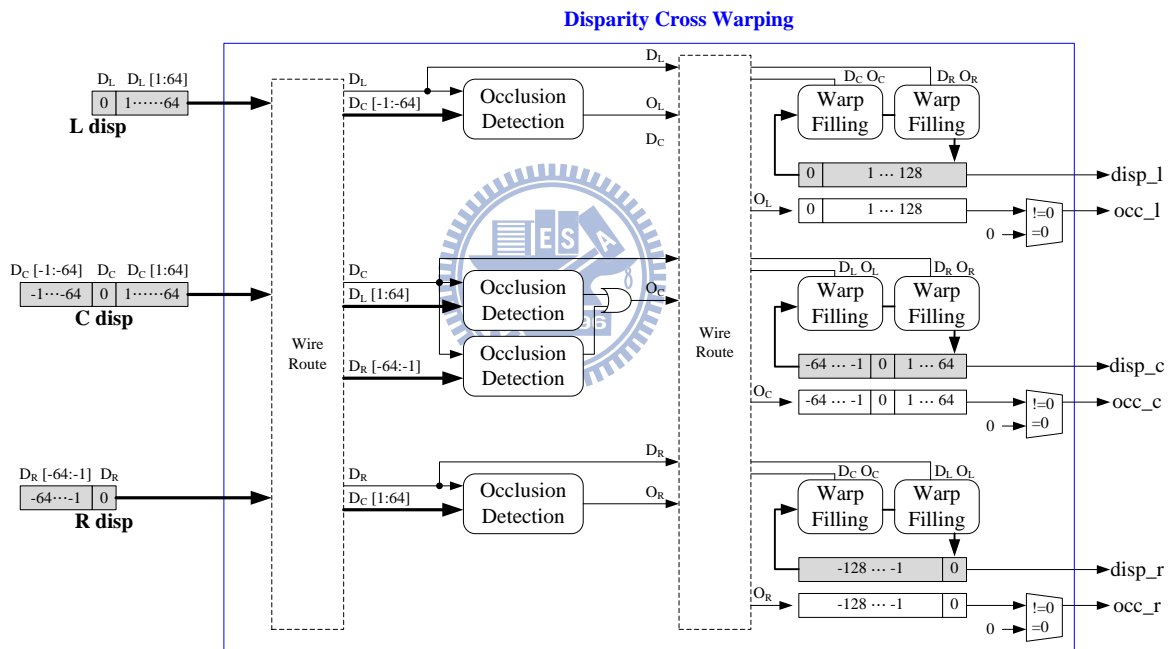


Figure VI-18 Architecture of the disparity cross warping module

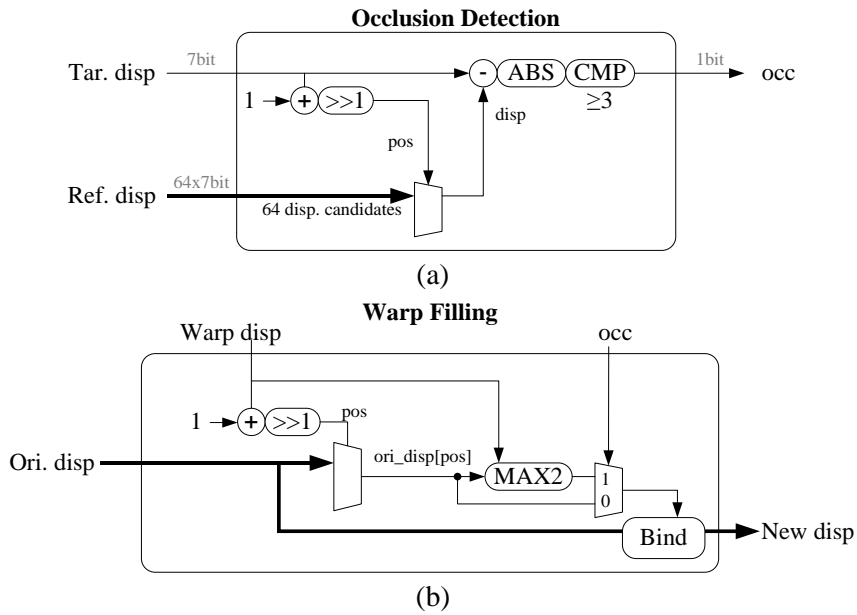


Figure VI-19 Architecture of the occlusion detection PE and the warp filling PE

2. Good Disparity Detection

With the disparity pixels generated by the disparity cross warping module, the good disparity detection module is to further find the reliable disparities for the next border and inside filling. Figure VI-20 shows the proposed architecture for the good disparity detection module. In which, the input disparities are shifted into the FIFO registers, and the double-LRC method is applied to find the good disparity. Unlike the single occlusion detection PE in the disparity cross warping module for side views, this module has two PEs because the good disparity should pass the stricter left-right check. Finally, the output good disparity labels are stored into the internal memory *lo_good* in Figure VI-17.

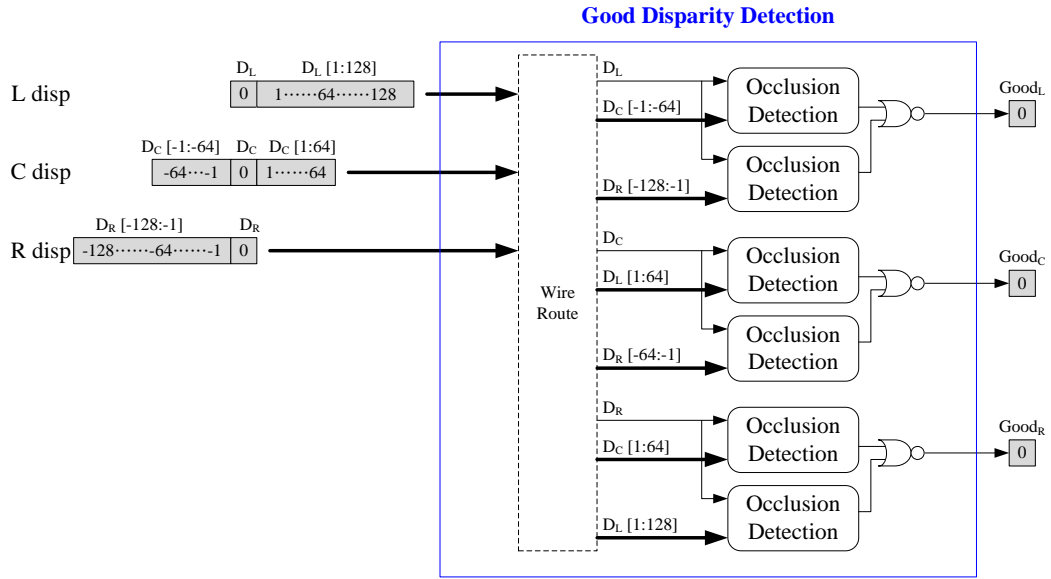


Figure VI-20 Architecture of the good disparity detection module

3. Border Filling and Inside Filling

The processing directions of occlusion filling are different in different views. For the left view, the border filling module processes a disparity row in the right-to-left direction, and the inside filling module processes the final disparity row in the left-to-right direction. For the right view, the two modules are performed in opposite directions to the left view. In addition, the processing direction of center view could follow the left view or right view.

Figure VI-21 (a) shows the architecture of border filling module that generates the new disparity for occlusion by the disparity vote PE according to the good disparity and occlusion maps. The disparity vote PE is identical to the window vote module in Figure VI-25. Its detailed architecture will be presented in the next stage. On the other hand, Figure VI-21 (b) and (c) shows the architecture of inside filling module for the center and side views. Their architecture is similar to the border filling module, and only the constraints for filling occlusion are different. After the processing of border filling and inside filling, the final low-resolution disparity rows are completed and written to the external memory for the last pipelining stage.

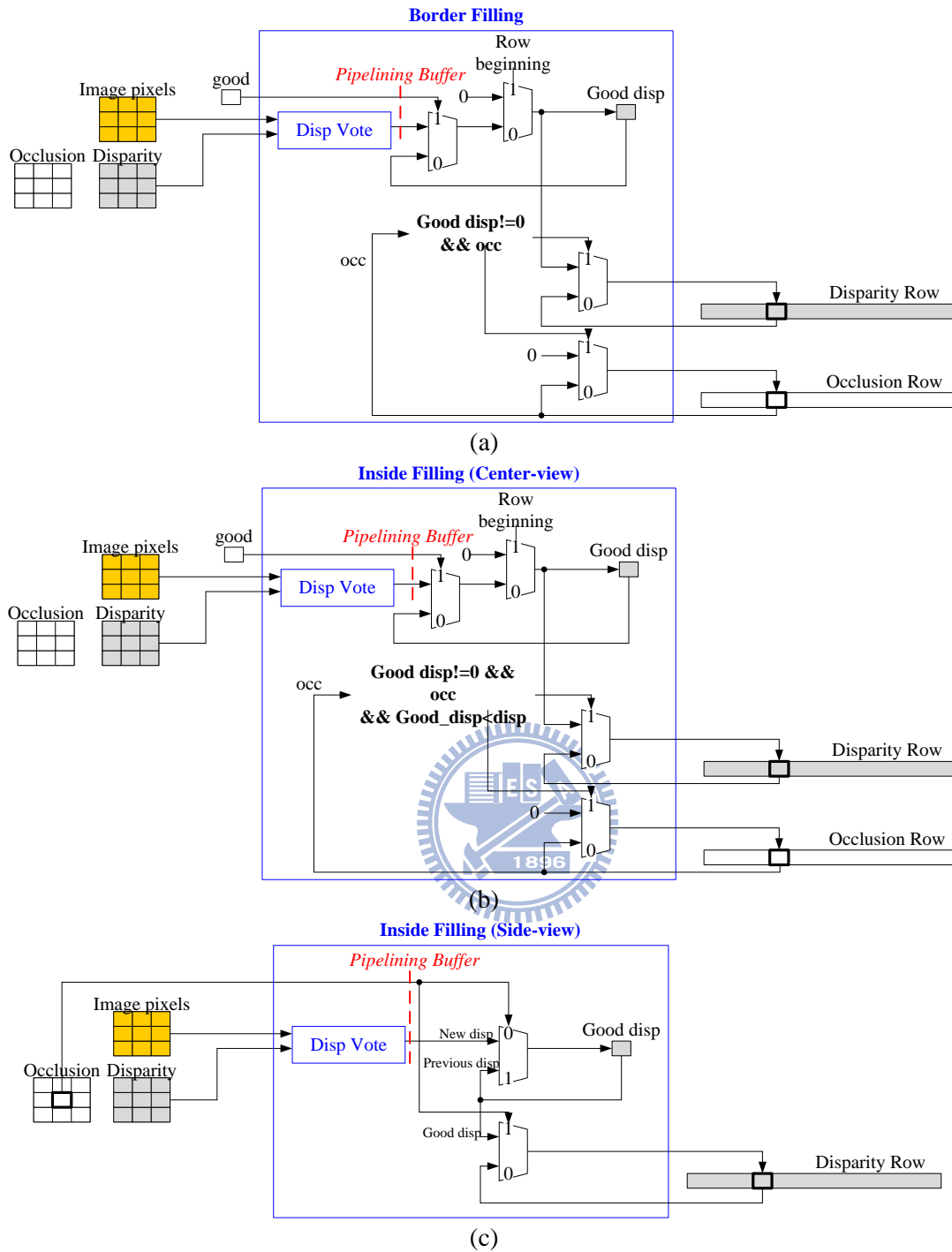


Figure VI-21 Architecture of border filling and inside filling modules

(a) border filling module, (b) inside filling module for center view, (c) inside filling module for side views

6.3.3 High-Resolution Disparity Estimation Stage

In the last pipelining stage, the low-resolution disparity maps are upsampled to the high-resolution ones by the joint bilateral upsampling, and refined by the window vote and the

still-edge preservation. Figure VI-22 shows the proposed architecture for this high-resolution disparity estimation pipelining stage, which consists of the data buffers and the three main modules. In which, the joint bilateral upsampling module fetches the guide high-resolution image and the low-resolution disparity from the external memory to calculate the high-resolution disparities. The new calculated disparities are stored into the internal memory hi_cur_disp for further processes.

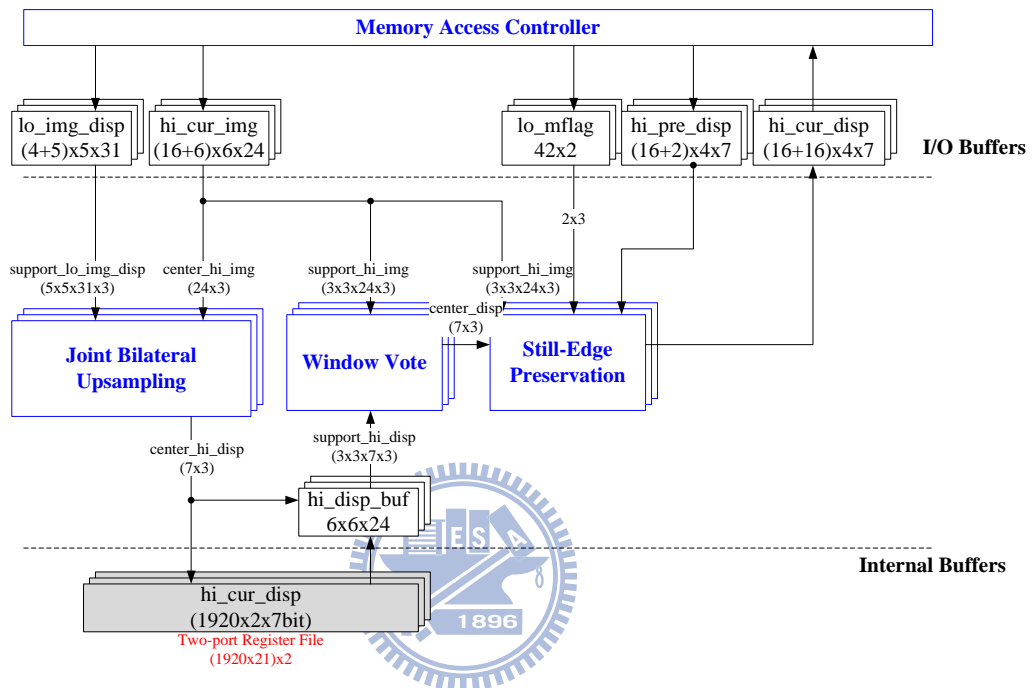


Figure VI-22 Architecture of the high-resolution disparity estimation stage

Then we consider the memory configuration in this stage as shown in Figure VI-23. The joint bilateral upsampling module could calculate 2×4 high-resolution disparities using 1 low-resolution disparity. At the same time, the window vote module refines the neighboring 2×4 disparities using the 4×6 disparities. Then, the still-edge preservation module masks the resultant 4×2 disparities for the temporal consistency. In the temporary output disparities, only the data between the joint bilateral upsampling and the window vote modules needs to be buffered, because the two modules are belong to filter-based process. To minimize the buffer size, we install a register file memory for the two disparity rows, and a 6×6 register array for the immediate accessed data. By the data configuration,

this pipelining stage could easily achieve the required throughput of 1 pixel/cycle with small internal memory.

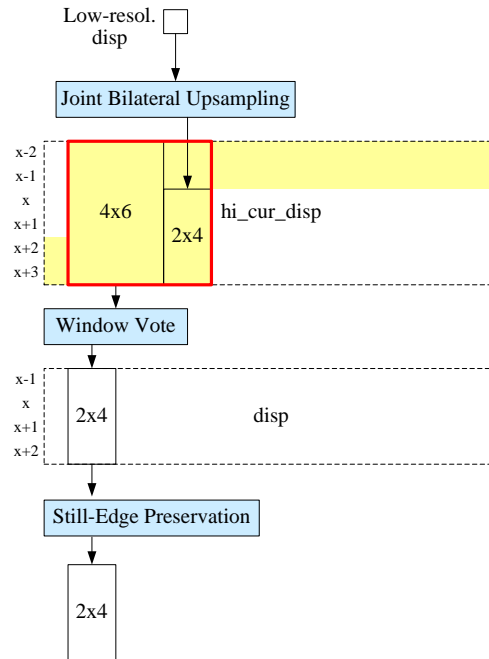


Figure VI-23 Memory configuration in the high-resolution disparity estimation stage

The detailed architecture of each module is introduced as follows.

1. Joint Bilateral Upsampling

The computation of the joint bilateral upsampling is defined in (IV-11). The architectural design approach proposed in Section 3.3 is not applied because that approach is helpful for large filter size, and the filter size in our algorithm is decreased to 5×5 . Thus, we adopt the straightforward architecture for the joint bilateral upsampling, and propose the low-hardware-cost multipliers for the range and spatial kernels. Figure VI-24 (a) shows the proposed architecture for the joint bilateral upsampling module, and Figure VI-24 (b) and (c) shows the proposed low-hardware-cost multipliers for the spatial and range kernels. In Figure VI-24 (a), a 5×5 low-resolution disparity window is used to compute one high-resolution disparity by considering a 5×5 low-resolution image window and a high-resolution pixel. For the computation in (IV-11), the product of the spatial kernel f and disparity is calculated by the proposed S_EXP multiplier in Figure VI-24 (b). The spatial kernel f is implemented by a S_EXP

table, and the multiplication is implemented by the adders and shifters. By the similar architecture, the proposed C_EXP multiplier in Figure VI-24 (c) is for the range kernel g and its multiplication. In addition, the summation and normalization in (IV-11) are implemented by adder-trees and the pipelined divider as shown in Figure VI-24 (a). Note that the pipelining stages are installed for the cut lines to break the critical paths.

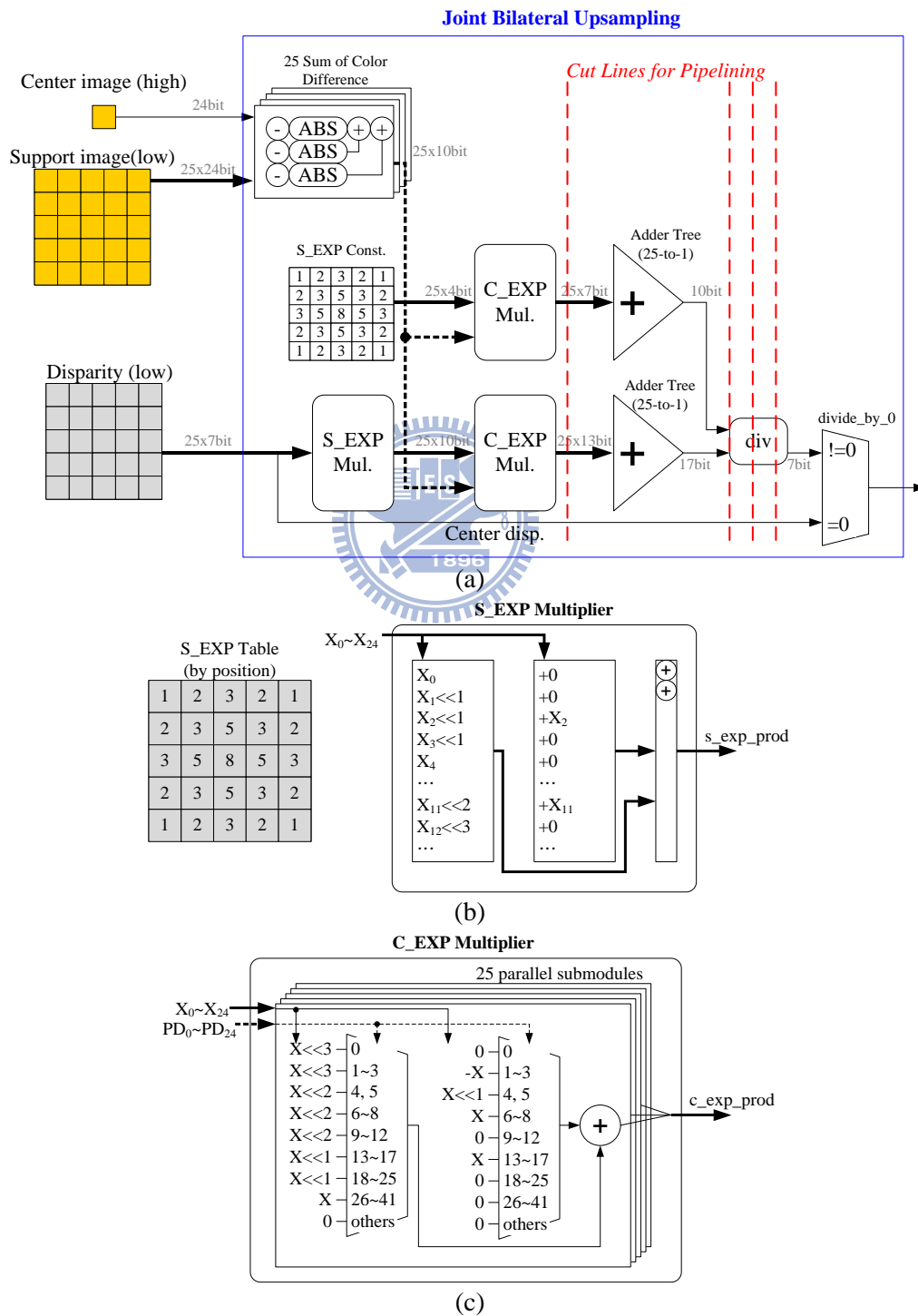


Figure VI-24 Architecture of the joint bilateral upsampling module

2. Window Vote

The window vote module is adopted in this stage for disparity refinement, and in the previous stage for occlusion filling. The computation of window vote method is defined in (IV-12) and (IV-13), and the corresponding architecture is shown in Figure VI-25. In which, the vote computation in (IV-13) is performed by the mask PE and 9 parallel vote PEs. Their architecture is shown in Figure VI-26. In the mask PE, each disparity in the 3×3 block is compared with other disparities. For the same disparity, the corresponding vote bit will be 1. Then, the vote bits for each target disparity are summed up by the vote PE. Finally, the disparity selection PE in Figure VI-25 chooses the disparity with the maximum vote as the resultant disparity. The resultant disparity is directly passed to the next still-edge preservation module.

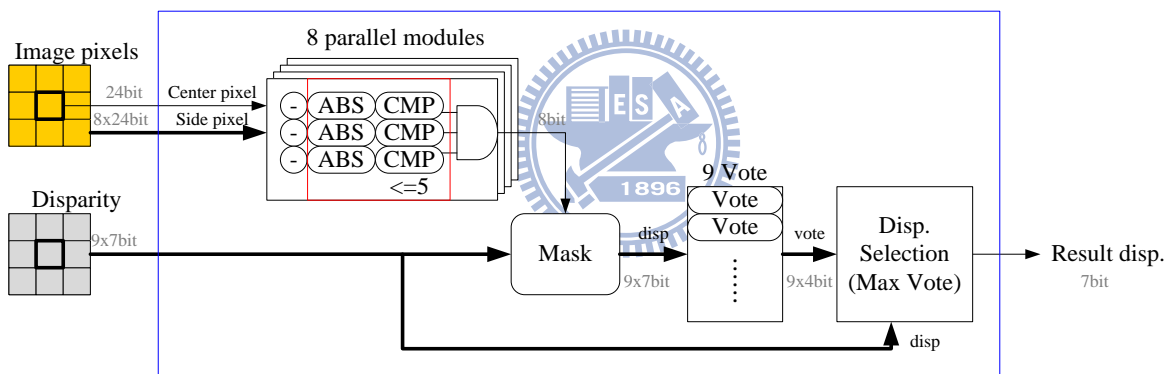


Figure VI-25 Architecture of the window vote module

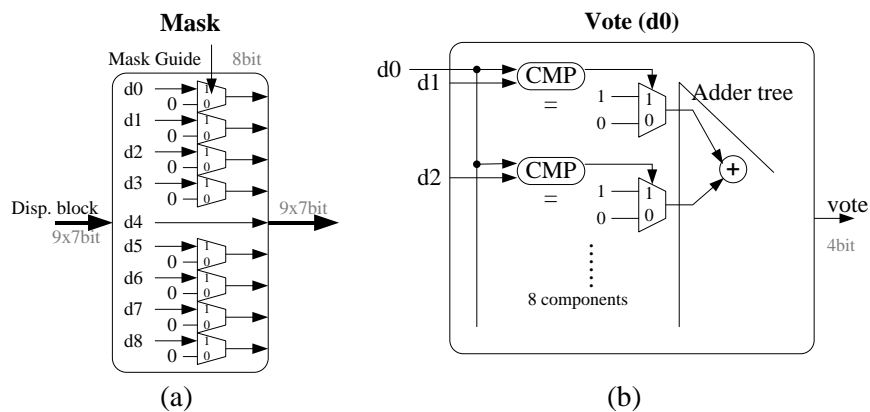


Figure VI-26 Architecture of the mask and vote PEs for the window vote module

3. Still-Edge Preservation

Figure VI-27 shows the architecture of still-edge preservation module, which replaces the current disparity with the previous disparity for the still-edge according to the edge and motion flag. The motion flag is fetched from the external memory, and the edge flag is computed by the Sobel filter in this module. The horizontal gradient g_x and vertical gradient g_y computed by the Sobel filter are used to decide the edge flag. If the pixel is no-motion and edge, the previous disparity would substitute the current disparity to be sent to external memory.

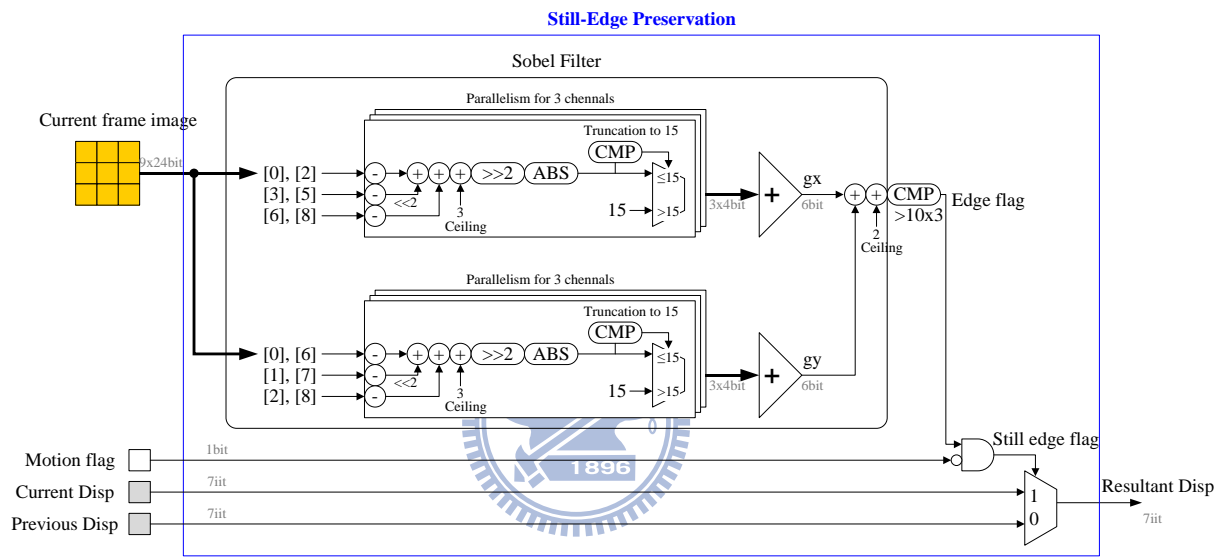


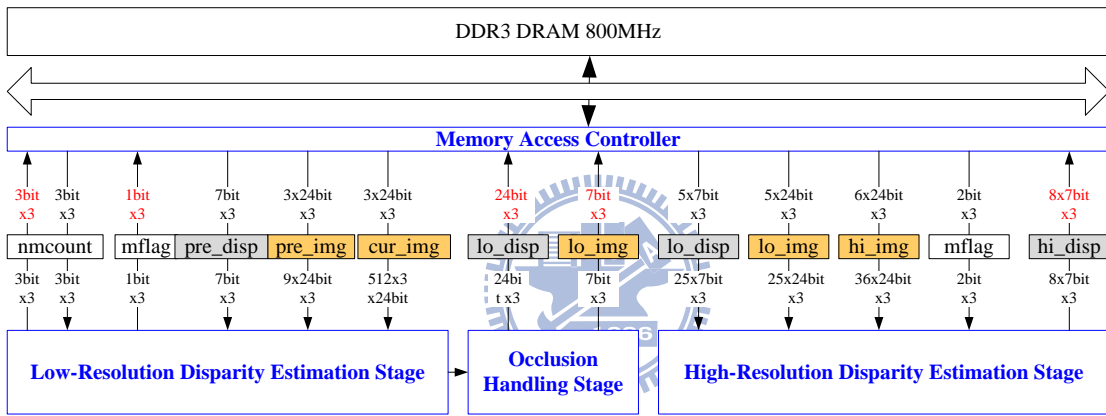
Figure VI-27 Architecture of the still-edge preservation module

6.4 External Memory Access

The computational modules are presented in previous section, and their throughputs are designed to fit the computational schedule in Figure VI-7. In this section, we additionally consider the external memory access of each module into the schedule. This section is organized as follows. First, we estimate the bandwidth requirement to determine the bus width. Then we present the external memory architecture and its data configuration. Finally, we describe the proposed external memory access schedule that could determine the required cycle count of our disparity estimation engine.

6.4.1 Bandwidth Requirement

First, we estimate the bandwidth requirement of the proposed disparity estimation engine. Figure VI-28 (a) shows the data width of access ports to the external memory. Corresponding to the previous computational schedule, Figure VI-28 shows an initial memory access schedule for computing 4 high-resolution disparity rows. In which, the peak of bandwidth usage would occur at the access of occlusion stage. For this peak interval, the estimated average bandwidth is estimated in Table VI-1. The total required bandwidth is 507 bits/cycle, and the budget bandwidth using 64-bit just satisfy the requirement. However, the average required bandwidth is an ideal value without considering the memory row miss. Thus, we choose the 128-bit for the system bus, and adopt the DDR3 SDRAM for the external memory.



(a)

		Unit: Cycle			
		← 1920	← 1920	← 1920	← 1920 →
Low-Resol. DE Stage	nmcount	3 data/4 cycles (R0)			
	nmcount	3 data/4 cycles (R0)			
	mflag	3 bits/4 cycles (R0)			
	pre_disp	3 disp/4 cycles (R0)			
	pre_img	18 pixels/4 cycles (R1, R0, R-1)			
	cur_img	18 pixels/4 cycles (R1, R0, R-1)			
Occ. Stage	lo_disp	3 disp/1 cycle (R-8)			
	lo_img	3 pixel/1 cycle (R-8)			
High-Resol. DE Stage	lo_disp	15 disp/8 cycles (R-12, R-16, R-20, R-24, R-28)			
	lo_img	15 pixels/8 cycles (R-12, R-16, R-20, R-24, R-28)			
	hi_img	36 pixels/8 cycles (R-17, R-18, R-19, R-20, R-21, R-22)			
	mflag	6 bits/8 cycles (R-16, R-20)			
	hi_disp	24 disp/8 cycles (R-18, R-19, R-20, R-21)			

(b)

Figure VI-28 Rough schedule for external memory access

(a) input and output ports to external memory, (b) rough external memory access schedule

Table VI-1 Estimated average external bandwidth for computing four disparity rows.

	Accessed Data	No. of Column	No. of Row	No. of View	Data Width (bit)	Iteration Count	Required Time (Cycle)	Bandwidth (bit/cycle)
Low-Resolution Disparity Estimation Stage	pre_disp	1	1	3	7	1	4	5
	pre_img	2	3	3	24	1	4	108
	cur_img	2	3	3	24	1	4	108
	mncount	1	1	3	3	2	4	5
	mflag	1	1	3	1	1	4	1
Occlusion Handling Stage	lo_disp	1	1	3	7	1	1	21
	lo_img	1	1	3	24	1	1	72
High-Resolution Disparity Estimation Stage	lo_img	1	5	3	24	1	8	45
	lo_disp	1	5	3	7	1	8	13
	hi_img	2	6	3	24	1	8	108
	mflag	1	2	3	1	1	8	1
	hi_disp	2	4	3	7	1	8	21
Total Required Bandwidth								507
Budget Bandwidth (DDR3 SDRAM 800MHz, 64-bit bus)								512
Budget Bandwidth (DDR3 SDRAM 800MHz, 128-bit bus)								1024

6.4.2 External Memory Architecture

For the above estimated bandwidth requirement, Figure VI-29 shows the architecture of external memory that consists of eight DDR3 SDRAMs [110] for 128-bit bus. One of the DDR3 SDRAMs has 8 banks, and one row has 1024 columns. In addition, the word width is 16-bit of each SDRAM module, and the data width of the merged SDRAMs would be 128-bit. According to the latency information in [110], the DDR3 SDRAMs could work at the highest frequency of 800 MHz. They could output data at the positive and negative edges of clock signal, and there are two transfers in one cycle. Thus, the external memory architecture can provide the bandwidth of $800M \times 128 \times 2$ bits (i.e. 25,600 Mbytes/s).

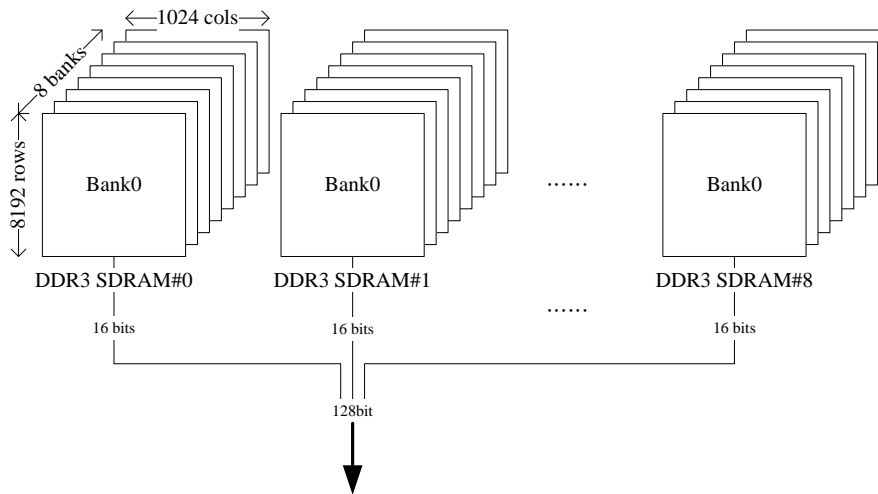


Figure VI-29 Architecture of external memory in our design

Figure VI-30 summarizes the data access latency of the external memory at the clock frequency of 800MHz. According to the associated latencies defined in [110], we could obtain the row miss latencies for different access types, and apply them to arrange the external memory access schedule.

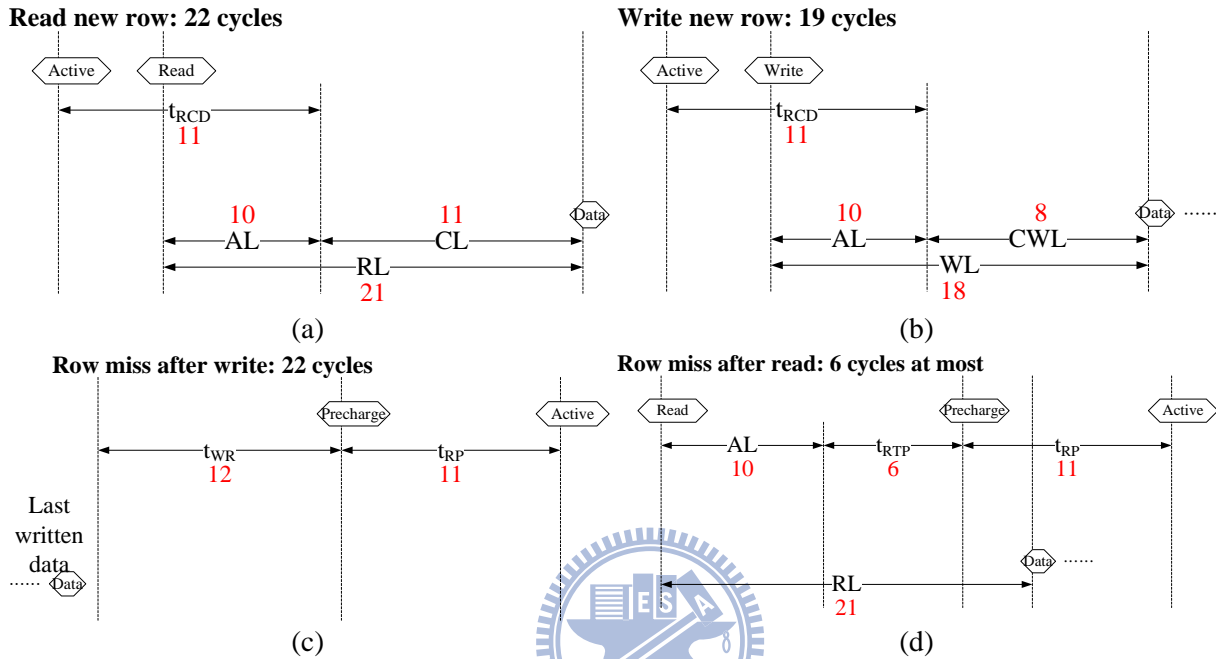


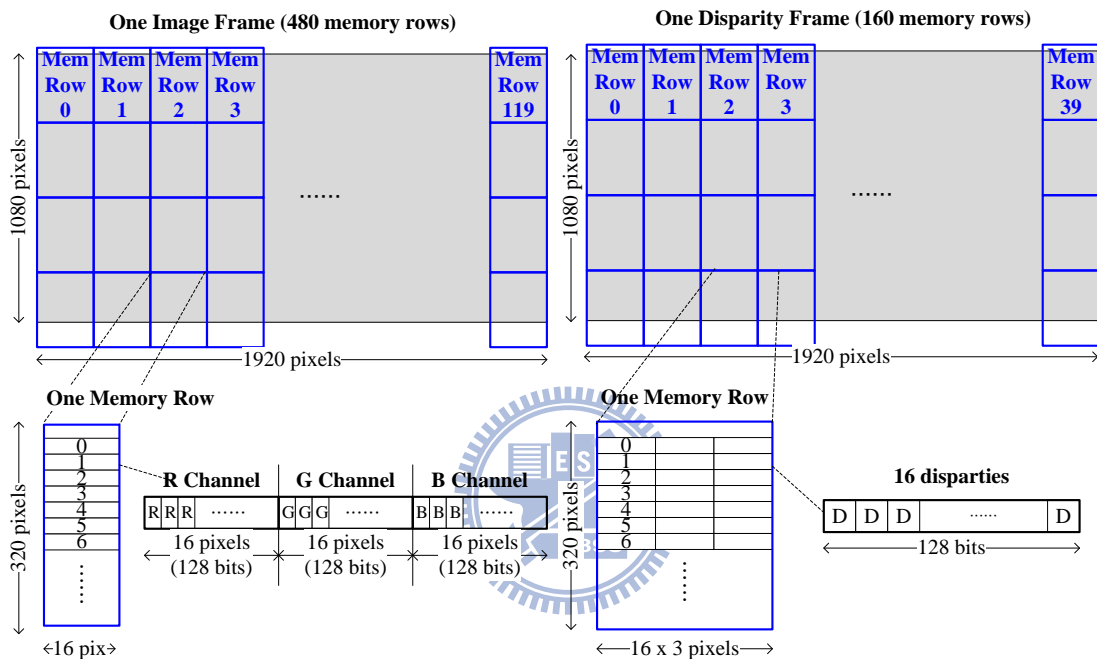
Figure VI-30 Read and write latency in the SDRAM model [110]

6.4.3 Data Configuration in External Memory

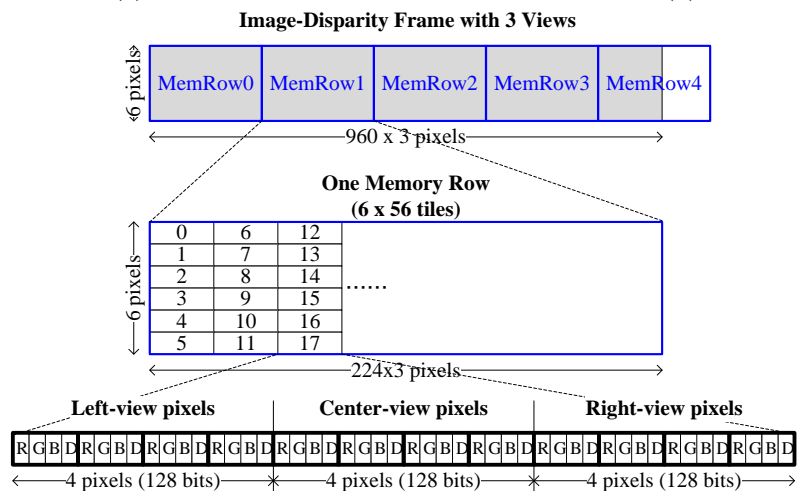
For the external memory, we propose the data configuration method as illustrated in Figure VI-31 to achieve efficient data access. The high-resolution videos and disparity maps are configured in the bank0 to bank5, and each view placed in one bank. The high resolution is mainly used by the motion detection, downsampled matching cost and the joint bilateral upsampling modules. Since their access are in the raster-scan order and cross multiple rows, 16 continuous pixels and disparities are packed, and neighboring pixel and disparity rows are in the same memory row to avoid frequent row miss.

For the low-resolution images and disparity maps are configured in the bank6, and their three view data are packed together as shown in Figure VI-31 (c). The low-resolution data are mainly accessed by the occlusion handling and the joint bilateral upsampling modules. Their access characteristic is similar to the above high-resolution data, so that they are configured with the same

manner. In addition, the disparity is bound with the image pixel as the image-disparity pixel because their accessed positions in image coordinate are identical. On the other hand, the low-resolution motion information is configured in bank7 as illustrated in Figure VI-31 (d) and (e). They are accessed by the motion detection and the still-edge preservation modules in the raster scan order without crossing multiple rows. Therefore, we place each motion information row into the same memory row.



(a) (b)



(c)

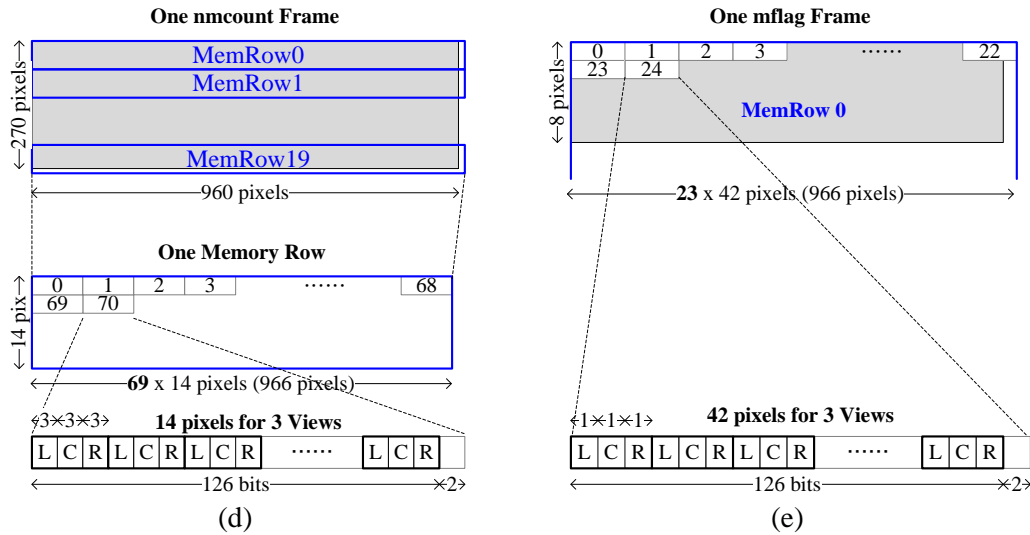


Figure VI-31 Data configuration in external memory

(a) high-resolution videos in bank 0, 1, 2, (b) high-resolution disparity maps in bank 3, 4, 5, (c) low-resolution image and disparity map in bank 6, (d) no-motion count in bank 7, (e) motion flag in bank 7.

6.4.4 External Memory Access Schedule

With the above mentioned data configuration in the external memory, we could further plan the external memory access schedule. Figure VI-32 shows the schedule of external memory access that is a hierarchy schedule from one frame to an access tile. By the schedule of access tile, the memory access controller can use a finite-state machine to read or write data for the main core computation. The order of data access follows the numbers in this figure. Note that each access block only uses the beginning cycles and preserves other cycles for row miss handling. Note that the available block at the bank 6 is preserved for the occlusion handling stage. We do not schedule the access of occlusion handling stage because its access period is very long and different from others. With the same reason, the motion information at the bank 7 is also not scheduled.

With this proposed external memory access schedule, the required data for one-frame computation could be completely accessed in 8.36Mcycles at the memory clock rate of 800MHz. In other words, the data throughput from external memory could achieve 95.6 frames/s. For the previous computational schedule, our main core could achieve 112.5 frames/s at the working frequency of 200 MHz. In other words, the performance bottleneck of our disparity estimation engine is the external

data access, instead of the computational speed. Nevertheless, the proposed disparity estimation engine could outperform our target throughput.

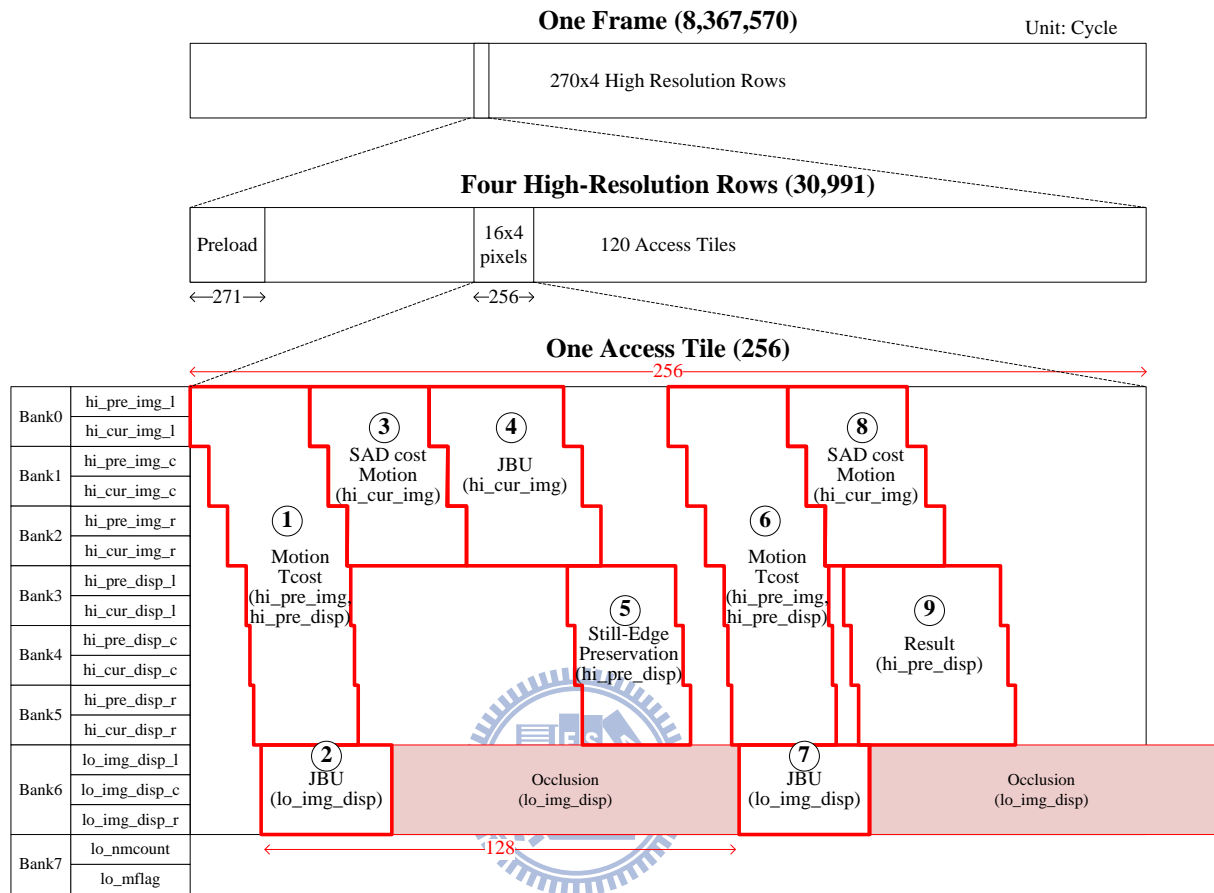


Figure VI-32 Schedule of external memory access for one HD1080p frame at 800MHz

6.5 Implementation Result

6.5.1 Hardware Cost

The proposed architecture of disparity estimation engine is implemented by the Verilog and synthesized using the UMC90nm technology process. Table VI-2 lists the performance of our disparity estimation engine. The proposed engine could use the three view HD1080p videos to calculate their corresponding three view disparity maps. The support disparity range could be 128 pixels. The required system memory is DDR3 SDRAM working at the clock frequency of 800MHz, and the system bus is 128-bit with the same clock frequency. The core module could achieve the throughput of

75.64G pixel-disparities/s by the logic cost of 1,645K gate counts and the memory cost of 59.4Kbytes. In other words, our disparity estimation engine could deliver 95 frames/s for three view HD1080p disparity maps.

Table VI-2 Performance of the proposed disparity estimation engine

I/O Function	Input Data	3 View HD1080p Videos
	Disparity Range (Pixel)	128
	Output Data	3 View HD1080p Disparity Maps
	Frame Rate (Frame/s)	95
System	External Memory	DDR3 SDRAM (800MHz)
	Bus Width (Bit)	128 (800MHz)
Core	Technology Process	UMC 90nm
	Clock Frequency	200MHz
	Gate-Count (Including Memory)	2,020K
	Gate-Count (Excluding Memory)	1,645K
	Internal Memory (Byte)	59.4K
	Throughput (Pixel-Disparity/s)	75.64G

Table VI-3 lists the internal SRAM usage for each pipelining stage. In which, the most usage is the low-resolution image buffers `lo_cur_img`, which is a shared buffer for the low-resolution disparity estimation stage and the occlusion handling stage. In addition, the SRAM usage of the high-resolution disparity estimation stage is also high due to the disparity row buffers for the joint bilateral upsampling and the window vote modules. The total gate-count for these internal SRAMs is about 375.6K.

Table VI-3 Internal SRAM usage in the proposed disparity estimation engine

		Memory Type	Word Num.	Word Width	Count	Size (Bit)
Low-Resol. DE Stage	lo_pre_disp	single-port	960	21	1	20,160
	lo_mval	single-port	960	24	1	23,040
	lo_min_cot	single-port	960	33	1	31,680
	lo_cur_disp	dual-port	960	21	2	40,320
	lo_cur_img	single-port	960	72	3	207,360
Occlusion Handling Stage	lo_occ	single-port	960	3	3	8,640
	lo_warp_disp	single-port	960	21	3	60,480
	lo_good	single-port	960	3	1	2,880
High-Resol. DE Stage	hi_cur_disp	two-port	1,920	21	2	80,640
Total						475,200

Table VI-4 lists the internal registers in each stage. Most of the registers are the access buffers in the I/O interface module. Because the registers are accessed by the main core with high data width,

they are not implemented by SRAM. In this table, the most register usage is the image data for the window-based SSAD in the low-resolution DE stage. It results from that the access to compute parallel matching costs in one cycle. The total register usage is 73Kbits, which is about 396K gate-counts.

Table VI-4 Internal registers in the proposed disparity estimation engine

		Row Num.	Word Num.	Word Width	Count	Size (Bit)
Low-Resol. DE Stage	hi_pre_img	3	16	24	3	3,456
	lo_nmcount	1	14	3	6	252
	lo_mflag	1	42	1	3	126
	hi_pre_disp	1	16	7	3	336
	hi_cur_img_l	3	146	24	1	10,512
	hi_cur_img_c	3	273	24	1	19,656
	hi_cur_img_r	3	146	24	1	10,512
Occlusion Handling Stage	ref_disp_buf	1	259	7	1	1,813
	warp_disp_buf	1	129	7	3	2,709
	ref_img_buf	3	3	24	3	648
	lo_img_disp	1	16	31	6	2,976
High-Resol. DE Stage	lo_img_disp	5	9	31	3	4,185
	hi_cur_img	6	22	24	3	9,504
	lo_mflag	2	42	1	3	252
	hi_pre_disp	4	18	7	3	1,512
	hi_cur_disp	4	32	7	3	2,688
	hi_disp_buf	6	6	24	3	2,592
Total						73,729

Table VI-5 lists the area of each module by the unit of gate count. In which, the half hardware cost is occupied by the window-based SSAD modules due to its parallel computation for matching costs. On the other hand, the horizontal cost diffusion has 17.1% hardware cost of whole core. That is because its convolution PE requires many parallel adders.

Table VI-5 Area of the computational logic

	Module	Gate Count	Percentage
Low-Resol. DE Stage	Motion Detection	19,058	1.5%
	Window-based SSAD, DPotts	616,725	49.4%
	Temporal Cost, Vertical Diffusion(Center-View)	54,541	4.4%
	Temporal Cost, Vertical Diffusion(Side-View)	102,077	8.2%
	Horizontal Diffusion (Computation)	213,495	17.1%
	Horizontal Diffusion (Registers)	4,794	0.4%
Occlusion Handling Stage	Warp Filling	31,688	2.5%
	Good Disparity Detection	5,210	0.4%
	Border and Inside Filling	36,174	2.9%
Occlusion Stage	Joint Bilateral Upsampling	125,457	10.0%
	Window Vote	30,318	2.4%
	Still-Edge Preservation	8,887	0.7%
Total		1,248,422	100%

Finally, Table VI-6 compares the previous implementation of real-time disparity estimation. For the GPU implementation, the previous work [43], [33] could deliver accurate disparity maps by the BP-based algorithm, but their throughputs are far from the requirement of real-time high-definition process. For the hardware design, Diaz *et al.* [111] implemented a high-throughput disparity estimation engine on FPGA but its disparity quality is not good enough for 3DTV applications due to its local disparity estimation approach. On the other hand, the ASIC implementation [10] could achieve real-time frame rate and requires low memory cost. But it supported frame resolution is only CIF. The other AISC implementation [33] could reach high frame rate for the VGA resolution, but it suffers from extremely high memory cost because of the BP algorithm. Compared to the related implementation, our disparity estimation engine could have the highest throughput with less hardware cost than the implementation [33] to satisfy the requirement of high definition 3DTV applications.

Table VI-6 Comparison of our design and previous implementation

	Yang [43]	Liang [33]	Diaz [111]	Chang [10]	Liang [33]	Our Design
No. Input View	2	2	2	2	2	3
No. Output View (Disparity Map)	1	1	1	1	1	3
Algorithm	Hierarchical BP	Tile-based BP	Phase Matching	Mini-Census ADSW	Tile-based BP	Cost Diffusion JBU
Frame Size	800×600	450×375	1280×960	352×288	640×480	1920×1080
Frame Rate (Frame/s)	0.67	1.68	52	42	58	95
Disparity Range (Pixels)	300	60	29	64	64	128
Implementation Method	GPU Nvidia Geforce 8800GTX	GPU Nvidia Geforce 8800GTS	FPGA Xilinx Vertex-II	ASIC UMC 90nm	ASIC UMC 90nm	ASIC UMC 90nm
Frequency (MHz)	-	-	65	95	185	200
Logic Area (Gate-Count)	-	-	-	562K	633K	1,645K
Memory Usage (Gate-Count)	-	-	-	-	1,871K	375K
	(9Mbyte)			(21.3Kbyte)		(59.4Kbyte)
Total Area	-	-	-	-	2,505K	2,020K
Throughput (Pixel-Disparity/s)	96M	17M	1,885M	272M	1,146M	75,644M

6.5.2 Disparity Quality

In this subsection, we demonstrate the disparity quality of disparity estimation engine using the same objective evaluation method in Chapter V. The first version of the algorithm for hardware design is called HW-DE algorithm, while the final version is called modified HW-DE algorithm. The only difference between the two algorithms is that the smoothness term in the cost diffusion process. Because the parameter τ_V could impact on the hardware cost, we change its value in the modified HW-DE algorithm to decrease hardware cost. In the following, we demonstrate the disparity quality change between the two algorithms. In addition, the evaluation results of DERS, HQ-DE, and HE-DE algorithms are also compared to the HW-DE algorithms.

Table VI-7 and Table VI-8 list the Y-PSNR evaluation results, and its corresponding column diagram is shown in Figure VI-33. Compared to the HE-DE algorithm, the HW-DE algorithm has the slight quality drop especially for the sequence LoveBird1. The disparity quality of modified HW-DE algorithm is approximate to HW-DE in the spatial distortion. For the other spatial distortion evaluation SSIM as shown in Table VI-9, Table VI-10, and Figure VI-34, the HW-DE and modified HW-DE algorithms have similar quality to the HE-DE algorithm. Finally, Table VI-11 and Table VI-12 lists the temporal distortion evaluation T_PSPNR, and its corresponding column diagram is shown in Figure VI-35. The evaluation results show that the HW-DE and the modified HW-DE algorithms have the same quality change, compared to the HE-DE algorithm. They suffer from quality degradation for the sequences LoveBird1 and Newspaper.

Table VI-7 Evaluation results of Y-PSNR for View0

	DERS	HQ-DE		HE-DE		HW-DE		Modified HW-DE	
	PSNR	PSNR	PSNR	PSNR	Δ PSNR	PSNR	Δ PSNR	PSNR	Δ PSNR
BookArrival	34.28	35.98	1.70	35.80	1.53	35.64	1.36	35.46	1.19
LoveBird1	32.45	32.63	0.18	31.53	-0.92	31.32	-1.13	31.09	-1.36
Newspaper	29.53	29.90	0.37	30.03	0.49	29.93	0.40	29.91	0.38
Café	N.A.	33.30	N.A.	33.22	N.A.	32.65	N.A.	32.52	N.A.
Kendo	N.A.	34.84	N.A.	34.88	N.A.	34.78	N.A.	34.76	N.A.
Balloons	N.A.	35.07	N.A.	34.91	N.A.	34.78	N.A.	34.83	N.A.
Champagne	25.32	27.63	2.31	31.07	5.75	30.74	5.42	30.63	5.31
Pantomime	36.46	35.94	-0.52	34.66	-1.80	36.54	0.08	36.64	0.18
Average	31.61	33.16	0.81	33.26	1.01	33.30	1.23	33.23	1.14

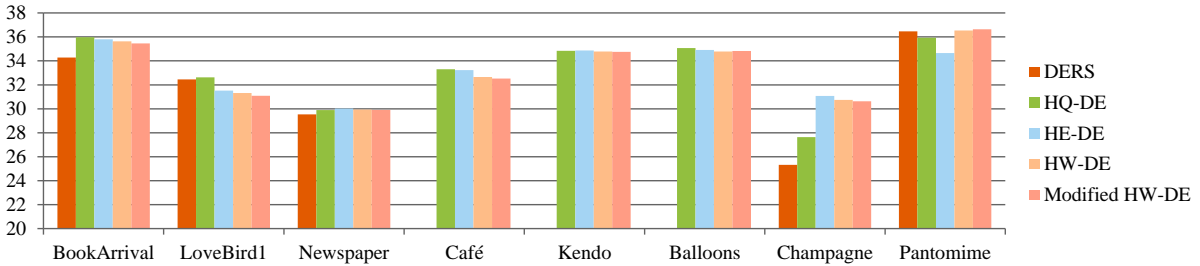
Unit: dB

Table VI-8 Evaluation results of Y-PSNR for View8

	DERS	HQ-DE		HE-DE		HW-DE		Modified HW-DE	
	PSNR	PSNR	PSNR	Δ PSNR	PSNR	PSNR	Δ PSNR	Δ PSNR	
BookArrival	35.87	35.68	-0.19	36.02	0.02	35.80	-0.07	35.62	-0.24
LoveBird1	29.31	27.53	-1.78	27.98	-1.08	27.67	-1.64	27.68	-1.63
Newspaper	31.86	31.29	-0.57	31.92	-0.10	31.75	-0.11	31.72	-0.14
Café	N.A.	32.87	-	33.04	-	32.70	N.A.	32.48	N.A.
Kendo	N.A.	35.75	-	36.36	-	36.15	N.A.	36.10	N.A.
Balloons	N.A.	35.24	-	35.58	-	35.35	N.A.	35.38	N.A.
Champagne	24.20	28.72	4.52	29.73	3.91	29.78	5.58	29.51	5.31
Pantomime	34.65	35.85	1.20	35.61	1.35	35.66	1.01	35.65	1.00
Average	31.18	33.11	0.82	33.28	1.08	33.11	0.95	33.02	0.86

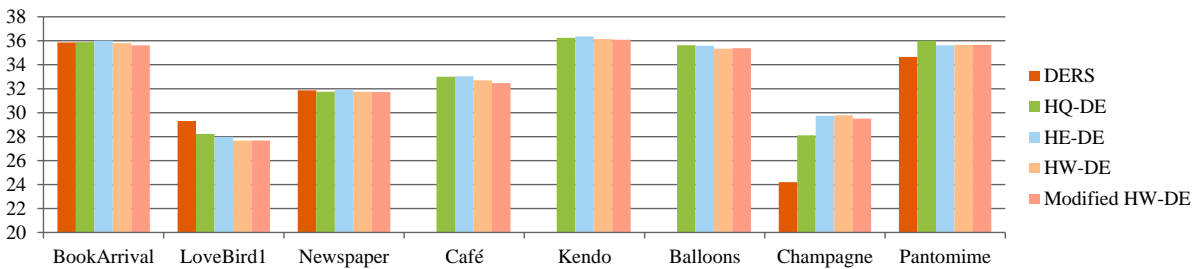
Unit: dB

Y-PSNR for View0 (dB)



(a)

Y-PSNR for View8 (dB)



(b)

Figure VI-33 Evaluation results of Y-PSNR

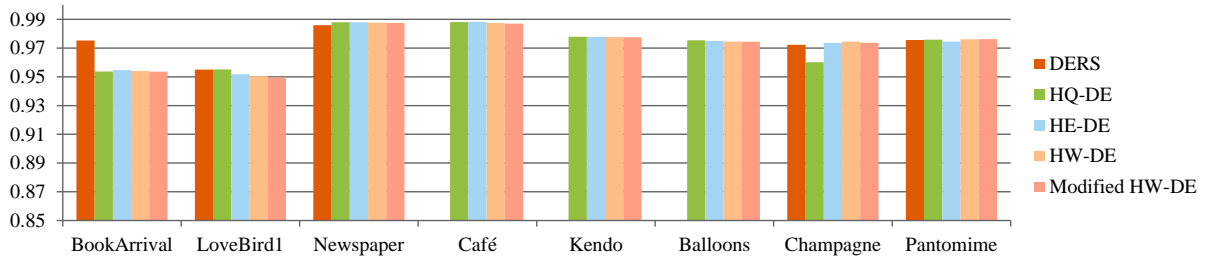
Table VI-9 Evaluation results of SSIM for View0

	DERS	HQ-DE		HE-DE		HW-DE		Modified HW-DE	
	SSIM	SSIM	SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM
BookArrival	0.98	0.95	-0.02	0.95	-0.02	0.95	-0.02	0.95	-0.02
LoveBird1	0.95	0.95	0.00	0.95	0.00	0.95	0.00	0.95	-0.01
Newspaper	0.99	0.99	0.00	0.99	0.00	0.99	0.00	0.99	0.00
Café	N.A.	0.99	-	0.99	-	0.99	N.A.	0.99	N.A.
Kendo	N.A.	0.98	-	0.98	-	0.98	N.A.	0.98	N.A.
Balloons	N.A.	0.97	-	0.97	-	0.97	N.A.	0.97	N.A.
Champagne	0.97	0.97	0.00	0.97	-0.01	0.97	0.00	0.97	0.00
Pantomime	0.98	0.98	0.00	0.97	0.00	0.98	0.00	0.98	0.00
Average	0.97	0.97	-0.01	0.97	0.00	0.97	0.00	0.97	0.00

Table VI-10 Evaluation results of SSIM for View8

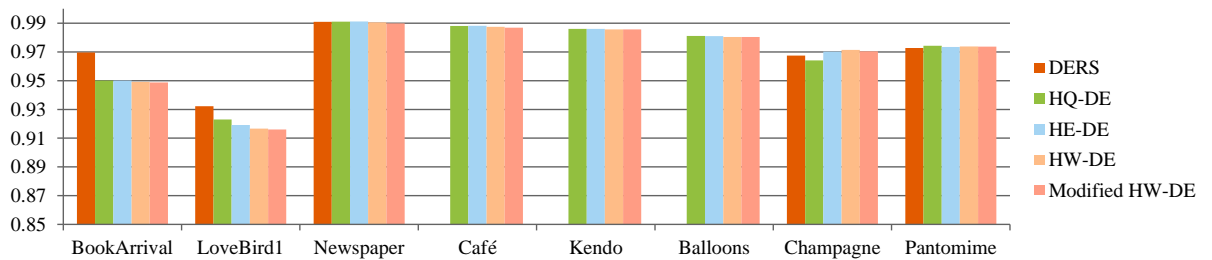
	DERS	HQ-DE		HE-DE		HW-DE		Modified HW-DE	
	SSIM	SSIM	SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM	SSIM	Δ SSIM
BookArrival	0.97	0.95	-0.02	0.95	-0.02	0.95	-0.02	0.95	-0.02
LoveBird1	0.93	0.92	-0.01	0.92	-0.01	0.92	-0.02	0.92	-0.02
Newspaper	0.99	0.98	-0.01	0.99	0.00	0.99	0.00	0.99	0.00
Café	N.A.	0.99	-	0.99	-	0.99	N.A.	0.99	N.A.
Kendo	N.A.	0.98	-	0.99	-	0.99	N.A.	0.99	N.A.
Balloons	N.A.	0.98	-	0.98	-	0.98	N.A.	0.98	N.A.
Champagne	0.97	0.97	0.00	0.97	0.00	0.97	0.00	0.97	0.00
Pantomime	0.97	0.97	0.00	0.97	0.00	0.97	0.00	0.97	0.00
Average	0.97	0.97	-0.01	0.97	-0.01	0.97	-0.01	0.97	-0.01

SSIM for View0



(a)

SSIM for View8



(b)

Figure VI-34 Evaluation results of SSIM

Table VI-11 Evaluation results of T_PSPNR (dB) for View0

	DERS	HQ-DE		HE-DE		HW-DE		Modified HW-DE	
	T_PSPNR	T_PSPNR	T_PSPNR	T_PSPNR	Δ T_PSPNR	T_PSPNR	Δ T_PSPNR	T_PSPNR	Δ T_PSPNR
BookArrival	52.96	53.60	0.64	52.94	0.64	52.93	-0.03	52.65	-0.31
LoveBird1	45.30	46.57	1.26	45.70	1.26	45.65	0.35	45.37	0.07
Newspaper	43.38	44.09	0.71	43.65	0.71	43.51	0.13	43.37	-0.01
Café	N.A.	46.59	-	47.83	-	46.38	N.A.	46.51	N.A.
Kendo	N.A.	48.08	-	48.15	-	48.12	N.A.	48.02	N.A.
Balloons	N.A.	49.99	-	49.93	-	49.87	N.A.	49.89	N.A.
Champagne	34.62	41.28	6.66	44.56	6.66	45.00	10.38	44.87	10.25
Pantomime	51.85	52.19	0.35	50.95	0.35	52.58	0.73	52.66	0.81
Average	45.62	47.80	1.92	47.96	1.94	48.01	2.31	47.92	2.16

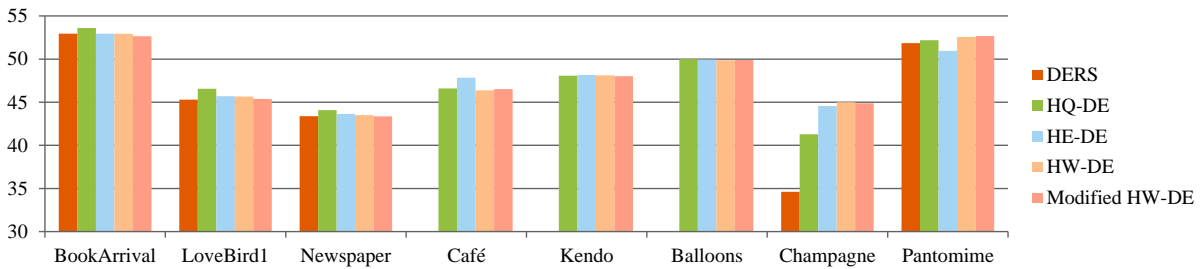
Unit: dB

Table VI-12 Evaluation results of T_PSPNR (dB) for View8

	DERS	HQ-DE		HE-DE		HW-DE		Modified HW-DE	
	T_PSPNR	T_PSPNR	T_PSPNR	T_PSPNR	Δ T_PSPNR	T_PSPNR	Δ T_PSPNR	T_PSPNR	Δ T_PSPNR
BookArrival	51.82	53.52	1.70	54.62	1.70	53.70	1.88	53.87	2.05
LoveBird1	43.33	44.70	1.37	43.84	1.37	43.27	-0.06	42.82	-0.50
Newspaper	47.92	47.96	0.04	47.82	0.04	47.19	-0.73	47.01	-0.91
Café	N.A.	46.86	-	46.85	-	45.96	N.A.	45.84	N.A.
Kendo	N.A.	50.58	-	50.81	-	50.66	N.A.	50.50	N.A.
Balloons	N.A.	49.76	-	49.90	-	49.85	N.A.	49.84	N.A.
Champagne	34.16	41.18	7.02	42.19	7.02	43.08	8.91	42.71	8.54
Pantomime	48.45	50.12	1.67	50.06	1.67	50.03	1.58	50.06	1.61
Average	45.14	48.09	2.36	48.26	2.57	47.97	2.32	47.83	2.16

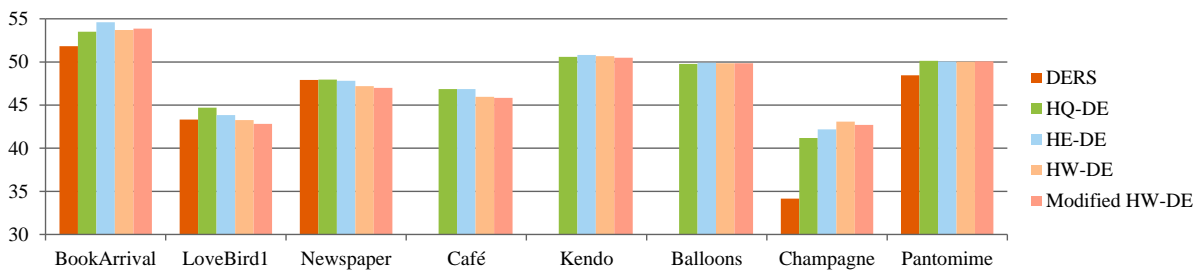
Unit: dB

T_PSPNR for View0 (dB)



(a)

T_PSPNR for View8 (dB)

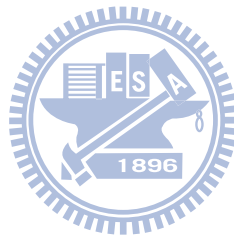


(b)

Figure VI-35 Evaluation results of T_PSPNR

6.6 Summary

In this chapter, we simplify the HE-DE algorithm by removing the de-noising filters, and improve the motion detection by considering the hardware cost. According to the HW-DE algorithm, we propose a high throughput disparity estimation engine using the three-pipelining-stage architecture and well-defined external memory access schedule. The implementation result shows that the proposed disparity estimation engine could achieve 95 frames/s for three view HD1080p disparity maps. The final quality evaluation shows that the disparity estimation engine only has slight quality drop in average.



VII Conclusion

7.1 Contribution

For the high definition 3DTV applications, the disparity estimation is one of the most important processes to generate disparity maps for view synthesis. The state-of-the-art DERS algorithm could provide high quality disparity maps but incurs high computational complexity. Because of its irregular and non-parallel graph-cut algorithm, it could not be accelerated to meet the high throughput requirement by software programming and hardware design.

To address the problem, this dissertation proposes the baseline disparity estimation algorithm that combines the belief propagation with the joint bilateral upsampling. The former has highly parallel computational characteristic, and the latter could reduce the computational resolution of disparity estimation.

Based on the baseline algorithm, we further propose the high-quality disparity estimation (HQ-DE) algorithm that could deal with the temporal consistency and occlusion problems to deliver high quality disparity maps. To accelerate the HQ-DE algorithm, we propose two fast algorithms by different strategies for different implementation. The first spare-computation disparity estimation (SC-DE) algorithm is suitable to software programming. That could reduce the computation of dense belief propagation to 13.4%, and the overall execution to 62.9%. The other hardware-efficient disparity estimation (HE-DE) algorithm is suitable to VLSI design, and could reduce the memory cost of original belief propagation to 0.00029% and achieve the approximate reduction of execution time to SC-DE algorithm. The objective evaluation results show that the proposed HQ-DE algorithm could deliver better disparity maps than the DERS algorithm, and the two fast algorithms has slight quality drop compared to the HQ-DE algorithm.

Following the HE-DE algorithm, we further simplify its computation to reduce the hardware cost in the algorithm level with slight quality drop, and deliver the hardware-based disparity estimation

(HW-DE) algorithm. By the architectural design techniques, we propose a disparity estimation engine that applies the three-stage pipelining architecture and parallel PEs to increase its throughput. The implementation result shows that our disparity estimation engine could achieve the throughput of 95 frames/s for the three view HD1080p disparity maps. Such the high throughput disparity estimation engine could be applied to high definition 3DTV systems.

7.2 Future Work

In this dissertation, the occlusion handling method and the evaluation results could be improved in the future work. For the occlusion problems, this dissertation fills the occlusion regions by the reliable disparities from the spatial and the inter-view domains. However, the disoccluded regions, which are visible only at one viewpoint, could not be filled well by the disparities from the two domains. To address this special case, we could detect the reliable disparities from the temporal domain. In other words, the reliable disparities would be at previous or next frames.

For the evaluation method, this dissertation adopts the common-used three objective evaluation methods which are compares the real captured videos with the synthesized videos using the proposed disparity maps. However, these evaluation methods are performed on 2-D videos, instead of 3-D videos. For the 3-D videos, the subjective evaluation method needs to be applied. Therefore, both the objective evaluation and the subjective evaluation methods should be used to assess the disparity quality.

In addition, the disparity map for scene change should be considered, especially for the temporal consistency and the sparse regions in the SC-DE algorithm. To deal with it, we could detect the frame with scene change according the total difference of successive frames, and initialize the motion information in the temporal consistency methods and the propagated cost cubes in the SC-DE algorithm.

Bibliography

Local Approach for Disparity Estimation

- [1] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling" *IEEE Trans. Pattern Anal. Mach. Intell.(TPAMI)*, no. 20, vol. 4, pp. 401-406, Apr. 1998.
- [2] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Conf. on Comput. Vision Pattern Recognition (CVPR '07)*, Jun. 2007.
- [3] N. Y.-C. Chang, Y.-C. Tseng, and T.-S. Chang, "Analysis of color space and similarity measure impact on stereo block matching," in *Proc. IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS'08)*, Dec. 2008, pp. 926-929.
- [4] J. Lu, G. Lafruit, and F. Catthoor, "Anisotropic local high-confidence voting for accurate stereo correspondence," in *Proc. SPIE Image Process.: Algorithm and Syst. VI*, vol. 68120, Jan. 2008.
- [5] K. Zhang, J. Lu, and G. Lafruit, "Scalable stereo matching with locally adaptive polygon approximation," in *Proc. IEEE Int. Conf. on Image Process. (ICIP '08)*, Oct. 2008, pp. 313-316.
- [6] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Trans. Circuits Syst. Video Technol.*, no. 19, vol. 7, pp. 1073-1079, Jul. 2009.
- [7] K.-J. Yoon and I.-S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 650-656, Apr. 2006.
- [8] M.-H. Ju and H.-B. Kang, "Constant time stereo matching" in *Proc. Int. Conf. on Machine Vision and Image Process. (IMVIP '09)*, Sep. 2009, pp. 13-17.
- [9] W. Yu, T. Chen, F. Franchetti, and J. C. Hoe, "High performance stereo vision designed for massively data parallel platforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1509-1519, Nov. 2010.
- [10] N. Y.-C. Chang, T.-H. Tsai, B.-H. Hsu, Y.-C. Chen, and T.-S. Chang, "Algorithm and architecture of disparity estimation with mini-census adaptive support weight," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 6, pp. 792-805, Jun. 2010.

Dynamic Programming

- [11] Y. Ohta and T. Kanade, "Stereo by intra- and inter- scanline search using dynamic programming," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, no. 7, vol. 2, pp. 139-154, Mar. 1985.

- [12] O. Veksler, "Stereo correspondence by dynamic programming on a tree," in *Proc. IEEE Conf. on Comput. Vision Pattern Recognition (CVPR'05)*, 2005, pp. 384-390.
- [13] Y. Deng and X. Lin, "A fast line segment based dense stereo algorithm using tree dynamic programming," in *Proc. European Conf. on Comput. Vision (ECCV'06)*, 2006, pp. 201-210.
- [14] C. Lei, J. Selzer, Y.-H. Yang, "Region-tree based stereo using dynamic programming optimization," in *Proc. IEEE Conf. on Comput. Vision Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2378-2385.

Graph-cut

- [15] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Proc. IEEE Int. Conf. on Comput. Vision (ICCV'01)*, vol. 2, Jul. 2001, pp. 508-515.
- [16] L. Ford and D. Fulkerson, *Flows in networks*, Princeton Univ. Press, 1962.
- [17] A. V. Goldberg, "A new approach to the maximum flow problem," *J. of the ACM*, vol. 35, pp. 921-940, 1988.
- [18] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 23, no. 11, pp. 1222-1239, Nov. 2001.
- [19] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 26, no. 9, pp. 1124-1137, Sep. 2004.
- [20] C.-W. Chou, J.-J. Tsai, H.-M. Hang, and H.-C. Lin, "A fast graph cut algorithm for disparity estimation," in *Proc. Picture Coding Symp. (PCS'10)*, Nagoya, Japan, Dec. 2010, pp. 326-329.
- [21] B. V. Cherkassky and A. V. Goldberg, "On implementing the push-relabel method for the maximum flow problem," *Algorithmica*, New York Inc.: Springer-Verlag, 1997, vol. 19, pp. 390-410.
- [22] A. DeLong and Y. Boykov, "A scalable graph-cut algorithm for N-D grids," in *Proc. IEEE Conf. on Comput. Vision Pattern Recognition (CVPR'08)*, Jun. 2008.
- [23] N. Y.-C. Chang and T.-S. Chang, "A scalable graph-cut engine architecture for real-time vision," in *Proc. VLSI design/CAD Symp.*, Hualien, Taiwan, 2007.

Belief Propagation

- [24] J. Sun, N.-N. Zhang, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 25, no. 7, pp. 787-800, Jul. 2003.

- [25] P. F. Felzenswalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comput. Vision (IJCV)*, vol. 70, no. 1, pp. 41-54, May 2006.
- [26] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov Random Fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 30, no. 6, pp. 1060-1080, Jun. 2008.
- [27] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. British Mach. Vision Conf. (BMVC)*, 2006.
- [28] S. Park C. Chen, and H. Jeong, "VLSI architecture for MRF based stereo matching," in *Proc. Int. Symp. on Syst., Architecture, Modeling and Simulation (SAMOS'07)*, Greece, Jul. 2007.
- [29] C.-C. Cheng, C.-K. Liang, Y.-C. Lai, H. H. Chen, and L.-G. Chen, "Analysis of belief propagation for hardware realization," in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS'08)*, Washington DC, USA, Oct. 2008, pp. 152-157.
- [30] C.-C. Cheng, C.-K. Liang, Y.-C. Lai, H. H. Chen, and L.-G. Chen, "Fast belief propagation process element for high-quality stereo estimation," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP'09)*, Taipei, Taiwan, Apr. 2009, pp. 745-748.
- [31] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR'09)*, Florida, USA, Jun. 2009, pp. 80-87.
- [32] C.-C. Cheng, C.-T. Li, C.-K. Liang, Y.-C. Lai, and L.-G. Chen, "Architecture design of stereo matching using belief propagation," in *Proc. IEEE Int. Symp. Circuits and Syst. (ISCAS'10)*, Jun. 2010, pp. 4109-4112.
- [33] C.-K. Liang, C.-C. Cheng, Y.-C. Li, L.-C. Chen, and H. H. Chen, "Hardware-efficient belief propagation," *IEEE Trans. Circuits Syst. Video Technol. (TCSVT)*, vol. 21, no. 5, pp. 525-537, May 2011.
- [34] S. C. Park and H. Jeong, "Memory-efficient iterative process for two-dimensional first-order regular graph," *Optics Letter*, vol. 33, no. 1, pp. 74-76, Jan. 2008.
- [35] T. Yu, R.-S. Lin, B. Super, B. Tang, "Efficient message representation for belief propagation," in *Proc. IEEE Int. Conf. on Comput. Vision (ICCV'07)*, Oct. 2007.
- [36] Y.-C. Tseng, N. Chang, and T.-S. Chang, "Low memory cost block-based belief propagation for stereo correspondence," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, Beijing, China, Jul. 2007, pp. 1415-1418.

- [37] M. P. Kumar and P. H. S. Torr, "Fast memory-efficient generalized belief propagation," in *Proc. European Conf. on Computer Vision (ECCV'06)*, vol. 3954, Austria, May 2006, pp. 451-463.
- [38] Y.-C. Tseng, N. Y.-C. Chang, and T.-S. Chang, "Block-based belief propagation with in-place message updating for stereo vision," in *Proc. IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS'08)*, Macau, China, Dec. 2008, pp. 918-921.
- [39] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and self-adapting dissimilarity measure," in *Proc. IEEE Int. Conf. on Pattern Recognition (ICPR'06)*, Sep. 2006, pp. 15-18.
- [40] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister, "Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 31, no. 3, pp. 1-13, Mar. 2009.
- [41] E. S. Larsen, P. Mordohai, M. Pollefeys, and H. Fuchs, "Temporally consistent reconstruction from multiple video streams using enhanced belief propagation," in *Proc. IEEE Int. Conf. on Comput. Vision (ICCV'07)*, Rio de Janeiro, Brazil, Oct. 2007.
- [42] K. Ogawara, "Approximate belief propagation by hierarchical averaging of outgoing messages," in *Proc. IEEE Int. Conf. Pattern Recognition (ICPR'10)*, Istanbul, Aug. 2010, pp. 1368-1372.
- [43] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR'10)*, Jun. 2010, pp. 1458-1465.
- [44] M. Sarkis and K. Diepold, "Sparse stereo matching using belief propagation," in *Proc. IEEE Int. Conf. on Image Process. (ICIP'08)*, San Diego, CA, Oct. 2008, 1780-1783.

Disparity Refinement Algorithms

- [45] G. Egnal and R. P. Wildes, "Detecting binocular half-occlusions: empirical comparisons of five approaches," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 24, no. 8, pp. 1127-1133, Aug. 2002.
- [46] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to analysis and automated cartography," *Commun. of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.
- [47] M. Gong, "Enforcing temporal consistency in real-time stereo estimation," in *Proc. European Conf. on Comput. Vision (ECCV'06)*, vol. 3953, 2006, pp. 564-577.

- [48] D. Min, S. Yea, Z. Arican, and A. Vetro, "Disparity search range estimation: forcing temporal consistency," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP'10)*, Dallas, Texas, May 2010, pp. 2366-2369.
- [49] R. Khoshabeh, S. H. Chan, T. Q. Nyuyen, "Spatio-temporal consistency in video disparity estimation," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP'11)*, Prague, Czech Republic, May 2011.
- [50] T.-W. Chen and S.-Y. Chien, "Bandwidth adaptive hardware architecture of K-means clustering for video analysis," *IEEE Trans. Very Large Scale Integr. Syst. (TVLSI)*, vol. 18, no. 6, pp. 957-966, Jun. 2010.

View Synthesis Algorithms and Implementation

- [51] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *Proc. SPIE Conf. on Stereoscopic Displays and Virtual Reality Systems*, vol. 5291, May 2004, pp. 93-104.
- [52] C. Vázquez, W. J. Tam, and F. Speranza, "Stereoscopic imaging: filling disoccluded areas in image-based rendering," in *Proc. SPIE Three-Dimensional TV, Video, and Display V*, vol. 6392, Oct. 2006, pp. 123-134.
- [53] C.-M. Cheng, S.-J. Lin, S.-H. Lai and J.-C. Yang, "Improved novel view synthesis from depth image with large baseline," in *Proc. IEEE Int. Conf. on Pattern Recognition (ICPR'08)*, Dec. 2008, pp.1-4.
- [54] L. Zhang and W. J. Tam, "Stereoscopic image generation based on depth images for 3D TV," *IEEE Trans. Broadcast.*, vol. 51, no. 2, pp. 191-199, Jun. 2005.
- [55] Y.-R. Horng, Y.-C. Tseng, and T.-S. Chang, "Stereoscopic image generation with directional Gaussian filter," in *Proc. IEEE Int. Symp. Circuits and Syst. (ISCAS'10)*, May-Jun. 2010, pp. 2650-2653.
- [56] W.-Y. Chen, Y.-L. Chang, S.-F. Lon, L.-F. Ding, and L.-G. Chen, "Efficient depth image based rendering with edge dependent depth filter and interpolation," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME'07)*, Jul. 2007, pp. 1314-1317.
- [57] Y. K. Park, K. Jung, Y. Oh, S. Lee, J. K. Kim, G. Lee, H. Lee, K. Yun, N. Hur, and J. Kim, "Depth-image-based rendering for 3DTV service over T-DMB," *Signal processing: Image communication*, vol. 24, no. 1-2, pp. 122-36, Jan. 2009.
- [58] S. Rogmans, J.-B. Lu, P. Bekaert, and G. Lafruit, "Real-time stereo-based view synthesis algorithms: a unified framework and evaluation on commodity GPUs," *Signal Processing: Image communication*, vol. 24, no. 1-2, pp. 49-64, Jan. 2009.


- [59] Y. Morvan, "Acquisition, compression and rendering of depth and texture for multi-view video," Ph.D. thesis, Eindhoven University of Technology, Netherlands, Apr. 2009.
- [60] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graphics, GPU, & Game Tools*, vol. 9, no. 1, pp.25-36, 2004.
- [61] P.-K. Tsung, P.-C. Lin, K.-Y. Chen, T.-D. Chuang, H.-J. Yang, S.-Y. Chien, L.-F. Ding, W.-Y. Chen, C.-C. Cheng, T.-C. Chen, and L.-G. Chen, "A 216fps 4096x2160 3DTV set-top box SoC for free-viewpoint 3DTV applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC'11)*, San Francisco, CA, Feb. 2011, pp. 124-126.
- [62] Y.-R. Horng, Y.-C. Tseng, and T.-S. Chang, "VLSI architecture for real time HD1080p view synthesis engine," to appear in *IEEE Trans. Circuits Syst. Video Technol. (TCSVT)*, vol. 21, no. 9, Sep. 2011.

Associated Algorithms to 3DVC

- [63] *Depth estimation reference software (DESR)*, version 4.0 [Online]. Available: http://wg11.sc29.org/svn/repos/MPEG-4/test/tags/3D/depth_estimation/DESR_4
- [64] *View Synthesis Reference Software (VSRS)*, version 3.5 [Online]. Available: http://wg11.sc29.org/svn/repos/MPEG-4/test/tags/3D/view_synthesis/VSRS_3_5
- [65] *Enhancement of temporal consistency for multi-view depth map estimation*, ISO/IEC JTC1/SC29/WG11, M15594, Jul. 2008.
- [66] *Depth estimation improvement for depth discontinuity areas and temporal consistency preserving*, ISO/IEC JTC1/SC29/WG11, M16048, Feb. 2008.
- [67] *The consideration of the improved depth estimation algorithm: the depth estimation algorithm for temporal consistency enhancement in non-moving background*, ISO/IEC JTC1/SC29/WG11, m16070, Jan. 2009.
- [68] *A soft-segmentation matching in Depth Estimation Reference Software (DESR) 5.0*, ISO/IEC JTC1/SC29/WG11, M17049, Xian, China, Oct. 2009.
- [69] D. Comaniciu and P. Meer, "Mean-shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vo. 24, no. 5, pp. 603-619, May 2002.
- [70] *Open Source Computer Vision* [Online]. Available: <http://opencv.willowgarage.com/wiki/>

Test Sequences and Evaluation Methods

- [71] *Description of exploration experiments in 3D video coding*, ISO/IEC JTC1/SC29/WG11, W11095, Kyoto, Japan, Jan. 2010.

- [72] D. Scharstien and R. Szeliski, *Middlebury Stereo Evaluation – Version 2* [Online]. Available: <http://vision.middlebury.edu/stereo/eval/>
- [73] D. Scharstien and R. Szeliski, “High-accuracy stereo depth maps using structured light,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR’03)*, vol. 1, Jun. 2003, pp. 195-202.
- [74] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process. (TIP)*, vol. 13, no. 4, pp. 600-612, Apr. 2004.
- [75] *Peak signal-to-perceptible-noise ratio tool: PSPNR 1.0*, ISO/IEC JTC1/SC29/WG11, M16584, London, UK, Jul. 2009.
- [76] *PSPNR Tool 2.0*, ISO/IEC JTC1/SC29/WG11, M16890, Xian, China, Oct. 2009.
- [77] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *The SSIM Index for Image Quality Assessment* [Online]. Available: <http://www.cns.nyu.edu/~lcv/ssim/>
- [78] *HHI test materials for 3D video*, ISO/IEC JTC1/SC29/WG11, M15413, Archamps, France, Apr. 2008.
- 
- Joint Bilateral Filter and Disparity Upsampling**
- [79] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, “Joint bilateral upsampling,” *ACM Trans. on Graphics (TOG)*, vol. 26, no. 3, article 96, Jul. 2007.
- [80] D. Chan, H. Buisman, C. Theobalt, and S. Thrun, “A noise-aware filter for real-time depth upsampling,” in *Proc. European Conf. on Comput. Vision Workshop on Multicamera and Multimodal Sensor Function Algorithms and Applications*, Oct. 2008, pp. 1-12.
- [81] A. K. Riemens, O. O. Gangwal, B. Barenbrug, and R.-P. M. Berretty, “Multi-step joint bilateral depth upsampling,” in *Proc. SPIE Visual Commun. and Image Process.*, vol. 7257, Jan. 2009.
- [82] O. P. Gangwal, E. Coezijn, and R.-P. Berretty, “Real-time implementation of depth map post-processing for 3D-TV on a programmable DSP (TriMedia),” in *Proc. IEEE Int. Conf. on Consumer Electronics (ICCE’09)*, Jan. 2009.
- [83] Q. Yang, K.-H. Tan, and N. Ahuja, “Real-time O(1) bilateral filtering,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR’09)*, Aug. 2009, pp. 557-564.
- [84] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dynamic-range images,” *ACM Trans. Graphics (TOG)*, vol. 21, no. 3, pp. 257-266, Jul. 2002.

- [85] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *Proc. European Conf. on Comput. Vision (ECCV'06)*, May 2006, pp. 568-580.
- [86] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Int. J. Comput. Vision*, vol. 81, no. 1, pp. 24-52, Jan. 2009.
- [87] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Trans. Graphics (TOG)*, vol. 26, no. 3, article 103, pp. 1-9, Jul. 2007.
- [88] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high dimensional filtering," *ACM Trans. Graphics (TOG)*, vol. 28, no. 3, article 21, Aug. 2009.
- [89] T. Q. Pham and L. J. van Vliet, "Separable bilateral filtering for fast video processing," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME'05)*, Jul. 2005.
- [90] T.-S. Huang, "Two-dimensional digital signal processing II: transforms and median filters," *Spring-Verlag*, New York, 1981, pp. 209-211.
- [91] F. Porikli, "Constant time $O(1)$ bilateral filtering," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR'09)*, Aug. 2008, pp. 1-8.
- [92] B. Weiss, "Fast median and bilateral filtering," *ACM Trans. Graphics (TOG)*, vol. 25, no. 3, pp. 519-526, Jul. 2006.
- [93] M.-H. Ju, and H.-B. Kang, "Constant time stereo matching," in *Proc. Int. Machine Vision and Image Processing Conf.*, Sep. 2009, pp. 13-17.
- [94] C. Charoensak and F. Satter, "FPGA design of a real-time implementation of dynamic range compression for improving television picture," in *Proc. IEEE Int. Conf. on Information Commun. and Signal Process. (ICICS'07)*, Dec. 2007.
- [95] T. Q. Vinh, J. H. Park, Y.-C. Kim, and S. H. Hong, "FPGA implementation of real-time edge-preserving filter for video noise reduction," in *Proc. IEEE Int. Conf. on Comput. and Elect. Eng. (ICCEE'08)*, Dec. 2008, pp. 611-614.
- [96] A. Gabiger, M. Kube, and R. Weigel, "A synchronous FPGA design of a bilateral filter for image processing," in *Proc. IEEE Ind. Electron. Conf. (IECON'09)*, Nov. 2009, pp. 1990-1995.
- [97] S.-K. Han, "An architecture for high-throughput and improved-quality stereo vision processor," M.S. thesis, Dept. of Electrical and Computer Engineering, Univ. of Maryland, 2010.
- [98] A. Wong. *NVIDIA GeForce 8800 GTX/GTS Tech Report* [Online]. Available: <http://www.techarp.com/showarticle.aspx?artno=358&pgno=0>
- [99] A. L. Shimpi and D. Wilson. *Nvidia's 1.4 billion transistor GPU: GT200 arrives as the GeForce GTX 280 & 260* [Online]. Available: <http://www.anandtech.com/show/2549>

Others

- [100] M. Tanimoto, "Free-viewpoint television", *Image and Geometry Processing for 3-D Cinematography*, Springer-Verlag, vol. 5, part 1, 2010, pp. 52-76.
- [101] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," *IEEE Signal Processing Mag.*, vo. 28, no. 1, pp. 67-76, Jan. 2011.
- [102] Q. Wei, "Converting 2D to 3D: a survey," Inform. and Commun. Theory Group, Faculty Elect. Eng., Math. and Comput. Sci., Delft Univ. of Technol., Netherlands, Research Assignment, Dec. 2005.
- [103] D. Hoiem, A. Stein, A. A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *Proc. IEEE Int. Conf. on Comput. Vision (ICCV'07)*, Oct. 2007.
- [104] D. Hoiem, A. Efros, and M. Hebert, "Recovering surface layout from an image," *Int. J. Comput. Vision (IJCV)*, vol. 75, no. 1, pp. 151-172, Oct. 2007.
- [105] D. Scharstien and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithm," *Int. J. Comput. Vision (IJCV)*, vol. 47, no. 1-3, pp. 7-42, May 2002.
- [106] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 25, no. 8, pp. 993-1008, Aug. 2003.
- [107] *Joint draft 6.0 on multiview video coding*, ISO/IEC JTC1/SC29 and ITU-T SG16 Q.6 JVT-Z209, Antalya, Turkey, Jan. 2008.
- [108] N. Matthews, X. Meng, P. Xu, and N. Qian, "A physiological theory of depth perception from vertical theory," *Vision Research*, vol. 43, no. 1, pp. 85-99, Jan. 2003.
- [109] J. C. A. Read and B. G. Cumming, "Does depth perception require vertical-disparity detectors?" *J. of Vision*, vol. 6, no. 12, pp. 1323-1355, Nov. 2006.
- [110] Micron Inc. *1Gb DDR3 SDRAM: MT41J128M8JP-125* [Online]. Available: <http://www.micron.com/get-document/?documentId=425>
- [111] J. Diaz, E. Ros, R. Carrillo, and A. Prieto, "Real-time system for high-image resolution disparity estimation," *IEEE Trans. Image Process. (TIP)*, vol. 16, no. 1, pp. 280-285, Jan. 2007.