Real-Time High-Resolution Multiple-Camera Depth Map Estimation Hardware and Its Applications

THÈSE N^{O} 6573 (2015)

PRÉSENTÉE LE 16 AVRIL 2015
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE SYSTÈMES MICROÉLECTRONIQUES
PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Abdulkadir AKIN

acceptée sur proposition du jury:

Prof. J.-Ph. Thiran, président du jury
Prof. Y. Leblebici, Dr A. Schmid, directeurs de thèse
Prof. L. Claesen, rapporteur
Prof. P. Frossard, rapporteur
Prof. J.-I. Guo, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

Nature is not on the surface,
it is in the depths.
Colors are an expression of these depths on the surface.
They rise from the roots of the world.

— Paul Cezanne

If one day,
real-time depth perception of machines becomes always better than human,
at that time,
intelligence and learning ability of human will not be needed anymore to develop technology.

— Kadir

To my family…

Acknowledgements

First of all, I would like to thank my thesis director Prof. Yusuf Leblebici who gave me the opportunity to pursue my research in Microelectronic Systems Laboratory (LSM). Prof. Leblebici always encouraged me to implement my creative ideas, and he always provided his aspiring guidance and advices. In addition, I am cordially thankful to him, since he established a laboratory with brilliant ambiance, friendship, collaboration and passion. I know that, the fabulous culture of this laboratory is an output of his intensive efforts, and the positive synergy of LSM led me to pursue successful research during my PhD studies.

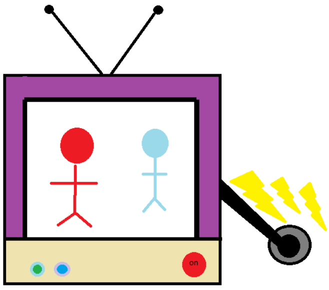
I would like to thank my thesis co-director Dr. MER. Alexandre Schmid especially for his guidance on the organization of my research. His suggestions on task scheduling and risk/benefit analysis kept me on the correct research path. In addition, his valuable comments on my papers significantly helped me to have attractive publications.

I would like to thank my thesis committee members, Prof. Jean-Philippe Thiran, Prof. Luc Claesen, Prof. Jiun In Guo and Prof. Pascal Frossard, who have evaluated this thesis and provided constructive comments. I would like to thank Melinda Mischler, Corine Degott and Patricia Vonlanthen for their support in the administrative works at EPFL. I would like to thank Alain Vachoux for his technical helps and advices.

During my research, I have collaborated with several PhD students, master students, interns and engineers on different aspects of my studies. I would like to present my grateful thanks to my these precious friends and collaborators. First, I would like to thank master students that I supervised: İpek Baz for the algorithm development of two- and three-camera disparity estimation (DE), Luis Gaemperle for the development of ethernet interface of DE system and development of graphical user interface (GUI), Raffaele Capoccia for the three-camera DE hardware implementation, Jonathan Narinx for the ASIC implementation of two-camera DE hardware, Jonathan Masur for the hardware implementation of free viewpoint synthesis, and Halima Najibi for the hardware implementation of the Caltech rectification algorithm. In addition, I would like to thank interns that I supervised: Gremaud Xavier Louis for the improvement of GUI, İrem Boybat for the algorithm development of two-camera DE and compressed rectification, Baris Hüseyin Atakan for the algorithm development of two-camera DE, Elif Erdede for the implementation of enhanced omnidirectional image reconstruction algorithm. Moreover, I would like to thank PhD students: Youngjoo Seo for the development of depth estimation based head/body tracking applications, Omer Cogal for his intensive collaboration on the development of ultra high resolution omnidirectional camera, and Kerem Seyid, Selman Ergünay, Hossein Afshari and Vladan Popovic for their useful technical helps, comments and advices. Furthermore, I would like to thank Tuğba Demirci for her co-supervision of Jonathan Narinx on ASIC implementation, Peter Brühlmeier for the design of all the PCB boards that are fabricated and used in the research, and Sylvain Hauser for the establishment of fabulous mechanical set-ups. While I was working with my collaborators, all the heavy work turned out just a fun thanks to their passion on their profession and friendship.

Being part of LSM gave me a chance to have great friends. Therefore I would like to extend my special thanks with former and current LSM members, especially Mustafa Kılıç, Gülperi Özsema, Can Baltacı, Zuhal Taşdemir, Şeniz Esra Küçük, Gözen Köklü, Clemens Nyffeler, Kiarash Gharibdoust, Nikola Katic, Mahsa Shoaran, Cosimo Aprile, Jury Sandrini, Gain Kim, Behnoush Attarimashalkoubeh, Reza Ranjandish, Elmira Shahrabi and Sebastian Rodriguez. I had a great time in Lausanne with my all friends. I would like to thank to Ece Boran, Ziya Köstereli, Ahmed Doğan, Nezihe Merve Gürel, Alp Yurtsever, Metin Kaycı, Ivan Beretta, Baran Gözcü, Florence Fraigneau, Emine Can, Enver Kılınç, Onur Yürüten, Ruben Braojos, Egeyar Bağcıoğlu, Elodie Lamothe, George Foustouskos and Özlem İpek for their friendship. The friendship of all these people made Lausanne a lovely city to me.

Finally, I would like to present my special thanks to my family, my mother Ayla, my father Ömer and my sisters Büşra, Şeyma and Elif Naz for their love and support during all my life. Particularly I want to thank again to my sister Elif Naz for the figure below that she drew using paint while she was 8 years old. While I was working on my thesis, she was upset that she could not have enough time to play with me. She told that she can help me to finish my works, then we would play some games together. I showed her the video results of my depth estimation system, then I asked her to draw a picture to summarize my work which I can put in my thesis. That was what I told just to make her busy, but now I keep my promise, here it is my beloved sister.



Lausanne, 6 March 2015 Kadir

Abstract

Depth information is used in a variety of 3D based signal processing applications such as autonomous navigation of robots and driving systems, 3D geographic information systems, object detection and tracking, computer games, 3D television, and free view-point synthesis. These applications require high accuracy and speed performances for depth estimation. Depth maps can be generated using disparity estimation methods, which are obtained from stereo matching between multiple images.

The computational complexity of disparity estimation algorithms and the need of large size and bandwidth for the external and internal memory make the real-time processing of disparity estimation challenging, especially for high resolution images. This thesis proposes a high-resolution high-quality multiple-camera depth map estimation hardware. The proposed hardware is verified in real-time with a complete system from the initial image capture to the display and applications. The details of the complete system are presented.

This thesis proposes binocular and trinocular hardware-oriented adaptive window size disparity estimation algorithms, which target high-resolution video with high-quality disparity results. The algorithms are carefully designed to be suitable to real-time hardware implementation by allowing efficient parallel and local processing while providing high-quality results.

The proposed binocular and trinocular disparity estimation algorithms are implemented in hardware. The hardware implementation of the proposed binocular disparity estimation algorithm can process 60 frames per second on a Virtex-5 FPGA at a 1024 × 768 XGA video resolution for a 128 pixel disparity range. The hardware implementation of the proposed trinocular disparity estimation algorithm can process 55 frames per second on a Virtex-7 FPGA for the same resolution and disparity range. The implementation details of these efficient binocular and trinocular disparity estimation hardware are presented. The proposed binocular disparity estimation hardware provides best quality compared to existing high-resolution disparity estimation hardware implementations.

Rectification is an important pre-processing step of the disparity estimation to remove the distortions in the lens and to solve the camera misalignments. A novel compressed-look up table based rectification algorithm and its real-time hardware implementation are presented. The proposed compression scheme is able to fit the look-up-tables into the on-chip BRAMs of the Virtex-5 FPGA. The low-complexity decompression process of the rectification hardware utilizes a negligible amount of LUT and DFF resources of the FPGA while it does not require the existence of external memory.

The first real-time high-resolution free viewpoint synthesis hardware utilizing three-camera disparity estimation is presented. The proposed hardware generates high-quality free viewpoint video at 55 frames per second using a Virtex-7 FPGA at a 1024 × 768 XGA video resolution for any horizontally aligned arbitrary camera positioned between the leftmost and rightmost physical cameras.

The full embedded system of the depth estimation is explained. The presented embedded system transfers disparity results together with synchronized RGB pixels to the PC for application development. Several real-time applications are developed on a PC using the obtained RGB+D results. The implemented depth estimation based real-time software applications are: depth based image thresholding, speed and distance measurement, head-hands-shoulders tracking, virtual mouse using hand tracking and face tracking integrated with free viewpoint synthesis.

The proposed binocular disparity estimation hardware is implemented in an ASIC. The ASIC implementation of disparity estimation imposes additional constraints with respect to the FPGA implementation. These restrictions, their implemented efficient solutions and the ASIC implementation results are presented.

In addition, a very high-resolution (82.3 MP) 360° × 100° omnidirectional multiple camera system is proposed. The hemispherical camera system is able to view the target locations close to horizontal plane with more than two cameras. Therefore, it can be used in high-resolution 360° depth map estimation and its applications in the future.

Keywords: Depth Estimation, Disparity Estimation, Rectification, Free Viewpoint Synthesis, Multiple Camera, Hardware Implementation, Real-Time, FPGA, ASIC.

Résumé

L'estimation de la profondeur (distance mesurée depuis l'observateur par rapport à chaque objet) est utilisée dans de multiples applications de traitement de signaux basés sur la 3D, comme par exemple les systèmes de pilotage automatique de robots et de véhicules, les systèmes d'information géographique 3D, la détection et le suivi d'objets, les jeux d'ordinateurs, la télévision 3D ou encore la synthèse d'images d'un point de vue virtuel. Ces applications nécessitent de grandes performances en précision et en vitesse dans l'estimation de profondeur de champ. En utilisant des méthodes d'estimation de la disparité obtenues à l'aide d'une comparaison entre plusieurs images stéréo, une cartographie complète de cette profondeur peut être calculée.

La complexité de calcul des algorithmes d'estimation de la disparité ainsi que le besoin de mémoires internes et externes de grande capacité et à forte bande passante rendent difficile le traitement en temps réel de l'estimation de la disparité, particulièrement pour les images à haute résolution. Cette thèse propose un hardware d'estimation de la cartographie de profondeur à haute résolution et de haute qualité utilisant plusieurs caméras. Ce dernier est vérifié en temps réel avec un système complet, depuis la capture de l'image initiale jusqu'à l'affichage et ses applications correspondantes. Tous les détails concernant ce système sont présentés.

Cette thèse propose des algorithmes binoculaires et trinoculaires d'estimation de la disparité orientés hardware à l'aide de fenêtres de tailles adaptatives qui ciblent la vidéo de haute résolution avec des résultats de haute qualité pour l'estimation de la profondeur. Les algorithmes sont soigneusement développés pour être adaptés à une implémentation en temps réel, permettant un calcul en parallèle efficace et local, tout en fournissant des résultats de haute qualité.

Les algorithmes d'estimation de disparité binoculaires et trinoculaires proposés sont implémentés en hardware. L'implémentation hardware de l'algorithme d'estimation de la disparité binoculaire proposé peut traiter 60 images par seconde sur une FPGA Virtex-5 dans une résolution vidéo XGA de 1024×768 pixels pour une plage de disparité de 128 pixels. L'implémentation hardware de l'algorithme d'estimation de la disparité trinoculaire proposé peut traiter 55 images par seconde sur une FPGA Virtex-7, pour la même résolution et la même plage de la disparité. Les détails de l'implémentation hardware de ces deux algorithmes sont présentés. Le matériel proposé délivre la meilleure qualité d'estimation de disparité comparée aux implémentations hardware existantes.

La rectification est une étape de prétraitement importante pour estimer la disparité afin de supprimer les distorsions dans les lentilles et de corriger l'alignement imparfait des caméras. Un nouvel algorithme, basé sur table de transcorrespondance (LUT) compressée, ainsi que son implémentation dans un hardware temps réel sont présentés. Un algorithme de compression est proposé et est capable d'accommoder les tables de transcorrespondance dans les BRAMs internes de la FPGA Virtex-5. Grâce à la faible complexité du processus de décompression, le hardware de la rectification utilise une quantité négligeable de LUTs et de DFFs de la FPGA, tout en ne nécessitant aucune mémoire externe.

Le premier hardware en temps réel et à haute résolution pour la synthèse de point de vue virtuel utilisant l'estimation de la disparité à trois caméras est présenté. Le hardware proposé génère une vidéo à point de vue virtuel de haute qualité à 55 images par seconde, utilisant la FPGA Virtex-7 à une résolution video XGA de 1024×768, pour toute caméra virtuelle alignée horizontalement entre les caméras physiques situées aux extrémités gauches et droites.

Le système embarqué complet d'estimation de la profondeur est présenté. Ce dernier transfère les résultats de manière synchrone avec les pixels RGB vers le PC pour le développement d'applications. Plusieurs applications temps réel sont développées sur PC en utilisant les résultats RGB+D obtenus. Les applications software implémentées sur l'estimation de profondeur de champ en temps réel sont les suivantes : seuillage d'image de profondeur de champ, mesure de la vitesse et de la distance, suivi de la tête, des mains et des épaules, souris virtuelle en utilisant le suivi de la main, suivi de la tête intégré avec la synthèse de point de vue virtuel.

Le hardware d'estimation de la profondeur binoculaire est implémenté dans un ASIC. Cette implémentation en ASIC impose des contraintes additionnelles par rapport à l'implémentation sur FPGA. Ces restrictions et leurs solutions efficaces, ainsi que les résultats de cette implémentation, sont présentés dans cette thèse.

De plus, un système omnidirectionnel 360° × 100° à caméras multiples de très haute résolution (82.3 MP) est proposé. Ce système de caméra hémisphérique est capable de voir les emplacements cibles proches du plan horizontal avec plus de deux caméras. Il peut donc être utilisé dans une estimation de profondeur de 360° à très haute-résolution dans le futur.

Mots clefs : Estimation de la profondeur, Estimation de la disparité, Rectification, Synthèse de point de vue virtuel, caméras multiples, implémentation hardware, temps réel, FPGA, ASIC.

Contents

Acknowledgements **i**

Abstract (English/Français) **iii**

List of figures **xi**

List of tables **xvii**

1 Introduction **1**
- 1.1 Large Angle of View Image Capture Systems 2
- 1.2 Depth Estimation Systems 4
- 1.3 Contribution of the Thesis 8
  - 1.3.1 Proposed Depth Map Estimation Systems 8
  - 1.3.2 Proposed Large Angle of View Image Capture and Reconstruction Systems 14
- 1.4 Thesis organization 14

2 State of the Art **15**
- 2.1 Hemispherical Multiple Camera Image Reconstruction Systems 15
- 2.2 Binocular Stereo Matching Algorithms and Their Hardware Implementations . 19
  - 2.2.1 Camera Calibration 22
  - 2.2.2 Rectification 26
  - 2.2.3 Matching Cost Computation 27
  - 2.2.4 Cost Aggregation 30
  - 2.2.5 Disparity Selection and Optimization 30
  - 2.2.6 Disparity Refinement 32
- 2.3 Trinocular Stereo Matching Algorithms and Their Hardware Implementations . 33
- 2.4 Thesis Goals 36

3 Binocular Adaptive Window Size Disparity Estimation Algorithm and Its Hardware Implementation **39**
- 3.1 Binocular Hardware-Oriented Adaptive Window Size Disparity Estimation Algorithm 39
  - 3.1.1 Window Size Determination 40

3.1.2 Disparity Voting 41
3.1.3 Disparity Refinement 43
3.2 Hardware Implementation of Proposed Binocular AWDE Algorithm 44
3.2.1 Overview 44
3.2.2 Data Allocation and Disparity Voting 46
3.2.3 Disparity Refinement 51
3.3 Iterative Refinement for the Enhanced AWDE implementation 52
3.4 Implementation Results 54

4 Trinocular Adaptive Window Size Disparity Estimation Algorithm and Its Hardware Implementation 63
4.1 Trinocular Hardware-Oriented Adaptive Window Size Disparity Estimation Algorithm 63
4.2 Trinocular Adaptive Window Size Disparity Estimation Hardware 65
4.3 Implementation Results 69

5 Compressed Look-Up-Table Based Rectification Algorithms and Their Hardware Implementations 73
5.1 Compressed Look-Up-Table based Rectification Algorithm 74
5.2 Real-Time De-Compression Hardware of CLUTR 82
5.3 Limitations of the CLUTR 85
5.4 Enhanced Compressed Look-Up-Table based Rectification Algorithm 88
5.5 Real-Time De-Compression Hardware of E-CLUTR 91
5.6 Real-Time Hardware of Caltech Rectification 94
5.7 Implementation Results 96

6 Embedded System for Depth Map Estimation 103
6.1 The Overview of the Depth Map Estimation System 103
6.2 Video Processing Hardware Cores and Circuits 104
6.3 Peripherals of the Embedded System 109
6.4 Embedded Software 116

7 Hardware and Software based Applications of Disparity Estimation 119
7.1 Free View Synthesis Hardware Using Trinocular Disparity Estimation 119
7.1.1 Hardware-Oriented Three-Camera Free Viewpoint Synthesis Algorithm 119
7.1.2 Real-Time Free Viewpoint Synthesis Hardware 122
7.1.3 Implementation Results 125
7.2 System GUI and Software Based Real-Time Applications 127
7.2.1 Capture and the display of the video 127
7.2.2 Front-end of the GUI 128
7.2.3 Software based real-time applications 129

8 ASIC Implementation of Binocular Disparity Estimation 133
- 8.1 Disparity Estimation ASIC Specifications . . . 133
- 8.2 Adaptation of Disparity Estimation Hardware for ASIC . . . 141
- 8.3 ASIC Design . . . 144
  - 8.3.1 Synthesis . . . 145
  - 8.3.2 Placement and Routing . . . 146
- 8.4 Test System . . . 151

9 High-Quality Omnidirectional Multi-Camera Systems 153
- 9.1 Enhanced Omnidirectional Image Reconstruction Algorithm and Its Real-Time Hardware Implementation . . . 153
  - 9.1.1 Equal Area Distribution Algorithm . . . 153
  - 9.1.2 Hardware Implementation of Equal Area Distribution Algorithm . . . 157
  - 9.1.3 Implementation Results . . . 162
- 9.2 Giga-Eye Camera . . . 163
  - 9.2.1 System Parameters and Requirements . . . 164
  - 9.2.2 System Architecture . . . 167
  - 9.2.3 Implementation Results . . . 170

10 Conclusion 175

A High-Resolution Visual Results of Proposed Systems 183
- A.1 Visual Results of the Depth Estimation System . . . 183
- A.2 Visual Results of the Giga-Eye Multiple-Camera System . . . 183

Bibliography 192

Abbreviations 193

Curriculum Vitae 197

List of Figures

1.1 Simplified block diagram of the complete depth map estimation system. 9

2.1 Ladybug3 multiple camera system 16
2.2 R7 multiple camera system of Google for Street View application 16
2.3 Panoptic camera 17
2.4 Hemispherical structure with multiple floors 18
2.5 (a) hemispherical surface showing θ and φ angles and unit vectors used in omnidirectional image reconstruction calculations, (b) ω vector and corresponding unit vectors; t (focus direction of the camera), u (vertical direction in the pixel representation of the camera), v (horizontal direction in the pixel representation of the camera) (c) 2D reconstruction using linear pixelization . 18
2.6 Inverse relationship between the depth and disparity. 19
2.7 Geometrical relationship between disparity and depth. 20
2.8 Distance measurement precision of disparity estimation based depth estimation. 21
2.9 Coordinate systems for pinhole camera model. 23
2.10 Mathematical alignment of the two cameras into an unique image plane. 26
2.11 Census transform computation. 29
2.12 Disparity selection. 31
2.13 Collinear placement of three cameras for disparity estimation 33
2.14 Triangular placement of three cameras for disparity estimation 33
2.15 Geometrical relationship between the cameras of the trinocular stereo system . 34

3.1 9 selected pixels in a block for BW-SAD calculation. 49 pixels in a block are searched in parallel in hardware. 40
3.2 49 selected pixels of adaptive windows (yellow (1): 7×7, green (2): 13×13 and blue (3): 25×25). 41
3.3 Examples for selecting 17 contributing pixels for 7×7, 13×13 and 25×25 window sizes during the disparity refinement process (yellow (1): 7×7, green (2): 13×13 and blue (3): 25×25). 43
3.4 Top level block diagram of the disparity estimation module. 45
3.5 Timing diagram of the system. 46
3.6 Block diagram of the Reconfigurable Data Allocation Module. 46

3.7 DFF Array and the Weaver (yellow: 7 × 7, green: 13 × 13 and blue: 25 × 25). . . . 47
3.8 Block diagram of the Reconfigurable Computation of Metrics. 49
3.9 Processing Scheme (“x” indicates 9 selected pixels in a block for BW-SAD calculations). 49
3.10 Disparity Refinement-Array of the Disparity Refinement Module (yellow (1): 7×7, green (2): 13 × 13 and blue (3): 25 × 25). 51
3.11 Processing Element of the Disparity Refinement Module. The Highest Frequency Selection Module includes seven of these DR-PE elements. 52
3.12 DR-Array of the Iterative Disparity Refinement Module (yellow line: 7 × 17 candidates for 7 × 7 window, green line: candidates for 13 × 13, and blue line: candidates for 25 × 25). 53
3.13 Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmark Clothes. Black regions in the ground truths are not taken into account for the error computations as explained in Middlebury evaluation. (a) left image (b) ground truth (c) DE result of AWDE (d) DE result of AWDE-IR . . 58
3.14 Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmark Art. Black regions in the ground truths are not taken into account for the error computations as explained in Middlebury evaluation. (a) left image (b) ground truth (c) DE result of AWDE (d) DE result of AWDE-IR 59
3.15 Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmark Aloe. Black regions in the ground truths are not taken into account for the error computations as explained in Middlebury evaluation. (a) left image (b) ground truth (c) DE result of AWDE (d) DE result of AWDE-IR 60
3.16 Visual disparity estimation results of AWDE and AWDE-IR algorithms for the 1024 × 768 resolution pictures captured by the implemented stereo camera system. The ground truth for these images is not available. (a) left image (b) right image (c) DE result of AWDE (d) DE result of AWDE-IR 61

4.1 Matching directions of the T-AWDE algorithm 64
4.2 Block diagram of the full system. 66
4.3 I^{2}C multiplexing hardware (resistors and level-shifters are not drawn) 67
4.4 Block diagram of trinocular DE hardware 68
4.5 Parallel processing scheme for two pairs 69
4.6 Processing element of the fusion module (PE-F). The fusion module includes 49 PE-F elements. 70
4.7 DE results obtained by MATLAB for Middlebury benchmarck image set “Bowling2” (a) Left Image (b) Center Image (c) Right Image (d) Ground Truth (black pixels are ignorable) (e) IR-AWDE for center-left (18.01%) (f) T-AWDE (9.41%) (g) IR- AWDE for center-right (15.60%) 71
4.8 Real-time snapshots captured by the proposed system. A ground truth for these images is not available. (a) Left Image (b) Center Image (c) Right Image (d) AWDE for center-left (e) T-AWDE (f) AWDE for center-right 72

5.1 Inverse mapping with fractional precision coordinates. Corners indicate integer pixel coordinates. 74
5.2 Forward mapping with fractional precision coordinates. 75
5.3 Flow-chart of the proposed compressed look-up-table based stereo image rectification process. 76
5.4 Third step of the compression flow (a) Due to the 3D rotation, some of the pixels in the rectified image cannot be related to their source pixels in the original image (b) selection of nearest source pixels from fractional inverse mapping (c) extraction of forward mapping with integer coordinates. 78
5.5 Integer coordinate precision forward mapping look-up-tables after the third step. Regular orders are shown with red ellipses (a) mapping of Y coordinates (b) mapping of X coordinates. 79
5.6 Coded regular orders after the third step (a) coded mapping of Y coordinates (b) coded mapping of X coordinates. 79
5.7 Look-up-tables after filling the NT pixels using the fourth step (a) mapping of Y coordinates (b) mapping of X coordinates. 79
5.8 Coded regular orders after filling the NT pixels using the fourth step (a) coded mapping of Y coordinates (b) coded mapping of X coordinates. 79
5.9 Visualization of the reason for the voids in the rectified image (a) inverse mappings with fractional coordinates (b) forward mapping with integer coordinate. 80
5.10 Filling the voids in the rectified image in the fifth step (a) finding the source location of a pixel at one row above the void (b) marking the source pixel as double targeted pixel. 80
5.11 Coding the behavior of breakpoints at the sixth step (a) coded mapping of Y coordinates (b) coded mapping of X coordinates. 80
5.12 Concatenation of the locations and behaviors at the seventh step (a) for the mapping of Y coordinates (b) for the mapping of X coordinates. 81
5.13 Utilization example of the proposed rectification hardware. 83
5.14 Top-level block diagram of the proposed rectification hardware of CLUTR. . . . 83
5.15 Block diagram of the proposed rectification hardware for decompressing the table of Y coordinates. Pipeline stages are presented with dashed lines. 84
5.16 Block diagram of the proposed rectification hardware for decompressing the table of X coordinates. 84
5.17 Visualization of the breakpoint frequency capacity of the X and Y coordinate mappings (a) breakpoints for the mapping of Y coordinates (b) breakpoints for the mapping of X coordinates 86
5.18 Visualization of the reason for the frequent breakpoints (a) finding the source locations of three pixels that are targeting one row above of the three consecutive voids (b) four break-points in consecutive five locations. 87
5.19 Visualization of the reason for the voids which can not be filled by CLUTR. . . . 88

5.20 Filling the voids in the rectified image in the fifth step (a) DT option of CLUTR (b) DT options of E-CLUTR. 89

5.21 Reducing the frequency of breakpoints using multiple DT options of E-CLUTR (a) finding alternative source locations for void pixels of rectified image (b) reduced frequency of breakpoints for the same row of the look-up-table of Y coordinates. 89

5.22 Vertically adjacent void pixels can be filled by E-CLUTR using multiple DT options. 90

5.23 Brief representation for the concatenation of the locations and behaviors it the seventh step of E-CLUTR for the mapping of Y coordinates. 91

5.24 Concatenation of the locations and behaviors at the seventh step for E-CLUTR for the mapping of Y coordinates (a) Concatenation scheme for DT options (b) Concatenation scheme for the breakpoint conditions for changing the last targeted row. 91

5.25 Concatenation of the locations and behaviors at the seventh step of E-CLUTR for the mapping of X coordinates. 91

5.26 Top-level block diagram of the proposed rectification hardware of E-CLUTR. . 92

5.27 Block diagram of the proposed rectification hardware for decompressing the table of X coordinates. 93

5.28 Block diagram of the proposed rectification hardware for decompressing the table of Y coordinates. Pipeline stages are presented with dashed lines. 93

5.29 Block diagram of the proposed hardware implementation for Caltech rectification algorithm. 95

5.30 Visual results of the first test using roughly aligned cameras: Original images have still distortions as observed near the lamp, bag, folder and cup; horizontal epipolar lines are displayed in red near the edge of these objects (a) left image (b) right image. 97

5.31 Visual results of the first test using roughly aligned cameras: The proposed CLUTR algorithm corrects distortions (a) left image (b) right image. 97

5.32 Visual results of the first test using roughly aligned cameras: Breakpoint locations obtained by CLUTR for the left image (a) breakpoints of the targeted Y coordinates; coded row-by-row. (b) breakpoints of the targeted X coordinates; coded column-by-column. 98

5.33 Visual results of the second test using excessively misaligned cameras: Original images have excessive distortions as observed on the lines (a) left image (b) right image. 99

5.34 Visual results of the second test using excessively misaligned cameras: E-CLUTR corrects distortions as observed on the lines (a) left image (b) right image. . . . 99

5.35 Visual results of the second test using excessively misaligned cameras: Breakpoint locations of the left image (a) breakpoints of the targeted Y coordinates; coded row-by-row. (b) breakpoints of the targeted X coordinates; coded column-by-column. 100

6.1 Demonstration of the system set-up. (a) close-up to the three cameras and the FPGA board (b) full demonstration system also including a PC and large-screen TV. 103
6.2 Block diagram of the full system setup. 105
6.3 RGB component values in function of the multiplied disparity values, used for the color representation. 108
6.4 Snapshot of the DE result of binocular disparity estimation system. The hardware is able to switch between to representations: (a) Gray scale disparity map (brighter color is closer, darker color is further), (b) Color mapped disparity map (hotter color is closer, colder color is further) 109
6.5 Block-diagram of the interface between the disparity estimation hardware and the DDR3 memory. 114
6.6 Block-diagram of the Output Selection circuit for three possible outputs. 114
6.7 Flowchart representation of the logic block functionality. 115
6.8 Top-level schematic of software accessible registers. 117

7.1 Definition of the three-camera free viewpoint synthesis setup. 120
7.2 The illustration of free viewpoint synthesis algorithm that utilizes trinocular disparity estimation. (Each square grid represents 4×4 pixels. Only one out of four column/row grids are represented to improve the clarity of the representation). 121
7.3 Block diagram of the free viewpoint synthesis system. 123
7.4 Rendering hardware (pipeline stages are not shown). 124
7.5 Real-time snapshots captured by the proposed system. (a) Left image (b) Center image (c) Right image (d)IR-AWDE for center-left (e) T-AWDE (f)IR-AWDE for center-right (g) synthesized free viewpoint image for an arbitrary camera located at q=(-0.5) (h) synthesized free viewpoint image for an arbitrary camera located at q=(0.5) (images best viewed in high resolution from the pdf files) 126
7.6 Screen shot of the stream viewer software running on Linux. 128

8.1 Depth estimation system that is utilizing two cameras 134
8.2 Modules present in the Disparity Estimation ASIC 136
8.3 top level inputs and outputs of the disparity estimation final ASIC. 136
8.4 Floorplan of the first design. 147
8.5 Floorplan of the second design. 148
8.6 Floorplan of the third design. 149
8.7 The three disparity estimation ASIC designs after P&R. For visual comparison, the proportions have been kept. (a)First design (b)Second design (c) Third design 150
8.8 The final ASIC design after P&R and placing the IO pads 151
8.9 Top-level diagram of the test system of ASIC. One VC707 board is connected to the test-board containing the ASIC package. 152
8.10 The emulation and verification set-up of the ASIC. 152

9.1 256 × 1024 resolution omnidirectional 2D reconstructions using a) *Linear pixelization* b) *Equal area distribution* 155
9.2 Captured images from two different cameras a) top camera b) side camera . . 155
9.3 Hemispheric structure for resolution calculation in terms of ppaa 156
9.4 Relationship between N and - angle for different methods of pixelization 156
9.5 Block diagram of omnidirectional image reconstruction algorithms 158
9.6 Block diagram of the Concentrator FPGA 159
9.7 Block diagram of the omnidirectional vision reconstruction unit inside the image processing and application block 160
9.8 θ and φ angle generation module architecture for the OIR and EOIR implementations 160
9.9 Architecture of the camera select and distance generation sub-block for the EOIR algorithm 160
9.10 Reconstructed 360° views 162
9.11 Top-level block diagram of the system hardware architecture 169
9.12 The complete omnidirectional imaging and recording system (Giga-Eye), overall system dimensions are 56x48x78 cm 171
9.13 Omnidirectional image obtained with the Giga-Eye system at 21.6 MP resolution showing the central campus square of EPFL, and two selected details (sub-regions) in this image. This omnidirectional image corresponds to one single frame of the 30 fps video obtained by the system. 172
9.14 Omnidirectional image obtained with the Giga-Eye system at 82.3 MP resolution. This omnidirectional image corresponds to one single frame of the 9.5 fps video obtained by the system. Flying plane and the moving car are shown in sub-windows172
9.15 Measured coverage map of the omnidirectional imaging system showing a high pixel redundancy especially close to the equator. The color labels indicate the number of the overlapping individual camera AOVs 173

List of Tables

3.1 Parameters of the AWDE 54
3.2 Disparity Estimation Performance Comparisons. Error rates (%) are provided compared to DE ground truths of the benchmark pictures. 55
3.3 Hardware Performance Comparison 57

5.1 Bit-size Parameters of the Caltech rectification hardware 95
5.2 Numerical results of the first test using roughly aligned cameras: PSNR (*dB*) with the rectified images produced by Caltech rectification algorithm 98
5.3 Numerical results of the first test using roughly aligned cameras: PSNR (*dB*) Comparison of the Disparity Estimation Results Using Different Disparity Estimation Algorithms 98
5.4 Numerical results of the second test using excessively misaligned cameras: PSNR (*dB*) with the rectified images produced by Caltech rectification algorithm . . . 100
5.5 Hardware Resource Comparison of the Rectification Hardware Implementations 101

8.1 ARM SRAMs Area Configuration Comparison (in μm^2) 137
8.2 Approximate power consumption for the different SRAM types of the ASIC design (in mW). 137
8.3 Register list and default values. *1 = always 1; 0 = always 0; d = programmable; – = read only 138
8.4 Core Registers descriptions 139
8.5 Area report of the three synthesized hardware. 146
8.6 Total power of the DE ASIC (in mW). 149
8.7 Dimensions of ASIC designs 150

9.1 Surface Area Coverage (%) with respect to θ angles (radians) for Possible Pixelization Methods. 154
9.2 Resolution values $[ppaa]$ for different θ angles 156
9.3 Implementation Results 162
9.4 Properties of the omnidirectional imaging system. 164
9.5 System Constraints to generate 30 fps 21.6 MP Omnidirectional Video. 166
9.6 System Constraints to generate 9.5 fps 82.3 MP Omnidirectional Video. 167

9.7 Comparison of the Giga-Eye with existing high-resolution omnidirectional camera systems. 174

1 Introduction

The stereoscopic depth perception is first described by the British scientist and inventor Charles Wheatstone in 1838. Wheatstone made a device called stereoscope to provide stereoscopic vision to a human observer. He used a pair of mirrors mounted with an angle of 45° with respect to the two drawings at the sides to reflect the drawings to the observer. He showed that brain makes the fusion of these drawings and perceives a three dimensional image. In 1850s, the American physician Oliver Wendell Holmes used glasses with two prismatic lenses to provide stereocopic vision. The invention of this portable device led to the development of rotating cardboard disks with image pairs which became a popular entertainment and virtual tourism tool until mid-20^{th} century.

In the very early times of the stereoscopy, pictures were taken by a single camera, moving it along its horizontal axis. Later, stereo cameras were manufactured following an increasing demand of stereo vision. In the late 19^{th} century, a stereo camera called Verascope was invented. In 1967, a color stereo camera called Stereo Realist was invented. This invention was followed by the development of 3D displayers for color images and videos.

Color anaglyph-based 3D glasses were popular in the late 20^{th} century. These glasses include two chromatically opposite colors (red and cyan) in each glass, which filter the left and right images separately for each eye. The brain operates the fusion and forms a three dimensional color scene.

In the late 20^{th} century, the artificial presence in a computer-generated or real-world environment, so-called virtual reality systems, became an important demand of medical, gaming and military applications and cinema industry. This demand was supported by the advances in digital photography and video which led to the development of advanced 3D vision and display systems.

Active shutter based glasses are mainly used today for 3D cinematography. Active shutter glasses works by rapidly showing two images to the left or right eye by blocking one of them. Meanwhile the image on the monitor shifts to the view of right or left eye synchronously to form stereoscopic view.

Head-mounted displays (HMD) are one of the popular 3D display devices available today. HMDs include optics and a wearable digital monitor to display computer generated video or movie transferred from a PC. HMDs may utilize active-shutter based multiplexing or side by side multiplexing to provide depth perception. HMDs are mainly used for military and video gaming applications to view the scene with 360° × 90° angle of view (AOV) in 3D.

3D TVs can utilize different type of methods to provide depth perception. The previously developed 3D TVs needed color anaglyph based 3D glasses for proper operation. Currently, most of the 3D TVs operate with active shutter based glasses. However, the requirement to wear 3D glasses is usually perceived uncomfortable which impedes any wide usage of stereoscopic 3D displays. Recently, autostereoscopic displays have emerged to provide glasses-free depth perception [1]. The autostereoscopic technology or so-called free viewpoint television (FVT) utilizes physical parallax barriers to display multiple views that provide a stereo pair regardless of the positions of the viewer. Although increasing the number of views decrease the limited distance range of these displays, broadcasting all the views is challenging in terms of bandwidth limitations. [2] demonstrates that standard 2D displays can act as 3D displays by deceiving human perception without using intricate autostereoscopic technique if the free view is generated efficiently considering the position of the head of the viewer. [2] presents the efficiency of this method using computer generated images and pre-computed depth map.

The emerging technology of virtual reality applications should be supported not only by the displays, but also from high-quality video capture, coding and processing technologies should be developed. Large AOV image capture systems and depth measurement are important requirements of advanced virtual reality and video processing applications, which should progress in parallel to the advances on display technologies.

1.1 Large Angle of View Image Capture Systems

In order to support the wide AOV requirement of virtual reality applications, recent research has mainly focused on developing complex lenses, mirrors and multiple-camera systems. Fish-eye lenses are major lens types that provide an AOV between 180° to 200° using a single camera. However, distortion in the edges of the images limits the use of single complex lenses.

In [3], a camera system with a convex mirror is developed in order to create 360° omnidirectional images. This method does not require intricate hardware or software for constructing omnidirectional images, and is thus widely cited in literature related to omnidirectional image reconstruction. In [4], the real-time panoromic image reconstruction hardware utilizing a system consisting of a convex mirror placed over a single camera is presented. The hardware presented in [4] provides 360° panoramic images of 3200×768 resolution at 40fps. The presented hardware fits into a Spartan 3 FPGA. Nevertheless, the resolution of omnidirectional images is limited to the resolution of a single camera. Moreover, it is incapable of showing the overall top view since the AOV of the camera is restricted to the area located below the mirror. Also, the convex mirror provides distortion at the edges of the image. Furthermore, the platform cannot be used for depth estimation and 3D reconstruction due to its single observation point feature.

An alternate method enabling wide AOV image acquisition is based on a single-camera multiple-lens optical sensors system [5–7]. Devices taking advantage of this technique and presenting very large AOVs are mainly inspired by the insect faceted eyes. Faceted insect eyes consist of hundreds to thousands channels, each called ommatidia [8]. These channels capture light within a defined angle and transmit it to light sensitive receptor cells. In [5], artificial insect eyes, cluster eyes and artificial apposition compound eyes (APCO) are fabricated using wafer-scale techniques. In [6], the AOV of eye clusters and APCOs is improved to 124° by using two additional micro-lenses in each channel, thereby causing significant distortion. In [7], a system named Krill-eye is presented. The Krill-eye utilizes 180 gradient-index lenses and hence, provides distortion-free 180° AOV. However, designing a Krill-eye composed of 180 lenses is not trivial, while the AOV remains significantly limited under 360°.

Multiple-camera systems are proposed as an alternative method, which enables obtaining wide AOVs. The synchronization of the cameras and the very large data bandwidth impose challenges for the implementation of these systems. Some camera array systems were developed for recording; the recordings were processed at a later time offline on a PC[9]. Many multiple-camera systems are only used for object tracking and detection, where the goal does not involve the creation of single omnidirectional images [10–12]. Each image obtained from multiple cameras should be appropriately combined in order to prevent unrealistic discontinuity. An image mosaic algorithm and its real-time hardware implementation are presented in [13]. This multi-camera system supports three cameras, and is capable of reaching an AOV smaller than 180°. The system operates in real-time benefiting from the utilization of digital signal processors (DSPs). Furthermore, the resulting omnidirectional image does not exhibit significant distortion or discontinuity.

In [14], a planar multiple-camera system composed of 100 cameras is presented as a solution to reach high resolution, high frame rates and high AOV. The presented platform consisting of 100 FPGAs is aimed to record large amounts of data to process offline, and thus provides limited local processing at the camera level. However, it reaches less than 180° AOV due to its planar structure.

A system composed of multiple cameras is presented in [15] targeting an immersive cockpit application. The system reaches a 150° horizontal and 110° vertical FOVs by utilizing eight 320×240 pixel CCD cameras, and a PC-based acquisition system. Another omnidirectional image capture device that uses six 1024×768 pixel CCD sensors that are placed in a cubic arrangement is reported in [16]. Each camera has a wide-angle lens, and a custom calibration and image generation method is developed. Currently, the Google Street View camera is one of the most well-known 360° imaging systems [17]. The system presented in [17] utilizes 15 5MP cameras to cover more than 80% of the full sphere with a panoramic image resolution of 8192×4096 by capturing pictures every 2.5 second.

Recently, a spherical multi-camera vision sensor called the Panoptic camera and a specific omnidirectional image reconstruction algorithm (OIR) have been proposed to enable 360° AOV with a high resolution [18]. The Panoptic camera is a biologically-inspired multi-camera vision sensor mimicking the eyes of flying insects where multiple imagers, each with a distinct focal point, are distributed over a hemisphere. The details of real-time hardware implementation of the OIR is presented in [19]. The system presented in [19] utilizes 40 cameras each with a 352×288 pixel resolution, and creates 256×1024 pixel resolution omnidirectional images in real-time. This video capture system can artificially make the viewer feel himself being at the place of the Panoptic camera using HMD devices or prospective future 360° × 90° hemispheric displays.

1.2 Depth Estimation Systems

Depth is a strong component of human vision. The stereoscopic imaging and utilization of glasses can provide depth perception to the user without the requirement of depth measurement. However, its measurement is required for many recent advanced virtual reality applications and 3D-based smart vision systems. Depth estimation is an algorithmic step in a variety of applications such as autonomous navigation of robot and driving systems [20], 3D geographic information systems [21], object detection and tracking [22], medical imaging [23], computer games and advanced graphic applications [24], 3D holography [25], 3D television [26], multiview coding for stereoscopic video compression [27], and disparity-based rendering [28]. These applications require high accuracy and speed performances for depth estimation.

Depth estimation can be performed by exploiting six main techniques: time-of-flight (TOF) camera, LIDAR sensors, radars, structured infrared light projection, learning based single camera algorithms and stereo camera. A TOF camera easily measures the distance between the object and camera using a sensor and projection of light pulse, circumventing the need of intricate digital image processing hardware [29]. However, it does not provide efficient results when the distance between the object and camera is high. Moreover, the resolution of TOF cameras is usually very low (200×200) [29] in comparison to the Full HD display standard (1920×1080). Multiple TOF systems can not be oriented to the same location due to interference of the multiple projections. Furthermore, their commercial price is much higher than the CMOS and CCD cameras. LIDAR sensors are similar to the TOF cameras, but they compute the depth image by using laser scanning mechanism [30]. LIDAR sensors are also very expensive compared to CMOS and CCD cameras. Due to laser scanning hardware, LIDAR sensors are heavy and bulky devices. Laser scanner based systems can measure the distance of the very far objects. However, the strength of the laser may damage the human eye if it is looking in the direction of the source of laser light for a long time. Radars utilize radio waves to measure the distance of the far and big size objects [31]. They are not useful to provide high resolution for close and small objects. Structured infrared light projection-based depth measurement systems such as Microsoft Kinect [32] and Structure Sensor of Occipital [33] provide high-quality results even for the textureless objects and in dark environment. However, they are not able to provide high quality results under sunlight due to interference of the infrared lights of the sun. Both Kinect and Structure Sensor can provide depth estimation results in VGA resolution (640×480). Their resolution is lower than the CMOS and CCD cameras since the pixel size of infrared sensors is large. Their depth measurement range is limited by the projection power of the infrared beamer. For example, Kinect is not able to measure the distance of the object if they are further than 7 m. In addition, multiple structured infrared light projection systems can not be oriented to the same location due to interference of the multiple projections. Single camera based depth estimation algorithms require very large training sets and learning approach [34]. Indeed, learning and intelligence are significant features of human vision that provides depth perception even with a single eye. However, implementing a high-quality and real-time depth estimation system is very challenging using a single camera since these algorithms are computationally intensive and require very large data sets. Consequently, in order to compute the depth map for real-time video processing applications, the majority of research focus on extracting the disparity information using two or more synchronized images taken from different viewpoints, using CMOS or CCD cameras [35].

Disparity estimation (DE) based depth estimation systems can measure the distance of the objects even if the objects are very close or far by mechanically adapting distance between the stereo cameras. Multiple DE systems can be used in same environment since they don't project any light. In addition to indoor environment, they can be also used in outdoor, since sunlight does not make interference problem. Possible damage of laser light on human eye is not a case for DE based depth measurement systems since they require only passive sensors. However, implementation of real-time and high-quality DE for high resolution (HR) video is challenging due to its computational complexity.

Many disparity estimation algorithms have been developed with the goal to provide high-quality disparity results. These are ranked with respect to their performance in the evaluation of Middlebury benchmarks [35]. Although top performer algorithms provide impressive visual and quantitative results [36–38], their implementations in real-time HR stereo video are challenging due to their complex multi-step refinement processes or their global processing requirements that demand huge memory size and bandwidth. For example, the AD-Census algorithm [36], currently the top published performer, provides successful results that are very close to the ground truths. However, this algorithm consists of multi disparity enhancement sub-algorithms, and implementing them into a mid-range FPGA is very challenging both in terms of hardware resource and memory limitations.

Various hardware architectures that are presented in literature provide real-time DE [39–45]. Some implemented hardware architectures only target CIF or VGA video [39–42]. The hardware proposed in [39] only claims real time for CIF video. It uses the Census transform [46] and currently provides the highest quality disparity results compared to real time hardware implementations in ASICs and FPGAs. The hardware presented in [39] uses the low-complexity Mini-Census method to determine the matching cost, and aggregates the Hamming costs following the method in [36]. Due to the high complexity of cost aggregation, the hardware proposed in [39] requires high memory bandwidth and intense hardware resource utilization, even for Low Resolution (LR) video. Therefore, it is able to reach less than 3 frames per second (fps) when its performance is scaled to 1024×768 video resolution and 128 pixel disparity range.

Real-time DE of HR images offers some crucial advantages compared to low resolution DE. First, processing HR stereo images increases the disparity map resolution which improves the quality of the object definition. Better object definition is essentially important for a variety of high-quality video processing applications such as object detection and tracking. Second, DE of HR stereo images offers the capacity to define the disparity with sub-pixel efficiency compared to the DE for LR image. Therefore, the DE for HR provides more precise depth measurements than the DE for LR.

Despite the advantages of HR disparity estimation, the use of HR stereo images presents some challenges. Disparity estimation needs to be assigned pixel by pixel for high-quality disparity estimation. Pixel-wise operations cause a sharp increase in computational complexity when the DE targets HR stereo video. Moreover, DE for HR stereo images requires stereo matching checks with larger number of candidate pixels than the disparity estimation for LR images. The large number of candidates increases the difficulty to reach real-time performance for HR images. Furthermore, high-quality disparity estimation may require multiple reads of input images or intermediate results, which poses severe demands on off-chip and on-chip memory size and bandwidth especially for HR images.

The systems proposed in [43–45] claim to reach real time for HR video. Still, their quality results in terms of the HR benchmarks given in [35] are not provided. [43] claims to reach 550 fps for 80 pixel disparity range at a 800×600 video resolution, but it requires high amount of hardware resources. In addition, [43] claims that their hardware implementation operates at 511 MHz using a Stratix IV FPGA, however without providing detail information related to the architecture and design that enable this high performance. A simple edge-directed method presented in [44] reaches 50 fps at a 1280×1024 video resolution and 120 pixel disparity range, but does not provide satisfactory DE results due to a low-complexity architecture. In [45], a hierarchical structure with respect to image resolution is presented to reach 30 fps at a 1920×1080 video resolution and 256 pixel disparity range, but it does not provide high quality DE for HR.

More than two cameras can be used to improve the depth map. As presented in [47], trinocular DE solves most of the occlusion problems present in a single-pair camera system since any occluded region in a matched stereo pair (center-left) is not occluded in the opposite matched pair (center-right). Moreover, the double-checking scheme of trinocular DE improves binocular DE results even for unoccluded regions and provides correct disparity results even if the object is located in the left or right edge of the center image.

A limited number of trinocular disparity estimation hardware implementations are presented in the literature [48, 49]. The hardware presented in [48] enables handling 52 fps on an Altera Cyclone-IV FPGA at a 640×480 video resolution. The hardware presented in [49] uses a triangular configuration of three cameras and enables handling 30 fps on a Xilinx Virtex-4 FPGA at a 320×240 video resolution for a 64 pixels disparity range.

Camera calibration and image rectification are important pre-processing parts of the DE. The stereo matching process compares the pixels in the left and right images and provides the disparity value corresponding to each pixel. If the cameras could be perfectly aligned parallel and the lenses were perfect, without any distortion, the matching pixels would be located in the same row of the right and left images. However, providing a perfect set-up is virtually impossible. Lens distortion and camera misalignments should be modeled and removed by internal and external stereo camera calibration and image rectification processes [50].

Many real-time stereo-matching hardware implementations [39, 43, 51] prove their DE efficiency using already calibrated and rectified benchmarks of the Middlebury evaluation set [52], while some do not provide detailed information related to the rectification of the original input images [45]. In a system that processes the disparity estimation in real-time, image rectification should also be performed in real-time. The rectification hardware implementation presented in [53] solves the complex equations that model distortion, and consumes a significant amount of hardware resources.

A look-up-table based approach is a straightforward solution that consumes a low amount of hardware resources in an FPGA or ASIC [54–56]. In [54–56], the mappings between original image pixel coordinates and rectified image pixel coordinates are pre-computed and then used as look-up-tables. Due to the significant amount of generated data, these tables are stored in an external memory such as a DDR or SRAM [54, 55]. Using external storage for the image rectification process may increase the cost of the disparity estimation hardware system or impose additional external memory bandwidth limitations on the system. In [56], the look-up-tables are encoded to consume 1.3 MB data for 1280×720 size stereo images with a low-complexity compression scheme. This amount of data requires at least 295 Block RAMs (BRAM) without considering pixel buffers, thus it can only be supported by the largest Virtex-5 FPGAs or other recent high-end FPGAs.

1.3 Contribution of the Thesis

In this thesis, novel, efficient and high-performance systems for depth map estimation and large AOV image capture are presented. The presented systems can be used for the development of advanced virtual reality applications and 3D-based video processing systems.

1.3.1 Proposed Depth Map Estimation Systems

This thesis proposes binocular and trinocular disparity estimation algorithms and their efficient hardware implementations. The systems that are utilizing two cameras and three cameras are verified in real-time. The simplified block diagram of the depth map estimation system that is utilizing three cameras is presented in Fig. 1.1. In this thesis, the details of the complete depth map estimation system from the initial image capture to the display and applications are presented.

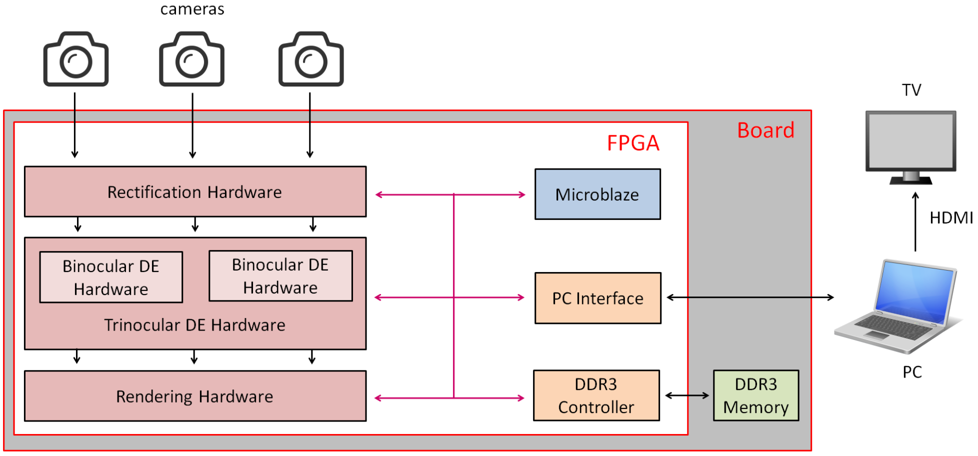


Figure 1.1: Simplified block diagram of the complete depth map estimation system.

Binocular Disparity Estimation Algorithm and Its Hardware Implementation

A binocular hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time high-resolution reconfigurable hardware implementation are presented [51]. The challenge of the disparity estimation is not only reaching real-time but also the development of a high quality algorithm. Therefore, the algorithm should be designed very carefully by considering all the details of its efficient hardware implementation. The main focus of the AWDE algorithm is its compatibility with real-time hardware implementation while providing high quality DE results for HR. The algorithm is designed to be efficiently parallelized and to require minimal on-chip memory size. The algorithm allows searching 49 pixel of the left image in the right image in parallel. The proposed AWDE algorithm combines the strengths of the Census Transform and the Binary Window SAD (BW-SAD) [60] methods, and thus enables an efficient hybrid solution for the hardware implementation. Although the low-complexity Census method can determine the disparity of the pixels where the image has a texture, mismatches are observed in textureless regions. Moreover, due to a 1-bit representation of neighboring pixels, the Census easily selects wrong disparity results. In order to correct these mismatches, our proposed AWDE algorithm uses the support of the BW-SAD, instead of using the complex cost aggregation method [36, 39].

The benefit of using different window sizes for different texture features on the image is observed from the DE results in [60], which inspired the dynamic window selection feature of the proposed AWDE algorithm. The selection of a large window size improves the algorithm performance in textureless regions while requiring higher computational load. However, the usage of small window sizes provides better disparity results where the image has a texture. Moreover, the use of BW-SAD provides better disparity estimation results than the SAD for the depth discontinuities [60]. The hardware presented in [60] is not able to dynamically change the window size, since it requires to re-synthesize the hardware for using different window sizes. In addition, the hardware presented in [60] does not benefit from the Census cost metric.

The proposed hardware of the AWDE algorithm provides dynamic and static configurability to have satisfactory disparity estimation quality for the images with different contents. It provides dynamic reconfigurability to switch between window sizes of 7×7, 13×13 and 25×25 pixels in run-time to adapt to the texture of the image. As a general rule, increasing the window size increases the algorithm and hardware complexity [60]. In the proposed AWDE algorithm, in order to provide constant hardware complexity over the three different window sizes, 49 neighbors are constantly sampled for different window sizes. Therefore, the proposed hardware architecture is able to reach the largest window size (25 × 25) among the hardware architectures implemented for DE [39–45, 60]. The proposed AWDE implementation utilizes a pixel intensity based refinement step to remove faulty disparity computations. In order to remove the faulty computations, the most frequent disparity value within the neighborhood is used.

The disparity estimation quality of the AWDE algorithm is improved using an iterative disparity refinement process. The proposed enhanced AWDE algorithm that utilizes Iterative Refinement (AWDE-IR) is implemented in hardware and its implementation details are presented [61]. Using the refinement process multiple times removes noisy computations more efficiently, and increases the disparity estimation quality.

The AWDE and AWDE-IR implementations provide static configurability to allow the user to change the disparity range, the strengths of Census and BW-SAD in hybrid cost computation, the closest and furthest expected distances and used color domain (Y, Cb or Cr) for disparity estimation. The architecture proposed in [43] is not able to provide configurability of disparity range since it is designed to search 80 disparity candidates in parallel, instead of providing parallelization to search multiple pixels in the left image. Therefore, a fixed amount of disparities is searched in [43], and changing the disparity range requires a redesign of their hardware.

The MATLAB implementation of the proposed AWDE-IR algorithm generates one single depth image at 1024×768 pixels resolution for a 128 disparity range in 1.5 hours using Intel i5 CPU at 2.67 GHz. Considering that 60 frames should be processed in every second, more than 3 days is required for a processor of the PC to generate one second of the depth video. The proposed hardware architectures for AWDE and AWDE-IR provides 60 frames per second at a 1024×768 XGA video resolution for a 128 pixel disparity range. Thereby, the proposed hardware implementations generates one second of the depth video in less than one second. The implemented disparity estimation hardware receives stream input of the stereo images and provides stream RGB+D output video. The hardware is carefully designed to do not require the existence of external memory for the disparity estimation computations. The latency of the hardware is just 0.84 ms for the same resolution and frame-rate, which allows to develop wide range of real-time depth estimation based applications. In addition, the proposed AWDE and AWDE-IR implementations provide better DE results than existing real-time high-resolution DE hardware implementations [43–45] for the tested HR Middleburry benchmarks [35]. Therefore, the proposed efficient AWDE hardware overcomes the main challenge of the disparity estimation process by both reaching real-time for HR video and providing very high quality depth estimation results, thanks to concurrent and careful design of the algorithm and the hardware.

Trinocular Disparity Estimation Algorithm and Its Hardware Implementation

A trinocular hardware-oriented adaptive window size disparity estimation (T-AWDE) algorithm and its hardware implementation are proposed to improve the disparity estimation quality of the AWDE and AWDE-IR implementations [62]. The T-AWDE algorithm and its hardware are the enhanced versions of the binocular AWDE-IR algorithm and its hardware implementation. The T-AWDE hardware generates a very high-quality depth map by merging two depth maps obtained from the center-left and center-right camera pairs. The T-AWDE hardware enhances disparity results by applying a double checking scheme which solves most of the occlusion problems existing in the AWDE-IR implementation while providing correct disparity results even for objects located in the left or right edge of the center image. The implemented T-AWDE hardware is the first hardware implementation that succeeds to provide real-time trinocular DE for HR video.

Rectification

A real-time disparity estimation system needs to perform real-time rectification which requires solving the models of lens distortions, image translations and rotations. Look-up-table based rectification algorithms allow image rectification without demanding high-complexity operations. However, they require an external memory to store large-size look-up-tables. In this thesis, an intermediate solution that compresses the rectification information to fit the look-up-table into the on-chip memory of a Virtex-5 FPGA is presented [63]. The proposed compressed look-up-table based rectification algorithm (CLUTR) can be used to rectify the stereo images if the lens distortion is not extreme and the cameras are not excessively misaligned.

In addition, in order to solve difficult camera alignment and distortion issues while maintaining the low complexity architecture, an enhanced version of the compressed look-up-table based rectification algorithm (E-CLUTR) and its real-time hardware are presented [64]. The low-complexity de-compression processes of CLUTR and E-CLUTR require a negligible amount of hardware resources for their real-time implementation and do not require the existence of external memory to store the look-up-tables. The capacity of CLUTR and E-CLUTR to fit the look-up-tables into the on-chip memory of the Virtex-5 FPGA is approximately six times and two times more efficient than [56], respectively, as a benefit of their efficient compression scheme.

Furthermore, the Caltech rectification algorithm [50] which does not benefit from look-up-tables is implemented in hardware, and its hardware resource consumption results are presented to improve the hardware comparison and to evidence the efficiency of CLUTR and E-CLUTR in an appropriate way. Hardware implementations of CLUTR and E-CLUTR require much less hardware resource than the hardware implementation of Caltech rectification while providing almost identical rectification results with very high PSNR results. Since the proposed high-quality compressed rectification implementations utilize negligible amount of hardware resources and do not require the existence of the external memory, they can be easily integrated into the state-of-the-art disparity estimation implementations which do not utilize rectification process due to its complexity.

Embedded System for Depth Map Estimation

The hardware architectures given in [39–45] present the main disparity estimation video processing cores but they do not reveal significant information about the other important peripherals of the complete embedded system such as softcore processor, its data bus, camera interface, display interface, external memory interface, DMA modules etc, and most of them claims verification only according to behavioral simulations. In this thesis, the full depth estimation embedded system that verifies the real-time functionality of the disparity estimation hardware is explained. The efficient communication and data exchange scheme of the system peripherals to transfer RGB+D output to a PC are presented. The proposed embedded system can be used to guide the authors of state-of-the-art disparity estimation hardware implementations to allow efficient real-time realization of their disparity estimation hardware.

Hardware and Software based Applications of Disparity Estimation

Depth map estimation can be used in a wide range of image and video processing applications. However, due to its challenge in real-time implementation, most of the applications of disparity estimation in state-of-the-art are realized by offline processing. This thesis presents several real-time hardware and software based applications of disparity estimation. The implemented applications conceptually prove that the high-quality and high-performance RGB+D outputs of the proposed real-time disparity estimation hardware can be used for enhanced 3D based video processing applications.

Free viewpoint synthesis is one of the important applications of depth map estimation. Free viewpoint synthesis is straightforward for computer generated images [2]. However, applying the same algorithms to real-world images is challenging since this process requires precise distance computation for every object. Many researches focus on developing high-quality free viewpoint synthesis algorithms [65, 66], while few results are published about real-time hardware implementation of free viewpoint synthesis [67, 68]. In this thesis, the first real-time high resolution free viewpoint synthesis system that utilizes three-camera disparity estimation hardware is presented [69]. The proposed hardware generates high-quality free viewpoint video at 55 frames per second using a Virtex-7 FPGA at a 1024×768 XGA video resolution for any horizontally aligned arbitrary camera positioned between the leftmost and rightmost physical cameras.

In this thesis, several software based real-time applications of disparity estimation are presented. The implemented software applications are operated at PC and visualized by the user-friendly graphical user interface (GUI) of the complete system. The implemented software applications are: Speed and distance measurement, depth based image thresholding, head-hands-shoulders tracking, virtual mouse using hand tracking, and face tracking integrated with free viewpoint synthesis. The implemented software applications prove that the proposed system can be used for advanced video processing applications where the depth computation is required. The proposed disparity estimation system can be utilized in many other 3D based video processing applications in the future.

ASIC implementation of Binocular Disparity Estimation

The proposed binocular disparity estimation hardware is implemented in an ASIC. In this thesis, the ASIC implementation details of the binocular DE hardware are presented. The ASIC is designed to be used as an accelerator for any complete system that requires stereoscopic depth computation. The ASIC solution for the developed hardware offers crucial advantages to the system compared to its FPGA implementation, such as less power consumption, faster performance and cost effectiveness. The ASIC implementation imposes some additional constraints with respect to the FPGA implementation. These restrictions mainly relate to the pin count, area and the usage of multiple clock domains. Therefore, several modifications are applied to the binocular DE hardware used for FPGA implementation. The specifications of the ASIC implementation are presented. The modifications applied to the binocular DE hardware are presented. The ASIC design and its possible utilization scheme are presented. The ASIC implementation can allow utilization of the depth estimation system in consumer electronic products such as mobile phones and wearable technologies.

1.3.2 Proposed Large Angle of View Image Capture and Reconstruction Systems

An enhanced version of the omnidirectional image reconstruction algorithm (EOIR) and its real-time hardware implementation are presented [57]. The EOIR algorithm provides homogeneous resolution over the entire reconstruction area. The proposed EOIR algorithm increases the realistic aspect of omnidirectional images captured by the Panoptic camera. The entire system provides the high bandwidth required to simultaneously process data originating from 40 cameras, and reconstruct omnidirectional images of 256×1024 pixels at 25 frames per second. Moreover, the hardware architecture is designed to provide flexibility in the selection of image resolution, AOV, contributing camera as well as algorithm choice between OIR and EOIR. Omnidirectional images are transmitted to the PC through a USB channel. The hemispherical 360° reconstruction can be viewed in real-time on a PC.

The omnidirectional video quality of the EOIR implementation is limited by the 352 × 288 resolution of the image sensors. Therefore, a very high-resolution multiple-camera omnidirectional video recording system called Giga-Eye is implemented using 5MP cameras. Giga-Eye records high-resolution images to reconstruct very-high resolution omnidirectional images in off-line processing [58]. The proposed Giga-Eye system is the highest resolution 360° omnidirectional camera that provides standard frame-rate video output (more than 25 fps) by its 21.6 MP video output capability at 30 fps. Moreover, Giga-Eye is the highest resolution 360° omnidirectional camera with its 82.3 MP output capability at 9.5 fps [59]. Giga-Eye is able to view the target locations close to horizontal plane with more than two cameras. Therefore, the presented video recording device can be used in ultra-high resolution omnidirectional video processing applications including depth map estimation and super-resolution in the future.

1.4 Thesis organization

This thesis is structured as follows. Chapter 2 presents the state-of-the-art of omnidirectional image reconstruction and depth map estimation. Chapter 3 presents the proposed binocular disparity estimation algorithms AWDE and AWDE-IR and their hardware implementations. The trinocular disparity estimation algorithm T-AWDE and its hardware implementation are presented in Chapter 4. Chapter 5 presents the compressed look-up-table based rectification algorithms CLUTR and ECLUTR and their hardware implementations. Chapter 6 presents the full embedded system of the disparity estimation hardware. The hardware and software based applications of the proposed disparity estimation hardware are presented in Chapter 7. The ASIC implementation of the binocular disparity estimation algorithm AWDE-IR is presented in Chapter 8. The proposed high-quality omnidirectional multiple-camera systems are presented in Chapter 9. The conclusion of the thesis is presented in Chapter 10. The high-quality visual results of the proposed systems are presented in Appendix A.

2 State of the Art

In this chapter, the state-of-the-art of 360° × 90° AOV multiple-camera omnidirectional image reconstruction systems and disparity estimation systems is presented. The algorithms and hardware features of these systems are explained. In addition, the goals which distinguish the research presented in this thesis from the state-of-the-art are presented.

2.1 Hemispherical Multiple Camera Image Reconstruction Systems

The visualization of a full 360° × 90° AOV scene forms the foundation enabling the emergence of novel applications in security systems, automotive platforms and mobile robots, realistic computer games, generation of street-level city maps and 3D cinematography.

The Ladybug3 camera [16] presented in Fig. 2.1 is an omnidirectional image capture device developed by Point Grey. Ladybug3 utilizes six 1024×768 pixel CCD sensors placed in a cubic arrangement to generate 360° × 90° AOV images in real-time. Each camera has a wide AOV lens. The cameras are controlled by a GUI. The stitching and blending processes are implemented in software on a PC. Due to utilization of only six image sensors, mutual image overlap is low. The PC operated software computes the calibration parameters of these images utilizing the low size of the overlaps. The details of the calibration process and the hardware system are not revealed by Point Grey, since Ladybug3 is a commercial product.

Google's Street View camera, R7 [17], is presented in Fig. 2.2. R7 utilizes 15 5MP cameras to cover more than 80% of the full sphere with a panoramic image resolution of 8192×4096 pixels by capturing pictures every 2.5 seconds. The multiple-camera image capture device is accompanied with a laser scanner to measure the distances. The hemispherical images

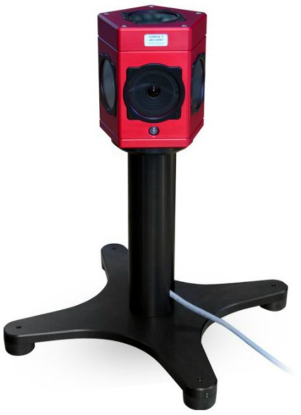


Figure 2.1: Ladybug3 multiple camera system

are reconstructed off-line. Distance measurement and 3D models of the buildings are used to remove parallax errors caused by depth discontinuity. Google utilizes R7 on fast moving vehicles, which results in blurring the original captured images. Therefore, information pertaining to the speed of the carrying vehicle and optical flow estimation are utilized to recover the images prior to the omnidirectional image reconstruction process.

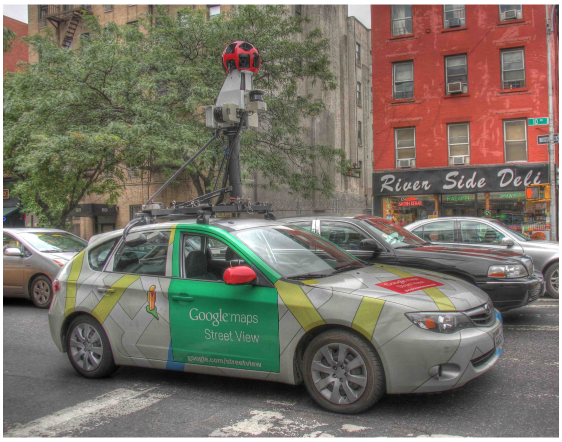


Figure 2.2: R7 multiple camera system of Google for Street View application

A spherical multi-camera vision sensor called Panoptic utilizes a specific omnidirectional image reconstruction algorithm (OIR) to construct $360° \times 90°$ AOV panoramic images in real-time [19]. The Panoptic camera is presented in Fig. 2.3. The system presented in [19] utilizes 40 cameras, each with a $352{\times}288$ pixel resolution, and creates $256{\times}1024$ pixel resolution omnidirectional images.

The hemispherical arrangement of Panoptic is fully covered with cameras following a systematic organization depicted in Fig. 2.4. The Panoptic camera has 7 floors to allow

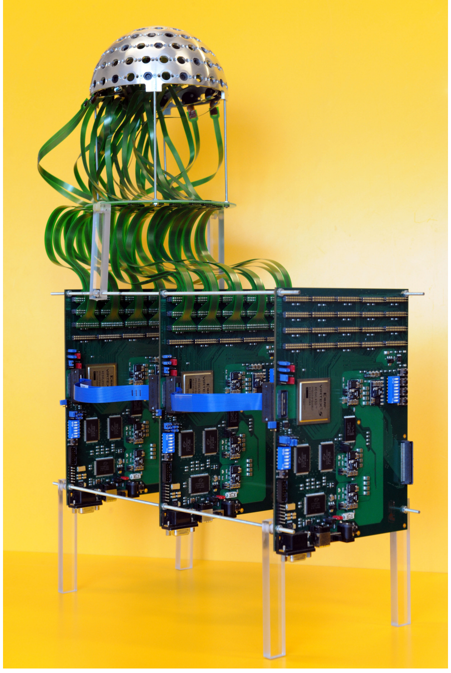


Figure 2.3: Panoptic camera

mounting 104 cameras. 40 of these locations are used to generate omnidirectional images. The AOVs of each camera are 53°and 43°, in the horizontal and vertical axis, respectively. The intrinsic and extrinsic calibrations of the cameras are computed using Caltech calibration tool [50].

The OIR algorithm assumes that the space located around the Panoptic camera is hemispherical, where the surface of the structure is divided into an equiangular grid with N_θ latitude and N_φ longitude pixels. Fig. 2.5a shows a hemispherical surface over which a linear pixelization scheme is applied, and Fig. 2.5c depicts its 2D reconstruction. The latter is divided into 256 and 1024 pixels for N_θ and N_φ respectively, which are linearly distributed. The direction of each pixel is described by a unit vector, ω with spherical coordinates $(\theta_\omega, \varphi_\omega)$, which is visually presented in Fig. 2.5b.

The vector t, which is shown in Fig. 2.5a, represents the focus direction, whereas u and v vectors stand for the vertical and horizontal directions of the camera in pixel representation. After the computation of ω, all cameras having ω in their angle of view are determined by processing the dot product of ω with t. Then, pixel grid locations onto which each ω projects are found. However, due to the fact that ω may not coincide with the exact pixels on the camera images, each projected pixel is interpolated with its adjacent pixels in order to extract the light intensity in that particular ω direction.

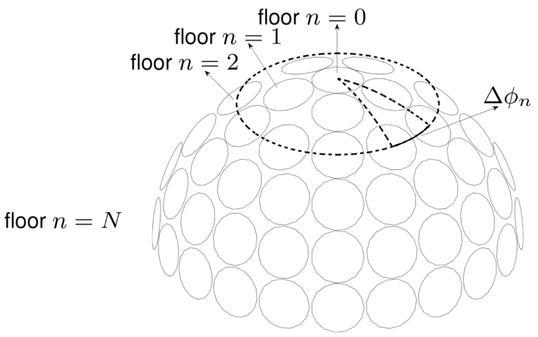
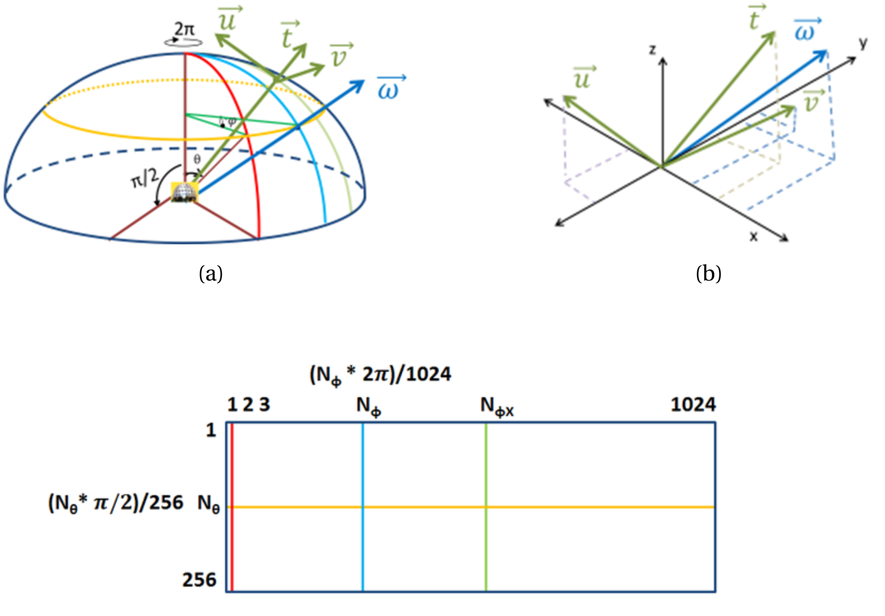


Figure 2.4: Hemispherical structure with multiple floors



(a)

(b)

(c)

Figure 2.5: (a) hemispherical surface showing θ and φ angles and unit vectors used in omnidirectional image reconstruction calculations, (b) ω vector and corresponding unit vectors; t (focus direction of the camera), u (vertical direction in the pixel representation of the camera), v (horizontal direction in the pixel representation of the camera) (c) 2D reconstruction using linear pixelization

Since the AOVs of the cameras may overlap, the final light intensity of each pixel is determined by assigning a *weight* to each contributing camera. The *weight* of a camera is calculated by computing its 2D distance to the pixel that is being considered, and taking its reciprocal.

Two optional schemes are presented to carry out the final interpolation process, namely nearest neighbor interpolation and linear interpolation. The nearest neighbor interpolation

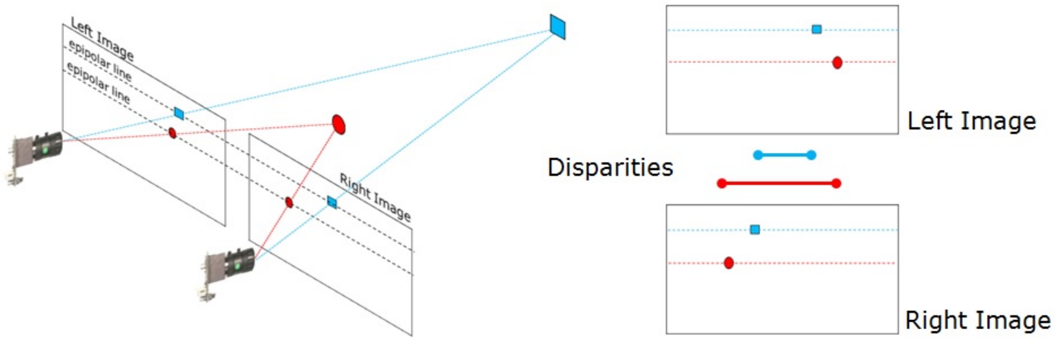


Figure 2.6: Inverse relationship between the depth and disparity.

method extracts pixels from the cameras that provide maximum weights, whereas the linear interpolation scheme linearly interpolates the light intensity extracted from each contributing camera.

2.2 Binocular Stereo Matching Algorithms and Their Hardware Implementations

In the human vision system, the disparity term describes the difference in location of corresponding points that are seen by the left and right eyes [35]. There is an inverse relationship between disparity and depth. This inverse relation is presented in Fig. 2.6. An object located further away provides a smaller horizontal location change in the left and right images, i.e. a smaller disparity result. Generally, horizontal displacements are used in disparity estimation, however the same principle can be applied as a vertical displacement if two cameras are located one top of each other.

In disparity estimation, the (x, y) coordinates of the disparity map are calculated by finding the relationship between a pixel coordinates (x, y) of the reference image (processed image) and its corresponding coordinates (x_1, y_1) in the matching image (searched image) [35]. The relationship is expressed in eq 2.1. In eq 2.1; s=±1 is a sign based on the choice of the reference and matching images to guarantee that the disparity value is positive. Thereby, the disparity of a pixel in the left or right image can be defined as ($d = x_l - x_r$).

$$x_1 = x + sd(x, y),\ y_1 = y \tag{2.1}$$

The disparity values can be converted to the depth Z by using geometrical relationships

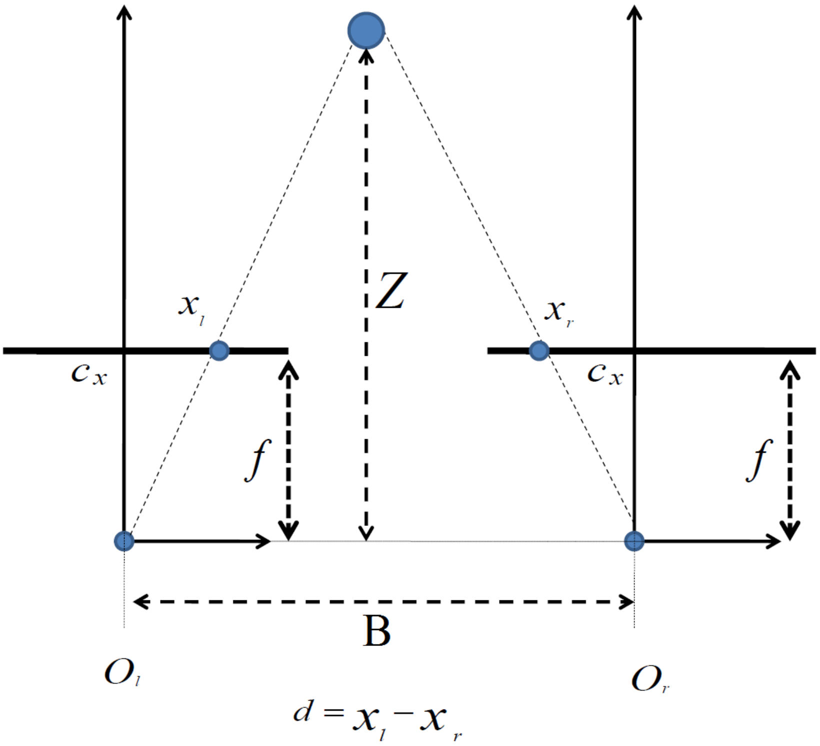


Figure 2.7: Geometrical relationship between disparity and depth.

between triangles [70], as presented in Fig. 2.7. The left and right image pixels coordinates are shown as (x_l, y_l) and (x_r, y_r), respectively. The centers of the projections are represented as O_l and O_r. (c_x, c_y) is the principal point. The focal length is represented by *f*. *B* indicates the distance between the cameras which is called the baseline. The mathematical relation between the depth and disparity is shown in eq. 2.2. The detailed information about the camera calibration methods to obtain *f* and (c_x, c_y) parameters are presented in subsection 2.2.1.

$$Z = \frac{fB}{x_l - x_r} \tag{2.2}$$

The theoretical distance precision limitations of the disparity estimation based depth estimation system is presented in Fig. 2.8. The precision values are presented in cm for a varying perpendicular depth values. Therefore, lower cm values of precision in the figure indicate better depth sensitivity, i.e. better precision. The precision values are generated for 1024×768 pixels resolution cameras that include 6 mm of lens. The maximum disparity range is considered as 255, and consecutive integer disparity values are considered as differentiable. As presented in Fig. 2.8, the distance measurement precision of the depth camera increases when the distance between the stereo cameras is increased. For example, when the baseline is 20 cm, the disparity estimation system can differentiate 0.5 cm of distance at 1 m, whereas, when the baseline is 10 cm, the disparity estimation system can differentiate 1.1 cm of distance at 1 m. The depth measurement precision reduces for increasing value of the depth for any constant baseline. For example, when the baseline is 20cm, the expected precision is 0.5 cm for an object at 1 m, whereas, the expected precision is 56 cm for an object at 10 m. Therefore, in order to increase the precision for far objects, the baseline should be increased. However, increasing the baseline reduces overlapping region between stereo images for close objects. Therefore, large baseline may not allow distance measurement for very close objects or increases the maximum required disparity range. For example, when the baseline is 50 cm and the maximum disparity range is 255, the system can not measure the distance of the object at 1 m, whereas, when the baseline is 10 cm, the disparity estimation system can measure the distance of the object at 1 m. Therefore, the baseline should be mechanically arranged according to expected distance range of the objects and target application. Although disparity estimation system can maintain distance measurement for any distance by changing the baseline, light projection based depth estimation systems are limited by the strength of the projected light source. For example, Kinect is not able to measure the distance of the object if they are further than 7 m. Laser scanner based systems can measure the distance of the very far objects, but in this case, the strength of the light may damage the human eye if it is looking in the direction of the source of laser light for a long time. On contrary, disparity estimation based depth estimation includes only passive sensors, which can not give any damage to human eye.

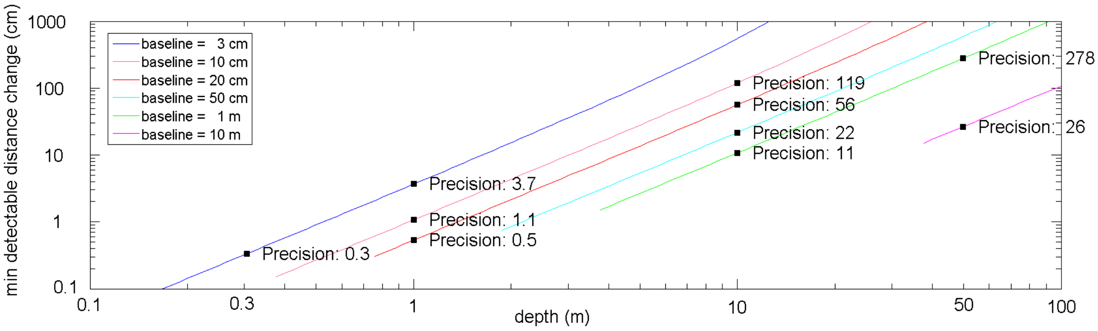


Figure 2.8: Distance measurement precision of disparity estimation based depth estimation.

Stereo matching algorithms are mainly classified into two categories, namely the local and global approaches. Local methods compute each pixel's disparity independently, based on the intensity similarity over the matching window. Depending on the algorithms, matching costs are aggregated over the matching window. Then the disparity value which gives the smallest cost is generally selected as the disparity of the pixel using a winner-take-all approach. In local methods, the selection of the stereo matching window size plays a significant role in the performance of the algorithm [60]. As presented in [60], a small window size provides good performance for high textured regions and object boundaries, whereas a large window size usually fails and blurs the object boundaries of disparity map. However, in low-textured regions, a large window size gives good results.

In global methods, the disparity estimation problem is considered as minimizing the energy function presented in eq. 2.3, which consists of data (E_{data}) and smoothness (E_{smooth}) terms, through various optimization techniques and weights (λ). The data term formulates the matching cost. The second term includes the smoothness assumption of the algorithm. The energy function can be minimized by different methods such as Morkov random fields, dynamic programming and graph-cut [35]. Although many global methods give better results than the local methods, they require very high computational load and memory bandwidth to minimize the energy function. Therefore, the use of global minimization-based disparity estimation algorithms is impractical in real-time hardware implementation, in consideration of hardware resource limitations.

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{2.3}$$

Based on the Middlebury taxonomy [35] of stereo algorithms, disparity estimation algorithms usually subdivide the stereo matching processes into the six consecutive steps: Calibration, Rectification, Matching Cost Computation, Cost Aggregation, Disparity Selection and Optimization, and Disparity Refinement. The details of these processes and their hardware implementations in the state-of-the-art developments are presented in following subsections.

2.2.1 Camera Calibration

In disparity estimation, epipolar line geometry is required to search a correspondence pixel in a horizontal line. Alignment mismatches between cameras in a stereo vision system make disparity estimation operation difficult by preventing the horizontal search assumption. Therefore, camera misalignments and distortions should be modeled and solved.

The camera calibration consists of estimating the mathematical relations that describe the projection of an object from 3D space to image space for each camera. These mathematical relations include parameters related to the physical characteristic (intrinsic parameters) as well as the position (extrinsic parameters) of each camera. The calibration can be performed using pictures of known and unknown structures. Chessboard can be used as known structure in calibration. By taking the picture of chessboard from a variety of angles and distances, it is possible to compute the relative orientations of the stereo cameras as well as their intrinsic parameters [70]. The calibration parameters can be obtained without using any known pattern. This process is called auto-calibration. Auto-calibration requires the detection of local features of the image [71]. Known structure-based calibration methods can be considered as more accurate since unknown patterns may cause ambiguity to the matching of the positions of the local features, especially in the presence of repetitive patterns or if there are not enough texture in the stereo images.

The pinhole camera model presented in Fig. 2.9 is used to describe the calibration procedure. In this model, light is envisioned as entering from the scene or a distant object, but only a single ray enters from any particular angle. This ray is then projected onto an imaging surface [70]. The pinhole camera can be modeled by using intrinsic and extrinsic parameters. Three orthogonal coordinate systems should be defined to clearly describe these parameters. As presented in Fig. 2.9, these three coordinate systems are the world coordinate system (X,Y,Z), the camera coordinate system (X_c,Y_c,Z_c), and the image coordinates system as (x, y).

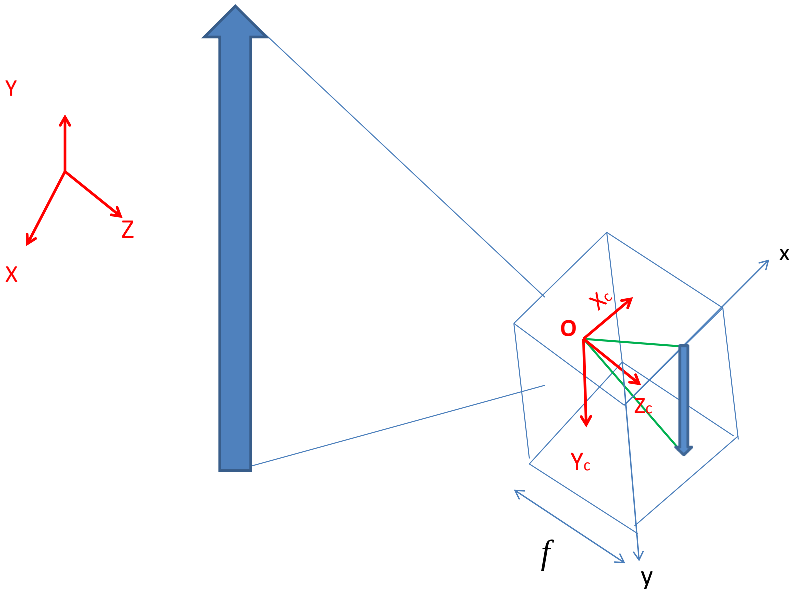


Figure 2.9: Coordinate systems for pinhole camera model.

The intrinsic parameters represent the internal characteristic of the camera. These are the focal length, the principal point and skew coefficient. According to the pinhole camera model, focal length can be defined as the distance from the pinhole aperture to the screen as presented in Fig. 2.9. The principal point can be defined as the coordinates of the center of the image in the image coordinate system. Expressed differently, it is a point which is located at the intersection of the optical axis with the image plane. The skew coefficient is the cosine of the angle between x and y coordinates, which is ideally equal to 0 since the axes are supposed to be orthogonal. Following these definitions, a camera matrix describing the transformation from 3D to 2D coordinate system, containing all internal characteristics of the camera can be formulated as given in eq. 2.4. In eq. 2.4, (c_x,c_y) and (f_x,f_y) represent the principal point and the focal lengths in pixel-related units. The intrinsic camera matrix parameters do not depend on the scene view. After the computation of intrinsic parameters, they can be used many times if the focus, optics or camera resolution are not changed.

$$A = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Lenses cause radial and tangential distortions on the images [70]. In the case of radial distortion, pixels located near the borders of images is noticeably distorted by the lenses of cameras. It is known that light rays further from the center of the lens are bent more than those closer to the center. Therefore, the radial distortion is 0 at the optical center of the image, and it increases towards the edges. The second distortion is the tangential distortion and it is mainly due to the lens that is not exactly parallel to the imaging plane. These radial and tangential distortions can be corrected using Brown's distortion model given in equations 2.5 and 2.6. The calibration process requires to determine the internal camera calibration parameters k_1, k_2, k_3 for radial distortion and p_1, p_2 for tangential distortions. In these equations, (x, y) is the position of a 3D point after applying camera transformation matrix, and ($x_{corrected}$, $y_{corrected}$) is the position of the 3D point in the 2D image after correcting the errors caused by the distortions.

$$\begin{aligned} r^2 &= x^2 + y^2 \\ x_{corrected} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + dx \\ y_{corrected} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + dy \end{aligned} \tag{2.5}$$

$$\begin{aligned} dx &= 2p_1 xy + p_2(r^2 + 2x^2) \\ dy &= 2p_1(r^2 + 2y^2) + 2p_2 xy \end{aligned} \tag{2.6}$$

After the pixel coordinates are corrected to cancel the lens distortions, the extrinsic parameters of the cameras should be computed. The geometrical model of the transformation between the camera coordinate system and the scene coordinate system can be determined after the computation of external calibration parameters [72]. These parameters are related to camera position and orientation. The 3×4 transformation matrix includes rotation parameters (r_{11}-…-r_{33}) and translation vector (t_1,t_2,t_3).

$$sm' = A[RT]M' \tag{2.7}$$

$$s\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.8}$$

The final transformation formula is presented in eq. 2.7 and 2.8. (X, Y, Z) are the scene coordinates of a 3D point in the world coordinate system (M). (u, v) are the coordinates in image plane (m) in number of pixels. R is the rotation matrix, T is the translation matrix and s is the scale factor.

The internal and external calibration equations can be reformulated to obtain a single equation presented in eq. 2.9. The Open-CV Calibration Toolbox [73] and Caltech's Matlab Calibration Toolbox [50] are user-friendly software implementations that are used to obtain the parameters of eq. 2.9.

$$\begin{aligned}
&x' = x/z \\
&y' = y/z \\
&x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [2p_1 x' y' + p_2(r^2 + 2x'^2)] \\
&y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [p_1(r^2 + 2y'^2) + 2p_2 x' y'] \\
&\text{where} \\
&r^2 = x'^2 + y'^2 \\
&\quad u = f_x * x'' + c_x \\
&\quad v = f_y * y'' + c_y
\end{aligned} \tag{2.9}$$

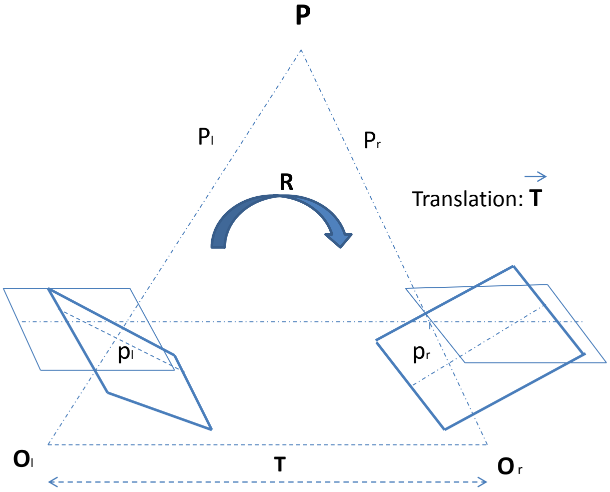


Figure 2.10: Mathematical alignment of the two cameras into an unique image plane.

2.2.2 Rectification

Image rectification is a transformation process applied to stereo images using the internal and external camera calibration parameters. The output of this process consists of row-aligned images. The final rectified images can be generated by calculating new transformation matrixes from the scene coordinates to image coordinate by rotating the old images around their optical center until the focal planes become coplanar [74], as shown in Fig. 2.10.

[50] and [73] attempt to maximize common viewing area by minimizing the amount of change of the reprojection produces for each of the two images. The rotation matrix R that gives the rotation from the right camera's image plane into the left camera's image plane is divided in half between the two cameras. As a result of this division, the distortions that come from reprojection are minimized. As a result of this process, the left and right images are rotated by a half, and the images are put into coplanar alignment.

The formula of the rotation matrix to obtain row aligned image planes is presented in eq. 2.10. In eq. 2.10, R_l is the rotation to be applied to the left image, R_r is the rotation to be applied to the right image. r_l and r_r are obtained by applying half a rotation clockwise and counterclockwise, so that $R = r_l^T r_r$. R_{epi} is the matrix to push the epipoles towards infinity, thereby epipolar lines become horizontal. R_l and R_r are used in eq. 2.7 to apply the projection.

$$\begin{aligned} R_l &= R_{epi} r_l \\ R_r &= R_{epi} r_r \end{aligned} \tag{2.10}$$

The equation to obtain R$_{epi}$ is presented in eq. 2.11 where T is the translation between the two cameras.

$$\begin{aligned} e_1 &= \frac{T}{||T||} \\ e_2 &= \frac{[-T_y T_x 0]^T}{\sqrt{T_x^2 + T_y^2}} \\ e_3 &= e_1 \times e_2 \\ R_{epi} &= \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix} \end{aligned} \tag{2.11}$$

A real-time rectification hardware should apply operations pertaining to rotation and translation, and should correct the lens distortions. The rectification process requires high-precision fractional operations, and thus a high amount of hardware resources. Therefore, look-up-table based rectification hardware implementations are presented in [54–56]. In this method, the mappings between the original image pixel coordinates and the rectified image pixel coordinates are pre-computed and then implemented into look-up-tables. Although the computational complexity of the look-up-table based approach is very low, using these tables requires a significant amount of memory bandwidth and size. For example, for the rectification of 1024×768 resolution stereo images with 6 bits fractional precision, the rectification map alone approximately requires 2×1024×768×2×(10+6) = 6 MB of memory space. This memory size is very high for regarding the on-chip memory capacity of mid-range FPGAs, which necessities the utilization of an external memory [54, 55]. The hardware presented in [56] encodes the difference between each element of the look-up-table to reduce the bitsize of the elements. This encoding method reduces the look-up-table size of 1280×720 size image from 7.5 MB to 1.3 MBs. Therefore, the look-up-tables of 1280×720 size images fit into the on-chip memory of a Virtex-5 FPGA.

2.2.3 Matching Cost Computation

In order to identify the corresponding pixels in stereo images, a matching cost should be computed for each candidate matchings. The absolute differences (AD), sum of absolute difference (SAD), squared differences (SD), sum of square difference (SSD) and Census transform are the most commonly used pixel-based matching costs [35]. These matching costs are explained in this sub-section.

Assuming that a pixel intensity in the gray-level left image is $I_l(x_l,y_l)$, and the candidate pixel in the gray-level right image is $I_r(x_l$-d,y) then the SD and AD matching costs between these pixels can be expressed as in eq. 2.12 and 2.13, respectively. The SAD and SSD matching costs can be calculated over the search window ((2w+1) × (2w+1)) centered at the (x_l,y) pixel. The SAD and SSD matching costs can be formulated as presented in eq. 2.14 and 2.15, respectively. Since the SSD operation requires multiplication operations and thus a high amount of hardware resources, it is not usually preferred for a hardware implementation.

$$SD(x_l, y, d) = (I_l(x_l, y) - I_r(x_l - d, y))^2 \tag{2.12}$$

$$AD(x_l, y, d) = (I_l(x_l, y) - I_r(x_l - d, y)) \tag{2.13}$$

$$SAD(x_l, y, d) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} |I_l(x_l) + i, y + j) - I_r(x_l + i - d, y + j)| \tag{2.14}$$

$$SSD(x_l, y, d) = \sum_{i=-w}^{w} \sum_{j=-w}^{w} (I_l(x_l) + i, y + j) - I_r(x_l + i - d, y + j))^2 \tag{2.15}$$

According to matching costs calculation using SAD, it is assumed that the 8-bit luminance of a pixel in a reference image and its correspondence pixel in the matching image are equal. The same procedure can be also applied for the matching of 24-bit RGB values as presented in [43]. However, applying the matching computations using RGB requires three times the matching computations using luminance values, but it does not significantly improve the disparity estimation results.

Using binary windows (BW) together with SAD computation increases the quality of the stereo matching for the object boundaries, which is presented in [60]. BW is computed for processed pixel by detecting neighboring pixels with same intensity values in a support window. BW is used to determine contributing pixels to the SAD computation. The computed BW-SAD metric provides high-quality results, especially for object boundaries.

The matching cost computation using SAD provides high-quality results using computer generated stereo images. However, SAD does not always provide high-quality results for real-world images since different color or intensity value can be delivered by two cameras observing the same object. Therefore, ambiguity occurs for the selection of the disparity with the minimum matching cost. Census transform is proposed to overcome this problem [35].

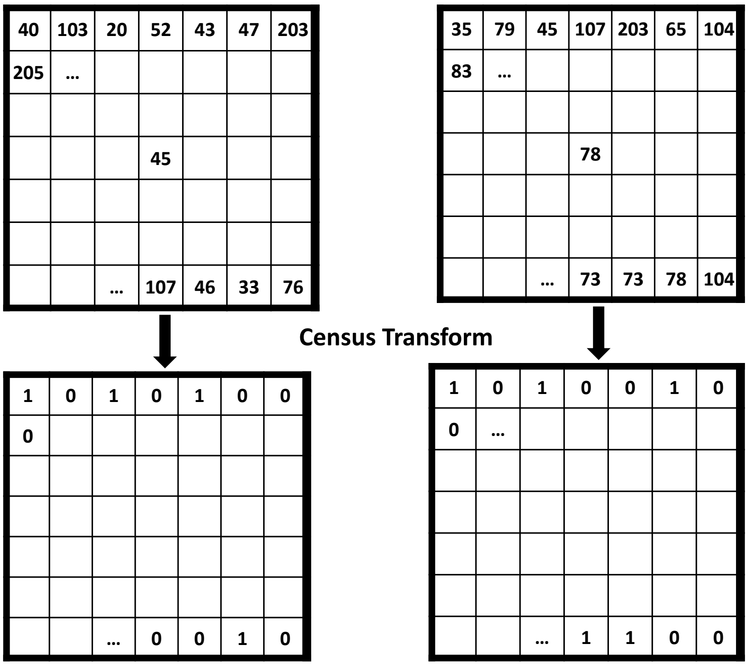


Figure 2.11: Census transform computation.

The Census transform [46] decreases the dependency of the stereo matching process to the sensitivity of the camera gains. Census transform codes the processed pixels according to their relation to the neighboring pixels. The algorithm assumes that even if one stereo image is relatively brighter to the other one, still the correlation between a pixel and its neighboring pixels should be regular. The Census transform is visualized in Fig. 2.11. A neighboring pixel that has an intensity value smaller than the center pixel intensity is coded as 1. A neighboring pixel that has an intensity value larger than or equal to the center pixel intensity is coded as 0. The hamming distance is used to compare the similarity between the two strings. The Hamming distance computes the number of different bits in two strings. The computational complexities of the Census transform and Hamming computation are much less than the SAD, since they require low bit-size adders, comparators and XOR computations. The hardware presented in [39] utilizes Census transform for the matching cost computations. The disadvantage of the Census transform is its cost representation sensitivity compared to SAD, which causes ambiguity for the selection of the best matching. For example, assuming 7×7 windows are used, the Hamming results can be between 0 and 48, while SAD can provide matching cost values between 0 and 12240. In addition, the Census transform is more sensitive than SAD to the existence of perfect calibration and rectification, which is usually challenging to obtain with real-world images.

The hardware presented in [44] utilizes edge detection method to reduce the computational load of matching cost computation compared to the SAD. It utilizes sobel edge detection to convert gray images to 1-bit pixel size images. Then, it performs XOR operations between the matching windows, and accumulates the 1-bit costs using adder-tree. Although [44] utilizes low amount of hardware resources, it does not provide high-quality disparity estimation results due to matching ambiguities caused by 1-bit representation of the pixels.

2.2.4 Cost Aggregation

In the cost aggregation, matching costs are summed and averaged over a support region [35]. The support region is generally three-dimensional in the sr_x-sr_y-d space, where sr_x and sr_y are the size of the support window and d is the disparity range. The support region should be generated for every processed pixel. The cost aggregation step smoothens the matching costs with ambiguity and provides a decrease in the disparity estimation mismatches [36]. However, the accumulation and averaging of all the cost values in a support region for every processed pixel require significant amount of hardware resources and memory bandwidth. Although cost aggregation step is generally used especially by the top performer algorithms [35], its usage is not suitable for hardware adaptable disparity estimation algorithms especially for high-resolution and high disparity range. The hardware presented in [39] utilizes cost aggregation, but it can perform in real-time only for CIF images (352×288).

2.2.5 Disparity Selection and Optimization

In most of the local processing based DE methods, the disparity value which gives the minimum cost over the disparity range is selected as the final disparity of a processed pixel. This operation is called winner-take-all (WTA). For example, if the matching costs are obtained as presented in Fig. 2.12, the WTA operation should determine 33 as the disparity of the processed pixel.

The WTA process can deliver incorrect disparity estimation results if there are strong local minima or if the global minima is very close to the matching cost of several other candidate disparity values. For this reason, in addition to the matching cost, confidence metrics are used to select the correct disparity result.

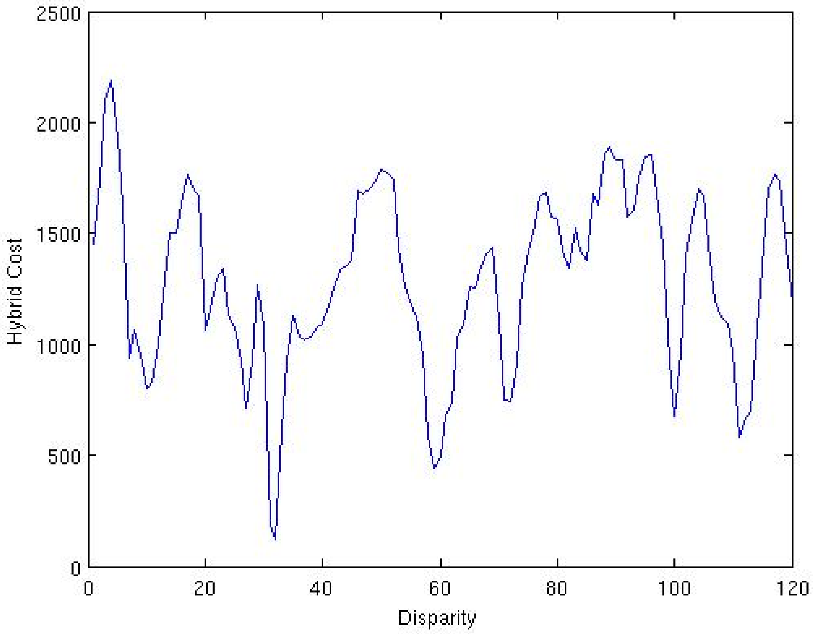


Figure 2.12: Disparity selection.

The confidence metrics can be generated using the matching costs [75, 76]. In the calculation of confidence metrics, c_1 represents the minimum matching cost and d_1 is the corresponding disparity value of the minimum matching cost. Furthermore, c_2 symbolizes the second minimum matching cost and d_2 is the corresponding disparity value of the second minimum matching cost. Different confidence metrics can be defined. Some of them are explained in this sub-section.

The sharpness around the global minimum can be measured by eq. 2.16. A candidate disparity value which does not provide sufficient sharpness is considered not-confident.

$$C_{CUR} = -2c(d_1) + c(d_1 - 1) + c(d_1 + 1) \tag{2.16}$$

Another confidence metric can be defined to provide information related to the strength of the global minimum compared to other local minimums. This metric can be defined as presented in eq. 2.17. Other strong costs within the disparity range causes ambiguity in the selected global minimum [75, 76].

$$C_{PKRN} = \frac{c_2}{c_1} \tag{2.17}$$

An alternate confidence metric can be calculated using two disparity maps. In the first map, the left image is used as the reference image and the right image is used as the matching image. In the second disparity map, the right image is used as the reference image and the left image is used as the matching image. These disparity maps are cross-checked to define the confidence. The hardware presented in [45] utilizes this confidence metric. Although this metric improves the DE quality, it doubles the computational cost of the disparity estimation hardware.

$$C_{LRC}(x, y) = -|d_1 - D_R(x - d_1, y)| \tag{2.18}$$

2.2.6 Disparity Refinement

Disparity refinement is a post-processing step to the disparity map estimation. Different methods to refine the calculated disparity map can be applied such as sub-pixel accurate disparity estimation and median filtering [35].

A low resolution of the input stereo images reduces the precision of the depth sensitivity. In order to increase the depth sensitivity of these images sub-pixel disparity estimation can be used [36].

Many disparity estimation algorithms do not provide satisfactory result in occluded regions. Using the left-right crosscheck method, left-to-right and right-to-left disparity maps are compared to detect the occluded regions and confident values. Afterwards, the occluded regions can be inpainted with confident disparity values [35].

The Median filter based approach is another method to refine the disparity maps. It can be used to filter the mismatches in disparity map. This smoothing method provides high-quality results when used along with object segmentation algorithms. However, the computational complexity and memory bandwidth requirements of the object segmentation process are high [35]. Therefore, this disparity smoothing method is not suitable to real-time hardware implementation.

2.3 Trinocular Stereo Matching Algorithms and Their Hardware Implementations

Trinocular disparity estimation is an extended version of binocular system. Trinocular vision system receives three separate views from three different viewpoints of the same scene. Hence, an occluded region in one of the binocular pairs may be visible in the other image pair. The three cameras can be located in the same horizontal epipolar line, i.e. collinear as presented in Fig. 2.13, or in a triangular placement as presented in Fig. 2.14.

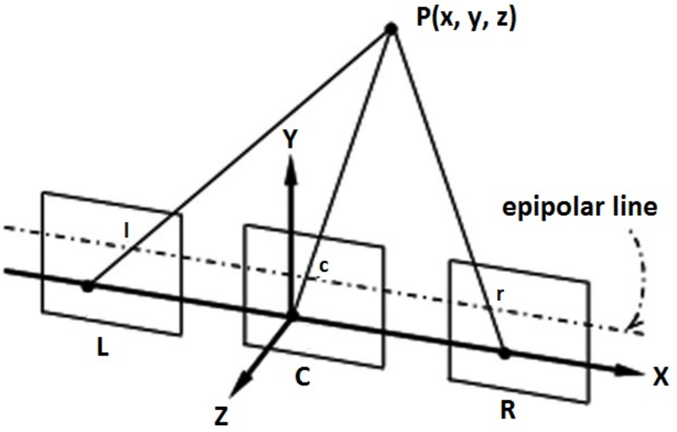


Figure 2.13: Collinear placement of three cameras for disparity estimation

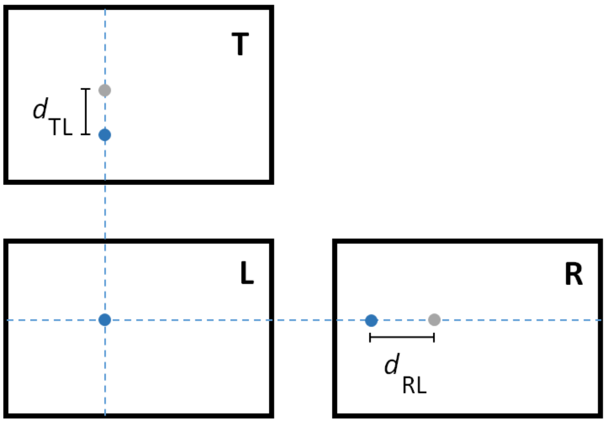


Figure 2.14: Triangular placement of three cameras for disparity estimation

In a trinocular system consisting of three horizontally alligned cameras, the center image is used as reference image and two binocular disparity estimation maps are calculated for both the center-right and center-left image pairs. The most important part of trinocular disparity estimation relates to find a fusion method to merge two binocular disparity maps into a robust and accurate trinocular disparity map. The main aim is to determine which binocular disparity map gives the best disparity result for the searched pixel.

The trinocular DE hardware presented in [48] enables handling 52 fps on an Altera Cyclone-IV FPGA at a 640×480 video resolution. It utilizes a hierarchical classifier to select the most promising disparity value. It computes a hierarchical classifier using information provided by the calculated cost curves and the spatial neighborhood of the pixels.

The hardware presented in [49] uses a triangular configuration of three cameras and enables handling 30 fps on a Xilinx Virtex-4 FPGA at a 320×240 video resolution for a 64 pixels disparity range. It solves the occlusion problem of the left-right camera pair using the up-down camera pair. Triangular allignment provides better results than collinear alignment especially for regions containing horizontal repetitive patterns. However, triangular allignment reduces the data reuse capability and increases the memory bandwidth requirement since the search directions are not symmetric considering the top-down and left-right binocular pairs.

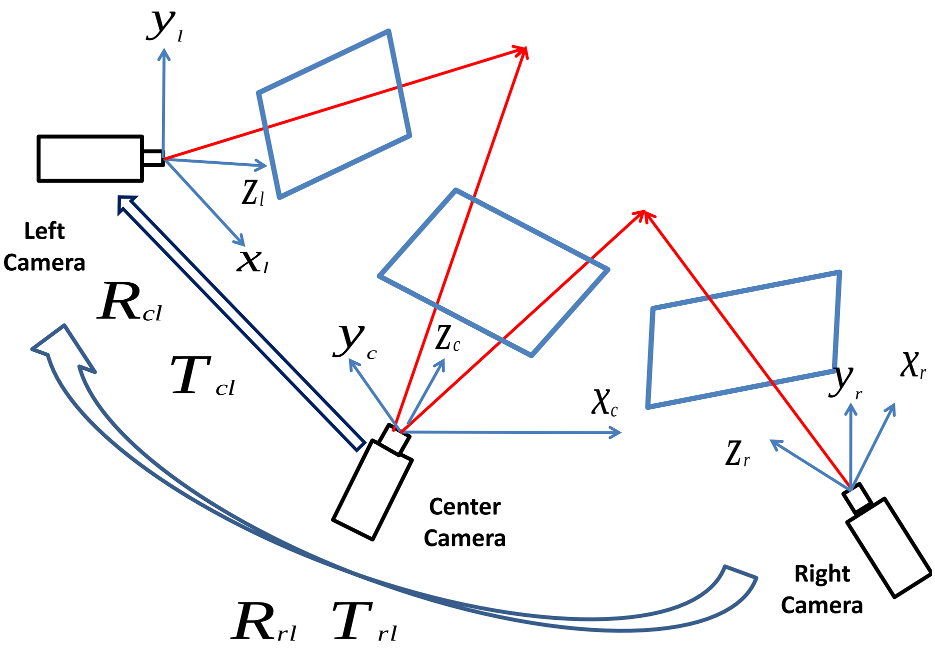


Figure 2.15: Geometrical relationship between the cameras of the trinocular stereo system

Camera calibration and rectification are also important pre-processing parts of the trinocular DE. In contrast to binocular stereo calibration and rectification, few works present multi-camera calibration [77–79] and rectification methods [80–82]. In multiple-camera calibration, one of the cameras is determined as the reference camera and binocular camera calibration procedures are applied between the reference camera and the other cameras in the system.

In a trinocular stereo system, firstly the two camera pairs are determined and then the positional relationships of the two camera pairs are obtained by using the binocular calibration parameters. The calibration process of a trinocular camera system is presented in Fig. 2.15. The left camera is the reference for the calibration process. The transformation matrixes from the world coordinate system to the image coordinate system for the left, center and right cameras are represented as presented in eq. 2.19, eq. 2.20 and eq. 2.21, respectively. R_l, R_c, and R_r are the rotation matrixes of the cameras. T_l, T_c and T_r are the translation matrixes of the cameras. Based on these transformation matrixes, the corresponding relationship between the left and right cameras and the corresponding relationship between the left and center cameras are expressed in eq. 2.22 and 2.23, respectively [77]. In eq. 2.22, R_{rl} and T_{rl} are the rotation and translation matrixes from the center camera coordinate system to the left camera system. In eq. 2.23, R_{cl} and T_{cl} are the rotation and translation matrixes from the right camera coordinate system to the left camera system.

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = M_l \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{l1} & r_{l2} & r_{1l3} & t_{lx} \\ r_{l4} & r_{l5} & r_{l6} & t_{ly} \\ r_{l7} & r_{l8} & r_{l9} & t_{lz} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.19}$$
$$M_l = [R_l T_l]$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = M_c \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{c1} & r_{c2} & r_{c3} & t_{cx} \\ r_{c4} & r_{c5} & r_{c6} & t_{cy} \\ r_{c7} & r_{c8} & r_{c9} & t_{cz} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.20}$$
$$M_c = [R_c T_c]$$

$$\begin{aligned}
\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} &= M_r \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{r1} & r_{r2} & r_{r3} & t_{rx} \\ r_{r4} & r_{r5} & r_{r6} & t_{ry} \\ r_{r7} & r_{r8} & r_{r9} & t_{rz} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\
M_r &= [R_r T_r]
\end{aligned} \tag{2.21}$$

$$\begin{aligned}
\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} &= R_l R_r^{-1} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + T_l - R_l R_r^{-1} T_r \\
R_{rl} &= R_l R_r^{-1} \\
T_{rl} &= T_l - R_l R_r^{-1} T_r
\end{aligned} \tag{2.22}$$

$$\begin{aligned}
\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} &= R_l R_c^{-1} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + T_l - R_l R_c^{-1} T_c \\
R_{cl} &= R_l R_c^{-1} \\
T_{cl} &= T_l - R_l R_c^{-1} T_c
\end{aligned} \tag{2.23}$$

Rectification is applied at the end of global calibration process. During the rectification, the three-camera coordinate systems are reprojected into a one-coordinate system according to multi-camera calibration parameters [80–82]. Firstly, a reference camera is chosen, and the common coordinate system is determined based on the reference camera. Then, all cameras in the system are reprojected into the common coordinate system [73].

2.4 Thesis Goals

The binocular disparity estimation algorithms in state-of-the-art either target very high-quality results but no real-time performance, or real-time implementation but they do not provide high-quality results for high-resolution video. The target of this thesis is carefully designing an high-quality disparity estimation algorithm considering the details of the hardware implementation such as minimizing hardware resource utilization and memory bandwidth, and maximizing the parallel and local processing. Thereby, high-quality and high-resolution real-time disparity estimation hardware can be used in advanced 3D-based video processing applications.

Many of the hardware implementations of the disparity estimation are not implemented in to ASIC, or their hardware implementations into an ASIC would be challenging since they require to access external memory. In this thesis, a streaming-input and streaming-output hardware implementation of the binocular DE which does not require the existence of external memory is targeted. Thereby, designing an efficent ASIC implementation of the binocular DE hardware is targetted. The targetted ASIC can significantly reduce the cost and power consumption compared to the disparity estimation hardware implemented for FPGA, which is important if massive production of the depth estimation system for consumer electronics is required.

A novel compressed look-up-table based rectification algorithm and its efficient hardware implementation are targetted. Consuming lower memory size than existing look-up-table based solutions while keeping the high-quality is targetted. Thereby, look-up-table based rectification can be utilized without using external memory, which can reduce the total cost of the depth estimation system.

Trinocular DE solves most of the occlusion problems in disparity estimation. Although few trinocular disparity estimation hardware implementations have been presented, none of them targets real-time for high-resolution video. This thesis targets the first high-resolution and high-quality trinocular disparity estimation hardware. Therefore, the targetted trinocular DE hardware can be used if removing the wrong estimation results in the occluded region is a significant necessity of a 3D-based video processing application.

Free-view synthesis is an important application of depth estimation which is mainly used for generating multi-views of glass-free TVs. The first high-resolution real-time free viewpoint synthesis hardware utilizing trinocular disparity estimation is targetted. Therefore, the targetted hardware can be used in glass-free 3D TVs to synthesize free view images in real-time.

Infrared projection based depth estimation devices such as Microsoft Kinect [32] and Structure Sensor of Occipital [33] are used for several 3D based video processing applications such as skeleton detection and gaming. However, Kinect and Structure Sensor do not work efficiently if there are multiple devices viewing the same scene, or they are used under the sun-light. Whereas, disparity estimation does not present these interference problems. In this thesis, the demonstration of the complete disparity estimation system with several 3D based video processing applications is targetted. In order to reach this goal, real-time software implementations of speed and distance measurement, depth-based image thresholding, head-hands-shoulders tracking, virtual mouse using hand tracking, and face tracking integrated with free viewpoint synthesis are targetted. The target of this thesis is to evidence that proposed disparity estimation system provides high-quality depth estimation results to be used in advanced real-time 3D-based video processing applications. Thereby, implementation of a depth estimation system that can be used both in indoor and outdoor, and even if in the case of utilization of multiple devices is targetted.

In addition, in this thesis, the implementation of the high-quality omnidirectional multiple-camera system is targetted. In order to reach this goal, improving the realist aspects of existing Panoptic camera is first targetted with an enhanced omnidirectional image reconstruction algorithm and its implementation. Since the omnidirectional video quality of the Panoptic camera is limited by the 352×288 resolution of the image sensors, an implementation of a novel high-resolution multiple-camera omnidirectional video recording system using 5MP cameras is targetted. Thereby, the goal is to implement the highest resolution 360° omnidirectional camera video record system which can be used in high-quality virtual reality applications. Moreover, the goal of the novel 360° omnidirectional camera system is to view the target locations close to horizontal plane with more than two cameras to allow its usage in depth map estimation in the future.

3 Binocular Adaptive Window Size Disparity Estimation Algorithm and Its Hardware Implementation

A hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time reconfigurable hardware implementation are presented in this Chapter [51]. The implemented hardware processes high resolution (HR) stereo video with high-quality disparity estimation results. In addition, the disparity estimation quality of the AWDE algorithm is improved using the iterative disparity refinement process. The proposed enhanced AWDE algorithm that utilizes Iterative Refinement (AWDE-IR) is implemented in hardware and its implementation details are presented [61].

3.1 Binocular Hardware-Oriented Adaptive Window Size Disparity Estimation Algorithm

The main focus of the AWDE algorithm is its compatibility with real-time hardware implementation while providing high quality DE results for HR. The algorithm is designed to be efficiently parallelized, to require minimal on-chip memory size and external memory bandwidth.

The term “block” is used in this Thesis to define the 49 pixels in the left image that are processed in parallel. The term “window” is used to define the 49 sampled neighboring pixels of any pixel in the right or left images with variable sizes of 7 × 7, 13 × 13 or 25 × 25. The pixels in the window are used to calculate the Census and BW-SAD cost metrics during the search process.

The algorithm consists of three main parts: window size determination, disparity voting, and disparity refinement. The parameters that are used in the AWDE algorithm are given in Section 3.4.

3.1.1 Window Size Determination

The window size of the 49 pixels in each block is adaptively determined according to the Mean Absolute Deviation (MAD) of the pixel in the center of the block with its neighbors. The formula of the MAD is presented in (3.1), where $\boldsymbol{c}$ is the center pixel location of the block and $\boldsymbol{q}$ is the pixel location in the neighborhood, N_c, of $\boldsymbol{c}$. The center of the block is the pixel located at block(4, 4) in Fig. 3.1. A high MAD value is a sign of high texture content and a low MAD value is a sign of low texture content. Three different window sizes are used. As expressed in (3.2), a 7 × 7 window is used if the MAD of the center pixel is high, and a 25 × 25 window is used if the MAD is very low.

$$MAD(\mathbf{c}) = \frac{1}{48} \times \sum_{\mathbf{q} \in N_c} |I_L(\mathbf{q}) - I_L(\mathbf{c})| \tag{3.1}$$

$$\text{Window Size} = \begin{cases} 7 \times 7 & \text{if } MAD(\mathbf{c}) > tr_{7\times 7} \\ 13 \times 13 & \text{else if } MAD(\mathbf{c}) > tr_{13\times 13} \\ 25 \times 25 & \text{otherwise} \end{cases} \tag{3.2}$$

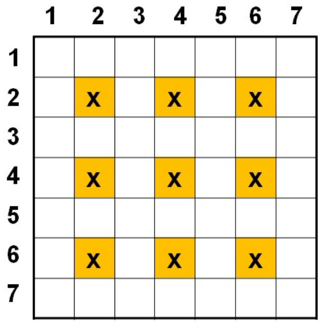


Figure 3.1: 9 selected pixels in a block for BW-SAD calculation. 49 pixels in a block are searched in parallel in hardware.

As a general rule, increasing the window size increases the algorithm and hardware complexity [60]. As shown in Fig. 3.2, in our proposed algorithm, in order to provide constant hardware complexity over the three different window sizes, 49 neighbors are constantly sampled for different window sizes. "1", "2" and "3" indicate the 49 pixels used for the different window sizes 7 × 7, 13 × 13 and 25 × 25, respectively. If the sampling of 49 pixels in a window is not applied and all the pixels in a window are used during the matching process, an improvement in the disparity estimation quality can be obtained. The overhead of computational complexity for this high-complexity case and the degradation of the DE quality due to sampling are presented in Section 3.4.

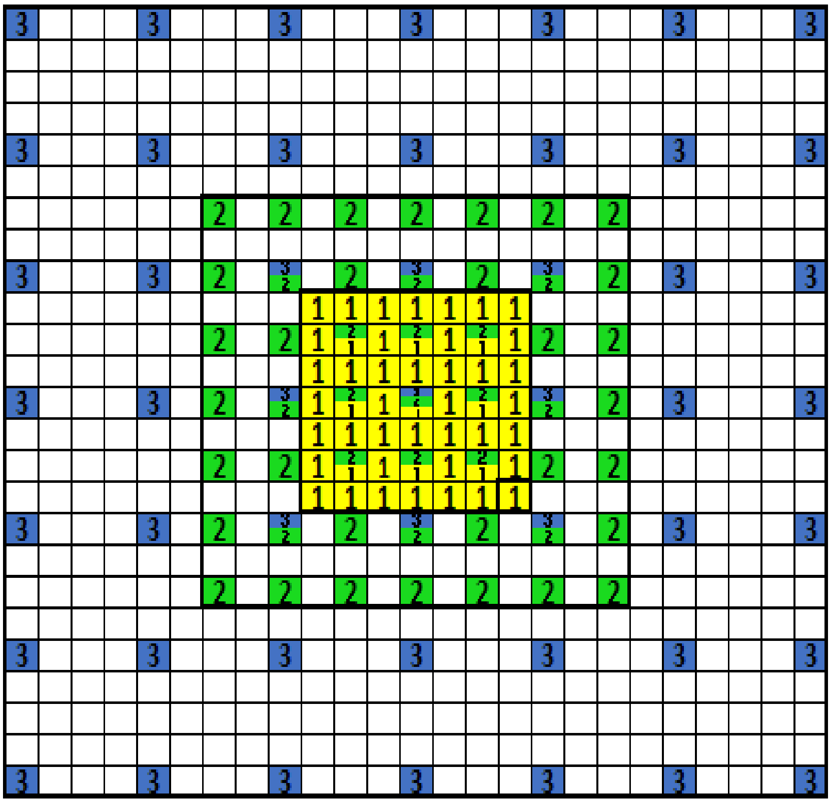


Figure 3.2: 49 selected pixels of adaptive windows (yellow (1): 7×7, green (2): 13×13 and blue (3): 25×25).

3.1.2 Disparity Voting

A hybrid solution involving the Binary Window SAD and Census cost computation methods is presented to benefit from their combined advantages. The SAD is one of the most commonly used similarity metrics. The use of BW-SAD provides better results than using the SAD when there is disparity discontinuity since it combines information about the shape of the object with the SAD [60]. However, the computational complexity of the BW-SAD is high, thus result of this metric is provided for nine of the 49 pixels in a block and they are linearly interpolated to find the BW-SAD values for the remaining 40 pixels in a block. The selected nine pixels for the computation of BW-SAD are shown in Fig. 3.1. The low complexity Census metric is computed for all of the 49 pixels of a block.

The formula expressing the BW-SAD for a pixel $\boldsymbol{p}$=(x, y) is shown in (3.3) and (3.4). The

BW-SAD is calculated over all pixels ***q*** of a neighborhood N_p, where the notation d is used to denote the disparity. The Binary Window, ***w***, is used to accumulate absolute differences of the pixels, if they have an intensity value which is similar to the intensity value of the center of the window. The multiplication with ***w*** in (3.4) does not cause significant computational load for the hardware since it is implemented as reset signal for the resulting absolute differences (AD). In the rest of the paper, the term, "*Shape*" is indicated by ***w***.

Depending on the texture of the image, the Census and the BW-SAD have different strengths and sensibility for the disparity calculation. To this purpose, a hybrid selection method is used to combine them. As shown in (3.5) and (3.6), an adaptive penalty (*ap*) that depends on the texture observed in the image is applied to the Hamming differences. Subsequently, the disparity with the minimum *Hybrid Cost* (HC) is selected as the disparity of a searched pixel. 2's order penalty values are used to turn the multiplication operation into a shift operation. If there is a texture on the block, the BW-SAD difference between the candidate disparities needs to be more convincing to change the decision of Census, thus a higher penalty value is applied. If there is no texture on the block, a small penalty value is applied since the BW-SAD metric is more reliable than the decision of Census.

$$\mathbf{w} = \begin{cases} 0 & \text{if } |I_L(\mathbf{q}) - I_L(\mathbf{p})| > threshold_w, \mathbf{q} \in N_p \\ 1 & \text{else} \end{cases} \tag{3.3}$$

$$BW\text{-}SAD(\mathbf{p}, d) = \sum_{\mathbf{q} \in N_p} |I_L(\mathbf{q}) - I_R(\mathbf{q} - d)| \cdot \mathbf{w} \tag{3.4}$$

$$HC(\mathbf{p}, d) = BW\text{-}SAD(\mathbf{p}, d) + \text{hamming}(\mathbf{p}, d) \times ap \tag{3.5}$$

$$ap = \begin{cases} ap_{7\times7} & \text{if} \quad \text{window size} == 7 \times 7 \\ ap_{13\times13} & \text{else if} \quad \text{window size} == 13 \times 13 \\ ap_{25\times25} & \text{else if} \quad \text{window size} == 25 \times 25 \end{cases} \tag{3.6}$$

3.1.3 Disparity Refinement

The proposed Disparity Refinement (DR) process assumes that neighboring pixels within the same *Shape* needs to have an identical disparity value, since they may belong to one unique object. In order to remove the faulty computations, the most frequent disparity value within the *Shape* is used.

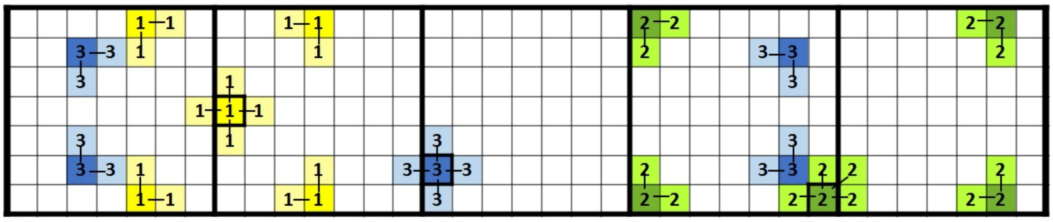


Figure 3.3: Examples for selecting 17 contributing pixels for 7×7, 13×13 and 25×25 window sizes during the disparity refinement process (yellow (1): 7×7, green (2): 13×13 and blue (3): 25×25).

As shown in Fig. 3.3, since the proposed hardware processes seven rows in parallel during the search process of a block, the DR process only takes the disparity of pixels in the processed seven rows. The DR process of each pixel is complemented with the disparities of 16 neighbor pixels and its own disparity value. Finally, the most frequent disparity in the selected 17 contributors is replaced with the disparity of that processed pixel.

The selection of these 17 contributors proceeds as follows. The disparity of the processed pixel and the disparity of its four adjacent pixels always contribute to the selection of the most frequent disparity. Four farthest possible *Shape* locations are pre-computed as a mask. If these locations are activated by *Shape*, the disparity values of these corner locations and their two adjacent pixels also contribute. Therefore, at most 17 and at least 5 disparities contribute to the refinement process of each pixel.

In Fig. 3.3, examples of the selection of contributing pixel locations are shown for three different window sizes. Considering the proposed contributor selection scheme, the pixels in the same row with the same window size have identical masks. The masks for the seven rows of a block and three window sizes are different. Therefore, 21 different masks are applied in the refinement process. These masks turn out to simple wiring in hardware.

Median filtering of the selected 17 contributors provides negligible improvement on the DR quality, but it requires high-complexity sorting scheme. The highest frequency selection is used for the refinement process since it can be implemented in hardware with low-complexity equality comparators and accumulators. The maximum number of contributors is fixed to 17 which provides an efficient trade off between hardware complexity and the disparity estimation quality.

3.2 Hardware Implementation of Proposed Binocular AWDE Algorithm

The efficient hardware implementation of the proposed hardware-oriented binocular AWDE algorithm is presented in this Section. The proposed hardware architecture of the AWDE algorithm enables handling 60 fps on a Virtex-5 FPGA at a 1024×768 XGA video resolution for a 128 pixel disparity. The proposed hardware provides dynamic and static configurability to have satisfactory disparity estimation quality for the images with different contents. It provides dynamic reconfigurability to switch between window sizes of 7×7, 13×13 and 25×25 pixels in run-time to adapt to the texture of the image. In addition, it provides static configurability to allow users to change the disparity range, the strengths of Census and BW-SAD in *HC* computation, the closest and furthest expected distances, the used color domain (Y, Cb or Cr), etc.

3.2.1 Overview

The top-level block diagram of the proposed reconfigurable disparity estimation hardware and the required embedded system components for the realization of the full system are shown in Fig. 3.4. The details of main real-time video processing hardware core of binocular disparity estimation is presented in this section. The details of the remaining embedded system components are presented in Chapter 6. The proposed Reconfigurable Disparity Map Estimation module involves 5 sub-modules and 62 dual port BRAMs. These five sub-modules are the Control Unit, Reconfigurable Data Allocation, Reconfigurable Computation of Metrics (RCM), Adaptive Disparity Selection (ADS) and Disparity Refinement. 31 of the 62 BRAMs are used to store 31 consecutive rows of the right image, and the remaining 31 BRAMs are used to store 31 rows of the left image. The dual port feature of the BRAMs is exploited to replace processed pixels with the new required pixels during the search process. The proposed hardware is designed to find the disparity of the pixels in the left image by searching candidates in the right image. The pixels of the right image are not searched in the left image, and thus cross-check of the DE is not applied.

The external memory bandwidth is an important limitation for disparity estimation of HR images. For example, the disparity estimation of a 1024 × 768 resolution stereo video at 60 fps requires 540 MB/s memory bandwidth considering loading and reading each image one time. The ZBT SRAM and DDR2 memories that are mounted on FPGA prototyping boards

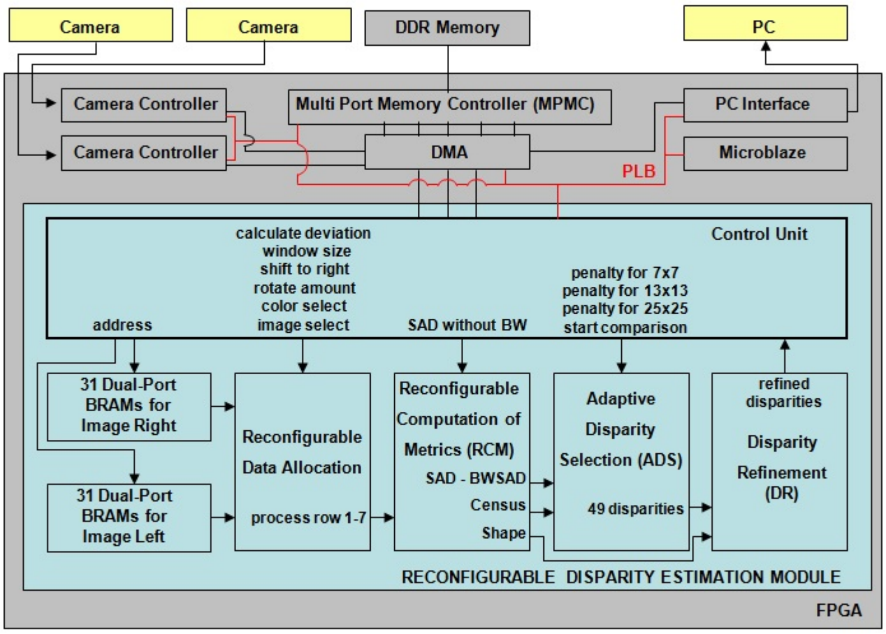


Figure 3.4: Top level block diagram of the disparity estimation module.

can typically reach approximately 1 GB/s and 5 GB/s, respectively. However, an algorithm or hardware implementation that requires multiple reads of a pixel from an external memory can easily exceed these bandwidth limitations. Using multiple stereo cameras in future targets or combining different applications in one system may bring external memory bandwidth challenges. The hardware in [39] needs to access external memory at least five times for each pixel. The hardware presented in [43] requires external memory accesses at least seven times for each pixel assuming that the entire data allocation scheme is explained. Our proposed memory organization and data allocation scheme require reading each pixel only one time from the external memory during the search process. In addition, it can be adapted to receive stream input and provide stream output without using external memory when the number of input buffer BRAMs are increased from 62 to 78 as presented in Section 6.

The system timing diagram of the AWDE is presented in Fig. 3.5. The disparity refinement process is not applied to the pixels that belong to the two blocks at the right and left edges of the left image. For the graphical visualization of the reconfigurable disparity computation process together with the disparity refinement process, the timing diagram is started from the process of the sixth block of the left image. As presented in Fig. 3.5, efficient pipelining is applied between the disparity refinement and disparity selection processes. Therefore, the disparity refinement process does not affect the overall system throughput but only increases the latency. The system is able to process 49 pixels every 197 clock cycles for a 128 pixel disparity search range. Important timings during the processes are also presented with dashed lines along with their explanations.

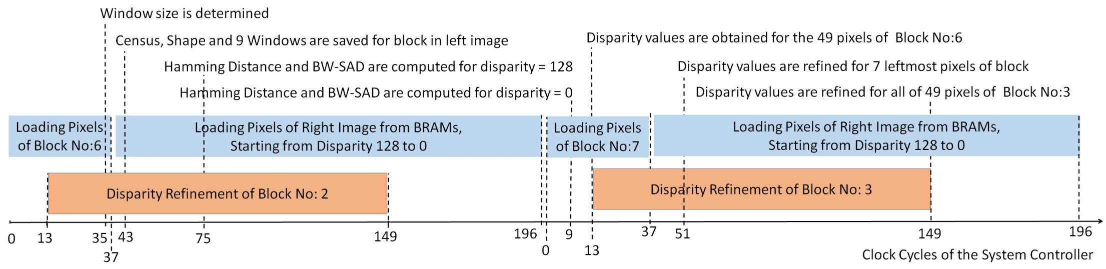


Figure 3.5: Timing diagram of the system.

3.2.2 Data Allocation and Disparity Voting

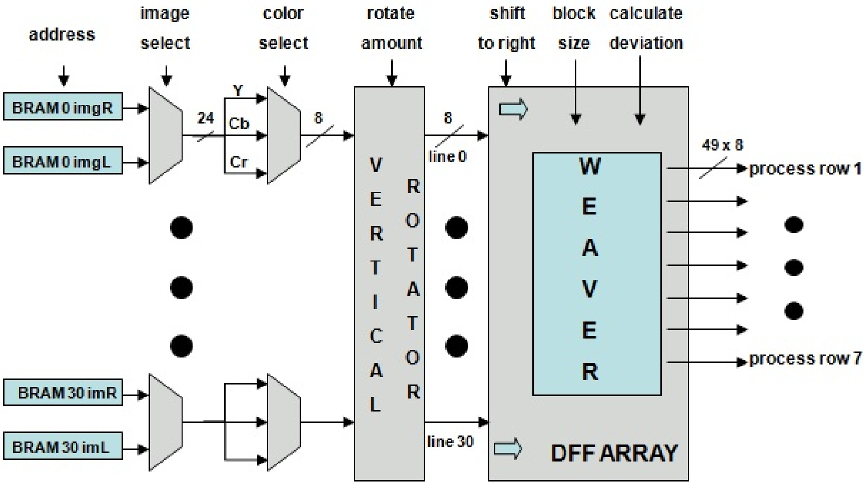


Figure 3.6: Block diagram of the Reconfigurable Data Allocation Module.

The block diagram of the Reconfigurable Data Allocation module is shown in Fig. 3.6. The data allocation module reads pixels from BRAMs, and depending on the processed rows, it rotates the rows using the Vertical Rotator to maintain the consecutive order. This process is controlled by the Control Unit through the *rotate amount* signal. The search process starts with reading the 31 × 31 size window of the searched block from the BRAMs of the left image. Therefore, the Control Unit sends the *image select* signal to the multiplexers that are shown in Fig. 3.6 to select the BRAMs of the left image. Moreover, the *color select* signal provides static configurability to select one of the pixel's components (Y, Cb or Cr) during the search process. This user-triggered selection is useful if the Y components of the pixels are not well distributed on the histogram of the captured images. While the windows of the searched block are loaded to the D flip-flop (DFF) Array, the RCM computes and stores the 49 Census transforms, 49 *Shapes* and 9 windows pertaining to the pixels in the block for the computation of BW-SAD.

The Census transforms and windows of the candidate pixels in the right image are also needed for the matching process. After loading the pixels for the computation of metrics for the 7 × 7 block, the Control Unit selects the pixels in the right image by changing the *image select* signal, and starts to read the pixels in the right image from the highest level of disparity by sending the address signals of the candidate pixels to the BRAMs.

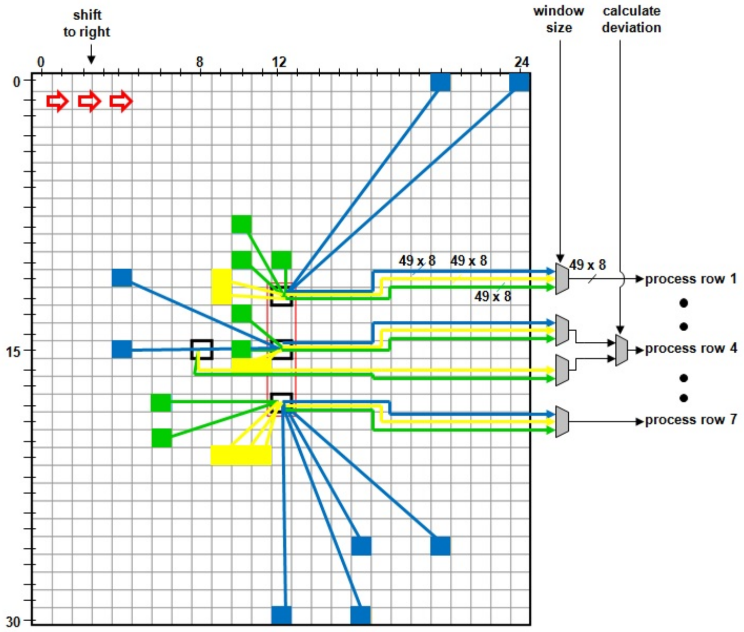


Figure 3.7: DFF Array and the Weaver (yellow: 7 × 7, green: 13 × 13 and blue: 25 × 25).

The disparity range can be configured by the user depending on the expected distance to the objects. Configuring the hardware for a low disparity range increases the hardware speed. In contrast, a high disparity range allows the user to find the depth of close objects. The architecture proposed in [43] is not able to provide this configurability since it is designed to search 80 disparity candidates in parallel, instead of providing parallelization to search multiple pixels in the left image. Therefore, a fixed amount of disparities is searched in [43], and changing the disparity range requires a redesign of their hardware.

The detailed block diagram of the DFF Array and the Weaver are shown in Fig. 3.7. They are the units of the system that provide the configurability of the adaptive window size. As a terminology, the term *weaving* is used to denote selecting 49 contributor pixels in different window sizes 7 × 7, 13 × 13 and 25 × 25 by skipping 1, 2 and 4 pixels respectively. Seven rows and one column are processed in parallel by the Weaver, and the processed pixels flow inside the DFF Array from the left to the right. Additionally, the weaving process is applied to the location (15, 8) of the DFF Array at the beginning of the search process only, to select the window size by computing the deviation of the center of the block from its neighbors for 7 × 7 and 13 × 13 windows.

The DFF Array is a 31 × 25 array of 8-bit registers shown in Fig. 3.7. The DFF Array has 25 columns since it always takes the inputs of the largest window size, i.e. 25 × 25, and it has 12+12+7=31 rows to process seven rows in parallel. While the pixels are shifting to the right, the Weaver is able to select the 49 components of the different window sizes from the DFF Array with a simple wiring and multiplexing architecture. Some of the contributor pixels of the windows for different window sizes are shown in Fig. 3.7 in different colors. The Weaver and DFF Array are controlled by the Control Unit through the *calculate deviation*, *window size* and *shift to right* signals. The Weaver sends seven windows to be processed by RCM as *process row 1* to *process row 7*, and each *process row* consists of 49 selected pixels.

A large window size normally involves high amounts of pixels and thus requires more hardware resources and computational cost to support the matching process. By using the proposed weaving architecture, even if the window size is changed, the windows only consist of 49 selected pixels. Therefore, the proposed hardware architecture is able to reach the largest window size (25 × 25) among the hardware architectures implemented for DE [39–45]. The adaptability of the window size between the small and large window sizes provides high-quality disparity estimation results for HR images.

During the weaving process of the 49 pixels in the block and the candidate pixels in the right image, the RCM computes the Census and *Shape* of these pixels in a pipeline architecture. The block diagram of the RCM is shown in Fig. 3.8. The process for each block starts by computing and storing the Census and *Shape* results for the 7 × 7 block. In Fig. 3.8, the registers are named as $Shape_{row\_column}$ and $Census_{row\_column}$. Since the BW-SAD is only applied for 9 of the 49 pixels, the BW-SAD computation sub-modules are only implemented in *process rows 2*, *4* and *6*.

The BW-SAD sub-module in Fig. 3.8 takes the *Shape*, registered window of the pixel in a block and the candidate window of the searched pixel as inputs, and provides the BW-SAD result as an output. The computation of the Hamming distance requires significantly less hardware area than the BW-SAD. Therefore, the Hamming computation is used for all of the 49 pixels in a block.

As shown in Fig. 3.8, when a new candidate Census for the *process row 1* is computed by the Census sub-module of the RCM, its Hamming distance with the preliminary computed seven $\text{Census}_{1\_[1:7]}$ of the block is computed by the seven Hamming sub-modules. The seven resulting Hamming Results of the *process row 1* are passed to the ADS module. Since this process also progresses in parallel for seven *process rows*, the proposed hardware is able to compute the Hamming distances of 49 pixels in a block in parallel. This parallel processing

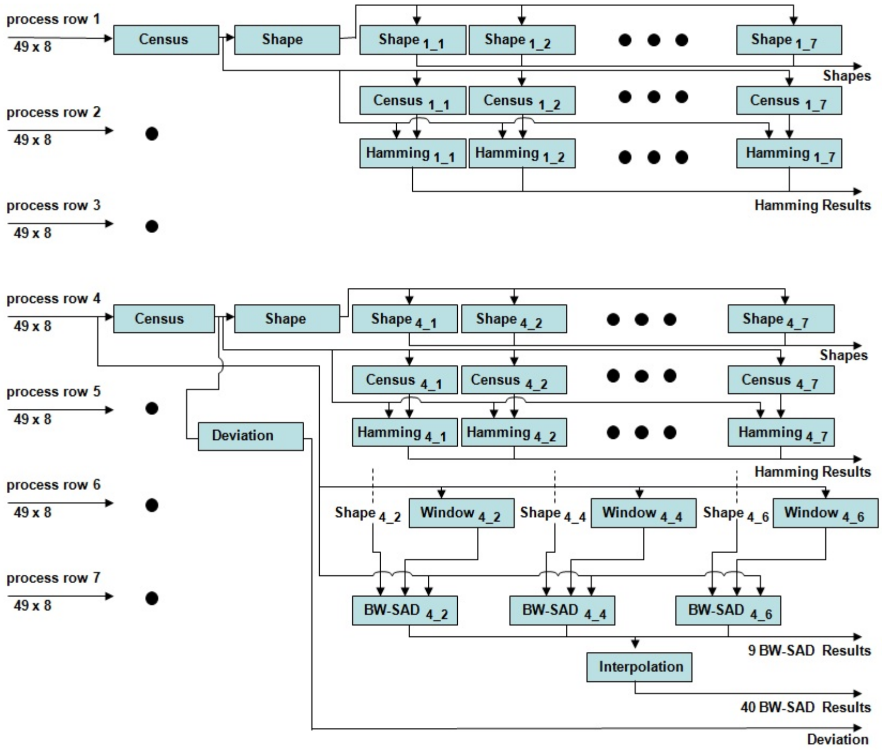


Figure 3.8: Block diagram of the Reconfigurable Computation of Metrics.

scheme is presented in Fig. 3.9. While the proposed architecture computes the Hamming distance for the left-most pixels of the block, the Hamming for disparity *d*, rightmost pixels of the block computes their Hamming for disparity *d+6*. Therefore, the resulting Hamming costs are delayed in the ADS to synchronize the costs. This delay is also an issue of the BW-SAD results and they are also synchronized in the ADS.

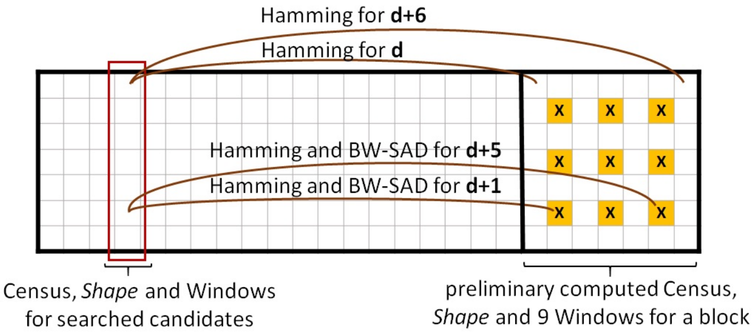

Hamming for **d+6**
Hamming for **d**
Hamming and BW-SAD for **d+5**
Hamming and BW-SAD for **d+1**
X
Census, *Shape* and Windows
for searched candidates
preliminary computed Census,
Shape and 9 Windows for a block

Figure 3.9: Processing Scheme ("x" indicates 9 selected pixels in a block for BW-SAD calculations).

The internal architecture of the Census transform involves 48 subtractors. The Census module subtracts the intensity of center from the 48 neighboring pixels in a window, and uses the sign bit of the subtraction to define 48-bit Census result. The *Shape* computation module reuses the subtraction results of Census module. The *Shape* module takes the absolute values of the subtraction results and compares the absolute values with the $threshold_w$. The Hamming computation module applies 48-bit XOR operation and counts the number of 1s with an adder tree.

The Deviation module shown in Fig. 3.8 only exists on the *process row 4* since it is only needed for the center of the 7 × 7 block to determine the window size. The module accumulates the absolute difference of the 48 neighboring pixels from the center. The Control Unit receives the deviation result of the 7 × 7 and 13 × 13 window sizes in consecutive clock cycles and determines window size. The mathematical calculation of the MAD requires dividing the total deviation by 48. In order to remove the complexity of the division hardware, the thresholds $tr_{7\times7}$ and $tr_{13\times13}$ are re-computed by multiplying them with 48 and compared with the resulting absolute deviations.

The use of BW-SAD provides better results than using the SAD in presence of disparity discontinuities [60]. However, if the processed image involves a significant amount of texture without much depth discontinuity, using the regular SAD provides better results. Especially for the 7 × 7 window size, using SAD instead of BW-SAD provides better visual results since it is the sign of significantly textured region. In order to take advantage of this property, dynamic configurability is provided to change the BW-SAD computation metric to the SAD computation for a 7 × 7 window. The SAD module computes the ADs and the result of ADs are stored in registers prior to accumulation. An active-low reset signal is used at the register of the AD to make its result 0, when the architecture is configured for the BW-SAD, and the respective *Shape* of the pixel in the block is 0. Otherwise, the AD register takes its actual value.

The ADS module, which is shown in Fig. 3.4 receives the Hamming results and the BW-SAD results from the RCM block and determines the disparity of the searched pixels. Since the BW-SAD results are computed for 9 of the 49 pixels, the RCM linearly interpolates these nine values to find the estimated BW-SAD results of the remaining 40 pixels in the block. Due to an efficient positioning of the nine pixels in a block, the linear interpolation requires a division by 2 and 4, which are implemented as shift operations.

The ADS module shifts the Hamming results of the candidate pixels depending on the 2's order adaptive penalty for the multiplication process as shown in formula (3.5). The ADS module adds the resulting Hamming penalty on the BW-SADs to compute *Hybrid Costs*. 49 comparators are used to select the 49 disparity results that point minimum *Hybrid Costs*.

3.2.3 Disparity Refinement

The DR module receives the 49 disparity results from the ADS and the *Shapes* of the 49 pixels of a block from the RCM and determines the final refined disparity values. As presented in Fig. 3.10, after the ADS module has computed 49 disparity values in parallel, it loads this data in to the DFF Array of the DR module (DR-Array). The DR-Array has a size of five blocks for the refinement process. The Control Unit enables the DFFs by using the *Load Disparity* signal when the 49 disparity outputs of ADS module are ready for the refinement process. In each cell of the DR-Array, the respective *Shape* of a pixel is loaded from the RCM using the *Load Shape* signal. DR-Array is designed to shift the disparity and *Shape* values from right to left to allocate data for the refinement processes.

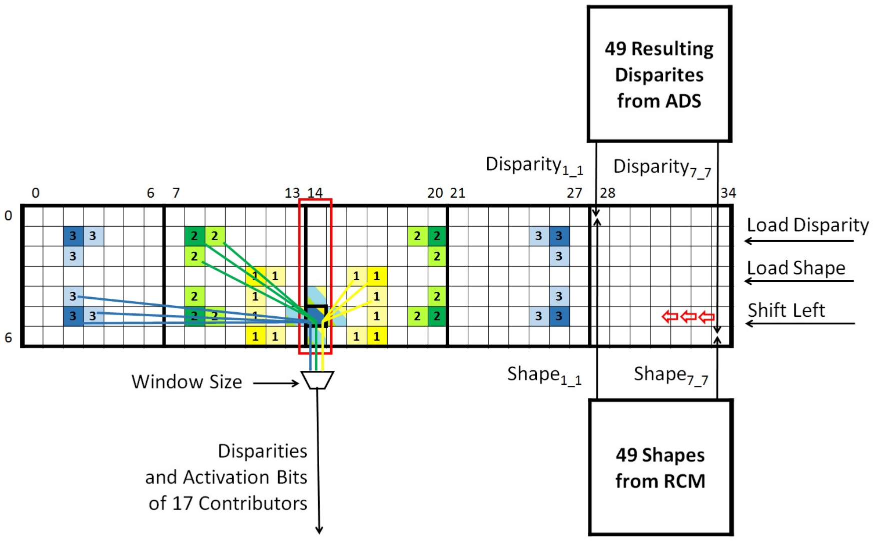


Figure 3.10: Disparity Refinement-Array of the Disparity Refinement Module (yellow (1): 7 × 7, green (2): 13 × 13 and blue (3): 25 × 25).

The DR hardware contains a Highest Frequency Selection (HFS) module that consists of seven identical Processing Elements (DR-PE). As presented in Fig. 3.10, DR-PEs are positioned to refine seven disparities in the 15th column of the DR Array in parallel while the disparity and *Shape* values shift through the DR-Array. The hardware architecture of a single DR-PE is presented in Fig. 3.11. The location of a single DR-PE is indictated in the 6th row of the DR-Array with a bold square.

In Fig. 3.10, while 17 disparity values are selected by the multiplexers, the *Shape* information corresponding to the four corners is also selected from the 49-bit *Shape* information of the processed pixel. The selected 4-bits inform the DR-PE which of these 12 disparity values on the corners will be used while computing the highest frequency disparity. These 4 bits of the *Shape* are called *activation bits* in Fig. 3.11. Each *activation bit* activates itself together

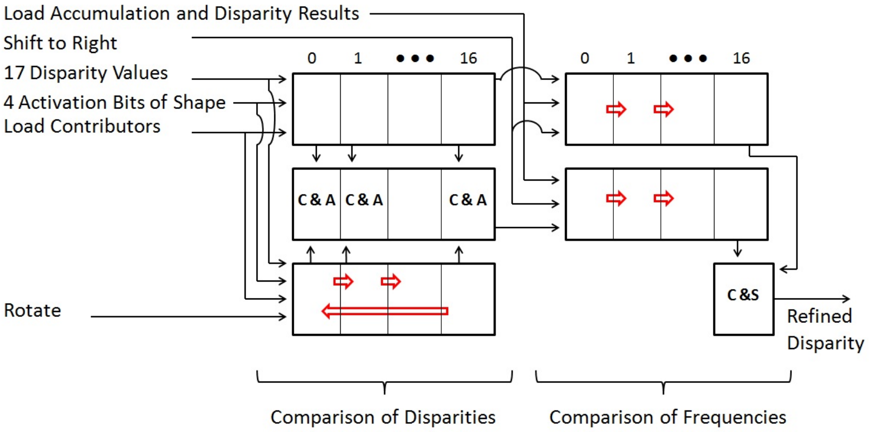

Load Accumulation and Disparity Results
Shift to Right
17 Disparity Values
4 Activation Bits of Shape
Load Contributors
Rotate
0
1
•••
16
C & A
C & A
C & A
0
1
•••
16
C&S
Refined Disparity
Comparison of Disparities
Comparison of Frequencies

Figure 3.11: Processing Element of the Disparity Refinement Module. The Highest Frequency Selection Module includes seven of these DR-PE elements.

with its two adjacent disparities. Since the center disparity and its four neighbors are always activated, the 17-bit activation information is loaded to the DR-PE together with the respective disparities.

As presented in Fig. 3.11, the DR-PE hardware consists of two parts: Comparison of Disparities and Comparison of Frequencies. In the Comparison of Disparities part, the 17-bit activation information and the 17 disparities are stored into two DFF Arrays. One of these DFF Arrays is used as a reference and the other one rotates to compare each disparity with the 16 other disparities. During the rotation process, 17 Compare and Accumulate (C&A) sub-modules compare the disparities in parallel. If the compared disparities are identical and both of them are activated, the values of the accumulators are increased by one. After 17 clock cycles, the values in the accumulators and their respective disparities are loaded into the DFF Array in the Comparison of Frequencies part of the DR-PE. In the pipeline architecture, at the same time, the Control Unit shifts the DR-Array to the left by one to load new 17 contributors to the DR-PE. The Compare and Select (C&S) sub-module compares the values of the accumulators to find the highest value in the accumulators, and selects the disparity with the highest frequency as the refined disparity. Since the DR process works in parallel with the other hardware modules of AWDE, it does not affect the throughput of the DE system if the disparity range is configured as more than 70.

3.3 Iterative Refinement for the Enhanced AWDE implementation

The intuition behind the proposed Iterative Refinement process of the IR-AWDE algorithm is identical to the DR process presented in the Section 3.1.3: neighboring pixels within the same *Shape* need to have an identical disparity value, since they may belong to one unique object.

Using the refinement process multiple times removes noisy computations more efficiently, and increases the disparity estimation quality.

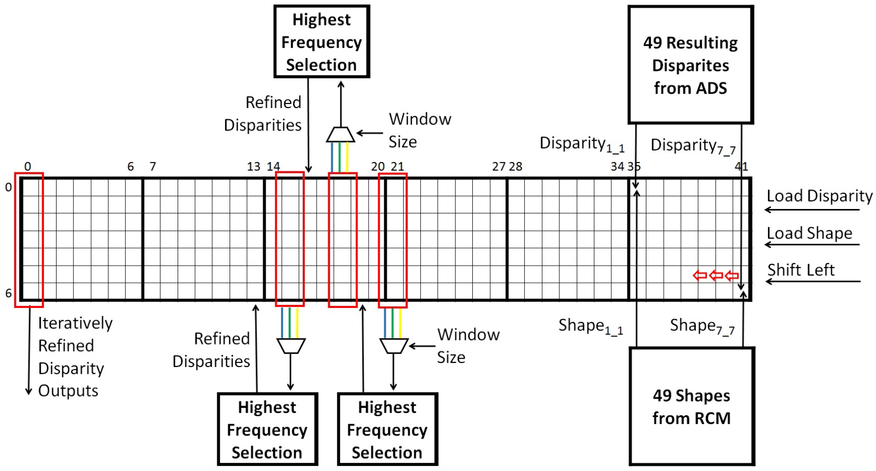


Figure 3.12: DR-Array of the Iterative Disparity Refinement Module (yellow line: 7 × 17 candidates for 7 × 7 window, green line: candidates for 13 × 13, and blue line: candidates for 25 × 25).

The iterative refinement hardware is presented in Fig. 3.12 which consists of an improved version of the DR hardware presented in Fig. 3.10. The proposed Iterative Refinement process utilizes three concatenated Highest Frequency Selection modules. Each HFS module includes seven identical DR-PEs, one of which is presented in Fig. 3.11. All DR-PEs receive 17 selected disparities from their own multiplexer. The DR-Array in Fig. 3.10 includes DFFs to keep record of the computed disparities for five blocks. Instead, for the IR, the size of the DFF-Array is increased to six blocks since the disparities need to be pipelined for longer duration. Moreover, the DR hardware presented in Fig. 3.10 provides the most frequent disparities as an output as the refined disparities. Instead, the HFS modules for the IR hardware write back the refined disparities on DR-Array. Writing back the most frequent disparities into the DR-Array provides an iterative refinement of the estimated disparities. Since the disparity results shift inside the DR-Array, refined disparities are overwritten 2 pixels left of the consecutive pixel location. For example, as presented in Fig. 3.12, while the HFS module refines the disparities of the seven pixels in column 21 of the DR-Array, the DR-Array shifts the disparity values 2 times. Therefore, the computed seven highest frequency disparities in the column 19 of the DR-Array are overwritten.

In addition to removing noisy computations, IR provides efficient results in assigning disparities of occluded regions. While searching pixels from the left image inside the right image, occluded regions appear on the left side of objects [35]. Consequently, wrong computations due to occlusion appear on the left sides of the objects in the image, which should be replaced by the correct disparities that are assigned to the left adjacent pixels of the occluded ones. The proposed iterative refinement process scans the estimated disparities from left to right. In addition, HFS modules receive updated disparities from their left since they are already overwritten by the refined ones. Therefore, this process iteratively spreads the correct disparities to the occluded regions while considering the object boundaries with the *Shape* information. While disparities shift inside the DR-Array, the leftmost disparities in the column 0 of the DR-Array are provided as the refined disparity value outputs of the IR Module.

3.4 Implementation Results

The reconfigurable hardware architecture of the proposed AWDE algorithm is implemented using Verilog HDL, and verified using Modelsim 6.6c. The Verilog RTL models are mapped to a Virtex-5 XCUVP-110T FPGA comprising 69k Look-Up Tables (LUT), 69k DFFs and 144 Block RAMs (BRAM). The proposed hardware consumes 59% of the LUTs, 51% of the DFF resources and 42% of the BRAM resources of the Virtex-5 FPGA. The proposed hardware operates at 190 MHz after place & route and computes the disparities of 49 pixels in 197 clock cycles for 128 pixel disparity range. Therefore, it can process 60 fps at a 1024 × 768 XGA video resolution.

The AWDE-IR is implemented to further improve the disparity estimation quality of AWDE using an efficient iterative refinement step. The hardware implementation of AWDE-IR is mapped to a same FPGA and verified using Modelsim 6.6c. The proposed AWDE-IR hardware consumes 70% of the LUTs, 63% of the DFF resources and 42% of the BRAM resources of the Virtex-5 FPGA. It can work at same speed performance due to the pipeline structure of the refinement process.

The parameters of the AWDE algorithm are shown in Table 3.1. Parameters are selected by sweeping to obtain high quality DE of HR images considering different features pertaining to the image content.

Table 3.1: Parameters of the AWDE

$tr_{7\times7}$	$tr_{13\times13}$	$ap_{7\times7}$	$ap_{13\times13}$	$ap_{25\times25}$	$threshold_w$
5	2	32	16	4	8

Table 3.2 and Table 3.3 compare the disparity estimation performance and hardware implementation results of the AWDE architecture with other existing hardware implementations that targets HR [43–45] and currently the highest quality DE hardware that targets LR [39]. These papers do not provide the disparity estimation quality results for the HR benchmarks of the Middlebury data-set [35]. Thus, we implemented [39, 43, 45] in software, and the software implementation of [44] is obtained from its authors. The DE results for the

Table 3.2: Disparity Estimation Performance Comparisons. Error rates (%) are provided compared to DE ground truths of the benchmark pictures.

	Tsukuba (288x384)	Venus (383x434)	Aloe (1110x1282)	Art (1110x1390)	Clothes (1110x1300)
Chang [39]	4.15	0.56	3.75	12.80	2.97
Ttofis [44]	13.21	4.56	8.88	32.18	7.67
Greisen [45]	12.42	4.14	8.65	23.46	5.30
Georgoulas [43]	12.38	15.20	6.97	23.75	9.15
Census7	26.05	30.80	20.36	45.39	21.80
Census13	18.19	18.83	11.21	31.65	9.36
Census25	15.94	15.38	10.41	29.66	7.16
BWSAD7	12.19	19.45	8.31	34.03	13.33
BWSAD13	11.23	15.16	7.13	28.57	9.27
BWSAD25	10.43	11.12	6.74	24.74	6.28
FWDE7	9.53	12.59	5.38	20.87	5.39
FWDE13	7.90	6.82	4.81	16.97	3.16
FWDE25	8.03	5.66	5.16	18.12	3.87
AWDE	7.64	5.33	4.94	16.33	2.89
AWDE-HC	7.47	4.73	4.92	16.17	2.95
AWDE-IR	6.53	5.01	4.30	14.47	2.94

Census and the BW-SAD metrics for different window sizes are also presented in Table 3.2. The comparisons of the resulting disparities with the ground-truths are done as prescribed by the Middlebury evaluation module. If the estimated disparity value is not within a ±1 range of the ground truth, the disparity estimation of the respective pixel is considered erroneous. 18 pixels located on the borders are neglected in the evaluation of LR benchmarks Tsukuba and Venus, and a disparity range of 30 is applied for all algorithms. 30 pixels located on the borders are neglected in the evaluation of HR benchmarks Aloe, Art and Clothes, and a disparity range of 120 is applied for all algorithms.

The Census and BW-SAD results that are shown in Table 3.2 are provided by sampling 49 pixels in a window. FW-DE indicates the combination of BW-SAD and Census for a fixed window size. The numbers terminating the name of the algorithms indicate the fixed window sizes of these algorithms.

Although the Census and the BW-SAD algorithms do not individually provide very efficient results, the combination of these algorithms into the FW-DE provides an efficient hybrid solution as presented in Table 3.2. For example, if a 7×7 window size and Census method are exclusively used for DE on the HR benchmark Art, 45.39% erroneous DE computation is observed from the result of Census7. Exclusively using a 7×7 window size and BW-SAD method for the same image yields 34.03% erroneous computation. However, if only a 7×7 window size is used combining the Census and BW-SAD methods, 20.87% erroneous computation is observed as presented in the result of FW-DE7. 20.87% erroneous computation is significantly smaller than 45.39% and 34.03%, which justifies the importance of combining the Census and BW-SAD into a hybrid solution. For the same image, using the FW-DE13 and FW-DE25 algorithms yields 16.97% and 18.12% erroneous DE computations, respectively. Combining the FW-DE7, FW-DE13 and FW-DE25 into a reconfigurable hardware with an adaptive window size feature further improves the algorithm results as demonstrated from the results of AWDE. AWDE provides 16.33% erroneous computation for the same image which is smaller than 20.87%, 16.97% and 18.12%, thus numerically emphasizing the importance of adaptive window size selection. The algorithmic performance of AWDE, 16.33%, is considerably better than the DE performance results of HR DE hardware implementations [44], [45] and [43] that provide 32.18%, 23.46% and 23.75% erroneous computations respectively for the same image.

If the sampling of 49 pixels in a window is not applied and all the pixels in a window are used during the matching process, the complexity of the AWDE algorithm increases by 12 times. The result of the high complexity version of the AWDE algorithm (AWDE-HC) is also provided in Table 3.2 for comparison. The AWDE-HC provides almost the same quality results as the AWDE. Considering the hardware overhead of AWDE-HC, the low complexity version of the algorithm, AWDE, is selected for hardware implementation, and its efficient reconfigurable hardware is presented.

Improving the results of AWDE is possible using the low complexity iterative refinement step as indicated from the results of AWDE-IR. AWDE-IR efficiently removes a significant amount of noisy computations by iteratively replacing the disparity estimations with the most frequent neighboring ones as can be observed from the results of Tsukuba, Venus, Aloe and Art. Moreover, IR does not require significant amount of additional computational complexity. Therefore, AWDE-IR is implemented in hardware for the further improvement of the disparity estimation quality.

The algorithm presented in [39] uses the Census algorithm with the cost aggregation method, and provides the best results for both LR and HR stereo images except the HR benchmark Clothes. As shown in Table 3.3, due to the high-complexity of cost aggregation, it only reaches 42 fps for CIF images, thereby consuming a large amount of hardware resource. If the performance of [39] is scaled to 1024 × 768 for a disparity range of 128, less than 3 fps can be achieved.

None of the compared algorithms that have a real-time HR hardware implementation [43–45] is able to exceed the DE quality of AWDE and AWDE-IR for HR images. The overall best results following the results of AWDE and AWDE-IR are obtained from [45]. The hardware

Table 3.3: Hardware Performance Comparison

Hardware	Technology	Image Resolution	DFF consumption	LUT consumption	Disparity Range	fps	Clock Speed (MHz)
Chang[39]	ASIC-90nm	352×288	562k Gates		64	42	95
Ttofis[44]	Virtex-5	1280×1024	31k	47k	120	50	100
Greis.[45]	Stratix-III	1920×1080	26k	54k	256	30	130
Georg.[43]	Stratix-IV	800×600	15k	146k	80	550	511
AWDE	Virtex-5	1024×768	35k	40k	128	60	190
		640×480			64	221	
		352×288			64	670	
AWDE-IR	Virtex-5	1024×768	43k	48k	128	60	190
		640×480			64	221	
		352×288			64	670	

presented in [45] consumes 20% of the 270k Adaptive LUT (ALUT) resources of a Stratix-III FPGA. It provides high disparity range due to its hierarchical structure. However, this structure easily causes faulty computations when the disparity selection finds wrong matches in low resolution.

The hardware implementation of [43] provides the highest speed performance in our comparison. However this hardware applies 480 SAD computations for a 7 × 7 window in parallel. The hardware presented in [43] consumes 60% of the 244k ALUT resources of a Stratix-IV FPGA. In our hardware implementation we only use 9 SAD computations in parallel for the same size window and this module consumes 16% of the resources of a Virtex-5 FPGA on its own. Therefore, the hardware proposed in [43] may not fit into 3 Virtex-5 FPGAs.

The visual results of the AWDE and AWDE-IR algorithms for the HR benchmarks Clothes, Art and Aloe are shown in Fig. 3.13, Fig. 3.14 and Fig. 3.15, respectively. The disparity map result of the AWDE algorithm for the 1024 × 768 resolution pictures taken by our stereo camera system is shown in Fig. 3.16. The proposed binocular disparity estimation hardware architectures provide both quantitative and visual satisfactory results and they reach real-time for HR.

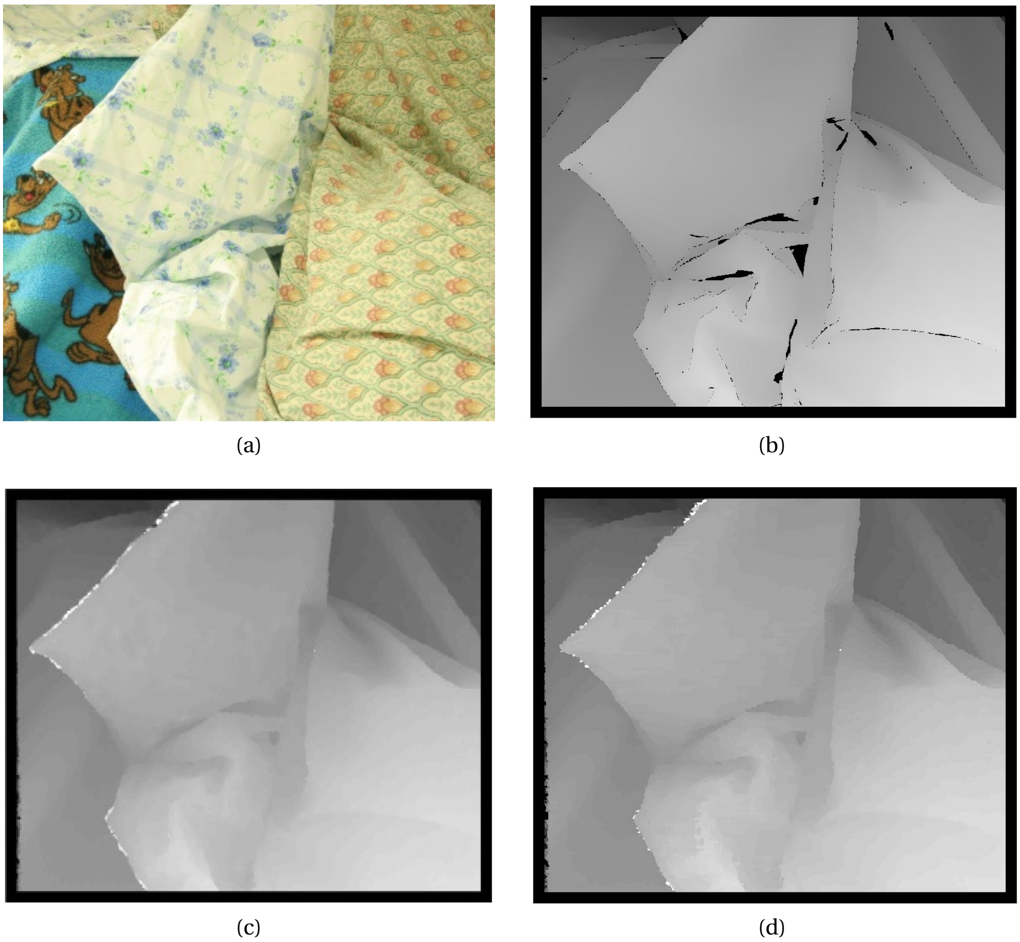
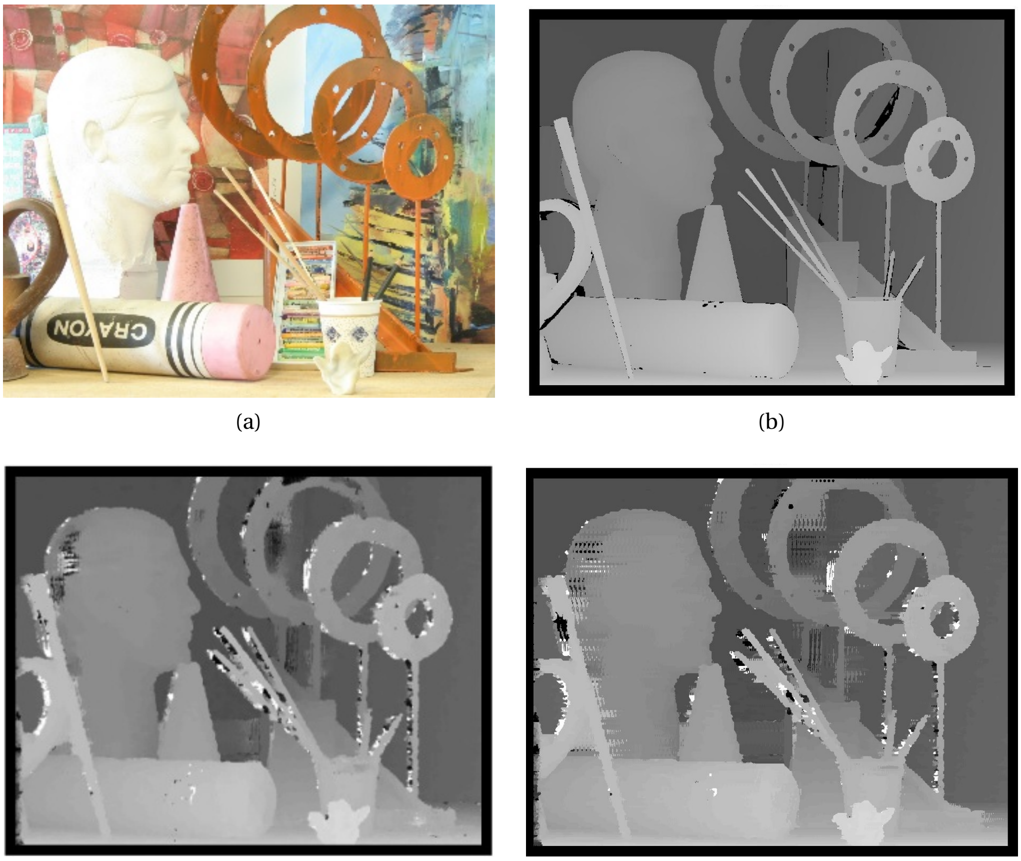
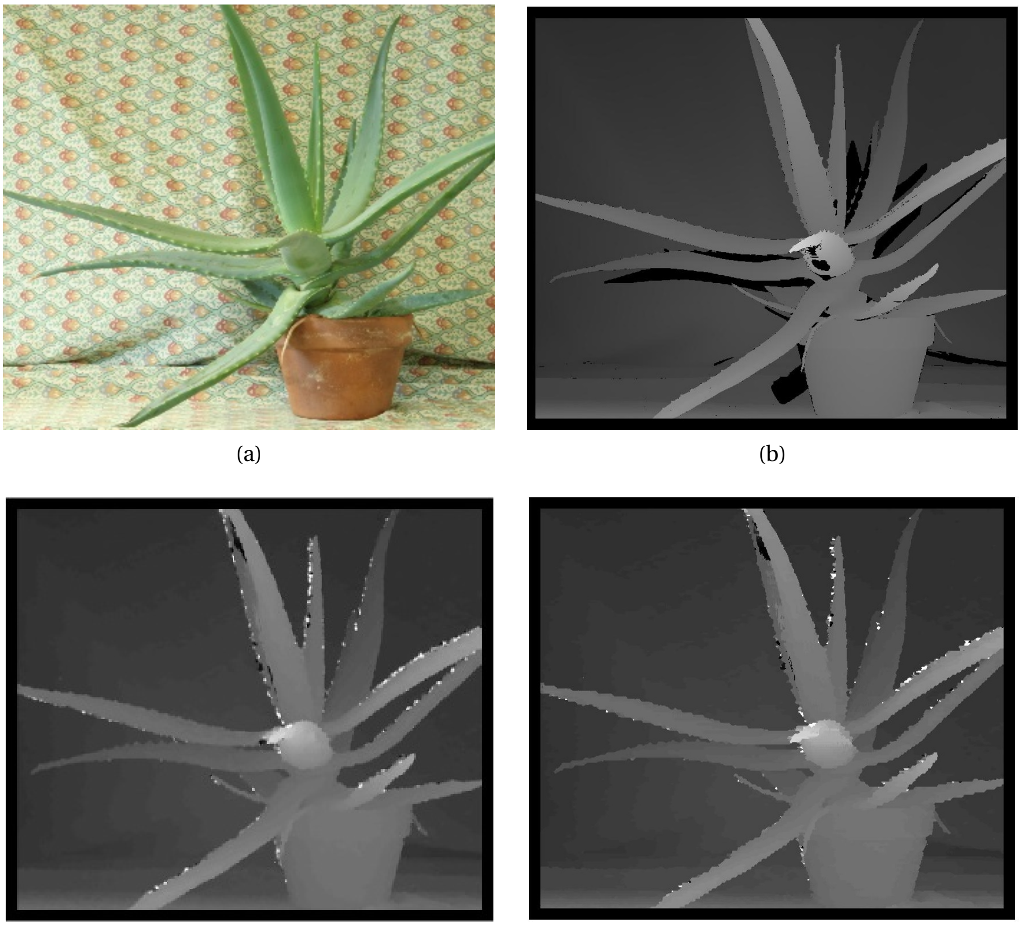

(a) (b) (c) (d)

Figure 3.13: Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmark Clothes. Black regions in the ground truths are not taken into account for the error computations as explained in Middlebury evaluation. (a) left image (b) ground truth (c) DE result of AWDE (d) DE result of AWDE-IR



(a) (b) (c) (d)

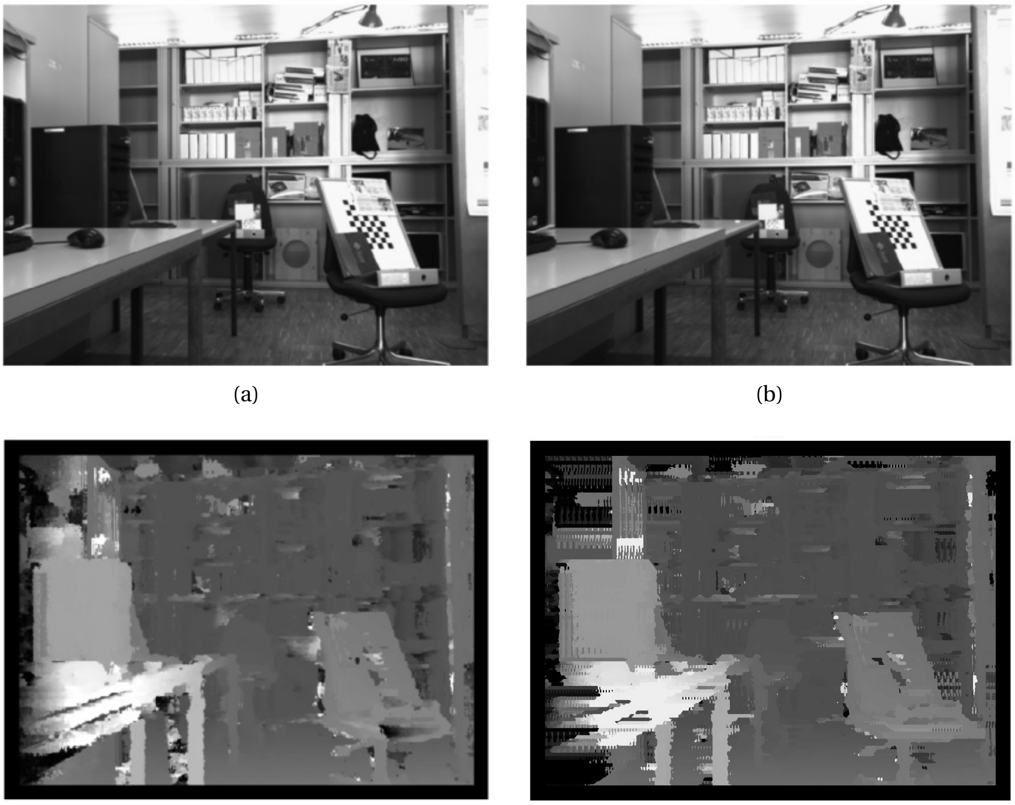
Figure 3.14: Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmark Art. Black regions in the ground truths are not taken into account for the error computations as explained in Middlebury evaluation. (a) left image (b) ground truth (c) DE result of AWDE (d) DE result of AWDE-IR



(a) (b)

(c) (d)

Figure 3.15: Visual disparity estimation results of AWDE and AWDE-IR algorithms for HR benchmark Aloe. Black regions in the ground truths are not taken into account for the error computations as explained in Middlebury evaluation. (a) left image (b) ground truth (c) DE result of AWDE (d) DE result of AWDE-IR



(a) (b) (c) (d)

Figure 3.16: Visual disparity estimation results of AWDE and AWDE-IR algorithms for the 1024 × 768 resolution pictures captured by the implemented stereo camera system. The ground truth for these images is not available. (a) left image (b) right image (c) DE result of AWDE (d) DE result of AWDE-IR

4 Trinocular Adaptive Window Size Disparity Estimation Algorithm and Its Hardware Implementation

This chapter presents a hardware-oriented trinocular adaptive window size disparity estimation (T-AWDE) algorithm and the first real-time trinocular disparity estimation (DE) hardware that targets high-resolution images with high-quality disparity results [88]. The proposed trinocular DE hardware is the enhanced version of the binocular AWDE implementation that is presented in Chapter 3. The T-AWDE hardware generates a very high quality depth map by merging two depth maps obtained from the center-left and center-right camera pairs. The T-AWDE hardware enhances disparity results by applying a double checking scheme which solves most of the occlusion problems existing in the AWDE implementation while providing correct disparity results even for objects located at left or right edge of the center image.

4.1 Trinocular Hardware-Oriented Adaptive Window Size Disparity Estimation Algorithm

The proposed T-AWDE algorithm is developed to support efficient parallel operations, to consume low hardware resources and to avoid the requirement of an external memory while providing very high-quality DE results. As presented in Fig. 4.1, while processing the trinocular DE for every pixel of the center image, the candidate disparities on the right side are searched for the center-left pair, and the candidate disparities on the left side are searched for the center-right pair. Therefore, in the T-AWDE algorithm, two disparity maps are calculated for the center-left and center-right pairs. The T-AWDE algorithm combines these two disparity maps to provide a very high quality DE result for the center camera. The proposed T-AWDE algorithm consists of six main parts: preprocessing, window size determination, matching cost calculation, disparity selection, fusion of the disparity maps, and iterative refinement (IR).

Image rectification is one of the most essential preprocessing parts of DE. The rectification process requires internal and external calibrations to model distortions of the lenses and the mechanical misalignment of the cameras. The Open-CV calibration toolbox [70] is used for external and internal calibrations. The Caltech rectification algorithm [50] is used to horizontally align the images captured from three cameras.

The window size determination, matching cost calculation and binocular disparity selection parts of the T-AWDE algorithm for each camera pair are identical to the AWDE algorithm. As presented in Fig. 4.1, the T-AWDE searches 49 pixels of the center image in parallel in candidate disparities of the right and left images. Each of these 49 pixels is independently searched, using their own window. The selection of a large window size improves the algorithm performance in textureless regions while requiring higher computational load. However, the usage of small window sizes provides better disparity results in regions where the image has a texture. The T-AWDE dynamically changes the windows size as either 7 × 7, 13 × 13 or 25 × 25 pixels to adapt to different texture features of the images. It utilizes the mean absolute deviation (MAD) to measure the local texture feature of the center image, and compares the MAD values with the threshold values to adaptively determine the window size. A constant hardware complexity over the three different window sizes is provided by constantly selecting 49 contributor pixels in different window sizes of 7 × 7, 13 × 13 and 25 × 25 pixels by skipping 1, 2 and 4 pixels, respectively. The T-AWDE adaptively utilizes the Census and binary-window sum of absolute difference (BW-SAD) matrixes as a hybrid solution. The computation of the Hybrid Cost (HC) provides high quality results for object boundaries and adapts to different light conditions of real images. Further details about window size determination, matching cost calculation and binocular disparity selection are presented in Chapter 3.

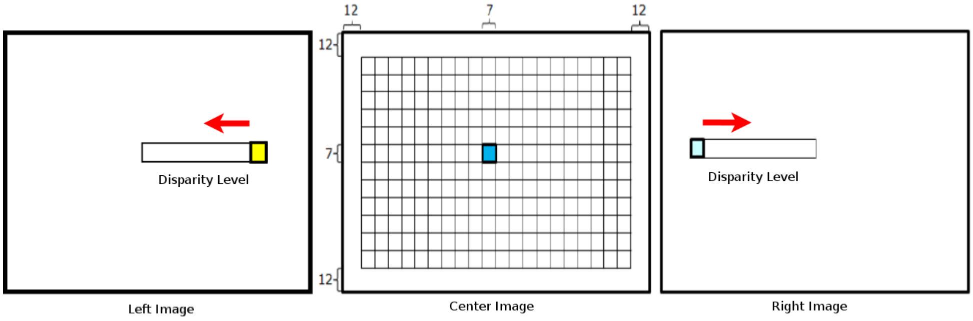


Figure 4.1: Matching directions of the T-AWDE algorithm

The fusion process includes two steps. In the first step, the T-AWDE compares HC values of the center-left and center-right pairs for every disparity, in order to select the one that exhibits the minimum cost as a disparity value. The winner-take-all (WTA) approach provides high quality results especially in occluded and low textured regions thanks to the strengths of the AWDE algorithm and the usage of three horizontally aligned cameras. In the next step of the fusion, a *confidence* metric is computed to further improve the DE results. The expression of the *confidence* metric is presented in 4.1. Here, c_1 represents the minimum matching cost and d_1 is the corresponding disparity value of the minimum matching cost. c_2 represents the second minimum matching cost and d_2 is the corresponding disparity value of the second minimum matching cost. (c_1, d_1) and (c_2, d_2) belong to the overall matching cost computation obtained from any of the stereo pairs. This metric identifies the strength of the global minima compared to local minima. Ambiguity in the selected minima is identified when two low-value costs are detected within the disparity range, while very different attached distances are computed. The *confidence* value of d_1 is then assigned as 0. The *confidence* metric is used in the disparity refinement process.

$$Confidence = \begin{cases} 0 & (c_1 - c_2) \leq (c_1/4) \ and \ (|d_1 - d_2|) > 5 \\ 1 & otherwise \end{cases} \tag{4.1}$$

The T-AWDE smoothens the computed disparity map using the brightness values of the neighboring pixels following the IR scheme prescribed in Chapter 3. The refinement process assumes that neighboring pixels with similar brightness values need to have identical disparity values, since they may belong to one unique object. In the refinement process, the disparities that are not frequently observed in the neighborhood are considered as faulty computations, and they are replaced by the most frequent disparity value that is computed in the neighborhood. This process is iteratively handled from the left to the right of the disparity image. In addition to the IR scheme presented in Chapter 3, the *confidence* metric is used in the IR process of T-AWDE algorithm. The unconfident disparity values are disregarded while determining the most frequent disparity value in order to prevent propagating them into the final DE result. Using the *confidence* metric during the IR process eliminates a significant amount of incorrect propagations, especially within low-textured regions.

4.2 Trinocular Adaptive Window Size Disparity Estimation Hardware

The block diagram of the full system that implements the T-AWDE algorithm is presented in Fig. 4.2. A Virtex-7 FPGA included in the VC707 Evaluation Board is used to prototype the developed hardware. All real time video processing computations are implemented in hardware. The PC is used as a display, to control the system and to obtain camera calibration parameters. The resulting disparity images are transferred to the PC using 1Gb raw Ethernet. A standard 2D TV is connected to the PC using HDMI to offer a better display. A MicroBlaze softcore is used to initialize cameras through I^2C, to control Ethernet and to communicate with the PC. A DDR3 memory is only used for Ethernet buffering. The details of main real-time video processing hardware core of trinocular disparity estimation is presented in this Section. The details of the other embedded system components are presented in Chapter 6.

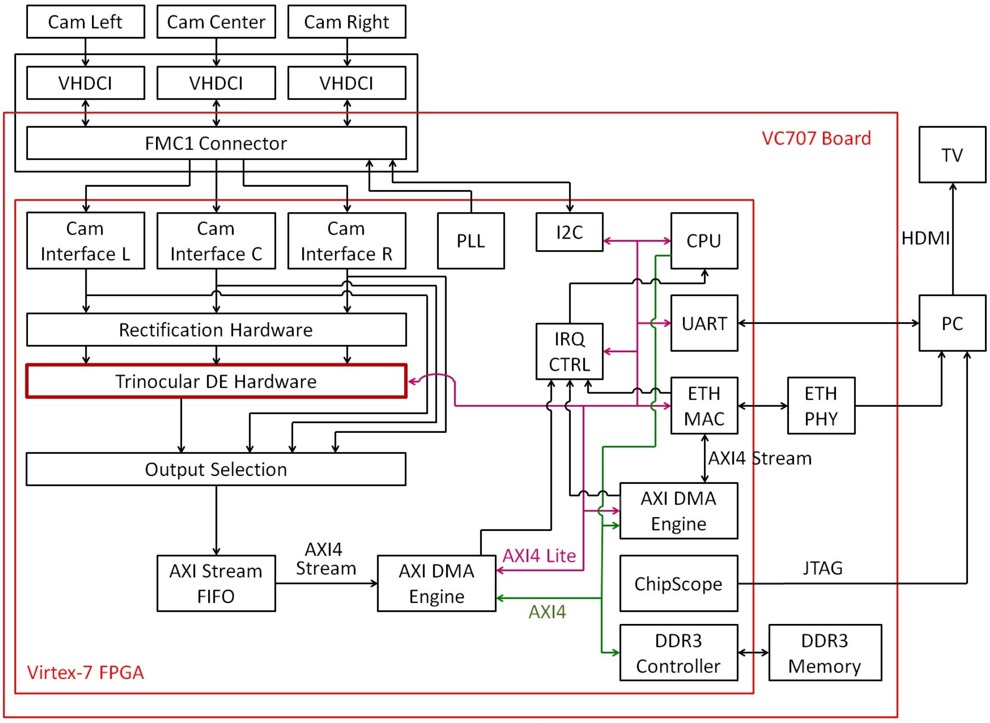


Figure 4.2: Block diagram of the full system.

The camera interface, rectification and disparity estimation blocks are designed to avoid using the DDR3 memory; thus they receive a streaming input and provide a streaming output. Therefore, the video processing core can be easily converted to a single ASIC. Eliminating the DDR3 memory from the video processing is a benefit of the efficient and hardware-oriented algorithm that only requires local processing, the utilization of local on-chip Block RAMs (BRAMs), and the perfect synchronization of cameras. Perfect synchronization of the cameras also always keeps fast moving objects in the same epipolar lines of the cameras to provide best DE quality for these objects. The system is able to deliver 32-bit pixel RGB+Disparity video of a center camera, or RGB video of any camera. Moreover, the system can be configured to deliver trinocular DE, center-left DE, or center-right DE results as a final disparity value to provide easy comparison of the results.

The two-cameras synchronization method presented in Chapter 9 is extended for a three-camera system. The perfect synchronization of cameras is achieved by providing a common clock source to the cameras from the FPGA, and simultaneously programming the cameras using the same I^2C module. Using the same clock source for the cameras is necessary to obtain an identical frame rate from the cameras. Simultaneous programming of the cameras is needed to start cameras exactly at the same time. The Xilinx I^{2}C IP and I^{2}C lines of the cameras are connected through multiplexers and tristate IO buffers as presented in Fig. 4.3. All three cameras have the same I^{2}C address. The MicroBlaze controls the select modes of the multiplexers using a 2-bit general purpose output (GPO). The multiplexing scheme allows to write the control registers of three camera sensors at the same time, to write to one of the selected camera, or to read from one of the selected camera. This connection method allows to concurrently write control registers of three cameras by ignoring the acknowledge signal. This transmission method does not conform the definition of I^{2}C. Nevertheless, according to our real-time tests, data is always correctly transmitted if I^{2}C is used at low frequency.

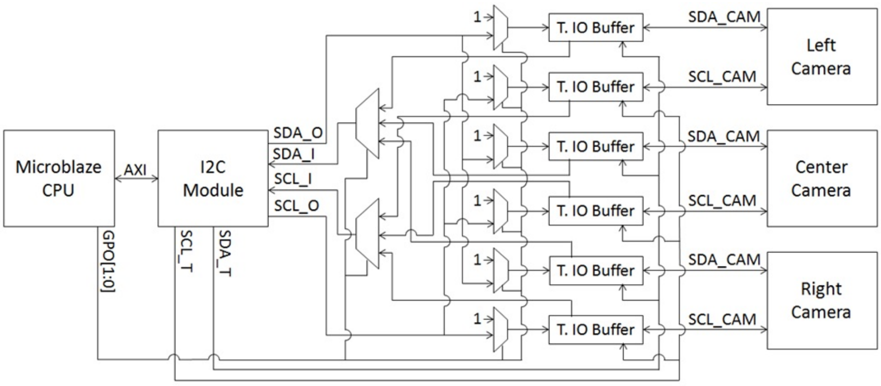


Figure 4.3: I^{2}C multiplexing hardware (resistors and level-shifters are not drawn)

The Caltech rectification hardware is implemented to solve the lens distortions and camera misalignments. The implemented rectification hardware utilizes 64 on-chip BRAMs for each camera. Each BRAM is used to buffer one row of the image. The rectification hardware processes the images of three cameras in parallel, and synchronously transfers rectified YCbCr images to the disparity estimation module. The proposed Caltech rectification hardware is presented in Chapter 5.

The block diagram of the trinocular DE hardware is shown in Fig. 4.4. The disparity estimation hardware buffers the input pixel values using 39 single-port BRAMs for each camera to realize a window-based matching scheme. The three-camera disparity estimation hardware is composed of two high-performance and high-quality binocular disparity estimators presented in Chapter 3. Each of these estimators includes three modules named as Reconfigurable Data Allocation (RDA), Reconfigurable Computation of Metrics (RCM) and Adaptive Disparity Selection (ADS). The Fusion module combines the DE outputs of the two estimators. The IR module smoothens DE computations and provides the final output of the trinocular DE. Finally, the 8-bit disparity values and the 24-bit RGB pixels of the left, center and right cameras are buffered and synchronized using 16 BRAMs for each channel before transferring this data to the Output Selection module that is presented in Fig. 4.2. The Controller generates read-addresses to the BRAMs, manages DE hardware modules to maintain their synchronous process, and interfaces with the MicroBlaze to apply user-programmable features that are provided from the GUI. In order to allow the programmability of the DE hardware, the controller includes software accessible registers. The hardware is configurable by the user who can select the maximum disparity range (maximum 255 is allowed), the disparity-start and disparity-end values to provide best DE quality at a certain distance interval, the strengths of the Census and BW-SAD metrics in the HC computation, and the resolution of depth images (maximum 1024×768 is allowed) to allow faster frame rates at lower resolution video.

The RDA module includes a vertical rotator, DFF-array and weaver sub-modules to arrange adaptive window sizes. The RCM computes the Census transforms, neighborhood information, BW-SAD and Hamming costs of the 49 parallel processed pixels. The ADS module receives the Hamming results and the BW-SAD results from the RCM, computes the HC values and determines the disparities of the 49 searched pixels pertaining to the center-left and center-right pairs.

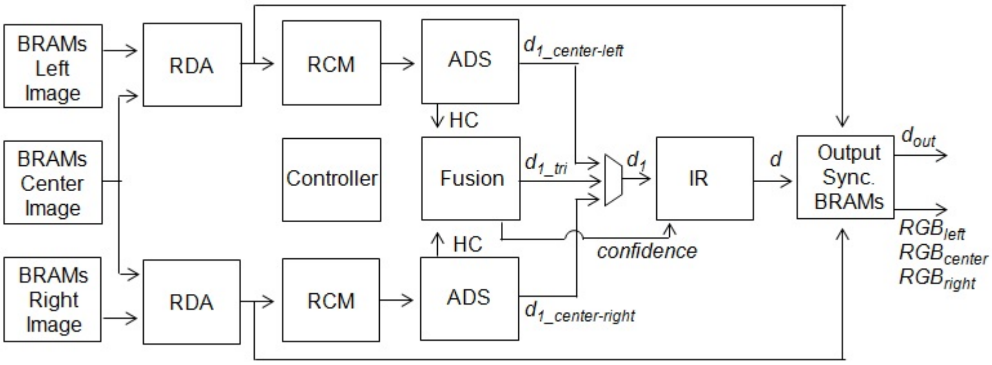


Figure 4.4: Block diagram of trinocular DE hardware

The parallel processing scheme of the T-AWDE hardware is presented in Fig. 4.5. The search process starts by reading the windows of the 49 processed pixels from the BRAMs of the center image. The RDA and RCM compute the Census and neighborhood information of the 49 processed pixels and permanently saves these values. Subsequently, the synchronous and symmetric scanning processes of candidate disparities start. The Controller starts to synchronously read the pixels in the right and left image from the highest level of disparity by sending the address signals of the candidate pixels to the BRAMs. The Controller sends addresses to the right image BRAMs in increasing order whereas it sends addresses to the left image BRAMs in decreasing order. Although the scanning process is symmetric, due to the square shape of the processed block, the Hamming and BW-SAD values of the pairs are not synchronously computed for the same disparity. For the center-right pair, while the proposed architecture computes the Hamming values for the left-most pixels of the block, the Hamming for disparity *d*, the rightmost pixels of the searched block computes their Hamming for disparity *d+6*. Whereas, for the center-left pair, while the proposed architecture computes the Hamming values for the left-most pixels of the block, the Hamming for disparity *d+6*, rightmost pixels of the block computes their Hamming for disparity *d*. The RCM applies additional pipelining to synchronize the matching cost. This synchronization process is necessary to allow fusion module to concurrently compare HC values obtained from pairs for the same disparity values.

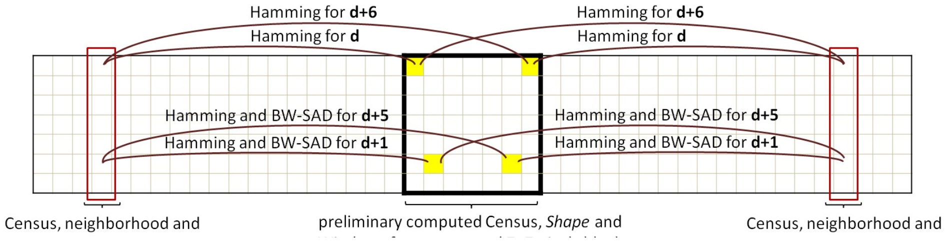

Hamming for **d+6**
Hamming for **d**
Hamming for **d+6**
Hamming for **d**
Hamming and BW-SAD for **d+5**
Hamming and BW-SAD for **d+1**
Hamming and BW-SAD for **d+5**
Hamming and BW-SAD for **d+1**
Census, neighborhood and
preliminary computed Census, *Shape* and
Census, neighborhood and

Figure 4.5: Parallel processing scheme for two pairs

The fusion module compares the HC values obtained from two pairs, and applies WTA to determine the trinocular disparity values of 49 pixels in parallel. Moreover it computes *confidence* values to be used for the IR process. The processing element of fusion hardware (PE-F) is presented in Fig. 4.6. The fusion hardware includes 49 PE-F to compute 49 trinocular disparities in parallel. The PE-F compares HC values obtained from two pairs to determine the c_1, d_1, c_2 and d_2 values using comparators and multiplexers. The PE-F computes mathematical operations of the *confidence* calculation presented in (1), and transfers d_1 and its respective *confidence* value to the disparity refinement module to compute the final disparity outputs.

The IR hardware of T-AWDE implementation is different from the hardware presented in Chapter 3, since the T-AWDE additionally utilizes a *confidence* metric. The IR module of the T-AWDE computes most frequent disparity value in the neighborhood and replaces noisy DE computations with the most frequent disparity values. If a particular disparity value is not deemed trustable, the value is removed from the neighborhood of other pixels by using logical AND gates. Using this method, only disparity values that are identified as confident propagate to the neighboring pixels.

4.3 Implementation Results

The proposed real-time trinocular DE hardware is implemented using Verilog HDL, and verified using Modelsim 10.1d. The Verilog RTL models are mapped to a Virtex 7 XC7VX485T FPGA comprising 607k Look-Up-Tables (LUT), 303k DFFs and 1030 BRAMs. The trinocular DE hardware consumes 25% of the LUTs, 11% of the DFF and 16% of the BRAM resources of

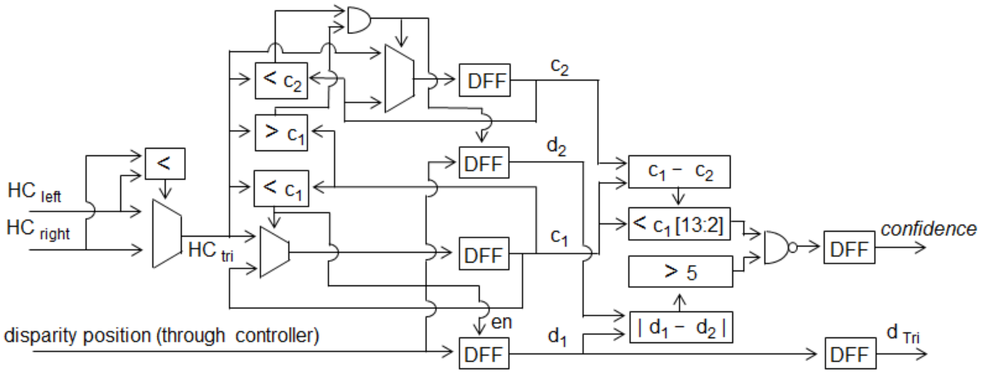
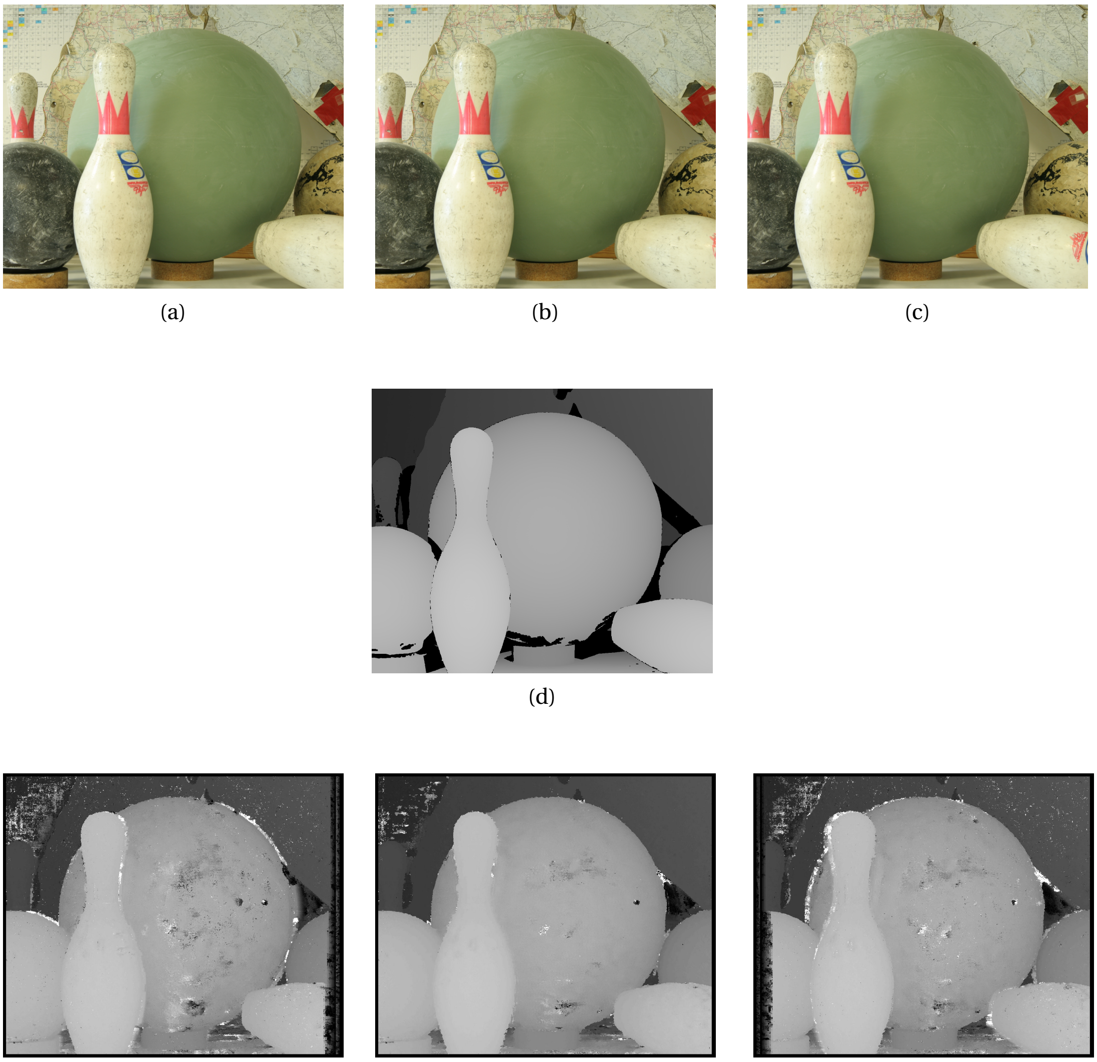


Figure 4.6: Processing element of the fusion module (PE-F). The fusion module includes 49 PE-F elements.

the FPGA. The proposed hardware operates at 175 MHz after place and route, and computes the disparities of 49 pixels in 198 clock cycles for a 128 pixel disparity range. Therefore, it can process 55 fps at a 768×1024 XGA video resolution. The system is functionally verified in real-time. Although a 55 fps performance is verified using Chipscope, the current display output of the system is 18 fps due to the bandwidth limitation of raw Ethernet output. The 55 fps performance of the hardware will be fully exploitable using USB3 or HDMI in the future versions.

The visual results of the T-AWDE and AWDE algorithms using the Middlebury benchmark image set Bowling2 (1276×1110) are obtained from MATLAB simulations and presented in Fig. 4.7. The comparisons of the resulting disparities with the ground-truth are realized as prescribed by the Middlebury evaluation module. Using the AWDE algorithm for center-left and center-right pairs yields 18.01% and 15.60% erroneous DE computations, respectively. Combining the binocular pairs into proposed trinocular DE improves the algorithm results as demonstrated in Fig. 4.7f. The proposed T-AWDE algorithm provides 9.41% erroneous computation for the same image thus numerically emphasizes the importance of proposed trinocular DE algorithm. 1024×768 resolution real-time snapshots captured by the proposed system are presented in Fig. 4.8. Models stand stable in front of the system while capturing multiple consecutive snapshots. The center-left, trinocular and center-right DE results are presented in Fig. 4.8d, 4.8e and 4.8f, respectively. The T-AWDE solves a significant amount of the occlusion and incorrect estimation errors exploiting the fusion of the DE results of the two pairs. Hence, the proposed T-AWDE system delivers high-quality results and realizes the first real-time trinocular DE hardware for high resolution.

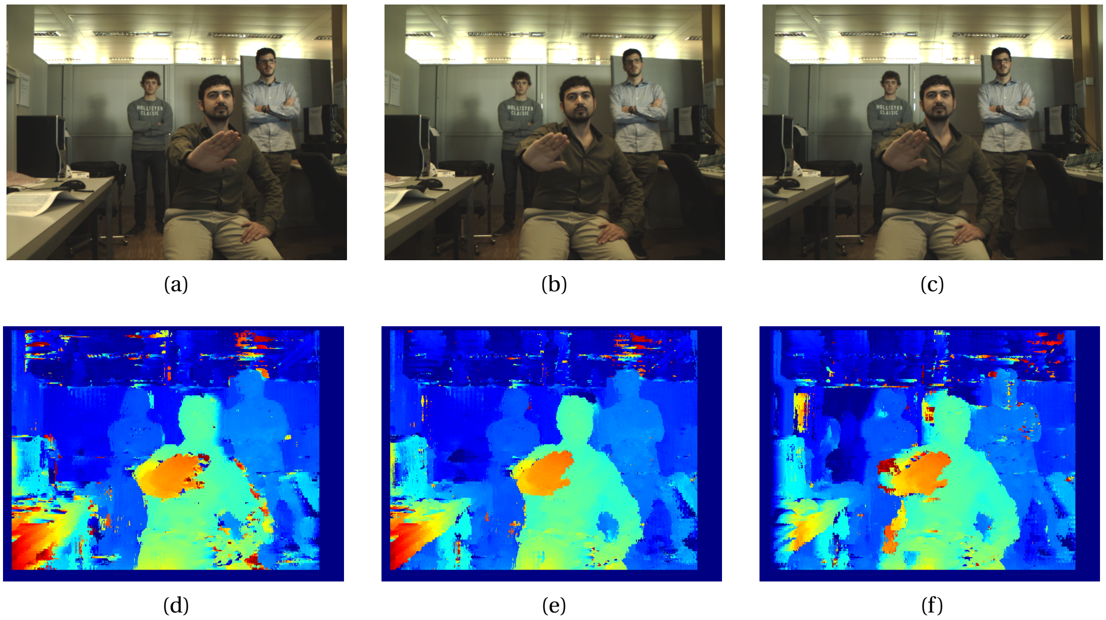


(a) (b) (c)

(d)

(e) (f) (g)

Figure 4.7: DE results obtained by MATLAB for Middlebury benchmarck image set “Bowling2” (a) Left Image (b) Center Image (c) Right Image (d) Ground Truth (black pixels are ignorable) (e) IR-AWDE for center-left (18.01%) (f) T-AWDE (9.41%) (g) IR- AWDE for center-right (15.60%)


(a) (b) (c)

(d) (e) (f)

Figure 4.8: Real-time snapshots captured by the proposed system. A ground truth for these images is not available. (a) Left Image (b) Center Image (c) Right Image (d) AWDE for center-left (e) T-AWDE (f) AWDE for center-right

5 Compressed Look-Up-Table Based Rectification Algorithms and Their Hardware Implementations

Stereo image rectification is a pre-processing step of disparity estimation intended to remove image distortions and camera misalignments to enable stereo matching along an epipolar line. A real-time disparity estimation system needs to perform real-time rectification which requires solving the models of lens distortions, image translations and rotations. Solving these complex equations requires a significant amount of hardware resources. Look-up-table based rectification algorithms allow image rectification without demanding high complexity operations. However, they require an external memory to store large size look-up-tables. In this chapter, a novel compressed look-up-table based rectification method is presented as an intermediate solution. The proposed method keeps the look-up-table based approach but compresses the look-up-tables to fit them into the on-chip memory of a Virtex-5 FPGA. In this chapter, first, a very low complexity compressed look-up-table based rectification algorithm (CLUTR) and its real-time hardware are presented. The implemented CLUTR hardware rectifies the stereo images if the lens distortion is not extreme and the cameras are not excessively misaligned. In order to solve more difficult camera alignment and distortion issues while maintaining the low complexity architecture, an enhanced version of the compressed look-up-table based rectification algorithm (E-CLUTR) and its real-time hardware are presented. The low-complexity de-compression processes of CLUTR and E-CLUTR require negligible amount of hardware resources for their real-time implementation. Furthermore, the Caltech rectification algorithm [50] which does not benefit from look-up-tables is implemented in hardware, and its hardware resource consumption results are presented to improve the hardware comparison and to evidence the efficiency of CLUTR and E-CLUTR much fairly.

5.1 Compressed Look-Up-Table based Rectification Algorithm

The CLUTR algorithm is proposed to compress the look-up-table based rectification information in order to fit them into the on-chip memory of a Virtex-5 FPGA. In this section, first standard look-up-table based rectification is briefly explained, then the proposed look-up-table compression scheme is presented.

In general, look-up-table based rectification methods can be distinguished by two different image warping flows: forward mapping and inverse mapping. Forward mapping computes the rectified target pixel locations based on the given pixel locations in the original image. Inverse mapping computes the original source pixel locations based on the given pixel locations in the rectified image. The mapping requires separate tables for X and Y coordinates, and for the right and left images. Therefore, four tables are needed. The formulations of forward and inverse mappings are presented in equations 5.1 and 5.2, respectively. In these equations, *ForwT* is the forward mapping table, *InvT* is the inverse mapping table, *Ori* represents the original image taken from the camera, *Rec* represents the rectified image. Y_{Rec}, X_{Rec}, Y_{ori} and X_{ori} represent the Y and X coordinates.

$$\begin{aligned} &Forward: (Y_{Rec}, X_{Rec}) = \left(ForwT_y(Y_{Ori}, X_{Ori}),\ ForwT_x(Y_{Ori}, X_{Ori})\right) \\ &Rec_{(y,\,x)} = linear\_interpolation\left(nearest\ neighbors\ of\ Rec_{(y,\,x)}\right) \end{aligned} \tag{5.1}$$

$$\begin{aligned} &Inverse: (Y_{Ori}, X_{Ori}) = \left(InvT_y(Y_{Rec}, X_{Rec}), InvT_x(Y_{Rec}, X_{Rec})\right) \\ &Rec(y, x) = linear\_interpolation\left(nearest\ neighbors\ of\ Ori_{(Y_{Ori},\, X_{Ori})}\right) \end{aligned} \tag{5.2}$$

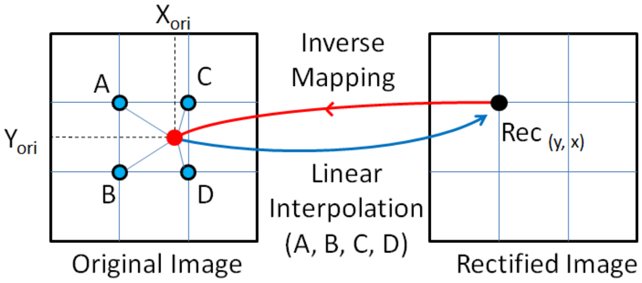


Figure 5.1: Inverse mapping with fractional precision coordinates. Corners indicate integer pixel coordinates.

A typical rectification process utilizes fractional pixel precision which requires the linear interpolation of four pixels. The linear interpolation schemes for inverse and forward

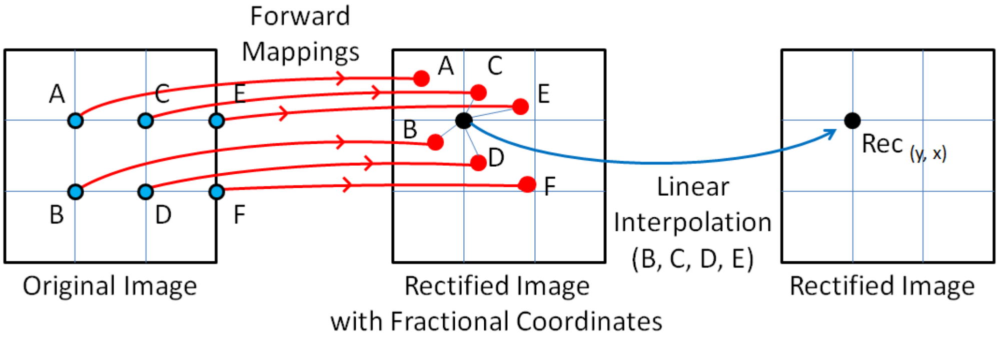

Forward Mappings
A
C
E
B
D
F
Original Image
Rectified Image with Fractional Coordinates
Linear Interpolation (B, C, D, E)
Rec (y, x)
Rectified Image

Figure 5.2: Forward mapping with fractional precision coordinates.

mappings are represented in Fig. 5.1 and Fig. 5.2, respectively. The linear interpolation process for forward mapping is more complex than the linear interpolation process of inverse mapping, since it requires additional computations and an intermediate memory consumption to find the closest target pixels in the rectified image. The look-up-table based rectification hardware architectures presented in [54–56] use inverse mapping due to its simplicity.

The size of the look-up-table depends on the size of the rectified image and the fractional precision. For example, for the rectification of 1024×768 resolution stereo images with 6 bits fractional precision, the rectification map alone requires approximately 6 MB of space in a memory. This amount of data is excessive to fit into the on-chip memory of a mid-range FPGA. Therefore, dumping look-up-tables into an external memory is preferred in the hardware implementations of [54, 55].

In contrast to the selection of the hardware implementations of [54–56], a forward mapping based rectification scheme is selected for the proposed CLUTR algorithm. In CLUTR, fractional precision is ignored. Ignoring fractional precision allows an efficient compression scheme. The negligible distortion in the rectified images originating from this simplification is analyzed in Section 5.7.

The compression scheme is presented in the flow-graph in Fig. 5.3. The proposed compressed rectification algorithm produces four compressed tables. The compression scheme requires eight steps. The details of steps 1-2 can be found in [50] and Chapter 2. The details of steps 3-8 are presented in this section.

In the third step, integer coordinate precision forward mapping is extracted from the fractional precision inverse mapping. The extraction scheme is demonstrated in Fig. 5.4. The example original and rectified pictures have a size of 4×5 pixels. First, inverse mapping is applied to

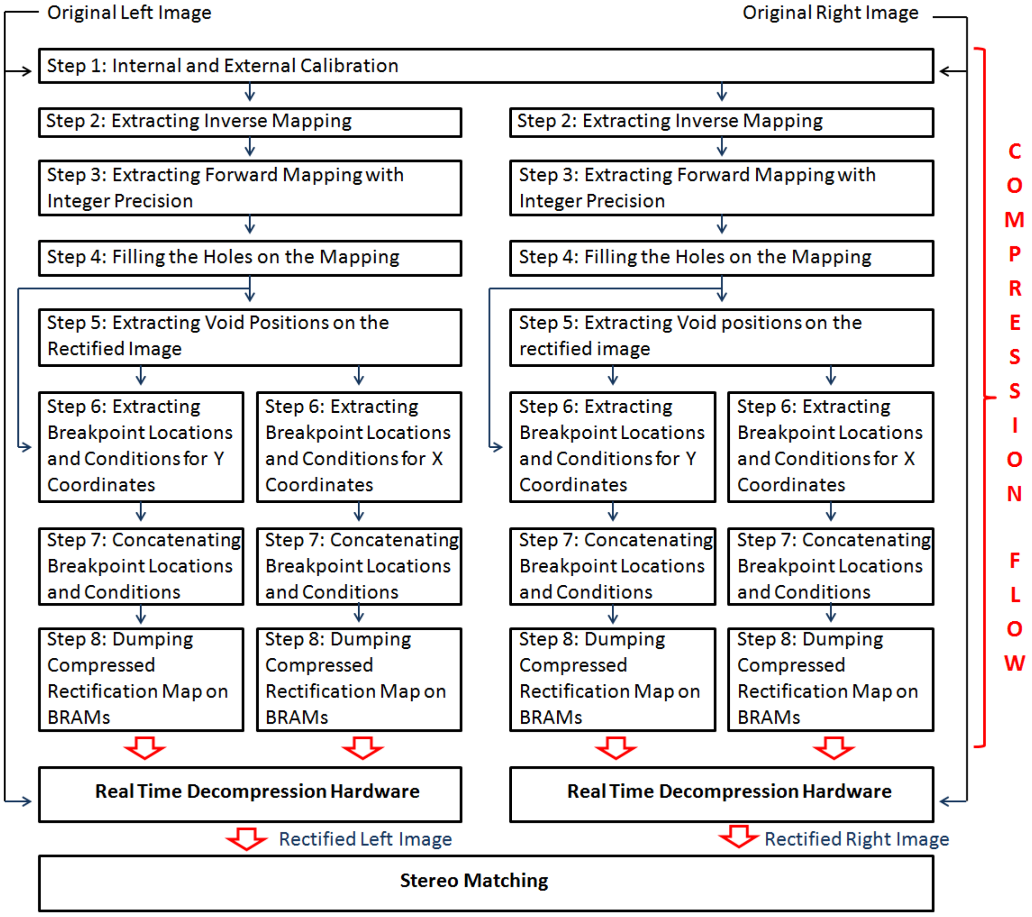

Original Left Image
Original Right Image
Step 1: Internal and External Calibration
Step 2: Extracting Inverse Mapping
Step 3: Extracting Forward Mapping with Integer Precision
Step 4: Filling the Holes on the Mapping
Step 5: Extracting Void Positions on the Rectified Image
Step 6: Extracting Breakpoint Locations and Conditions for Y Coordinates
Step 6: Extracting Breakpoint Locations and Conditions for X Coordinates
Step 7: Concatenating Breakpoint Locations and Conditions
Step 8: Dumping Compressed Rectification Map on BRAMs
Real Time Decompression Hardware
Rectified Left Image
Rectified Right Image
Stereo Matching
COMPRESSION FLOW

Figure 5.3: Flow-chart of the proposed compressed look-up-table based stereo image rectification process.

find the fractional source pixel locations of all pixels in the rectified image. Due to the 3D rotation, some of the pixels in the rectified image cannot be related to their source pixels in the 4×5 original image, as shown in Fig. 5.4a. The nearest integer coordinates of all fractional source coordinates are computed, and they are targeted onto the integer pixel coordinates in the rectified image, as presented in Fig. 5.4b and 5.4c. Thus one-to-one mapping is provided in the third step.

The integer pixel precision forward mapping extracted for the example picture in Fig. 5.4b yields the look-up-tables of X and Y coordinates shown in Fig. 5.5. The pixels that are not targeted to any location are identified with NT. *Ori(2,2)* and *Ori(2,3)* are adjacent pixels, and both of them target “row no 2” of the rectified image; *Ori(2,2)* and *Ori(3,2)* are adjacent pixels and both target “column no 1” of the rectified image. This regular order is more apparent with higher resolution images. According to our experiments with a 1024×768 image, up to 220 times repetition of a single target coordinate is observed in the integer precision forward mapping table of X coordinates.

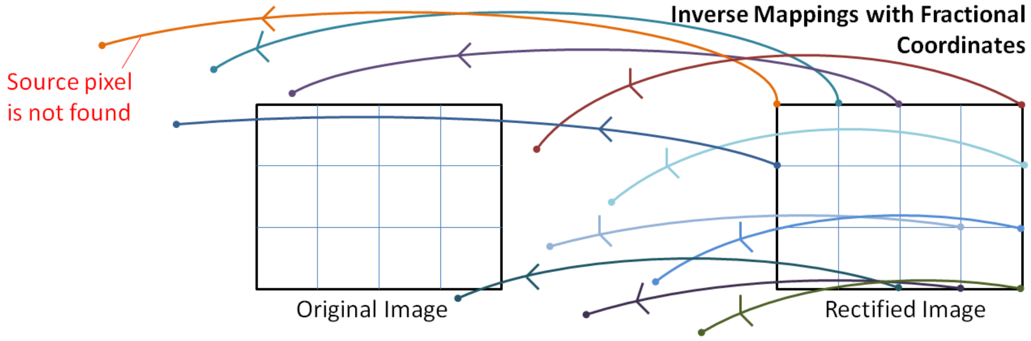
The method governing compressed rectification is similar to the run-length encoding technique [89]. In the proposed coding scheme, instead of coding the run-length of the regular order, the locations where the regular order changes are encoded. These locations are called breakpoints. Moreover, the proposed scheme includes additional specific techniques to compress the integer precision forward mapping efficiently.

The regular order of the Y coordinate mapping is encoded following a row-by-row scheme, and the regular order of the X coordinate mapping is encoded following a column-by-column scheme. The resulting look-up-tables after encoding Fig. 5.5a and Fig. 5.5b are presented in Fig. 5.6a and Fig. 5.6b. In Fig. 5.6a, the elements of the compressed table are represented as (column number, new value in row). In Fig. 5.6b, the elements of the compressed table are represented as (row number, new value in column).

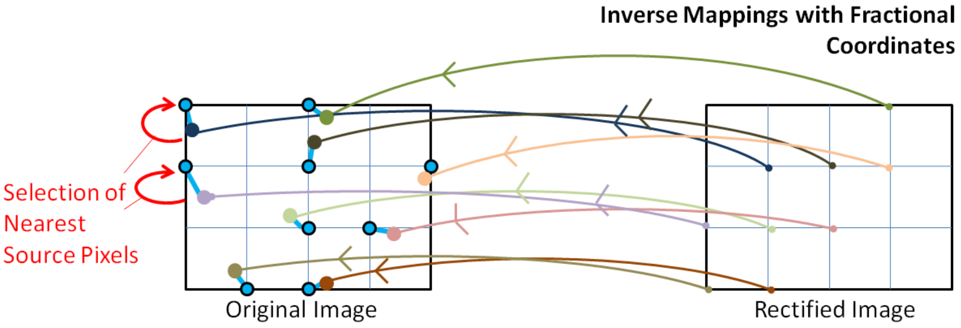
The high number of NT pixels dramatically increases the number of breakpoints. This issue becomes more pronounced for high resolution images. Therefore, the fourth step of the compression algorithm fills the NT pixel locations to keep the regular order. In order to fill the NT pixel locations, the same order is repeated vertically and horizontally for Y and X locations, respectively. After the fourth step, Fig. 5.5 is transformed into Fig. 5.7, and Fig. 5.6 into Fig. 5.8.

After the first two steps, two or more source fractional coordinates can have the same pixel coordinate in the original image as their nearest neighbor, as presented in Fig. 5.9a. However, after step three and four, every integer pixel coordinate of the original image is targeted to a single coordinate in the rectified image. Consequently, some pixels in the rectified image may be void, as presented in Fig. 5.9b. These pixels will remain as void, i.e. black pixels, in the rectified image if they are not filled. The fifth step is applied to fill these voids. As shown in Fig. 5.10, the pixels on the original image which target the pixel coordinates that are located on the row above these voids are marked. Marked pixels are used to fill the voids as source pixels which have double targets (DT). Thus, DT pixels are used to concurrently target two vertically neighboring pixels in the rectified image.

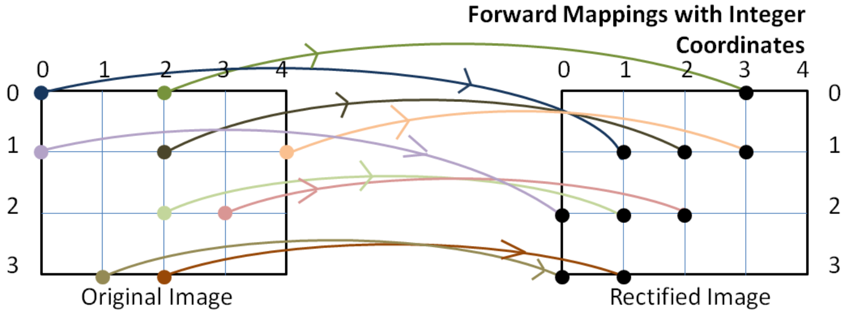
The sixth step of the algorithm extracts the breakpoint locations and analyzes the behavior of the breakpoints. As shown in Fig. 5.8, the difference between the new and previous target locations equals plus or minus one, which can be encoded consuming less data than encoding the exact integer coordinates. An example of coding the behavior of the cells in Fig. 5.8 is presented in Fig. 5.11 as *(location, behavior)*. The initialization coordinates are provided in the first column of the look up table for Y coordinates, and in the first row of the look up table for X coordinates. The next breakpoint values are identified with ±1. Moreover, dummy


Inverse Mappings with Fractional Coordinates
Source pixel is not found
Original Image
Rectified Image

(a)

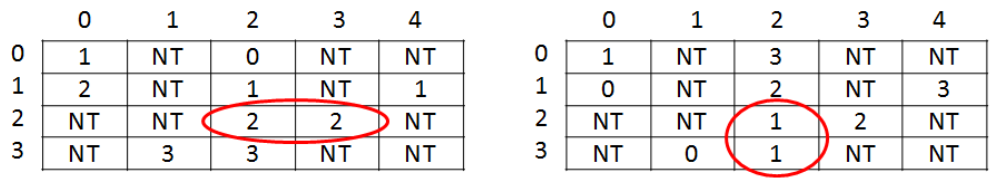

Inverse Mappings with Fractional Coordinates
Selection of Nearest Source Pixels
Original Image
Rectified Image

(b)



(c)

Figure 5.4: Third step of the compression flow (a) Due to the 3D rotation, some of the pixels in the rectified image cannot be related to their source pixels in the original image (b) selection of nearest source pixels from fractional inverse mapping (c) extraction of forward mapping with integer coordinates.



	0	1	2	3	4
0	1	NT	0	NT	NT
1	2	NT	1	NT	1
2	NT	NT	2	2	NT
3	NT	3	3	NT	NT

(a)

	0	1	2	3	4
0	1	NT	3	NT	NT
1	0	NT	2	NT	3
2	NT	NT	1	2	NT
3	NT	0	1	NT	NT

(b)

Figure 5.5: Integer coordinate precision forward mapping look-up-tables after the third step. Regular orders are shown with red ellipses (a) mapping of Y coordinates (b) mapping of X coordinates.

0	0, 1	1, NT	2, 0	3, NT	
1	0, 2	1, NT	2, 1	3, NT	4, 1
2	0, NT	2, 2	4, NT		
3	0, NT	1, 3	3, NT		

(a)

0	1	2	3	4
0, 1	0, NT	0, 3	0, NT	0, NT
1, 0	3, 0	1, 2	2, 2	1, 3
2, NT		2, 1	3, NT	2, NT

(b)

Figure 5.6: Coded regular orders after the third step (a) coded mapping of Y coordinates (b) coded mapping of X coordinates.

	0	1	2	3	4
0	1	1	0	0	0
1	2	2	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3

(a)

	0	1	2	3	4
0	1	0	3	2	3
1	0	0	2	2	3
2	0	0	1	2	3
3	0	0	1	2	3

(b)

Figure 5.7: Look-up-tables after filling the NT pixels using the fourth step (a) mapping of Y coordinates (b) mapping of X coordinates.

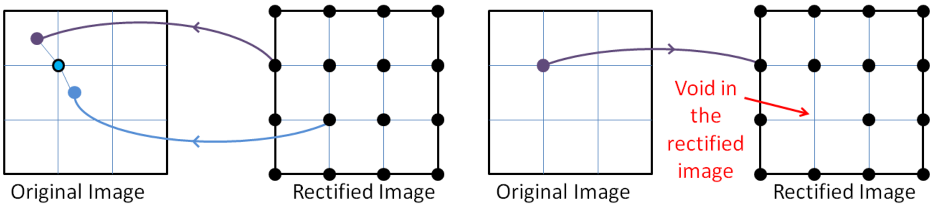
0	0, 1	2, 0
1	0, 2	2, 1
2	0, 2	
3	0, 3	

(a)

0	1	2	3	4
0, 1	0, 0	0, 3	0, 2	0, 3
1, 0		1, 2		
		2, 1		

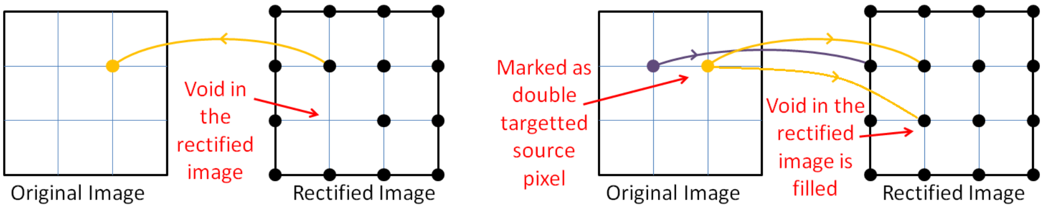
(b)

Figure 5.8: Coded regular orders after filling the NT pixels using the fourth step (a) coded mapping of Y coordinates (b) coded mapping of X coordinates.



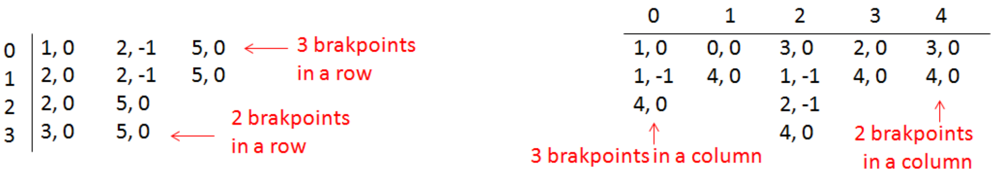
(a) (b)

Figure 5.9: Visualization of the reason for the voids in the rectified image (a) inverse mappings with fractional coordinates (b) forward mapping with integer coordinate.


Original Image
Void in the rectified image
Rectified Image
Marked as double targetted source pixel
Original Image
Void in the rectified image is filled
Rectified Image

(a) (b)

Figure 5.10: Filling the voids in the rectified image in the fifth step (a) finding the source location of a pixel at one row above the void (b) marking the source pixel as double targeted pixel.



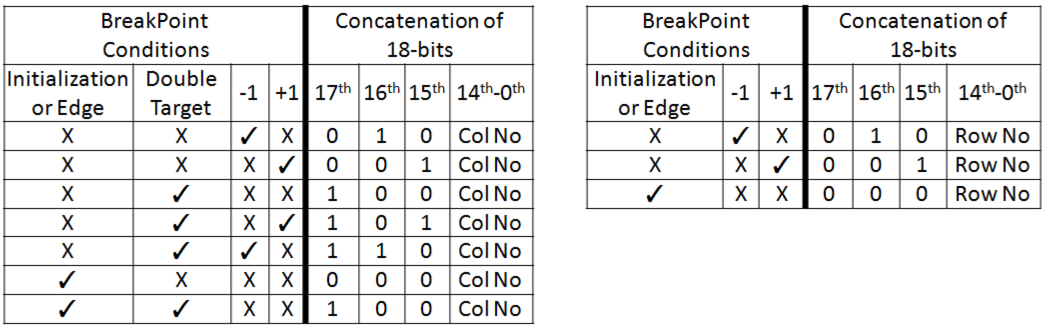
0	1, 0	2, -1	5, 0
1	2, 0	2, -1	5, 0
2	2, 0	5, 0	
3	3, 0	5, 0	

0	1	2	3	4
1, 0	0, 0	3, 0	2, 0	3, 0
1, -1	4, 0	1, -1	4, 0	4, 0
4, 0		2, -1		
		4, 0		

3 brakpoints in a row
2 brakpoints in a row
3 brakpoints in a column
2 brakpoints in a column

(a) (b)

Figure 5.11: Coding the behavior of breakpoints at the sixth step (a) coded mapping of Y coordinates (b) coded mapping of X coordinates.



BreakPoint Conditions				Concatenation of 18-bits			
Initialization or Edge	Double Target	-1	+1	17th	16th	15th	14th-0th
X	X	✓	X	0	1	0	Col No
X	X	X	✓	0	0	1	Col No
X	✓	X	X	1	0	0	Col No
X	✓	X	✓	1	0	1	Col No
X	✓	✓	X	1	1	0	Col No
✓	X	X	X	0	0	0	Col No
✓	✓	X	X	1	0	0	Col No

(a)

BreakPoint Conditions			Concatenation of 18-bits			
Initialization or Edge	-1	+1	17th	16th	15th	14th-0th
X	✓	X	0	1	0	Row No
X	X	✓	0	0	1	Row No
✓	X	X	0	0	0	Row No

(b)

Figure 5.12: Concatenation of the locations and behaviors at the seventh step (a) for the mapping of Y coordinates (b) for the mapping of X coordinates.

breakpoints are inserted at the edges of the image to simplify the hardware implementation. Dummy insertions are represented by (5,0) and (4,0) in Fig. 5.11a and Fig. 5.11b, respectively.

In the seventh step, the locations and behaviors of the breakpoints are concatenated and stored in a data array. Every BRAM in a Virtex-5 FPGA has 1024 addresses and it can be configured to store one array composed of 1024×36bits or two arrays composed of 1024×18bits. The BRAMs of the FPGAs are configured to store 18-bits in each address in the proposed concatenation scheme. As shown in Fig. 5.12, 3-bits are used for coding the behaviors, and the remaining 15-bits encode the locations of the breakpoints. Therefore, the proposed concatenation scheme can be applied to an image that has a resolution lower than 32767×32767 pixels. In Fig. 5.12a, DT and changing the last targeted row by ±1 are independent breakpoint conditions of Y coordinates, which can be applied to source pixels, concurrently or separately. Therefore, the "X" symbol in the -1 and +1 columns of Fig. 5.12 implies keeping the last targeted coordinate.

The number of breakpoints in every row of the Y table and the number of breakpoints in every column of the X table depend on the distortion of the lens, the resolution of the image sensor and the mechanical misalignment. The experimental setup used for the development of CLUTR algorithm consists of 1024×768 resolution cameras. In the experiments, cameras are aligned in parallel configuration without using any sensitive mechanical placement tool. At most 21 breakpoints are observed in any given row of Y tables, and at most 17 breakpoints are observed in any given column of X tables. Data arrays of CLUTR are created for 24 possible breakpoint locations for Y tables and 20 possible breakpoint locations for X tables to support more challenging distortion conditions. Therefore, storing the X and Y tables for the right and left images requires 38 BRAMs which can even be supported by low cost FPGAs. The data arrays that are programmed into the BRAMs are converted into coefficient (COE) files using MATLAB.

In the eighth step, 38 BRAMs are instantiated as single port ROMs. The pre-computed compressed rectification maps are programmed into the BRAMs using the Xilinx ISE 12.4 and COE files.

5.2 Real-Time De-Compression Hardware of CLUTR

The de-compression process is simpler than the off-line compression process in terms of computational complexity. The proposed rectification module can be used as a hardware accelerator taking place between the camera interface hardware and the on-chip memory controller, as shown in Fig. 5.13. The rectification module is used for the left and right cameras separately. The rectification module processes source pixel values as $Ori_{(Yori,\, Xori)}$ and the respective source row and source column coordinates as Y_{Ori} and X_{Ori}. The rectification module computes the target row and target column coordinates as Y_{Rec} and X_{Rec}, and the 1-bit DT signal to identify double targeted locations. $Ori_{(Yori,\, Xori)}$ is delayed for 6 clock cycles and $Rec_{(Yrec,\, Xrec)}$ is given as an output. Thanks to the pipelined structure of the hardware, inputs can be consecutively received and outputs can be consecutively provided.

The top-level block diagram of the rectification module is presented in Fig. 5.14. The rectification module involves $(768\times24\times18)/(1024\times36) = 9$ BRAMs to store the compressed table of Y coordinates and $(1024\times20\times18)/(1024\times36) = 10$ BRAMs to store the compressed table of X coordinates. Half of 1 additional BRAM is used to store the last breakpoint locations and the last target X coordinates of the row which is located above the row currently being processed.

The block diagram of the decompression hardware of Y coordinates is presented in Fig. 5.15. The hardware resets itself every time X_{Ori} is equal to zero which implies that the first pixel of a new row is fetched from the camera. The target Y coordinate of the first incoming pixel in a new row is loaded from the ROM and written to the output register of Y_{Rec}. For every consecutive pixel, X_{Ori} is compared to the coordinate of the next breakpoint which is loaded from the ROM. When a breakpoint is reached, the Y_{Rec} value is changed using a multiplexer depending on the coded behaviors of the breakpoints. Meanwhile, the hardware loads the coordinate of the next breakpoints to compare with the upcoming X_{Ori}.

The block diagram of the decompression hardware of X coordinates is presented in Fig. 5.16. Pixels are supplied by the camera row-by-row, whereas the X coordinates are compressed column-by-column. This situation causes one important difference between the de-compression hardware architectures of the X and Y tables. When the camera provides

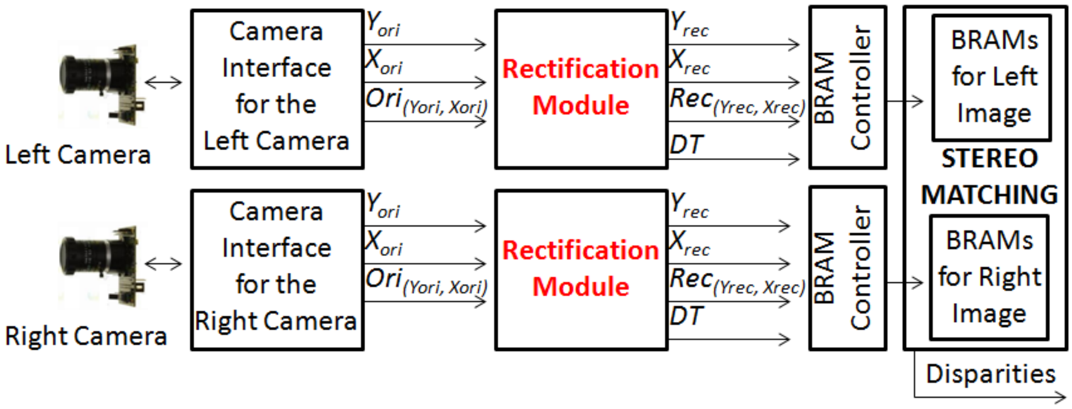


Figure 5.13: Utilization example of the proposed rectification hardware.

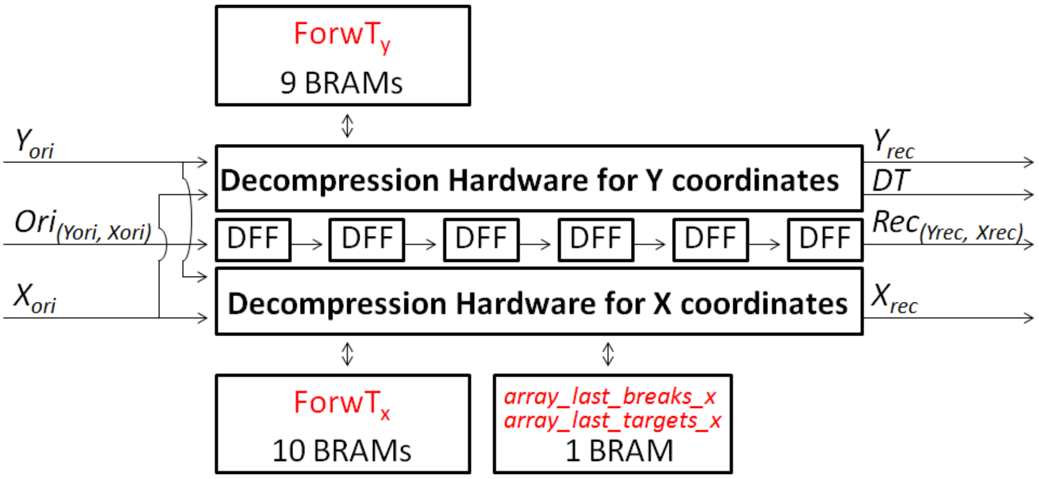


Figure 5.14: Top-level block diagram of the proposed rectification hardware of CLUTR.

pixels of a new row, the de-compression hardware needs to keep record of the previous X_{Rec} coordinates and the last checked breakpoint address in the ROM for the respective column of the previous row. Two 1×1024 size data arrays are needed to store this information. These arrays are named *array_last_break_x* and *array_last_target_x* in Fig. 5.14. These arrays are concatenated for respective column coordinates of the original image, and stored into one half of the 1 BRAM, which is named *X_last_data_BRAM* in Fig. 5.16. The values in *X_last_data_BRAM* are replaced with the new ones when a breakpoint is reached for the respective X_{Ori}. The de-compression hardware of the Y coordinates does not comprise these arrays because Y coordinates are compressed row-by-row. Therefore, the last Y_{Rec} can be directly used for computing the next Y_{Rec} of the next pixel in the same row of the original image. The decompression hardware of X coordinates operates in a similar fashion as the decompression hardware of Y coordinates, with the exception of keeping record of the information about the previous row.

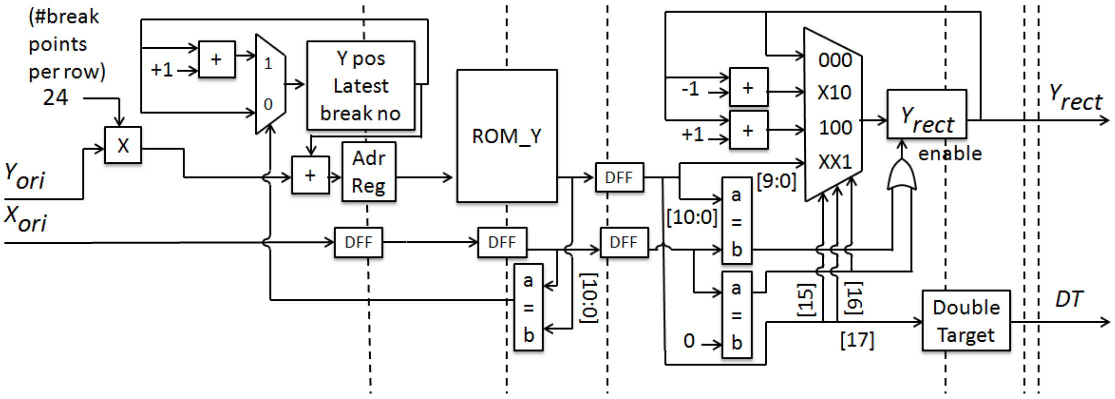


Figure 5.15: Block diagram of the proposed rectification hardware for decompressing the table of Y coordinates. Pipeline stages are presented with dashed lines.

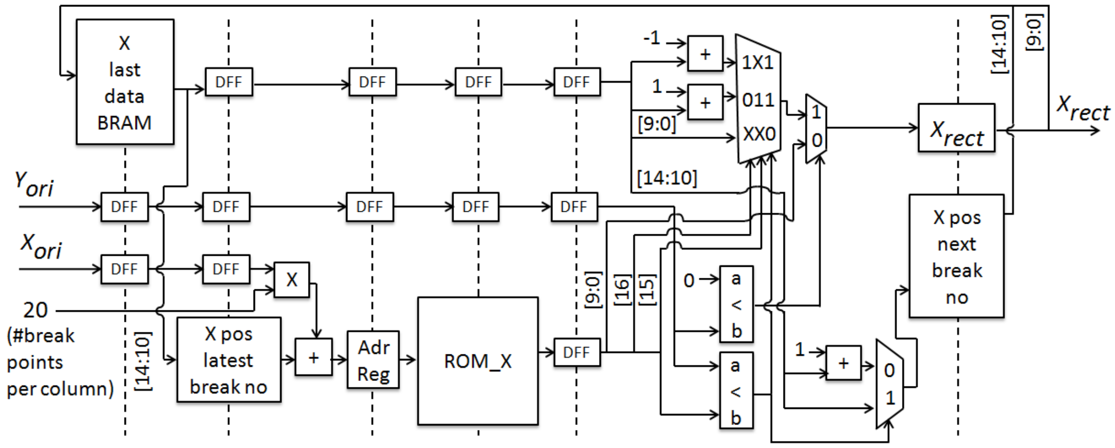


Figure 5.16: Block diagram of the proposed rectification hardware for decompressing the table of X coordinates.

The proposed rectification hardware can be used in any stereo-matching system. The stereo matching process can be started when the required amount of rows is buffered in the BRAMs of the stereo matching hardware. Processed rows in these BRAMs can be overwritten by new rows during the stereo matching process.

The hardware architectures presented in [55, 56] require large pixel buffers due to the inverse mapping scheme. The proposed de-compression does not need large pixel buffers between the camera interface and the rectification modules. In contrast, the hardware requires these pixels buffers for the rectified image. However, typically DE hardware implementations already include BRAMs to buffer the pixels [39, 43, 45, 51]. Therefore, these buffers can be used for the proposed de-compression hardware. Thus, using the proposed rectification hardware on a complete DE system may not need additional large pixel buffers.

5.3 Limitations of the CLUTR

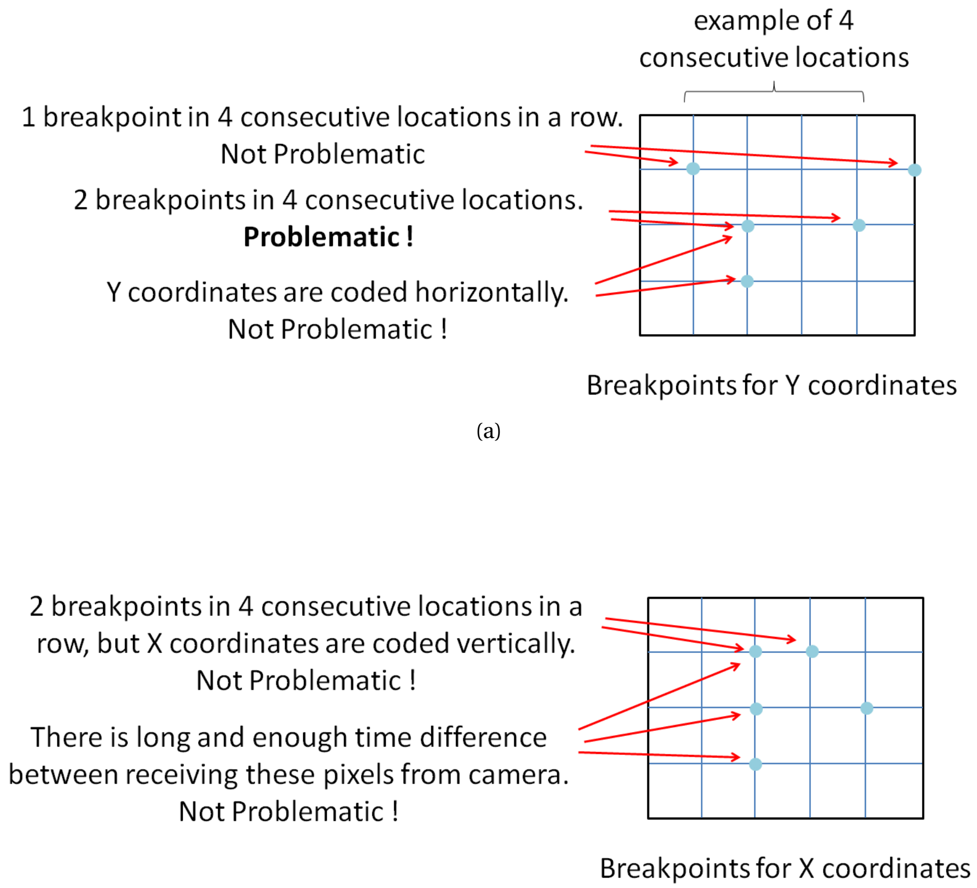
In an ideal case, i.e. where the cameras are perfectly parallel and lenses do not have distortion, two breakpoints are required for every row of the look-up-table for Y coordinates and two breakpoints are required for every column of the look-up-table for X coordinates. One of these two breakpoints is needed to define the initial breakpoint location to target coordinate 0, and the other one is needed to define the final target location as the horizontal or vertical size of the image. In the classical case of a real-time working environment, the mechanical set-up of the stereo-matching system should be carefully designed to be close to an ideal case. Still the main goal of the rectification consists of solving lens distortions and sensitive mechanical misalignments.

According to tests applied to the CLUTR algorithm and hardware, the pixel location difference of two consecutive breakpoints typically reaches more than 15 pixels and the number of breakpoints is smaller than the pre-defined breakpoint capacity of the ROMs of CLUTR. Therefore, CLUTR successfully rectifies the images when the lens distortion and the mechanical misalignments are not excessive. However, unusual conditions bring limitations on the CLUTR hardware.

Two important limitations of the CLUTR hardware must be considered to maintain its suitability to rectify challenging situations. The first limitation relates to the capacity of the ROMs to store a sufficient number of breakpoints. The second limitation relates to the frequency of the breakpoints.

The limitation caused by the number of breakpoints is mainly due to the mechanical misalignment of the cameras. In order to identify the limit of the breakpoint storage capacity of ROMs, two cameras are manually rotated approximately 3 degrees around opposite directions of all rotational axis. This test can be considered as an excessive misalignment of a carefully designed mechanical setup of the stereo-matching system. Using the compression scheme of CLUTR, 43 breakpoints are needed in the look-up-table of X coordinates for one column, and 69 breakpoints are needed in the look-up-table of Y coordinates for one row. The pre-defined breakpoint capacity of CLUTR does not support this condition. Overcoming this first limitation is straightforward to achieve by increasing the size of ROMs to store more breakpoints.

CLUTR supports rectification if the two breakpoints of the Y coordinates have at least 4 pixel position difference. When a breakpoint location is reached, the hardware needs to read the next breakpoint from the ROM. The address computation and reading the next breakpoint from the ROM operations consume 4 clock cycles. The camera continues to send pixels and the camera controller increases X_{Ori} during the address computation and reading breakpoints


example of 4
consecutive locations
1 breakpoint in 4 consecutive locations in a row.
Not Problematic
2 breakpoints in 4 consecutive locations.
Problematic !
Y coordinates are coded horizontally.
Not Problematic !
Breakpoints for Y coordinates
2 breakpoints in 4 consecutive locations in a row, but X coordinates are coded vertically.
Not Problematic !
There is long and enough time difference between receiving these pixels from camera.
Not Problematic !
Breakpoints for X coordinates

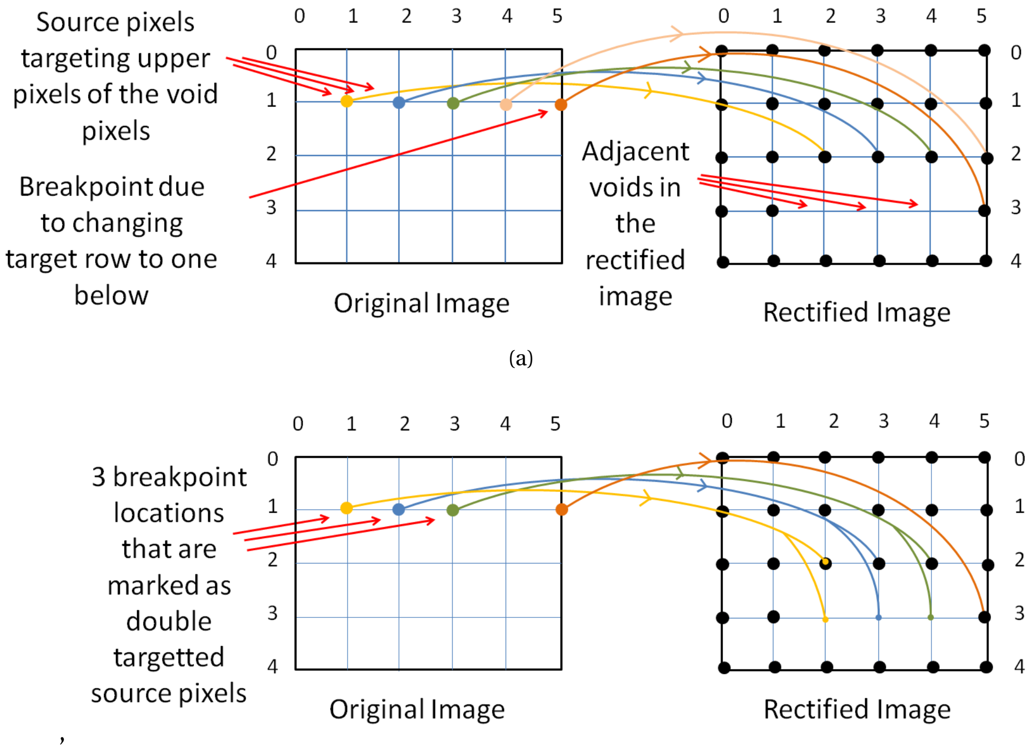
(a)

(b)

Figure 5.17: Visualization of the breakpoint frequency capacity of the X and Y coordinate mappings (a) breakpoints for the mapping of Y coordinates (b) breakpoints for the mapping of X coordinates

from the ROM. Therefore, if there are multiple breakpoints in 4 consecutive pixels, CLUTR is not able to apply a breakpoint condition to those pixels. Hence, the limits of the CLUTR hardware to successfully rectify stereo images is exceeded if breakpoints are frequent, i.e. if two breakpoints of Y coordinates in a row have less than 4 pixel position difference. Since the breakpoints of X coordinates are coded column by column but the camera sends pixels row by row, the time to process consecutive breakpoints in same column and consecutive rows is sufficiently long. Therefore, frequent breakpoints in the same column of the look-up-table of X coordinates do not cause a limitation. The frequency limitation of CLUTR is visually explained in Fig. 5.17.

The main reason for the occurrence of frequent breakpoints is the high number of adjacent void pixels, which is caused by excessive camera misalignment or lens distortion. An example


Source pixels targeting upper pixels of the void pixels
Breakpoint due to changing target row to one below
0 1 2 3 4 5
0 1 2 3 4
Original Image
Adjacent voids in the rectified image
Rectified Image
3 breakpoint locations that are marked as double targetted source pixels
Original Image
Rectified Image

(a)

(b)

Figure 5.18: Visualization of the reason for the frequent breakpoints (a) finding the source locations of three pixels that are targeting one row above of the three consecutive voids (b) four break-points in consecutive five locations.

of one such challenging condition is presented in Fig. 5.18. As shown in Fig. 5.18b, 4 out of 5 consecutive pixels are marked as breakpoints. 3 out of these 4 breakpoints are DT breakpoints which are coded in the look-up-table of Y coordinates, and the other one is located at *Ori(1,5)* which requires changing the target row number from 2 to 3. This challenging example case exceeds the limits of CLUTR, since CLUTR is not able to apply a breakpoint condition if there are multiple breakpoints in 4 consecutive pixels.

Adjacent void pixels may occur not only horizontally but also vertically. Vertically adjacent void pixels may cause void pixels, which cannot be filled by the DT feature of CLUTR. As visualized in Fig. 5.19, DT pixels can fill the voids located one row below the targeted pixel in the rectified image. If there are two voids which are vertically adjacent, the void located below can not be filled by CLUTR since the pixel above is not targeted directly by any source pixel.

Another limitation of CLUTR is related to the usage of ROMs for the hardware implementation which increases the off-line processing duration. Using ROMs is suitable to demonstrate the

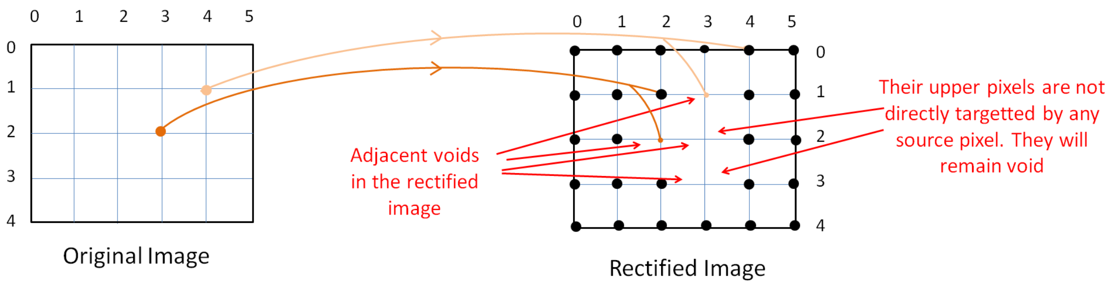

0 1 2 3 4 5
0 1 2 3 4
Original Image
0 1 2 3 4 5
0 1 2 3 4
Rectified Image
Adjacent voids in the rectified image
Their upper pixels are not directly targetted by any source pixel. They will remain void

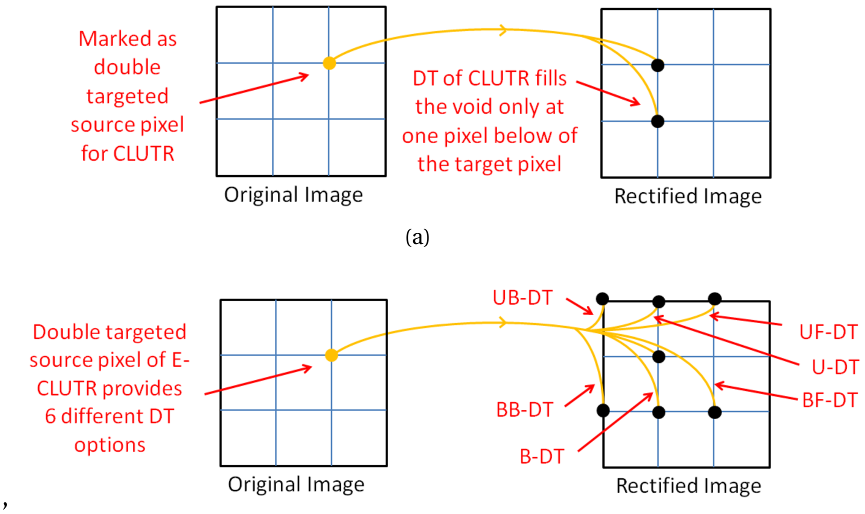
Figure 5.19: Visualization of the reason for the voids which can not be filled by CLUTR.

efficiency of compressed look-up-table based rectification. However, after each adjustment of the camera settings and alignments, creating new compressed tables of the hardware requires re-synthesis and place & route of the implementation. Thus initializing the CLUTR hardware takes a long time.

5.4 Enhanced Compressed Look-Up-Table based Rectification Algorithm

The E-CLUTR algorithm and its hardware implementation are designed to overcome the limitations of CLUTR while maintaining the low complexity decompression scheme. The limitations of CLUTR are mainly solved by improving the design of the decompression hardware. Moreover, algorithmic enhancements are applied to further improve the efficiency of the compression scheme to handle challenging lens distortions and mechanical misalignments.

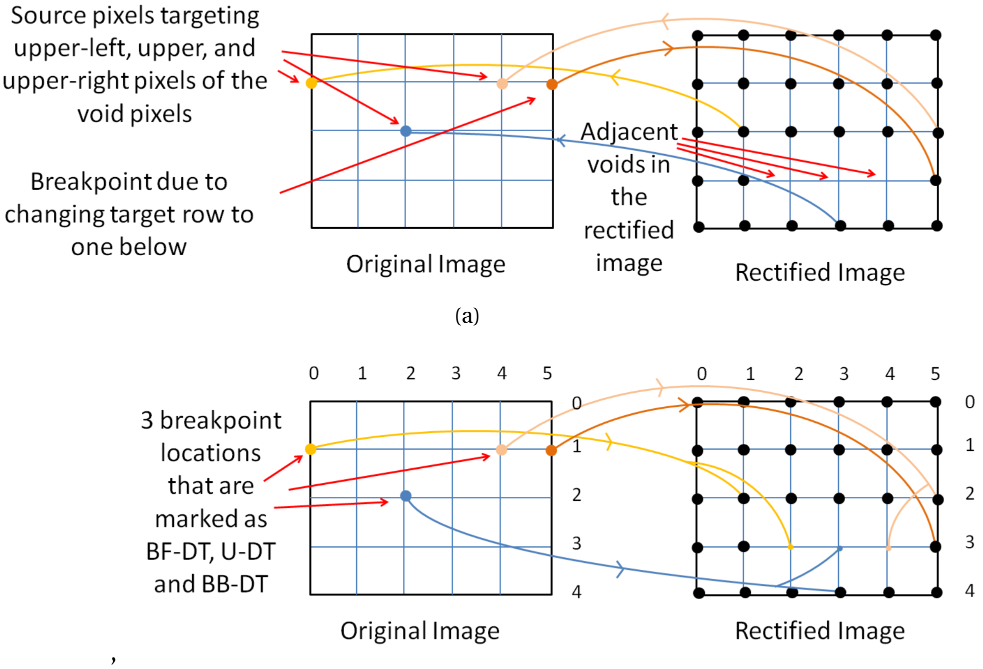
Algorithmic enhancements are explained in this Section. The flow-chart of the compression scheme of E-CLUTR is identical to the flow-chart presented in Fig. 5.3. The steps 5, 6 and 7 that are shown in Fig. 5.3 are enhanced in E-CLUTR. The algorithmic enhancement for the compression scheme is proposed to reduce the frequency of DT breakpoints. In order to decrease the amount of consecutive breakpoints in a row, the condition type of breakpoints for filling the voids are improved. As explained in Section 5.1 in step 5, DT breakpoints are used to fill the voids that are located one pixel below the targeted pixel. To avoid any confusion, the DT condition of CLUTR is renamed as below-DT (B-DT) in E-CLUTR. In addition to B-DT, below-backward-DT (BB-DT), below-forward-DT (BF-DT), upper-DT (U-DT), upper-backward-DT (UB-DT) and upper-forward-DT (UF-DT) breakpoint conditions are defined in E-CLUTR. Using extra DT conditions, the source pixel can be targeted not only to one pixel below the target, but additional options are provided to fill any of 6 possible neighbors of the targeted pixel of the rectified image. These additional options are visualized in Fig. 5.20.


Marked as double targeted source pixel for CLUTR
DT of CLUTR fills the void only at one pixel below of the target pixel
Original Image
Rectified Image
Double targeted source pixel of E-CLUTR provides 6 different DT options
UB-DT
UF-DT
U-DT
BF-DT
BB-DT
B-DT
Original Image
Rectified Image
,

(a)

(b)

Figure 5.20: Filling the voids in the rectified image in the fifth step (a) DT option of CLUTR (b) DT options of E-CLUTR.


Source pixels targeting upper-left, upper, and upper-right pixels of the void pixels
Breakpoint due to changing target row to one below
Adjacent voids in the rectified image
Original Image
Rectified Image
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3 4
3 breakpoint locations that are marked as BF-DT, U-DT and BB-DT
Original Image
Rectified Image
,

(a)

(b)

Figure 5.21: Reducing the frequency of breakpoints using multiple DT options of E-CLUTR (a) finding alternative source locations for void pixels of rectified image (b) reduced frequency of breakpoints for the same row of the look-up-table of Y coordinates.

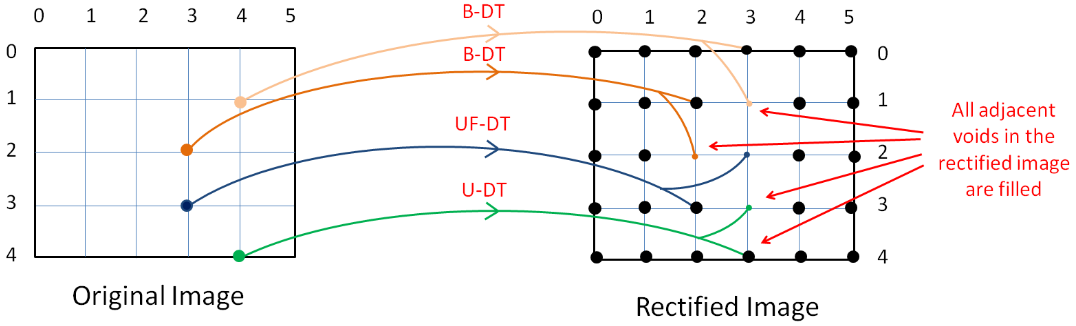


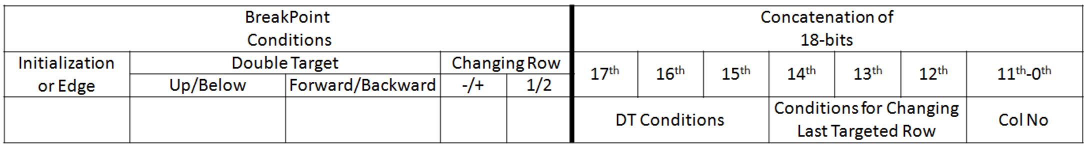
Figure 5.22: Vertically adjacent void pixels can be filled by E-CLUTR using multiple DT options.

As presented in Fig. 5.21, the frequency of breakpoints in the same row is reduced compared to Fig. 5.18, by using BF-DT, BB-DT and U-DT breakpoint conditions. As presented in Fig. 5.19, vertically adjacent void pixels are problematic for CLUTR. However, these voids can be filled by multiple DT options of E-CLUTR as presented in Fig. 5.22.

In step 6, the breakpoints are coded considering the new DT breakpoint conditions. In the challenging example, there should be support of 2 consecutive breakpoints at least in 3 consecutive pixel coordinates, as presented for the breakpoints at *Ori(1,4)* and *Ori(1,5)*. Therefore, the algorithmic enhancement requires the support of at least 2 breakpoints for 3 consecutive locations as an additional constraint. The hardware based enhancement to provide this support is explained in Section 5.5.

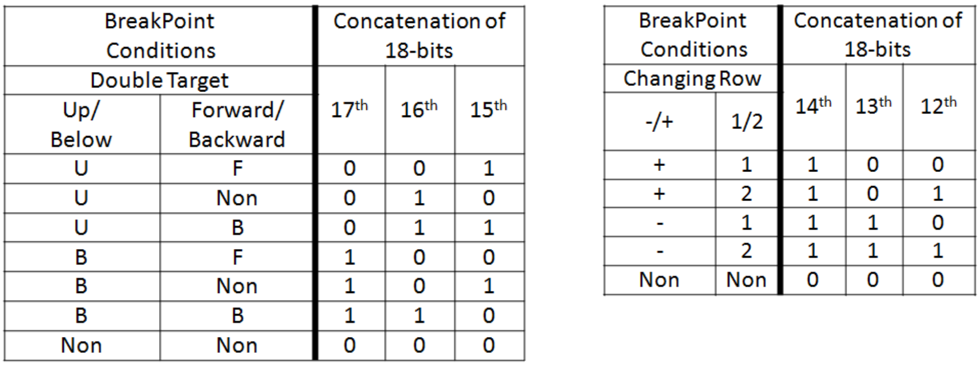
Black pixels occur at the borders of the rectified image. These stem from the 3-D rotation of the original image and the mapping of the rectified image to the original resolution. Consequently, the effective resolution slightly decreases in the rectified image [50]. Due to this fact, changing row and column coordinates stay in the range of ±1. However, this range may not be guaranteed for all possible extreme conditions. Thus, a generic solution should cover all possible extreme situations. In order to cover the cases that create a situation beyond the challenging camera misalignment tests, ±2 row and ±2 column coordinate change options are included as a breakpoint condition in step 6 of E-CLUTR.

The concatenation scheme presented in Fig. 5.12 of CLUTR is modified for E-CLUTR as presented in Fig. 5.23, Fig. 5.24 and Fig. 5.25 using improved breakpoint conditions. The breakpoint conditions for DT conditions and row changing can be applied concurrently or separately to the source pixel. Consequently, $7 \times 5 = 35$ different conditions occur for the concatenation of the conditions for the breakpoints of Y coordinates. A brief concatenation scheme of the Y coordinates is presented in Fig. 5.23. The concatenation scheme for DT codes and row changing are separately presented in Fig. 5.24a and Fig. 5.24b. The concatenation scheme of the X coordinates is presented in Fig. 5.25.



BreakPoint Conditions					Concatenation of 18-bits						
Initialization or Edge	Double Target		Changing Row		17th	16th	15th	14th	13th	12th	11th-0th
	Up/Below	Forward/Backward	-/+	1/2							
					DT Conditions			Conditions for Changing Last Targeted Row			Col No

Figure 5.23: Brief representation for the concatenation of the locations and behaviors it the seventh step of E-CLUTR for the mapping of Y coordinates.



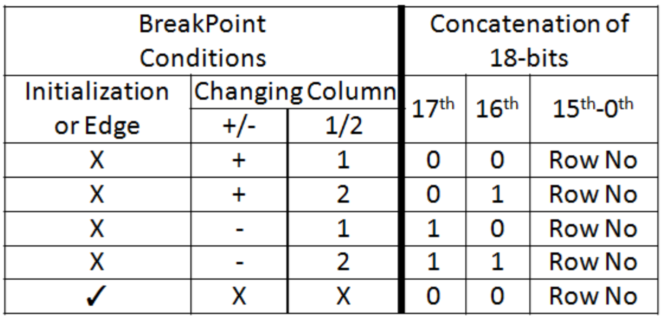
BreakPoint Conditions		Concatenation of 18-bits		
Double Target				
Up/ Below	Forward/ Backward	17th	16th	15th
U	F	0	0	1
U	Non	0	1	0
U	B	0	1	1
B	F	1	0	0
B	Non	1	0	1
B	B	1	1	0
Non	Non	0	0	0

(a)

BreakPoint Conditions		Concatenation of 18-bits		
Changing Row				
-/+	1/2	14th	13th	12th
+	1	1	0	0
+	2	1	0	1
-	1	1	1	0
-	2	1	1	1
Non	Non	0	0	0

(b)

Figure 5.24: Concatenation of the locations and behaviors at the seventh step for E-CLUTR for the mapping of Y coordinates (a) Concatenation scheme for DT options (b) Concatenation scheme for the breakpoint conditions for changing the last targeted row.



BreakPoint Conditions			Concatenation of 18-bits		
Initialization or Edge	Changing Column		17th	16th	15th-0th
	+/-	1/2			
X	+	1	0	0	Row No
X	+	2	0	1	Row No
X	-	1	1	0	Row No
X	-	2	1	1	Row No
✓	X	X	0	0	Row No

Figure 5.25: Concatenation of the locations and behaviors at the seventh step of E-CLUTR for the mapping of X coordinates.

5.5 Real-Time De-Compression Hardware of E-CLUTR

The top-level block diagram of the E-CLUTR module is presented in Fig. 5.26. Data arrays of E-CLUTR are created for 80 possible breakpoint locations for Y tables and 50 possible breakpoint locations for X tables in order to support very challenging distortion conditions.

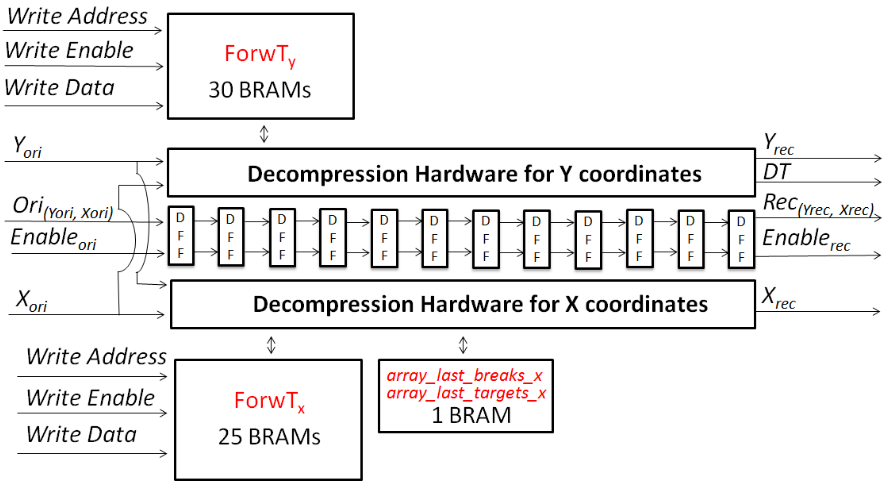


Figure 5.26: Top-level block diagram of the proposed rectification hardware of E-CLUTR.

The rectification module involves (768×80×18)/(1024×36) = 30 BRAMs to store the compressed table of Y coordinates and (1024×50×18)/(1024×36) = 25 BRAMs to store the compressed table of X coordinates. As presented in Figure 26, the ROMs are converted to RAM to ease the initialization of the look-up-tables of E-CLUTR after changing the camera settings. The decompression hardware for the X coordinates is presented in Fig. 5.27. The decompression hardware for Y coordinates is presented in Fig. 5.28.

As presented in Fig. 5.27, the decompression hardware of E-CLUTR pertaining to X coordinates is similar to the hardware used in CLUTR. The multiplexing stage is adapted to provide the ±2 target pixel column change feature for the X breakpoints.

The decompression hardware of E-CLUTR pertaining to Y coordinates is redesigned to support frequent breakpoints and the six different DT options. As presented in Fig. 5.28, the E-CLUTR hardware reads the first six breakpoints from the RAM as soon as the camera starts to send a new row. The first breakpoint is used to initialize the target row and the next five breakpoints are buffered in a local cache. Whenever a new breakpoint location is reached, the next breakpoint location is read from the RAM and the cache shifts the existing upcoming breakpoint locations. Using this local cache of the breakpoints, the original pixel coordinates can be compared to the pixel locations in the cache. Therefore, the breakpoint conditions can be applied to all passing pixels even if the breakpoints are frequent.

The multiplexing stage for the computation of the next target row is improved to provide the ±2 target pixel row change feature. Moreover, the hardware sends the 3-bit DT condition to the BRAM controller together with the pipelined source pixel and its target locations Y_{Rec} and X_{Rec}, synchronously.

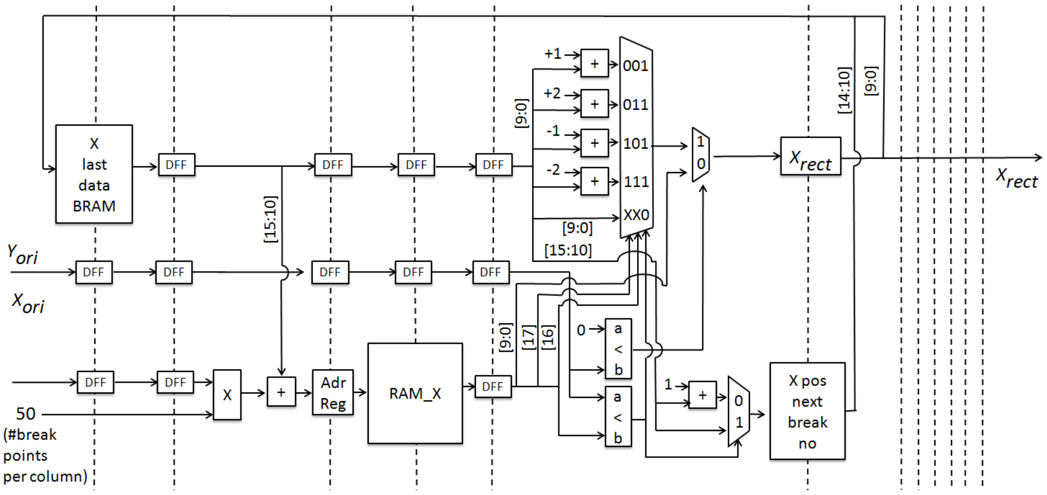


Figure 5.27: Block diagram of the proposed rectification hardware for decompressing the table of X coordinates.

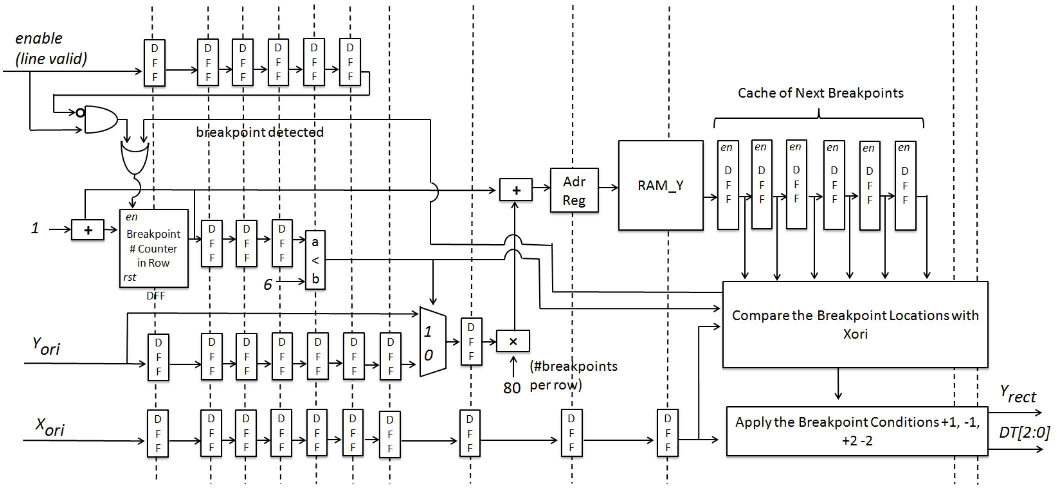

enable (line valid)
DFF
breakpoint detected
1
+
en Breakpoint # Counter in Row rst
DFF
a < b
6
Y_{ori}
X_{ori}
1
0
×
80
(#breakpoints per row)
Adr Reg
RAM_Y
Cache of Next Breakpoints
en
Compare the Breakpoint Locations with Xori
Apply the Breakpoint Conditions +1, -1, +2 -2
Y_{rect}
DT[2:0]

Figure 5.28: Block diagram of the proposed rectification hardware for decompressing the table of Y coordinates. Pipeline stages are presented with dashed lines.

The BRAM controller that is shown in Fig. 5.13 writes the pipelined source pixels to the decompressed target row and target column coordinates. E-CLUTR hardware is verified by merging it with the binocular DE hardware presented in Chapter 3, which buffers pixels of rows in its own separate BRAMs. Y_{Rec} is used to select and enable the BRAM to write target pixel. X_{Rec} is used to determine the write address of the enabled BRAM of the DE hardware. If a DT condition exists, the same source pixel is concurrently written to two BRAMs by enabling two BRAMs that buffer two consecutive rows. If the DT condition is pointing into forward or backward positions, the address port of the BRAM that is targeted by the DT condition receives the computed target BRAM address ±1.

5.6 Real-Time Hardware of Caltech Rectification

The Caltech rectification algorithm [50] which does not benefit from look-up-tables is implemented in hardware to evidence the efficiency of CLUTR and E-CLUTR implementations much fairly. The camera rotation and lens distortion parameters are computed in a PC using Caltech calibration software and transferred to the proposed rectification hardware through UART communication.

The top-level block diagram of the proposed Caltech rectification hardware for a single camera is presented in Fig. 5.29. The two-camera disparity estimation hardware includes two of these modules working in parallel and the three camera disparity estimation module includes three of the modules working in parallel. The top-level rectification module receives the YCbCr pixel data from the camera interface. The Calibration Parameters module receives the camera calibration parameters that are provided from the PC and stores them in DFF array. The control module counts the rectified pixel row and column locations, synchronizes the internal operations of the sub-modules of rectification hardware, generates addresses to read and write the on-chip BRAMs. The Rectification Index and Apply Distortion hardware modules proceed the mathematical operations to compute the fractional source pixel locations of the processed pixels of the rectified image. The Interpolation hardware module performs linear interpolation between the four neighbors of the pointed fractional source pixel location and provides the output rectified pixel YCbCr value.

The hardware is a fully parametrized and pipelined implementation of the Caltech rectification algorithm. The hardware provides one rectified pixel location, Y_{Rec} and X_{Rec}, and its pixel value, $Rec_{(Yrec,\, Xrec)}$, for each clock cycle. A fixed-point implementation is used to provide fractional precision. The Caltech rectification algorithm is very sensitive to the fractional precision of the variables. The parameters that define the sizes of the signals and their fixed-point precision are determined to provide high quality results. Still, some parameters for the fixed-point precision can be increased to guarantee better results, especially in cases where the multiple-camera setup is very defective. However, increasing the fixed-point precision significantly increases the hardware resource consumption. The bit-size for the main input parameters of the hardware are provided in Table 5.1. The equations related with these parameters can be consulted in [50].

The Caltech rectification hardware buffers the incoming camera data in the 64 dual-port BRAMS. The Rectification Index module solves image rotation equations. The Apply Distortion module includes computationally intensive operations for solving lens distortion. The Apply Distortion module involves high precision and sensitive division and multiplication operations.

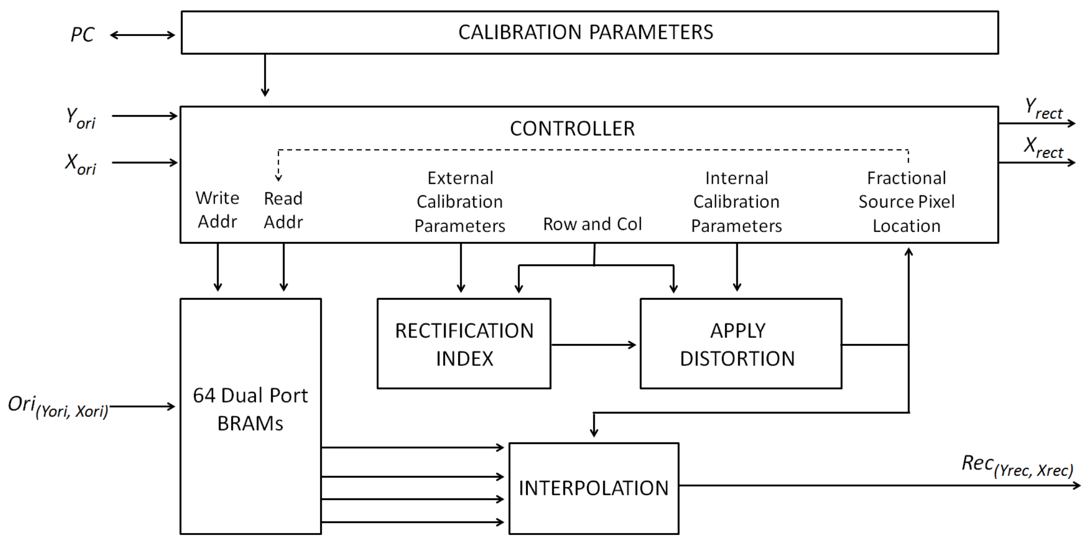


Figure 5.29: Block diagram of the proposed hardware implementation for Caltech rectification algorithm.

Therefore, it takes part as the main hardware resource consuming module of the rectification hardware. After the fractional source pixel coordinate of the processed rectified pixel is calculated, the four neighboring pixels of the computed source pixel location are read from the dual-port BRAMS in parallel. These four pixels are linearly interpolated and the output is provided as a rectified YCbCr image pixel. The Interpolation module utilizes the fractional precision computed by the Apply Distortion module to perform linear interpolation between four pixels. The interpolation of the pixel values for the Y, Cb and Cr channels are processed in parallel.

Table 5.1: Bit-size Parameters of the Caltech rectification hardware

Input Parameter	**# Total Bits**	**# Bits of Fractional Precision**
P_new	28	26
rays	28	26
cx/cy	16	6
k	12	10
f	16	2
intermediate signals	-	16

5.7 Implementation Results

The proposed rectification hardware architectures of CLUTR and E-CLUTR are implemented using Verilog HDL, and verified using Modelsim 10.1d. The Verilog RTL models are mapped to a Virtex 5 XCUVP-110T FPGA comprising 69k Look-Up-Tables (LUT), 69k DFFs and 148 BRAMs. One rectification module of CLUTR consumes 0.32% of the LUTs, 0.28% of the DFF resources and 14% of the BRAM resources of the Virtex-5 FPGA. One rectification module of E-CLUTR consumes 0.63% of the LUTs, 0.51% of the DFF resources and 38% of the BRAM resources of the Virtex-5 FPGA. The proposed E-CLUTR hardware operates at 212 MHz after place & route. Therefore, it can process up to 269 fps at a 1024×768 XGA video resolution. In addition, the proposed rectification hardware of CLUTR and E-CLUTR are merged with the DE hardware presented in Chapter 3. The merged DE systems are also verified using Modelsim 10.1d.

The proposed rectification hardware of CLUTR and E-CLUTR do not need the support of external memory if the cameras are synchronized. The cameras can be perfectly synchronized by driving the cameras with same clock source and using one common I^2C module for the initialization of the cameras as explained in Chapter 9.

The proposed compression and decompression algorithms are evaluated using the pictures taken by the stereo camera system presented in Chapter 3. Two different tests are applied to prove the quality of the CLUTR and E-CLUTR implementations. In the first test, cameras are roughly alligned since in the classical case of a real-time working environment, the mechanical set-up of the stereo-matching system should be carefully designed to be close to an ideal case. As a second test, two cameras are manually rotated approximately 3 degrees around opposite directions of all rotational axis. The second test can be considered as an excessive misalignment of a carefully designed mechanical setup of the stereo-matching system.

The 1024×768 size original left and right pictures that are taken for the first test are shown in Fig. 5.30. The original images are rectified using the full precision MATLAB implementation of the Caltech rectification algorithm [50] and the proposed CLUTR algorithm. The rectification results of the CLUTR are presented in Fig. 5.31. The breakpoint locations pertaining to the X and Y coordinates of the left image are presented in Fig. 5.32. The result of the E-CLUTR algorithm is not provided separately for the first test since CLUTR and E-CLUTR provide identical visual and numerical results when the cameras are not extremely misaligned.

The PSNR between the rectification results of the CLUTR and Caltech rectification algorithms are evaluated in Table 5.2. The PSNR of the left image is 42.67 dB, and the PSNR of the right image is 41.87 dB. Generally, a PSNR larger than 30 dB is considered acceptable to the human

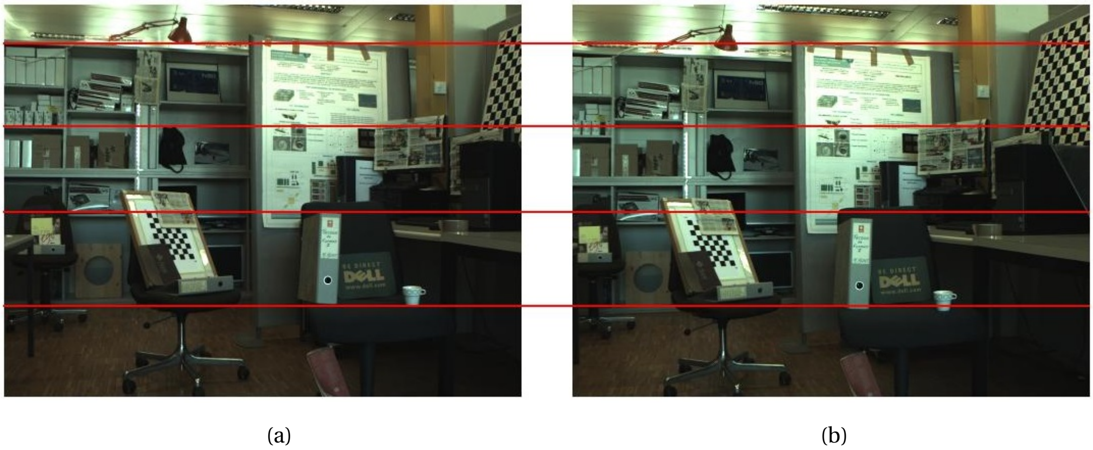

(a) (b)

Figure 5.30: Visual results of the first test using roughly aligned cameras: Original images have still distortions as observed near the lamp, bag, folder and cup; horizontal epipolar lines are displayed in red near the edge of these objects (a) left image (b) right image.

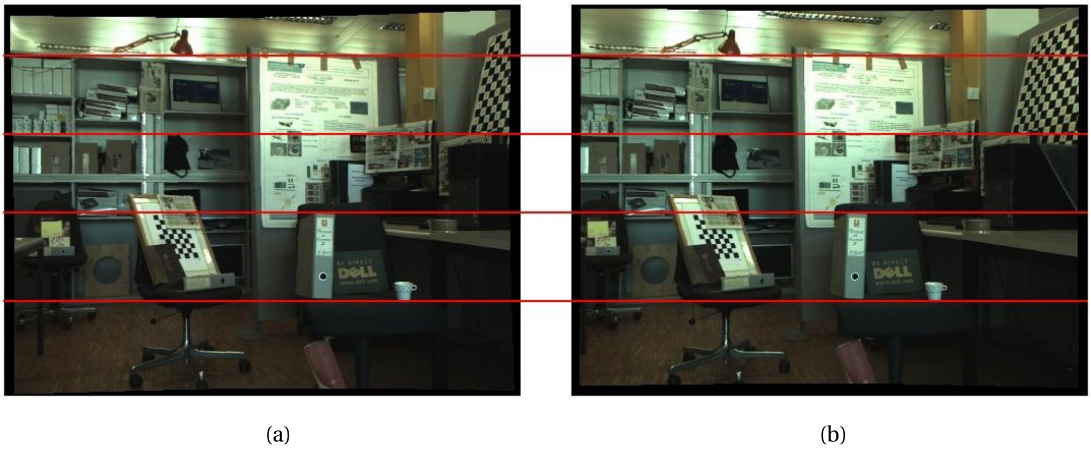

(a) (b)

Figure 5.31: Visual results of the first test using roughly aligned cameras: The proposed CLUTR algorithm corrects distortions (a) left image (b) right image.

eye. Therefore, CLUTR provides very high quality rectification results. The PSNR between the original images and the rectification results of Caltech are also provided in Table 5.2 for comparison.

The performance loss of CLUTR is also evaluated for different DE algorithms. The DE algorithms that are implemented on real-time hardware are used for the evaluation [39, 43, 45, 51]. The DE results obtained using the images that are rectified by Caltech rectification algorithm are assumed as the respective ground truths of the DE algorithms. These ground truths are compared with the DE results of the respective algorithms using the images that are rectified by CLUTR. The PSNR results are provided in Table 5.3. 120 and 255 are applied as a disparity range (DR) and the respective DRs are used as peak signals for PSNR calculations. CLUTR provides 32.87 dB and 27.94 dB PSNR for the AWDE algorithm [51] and Greisen et al. [45], for a 255 DR, respectively. Therefore, the proposed CLUTR algorithm has an insignificant effect on the quality of the DE, and it can be used in different DE systems. The PSNR between the DE results using the original images and the DE results using the rectified images of Caltech are also provided in Table 5.3 for the comparison.

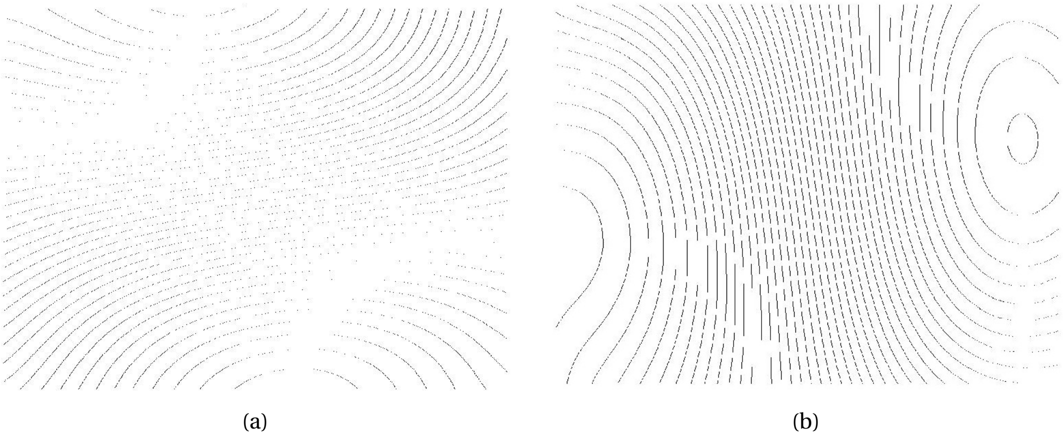

(a) (b)

Figure 5.32: Visual results of the first test using roughly aligned cameras: Breakpoint locations obtained by CLUTR for the left image (a) breakpoints of the targeted Y coordinates; coded row-by-row. (b) breakpoints of the targeted X coordinates; coded column-by-column.

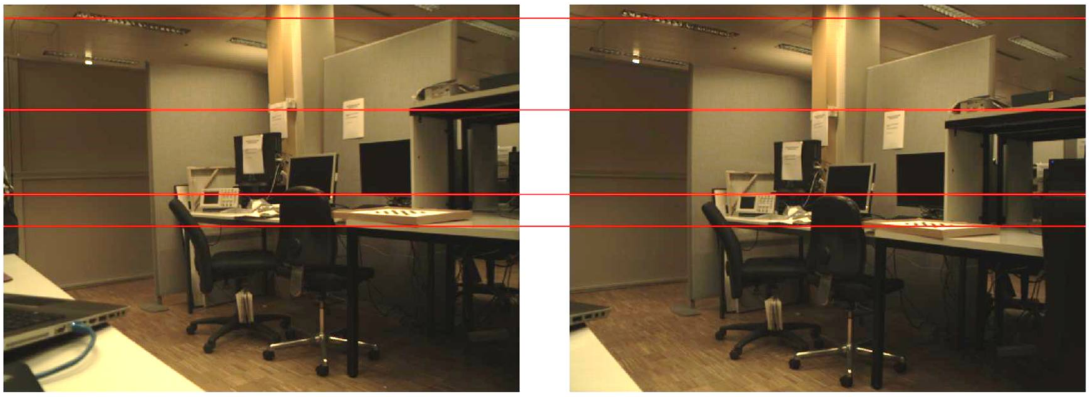
Table 5.2: Numerical results of the first test using roughly aligned cameras: PSNR (*dB*) with the rectified images produced by Caltech rectification algorithm

	Comparison with Rectified Left Image [50]	**Comparison with Rectified Right Image [50]**
Original Image	15.69	16.08
Proposed (CLUTR)	42.67	41.87

Table 5.3: Numerical results of the first test using roughly aligned cameras: PSNR (*dB*) Comparison of the Disparity Estimation Results Using Different Disparity Estimation Algorithms

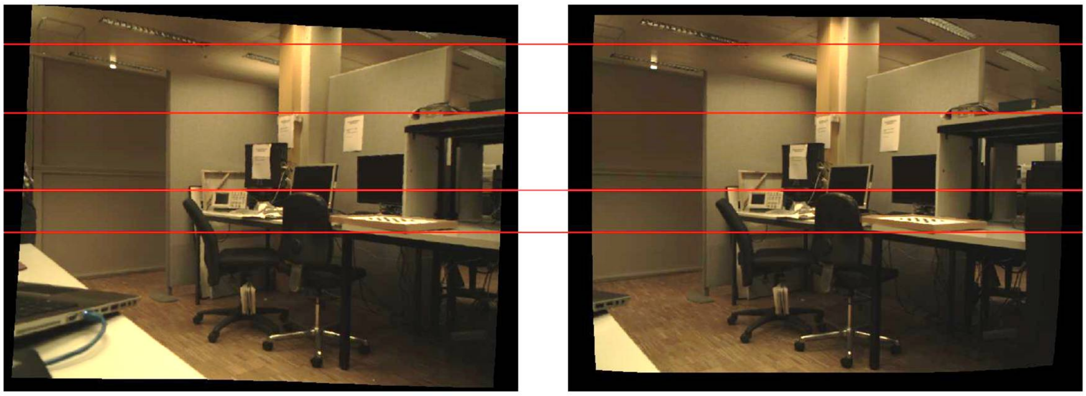
	DE using [50] vs. DE using CLUTR		**DE using [50] vs. DE using Original Images**	
	DR=120	DR=255	DR=120	DR=255
Mini-Census [39]	29.98	32.49	12.01	11.70
Georgulas [43]	28.72	32.31	12.39	13.44
AWDE [51]	29.95	32.87	12.93	13.65
Greisen [45]	26.30	27.94	11.20	10.79

The 1024×768 size original left and right pictures that are taken for the second test is shown in Fig. 5.33. The pictures in Fig. 5.33 are taken under a camera misalignment condition that exceeds the limits of CLUTR implementation. The original images in Fig. 5.33 are rectified using the Caltech rectification algorithm and the proposed E-CLUTR algorithm. The rectification results of the E-CLUTR are presented in Fig. 5.34. The extreme rotation of the rectified image can be visually observed in Fig. 5.34. The breakpoint locations pertaining to the X and Y coordinates of the left image are presented in Fig. 5.35. The result of CLUTR algorithm is not provided separately for the second test since the excessive camera misalignment of this test case exceeds the capacity of the CLUTR hardware implementation.



(a) (b)

Figure 5.33: Visual results of the second test using excessively misaligned cameras: Original images have excessive distortions as observed on the lines (a) left image (b) right image.



(a) (b)

Figure 5.34: Visual results of the second test using excessively misaligned cameras: E-CLUTR corrects distortions as observed on the lines (a) left image (b) right image.

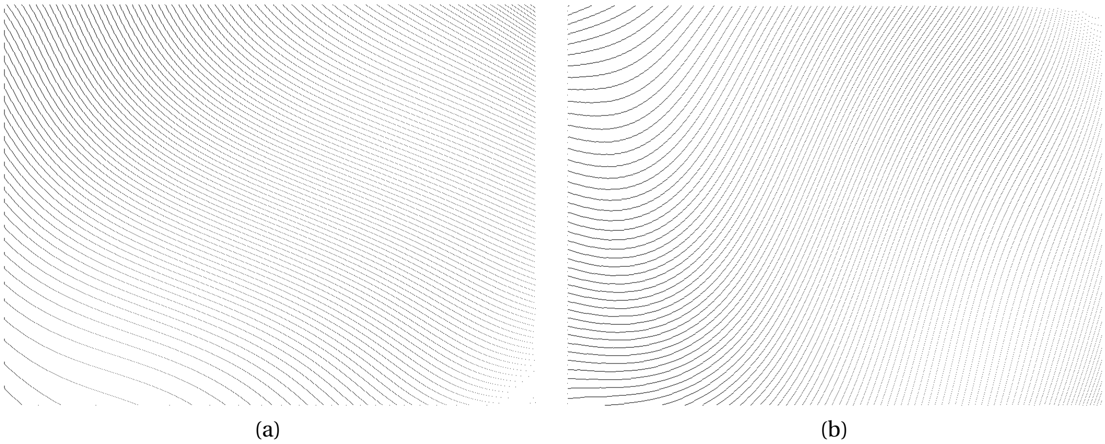


Figure 5.35: Visual results of the second test using excessively misaligned cameras: Breakpoint locations of the left image (a) breakpoints of the targeted Y coordinates; coded row-by-row. (b) breakpoints of the targeted X coordinates; coded column-by-column.

The PSNR between the rectification results of E-CLUTR and Caltech rectification algorithm are evaluated in Table 5.4. The PSNR of the left image is 43.10 dB, and the PSNR of the right image is 42.02 dB. Therefore, E-CLUTR provides very high quality rectification results even though cameras are excessively misaligned. The PSNR between the original images and the rectification results of Caltech are also provided in Table 5.4 for comparison.

Table 5.4: Numerical results of the second test using excessively misaligned cameras: PSNR (*dB*) with the rectified images produced by Caltech rectification algorithm

	Comparison with Rectified Left Image [50]	**Comparison with Rectified Right Image [50]**
Original Image	17.99	19.34
Proposed (E-CLUTR)	43.10	42.02

The hardware implementation of the CLUTR and E-CLUTR are compared with the stereo image rectification hardware implementations in Table 5.5. The hardware architecture of [53] requires a significant amount of hardware resources to support complex operations for solving the lens distortion models. Hardware architectures of look-up-table based implementations [54] and [55] require an external memory. Combining the CLUTR with a BRAM controller or E-CLUTR with a BRAM controller consumes less LUT and DFF resources than [53–55]. The DFF and LUT consumption of [56] is not available (NA). Nevertheless, the capacity of CLUTR and ECLUTR to fit the look-up-tables into the on-chip memory of the Virtex-5 FPGA is approximately six times and two times more efficient than [56], respectively, as a benefit of their efficient compression scheme. Moreover, hardware resource consumption results of the presented high precision hardware of the Caltech rectification algorithm is provided in Table 5.5 for comparison. The hardware implementations of CLUTR and E-CLUTR require much less hardware resource than the hardware implementation of Caltech rectification while providing almost identical rectification results with very high PSNR results.

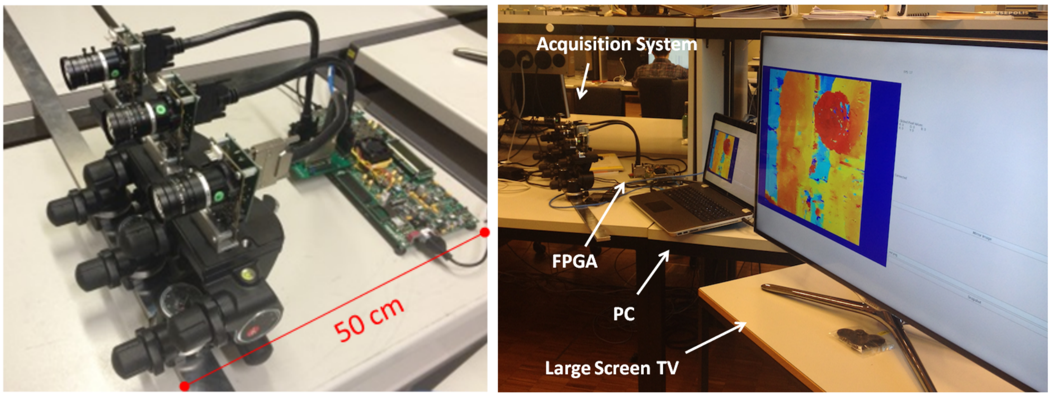
The hardware resource consumption of E-CLUTR is higher than CLUTR. However, if the cameras are extremely misaligned, the limitations of CLUTR can be exceeded. In these extreme conditions, E-CLUTR still supports rectification. Whereas, using CLUTR hardware can be more profitable if the stereo cameras are carefully aligned.

Table 5.5: Hardware Resource Comparison of the Rectification Hardware Implementations

	Device	**Resolution**	**LUT**	**DFF**	**On-Chip Memory (KB)**	**External Memory**
[53]	Virtex-4	752x480	3418	5932	0	✓
[54]	Virtex-E	640x512	2459	2075	99	✓
[55]	Spartan-2	640x480	≈2396	≈2396	16	✓
[56]	Virtex-5	1280x720	NA	NA	1300	X
Caltech Hardware	Virtex-5	1024x768	24384	25346	192	X
CLUTR	Virtex-5	1024x768	227	197	90	X
2x(CLUTR+ BRAM Contr.)	Virtex-5	1024x768	784	427	176	X
E-CLUTR	Virtex-5	1024x768	434	350	252	X
2x(E-CLUTR+ BRAM Contr.)	Virtex-5	1024x768	2278	956	500	X

6 Embedded System for Depth Map Estimation

In this chapter, the embedded system of the depth estimation is explained. The efficient communication and data exchange scheme of the system peripherals are explained. In following sections, first, the general overview of the complete system is presented. Secondly, the overview of the video processing hardware cores and circuits is presented. Thirdly, embedded system peripherals other than video processing cores are explained. Lastly, the embedded software operated in softcore processor is explained.



(a) (b)

Figure 6.1: Demonstration of the system set-up. (a) close-up to the three cameras and the FPGA board (b) full demonstration system also including a PC and large-screen TV.

6.1 The Overview of the Depth Map Estimation System

The system setup of the depth map estimation is shown in Fig. 6.1. The complete system in Fig. 6.1 consists of three horizontally aligned XGA (1024×768) resolution cameras, a Xilinx VC707 Evaluation Board, a PC and a TV. A Virtex-7 FPGA included in the VC707 board is used to prototype the developed rectification hardware, disparity estimation hardware, rendering hardware and the system peripherals that interface with cameras, a PC and video processing cores.

In the following sections, the full embedded system obtained after merging the separate hardware modules is explained. Fig. 6.2 shows a block-diagram of the full depth estimation system. All real-time video processing computations are implemented in hardware. The PC is used as a display, to control the system, to obtain camera calibration parameters and to developed software based real-time applications. The resulting disparity images are transferred to the PC using 1Gb raw Ethernet. A standard 2D TV is connected to the PC using HDMI to offer a better display. A MicroBlaze softcore (CPU) is used to initialize cameras through an I^2C Serial Interface, to control Ethernet and to communicate with the PC. A DDR3 memory is only used for Ethernet buffering. The presented block diagram in Fig. 6.2 includes three cameras. The embedded system for two cameras is almost identical to the three camera version. The minor differences of the embedded system between two and three camera systems are explained in this section. The same system architecture can be also implemented using Virtex-5 FPGA. Virtex-7 FPGA is selected to prototype the proposed complete system since it includes more hardware resources than Virtex-5, which helps placing and routing the hardware components more efficiently.

6.2 Video Processing Hardware Cores and Circuits

As presented in Fig. 6.2, three Aptina image sensors are connected to the VC707 board via an FMC1 connector, piggyback boards and VHDCI cables. The IO pins of the Virtex-7 FPGA operates at 1.8V, and the Aptina Cameras work at 3.3V. Piggyback boards are designed to shift to the different voltage level, to increase the strength of the signals, and to route the signals to VHDCI connectors. Each of the three cameras has its own piggyback board (Camera-VHDCI) and one piggyback board is connected to the FMC1 connector (FMC-VHDCI). The Camera-VHDCI boards that are connected to image sensors are identical.

The IOs of the image sensors are not powerful enough to successfully drive 40cm VHDCI cables for single ended *pixel data*, *hsync*, *vsync* and *pixel clock* signals. Therefore, reflection, voltage drop and noise problems are observed on the signals, and thus on the images. In order to solve this noise issue, a buffer chip, NXP 74LVT16244BDGG, is placed on the Camera-VHDCI board to increase the strength of the camera signals before transferring them to the VHDCI cable.

The FMC-VHDCI board includes three VHDCI connectors and level-shifter chips. Three TI PCA9306DCUR and four TI SN74AVC16T245DGGR chips are used to shift the signal voltage levels between 1.8V and 3.3V. The bi-directional PCA9306DCUR chips are used for shifting voltage levels of I^2C signals. Three of the four SN74AVC16T245DGGR chips are used for

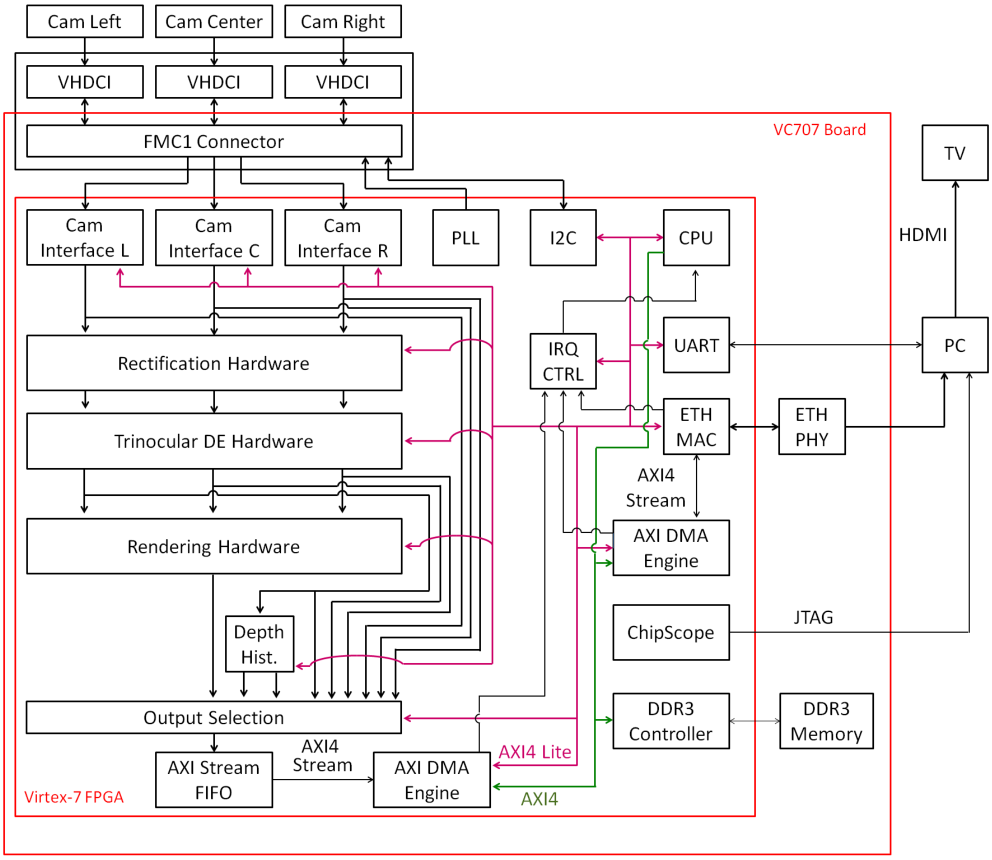


Figure 6.2: Block diagram of the full system setup.

shifting voltage levels of *pixel data*, *hsync*, *vsync*, *pixel clock* signals which are directed from three Cameras to FPGA. One of the SN74AVC16T245DGGR chips is shared between three cameras for shifting the voltage level of the *camera clock* and *camera reset* signals which are directed from FPGA to Cameras.

The trinocular depth estimation system presented in Fig. 6.2 includes three Camera Interface modules. The binocular depth estimation system includes two Camera Interface modules. The Camera Interface module receives 10-bit Bayer format *pixel data*, *hsync*, *vsync* and *pixel clock* signals, counts the row and column numbers and applies demosaicing to convert Bayer images into YCbCr and RGB formats. The Camera Interface module outputs pixels in 24-bit YCbCr and 24-bit RGB pixel formats, and their respective column and row numbers. Moreover, the Camera Interface module computes average Red, Green and Blue values in the images to be used in auto color-gain and auto shutter-width corrections.

The nearest neighbor interpolation method is used as a demosaicing algorithm. The nearest neighbor interpolation requires to access at most 8 adjacent pixels of the processed pixel.

Therefore, in addition to the processed row, its upper and below rows should be buffered. The Camera Interface module utilizes 4 dual-port BRAMs to buffer 4 rows of the Bayer image. Buffered pixels are read back from the 4 BRAMs in parallel. The read pixels are transferred to the 4×3 pixels size systolic flip-flop array. The processed pixel and its 8 neighbors are selected from the flip-flop array. Afterwards, the nearest neighbor interpolation operations are processed to interpolate not-existing red, green or blue values of the processed pixel. The YCbCr values of the pixels are computed upon the completion of Bayer to RGB conversion. The RGB to YCbCr convertion operations that are given in eq. 6.1 are implemented with 10-bit precision fixed point precision. All the operations of the camera interface module are implemented in pipeline. Camera Interface module receives stream-input and provides stream-output. Therefore, it does not utilize external memory for the demosaicing operations. The throughput of the camera interface module is one pixel per clock cycle. During the demosaicing process, the camera interface module computes average red, green and blue values in the image. These average values are computed for every image and transferred to the softcore processor to be used for auto color-gain and shutter-width corrections of the image sensors. The RGB outputs of the camera interface module are transfered to the Output Selector module to be displayed as the original images of the cameras. YCbCr outputs of the Camera Interface module are transferred to the rectification hardware to be used in depth estimation computations.

$$\begin{aligned} Y &= (0.257 \times R) + (0.504 \times G) + (0.098 \times B) + 16 \\ Cb &= -(0.148 \times R) - (0.291 \times G) + (0.439 \times B) + 128 \\ Cr &= (0.439 \times R) - (0.368 \times G) - (0.071 \times B) + 128 \end{aligned} \tag{6.1}$$

The trinocular system includes three rectification hardware modules to solve the lens distortions and camera misalignments of three cameras. The binocular depth estimation system includes two Rectification hardware modules. In Chapter 5, CLUTR, E-CLUTR and Caltech Rectification hardware modules are presented. The system can operate with any of these rectification hardware modules since their inputs and outputs are compatible. The rectification hardware receives the YCbCr value of an original image pixel and its respective row and column coordinates from the Camera Interface module as an input, and provides rectified pixel row and column coordinates and the YCbCr value as an output. The DT signal is provided as an output of the CLUTR and E-CLUTR hardware. The proposed implementation of Caltech rectification algorithm always transfers ’0’ as a DT value since it utilizes backward rectification scheme and does not require the double-target process. The output pixels of the rectification hardware are buffered into the BRAMs of the disparity estimation module.

The compressed look-up-table information of CLUTR hardware is dumped into the ROMs during the synthesis of the complete hardware. The calibration parameters can be dynamically transferred to the E-CLUTR and Caltech rectification hardware through the GUI for every system initialization. The external and internal calibration parameters are automatically computed by the system GUI using the multiple snapshots of the checkerboard. The calibration process through the system GUI is explained in Section 6.4.

The BRAM controller of the disparity estimation hardware module receives the YCbCr rectified pixel value, its respective column and row numbers and the DT value. The BRAM controller maps the pixels to the input pixel buffer BRAMs of the DE hardware. The number of the BRAMs for buffering the input pixels of the DE hardware module depends on the rectification hardware and the usage of the external memory. The BRAM controller is blind to the type of the rectification method or the existence of external memory for the video processing. The number of BRAMs for buffering the input pixels of the DE hardware module, *# DEinputBRAMs*, is provided as a parameter of the BRAM controller. When the BRAM controller receives a row number between 0 and 767, it performs a mod operation using the *# DEinputBRAMs* to compute the target BRAM. The column number is used as a write address of the BRAMs. If DT is activated by the rectification hardware, the BRAM controller computes the DT location, and maps the pixel to the BRAM in parallel as second target location. When the first 31 rows of the all rectified images are mapped to the BRAMs, the BRAM Controller generates a *ready* signal to inform the Controller of the DE module to start the DE process. Afterwards, the *ready* signal is generated upon the transfer of every seven rows to inform the Controller of the DE module to continue the DE process of the next 7 rows. In addition, when the last pixels of the rectified image is mapped to the BRAMs, the BRAM controller informs the Controller module of the DE module with a *transmitted* signal to prepare the DE module for finalizing the DE process of an image and to place it in wait-state for the next *ready* signal. Extracting and sending *ready* and *transmitted* signals to the Controller of DE module is important especially when the forward mapping based compressed rectification hardware module is used, since in this case the first and last transmitted pixels may not be the pixels at the corners of the rectified image.

The DE hardware module computes 8-bit disparity results in real-time. The hardware presented in Chapter 4 is used for trinocular DE, and the hardware presented in Chapter 3 is used for binocular DE. The DE hardware is configurable through the GUI to adapt to targetted software application. The hardware is configurable by the user who can select the maximum disparity range (a maximum of 255 is allowed), the disparity-start and disparity-end values to provide best DE quality at a certain distance interval, the strengths of the Census and BW-SAD metrics in the HC computation, and the resolution of depth images (maximum 1024×768 is allowed) to allow faster frame rates at lower resolution video. The output 8-bit disparity values and the 24-bit YCbCr pixels of cameras are buffered and synchronized using 16 BRAMs for each channel. YCbCr pixels are converted back to the RGB pixels. Therefore the DE hardware transfers 24-bit YCbCr and RGB pixels of the cameras and 8-bit disparity values to the Output Selection and rendering modules. The synchronization of pixel color values is required for rendering hardware. In addition, output synchronization is important to transmit RGB+D pixel format to the PC to implement wide range of software applications.

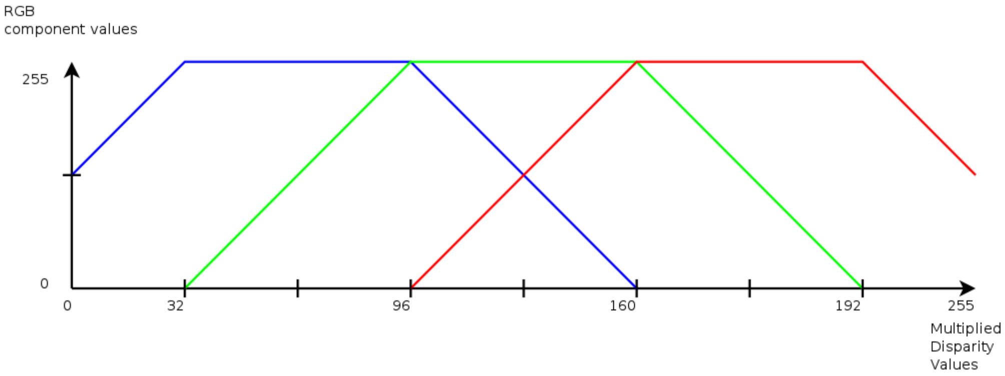


Figure 6.3: RGB component values in function of the multiplied disparity values, used for the color representation.

The Depth Histogram module presented in Fig. 6.2 visually improves the display of the depth results but does not refine the depth measurement. The system can be configured for a disparity range between 0 and 255. Therefore, the resulting disparity maps are represented as 8-bit gray scale images by default. However, if the searched disparity range is configured between 0 and 128, visually recognizing the distance difference is not easy on monitor due to a poor distributed histogram, since many pixels will seem close to the black color. Therefore, if the searched disparity range is configured between 0 and 128, the microblaze informs the Depth Histogram module to multiply the disparity values by 2 to obtain a visually better histogram. Since the disparity values are all smaller than 128, their multiplied results will still be 8-bits. The GUI is already informed about this multiplication since the disparity range can be configured through the GUI. Therefore, it considers this multiplication operation while computing the distances and realizing software applications. However, still all the pixels are in gray values, and thus multiplying by two may not always provide a visually better histogram. In order to improve the visual depth perception, RGB channels of the display should be used. The format of the display can be selected as either RGB or Gray from the GUI. The implemented Depth Histogram hardware maps the multiplied or original disparity values in the range 0 to 255 to an RGB color gradient. In the resulting image, hotter colors i.e. colors close to the red spectrum, represent objects that have a high disparity and cold colors i.e. colors close to the blue spectrum represent objects that have a low disparity. Fig. 6.3 shows the function that determines the RGB values of the different disparity values. The Depth Histogram hardware includes multiplexers, multipliers and adders to realize conversion to the RGB values. Figure 6.4 shows both a gray scale and a color version of a disparity map. The differences between the orange, yellow and green portions of the image are much easier to identify on the color version than the gray scale one.

The rendering hardware synthesizes free view images for any horizontally aligned arbitrary camera positioned between the leftmost and rightmost physical cameras. The rendering

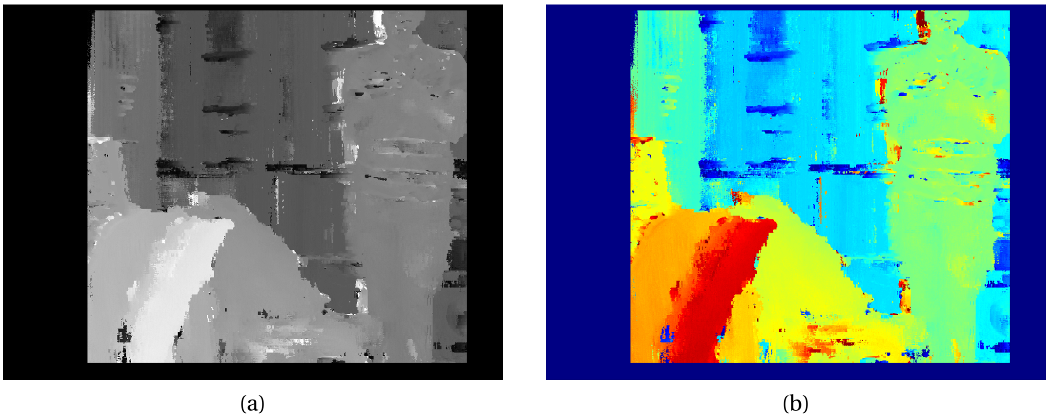

(a) (b)

Figure 6.4: Snapshot of the DE result of binocular disparity estimation system. The hardware is able to switch between to representations: (a) Gray scale disparity map (brighter color is closer, darker color is further), (b) Color mapped disparity map (hotter color is closer, colder color is further)

hardware utilizes the disparity values, RGB pixel values of the cameras and the arbitrary camera location q to synthesize free view images. The arbitrary camera location q is dynamically sent from the GUI. The details of the rendering hardware are presented in Chapter 7.1.

6.3 Peripherals of the Embedded System

The embedded system is designed to transfer image data and disparity maps to the PC. In this subsection, the system peripherals of the complete embedded system other than the video processing cores presented in sub-section 6.2 are explained. The data transfer and communication schemes of the system peripherals are detailed.

The AXI bus is the main connection path between the peripherals of the complete system-on-chip. Two AXI buses are used in the system to speed up the processing of the softcore by reducing the bus based bandwidth limitations. These two buses are named as AXI4 and AXI4 Lite as presented in Fig. 6.2. The microblaze is the only master of AXI4 Lite. The microblaze utilizes the AXI4 Lite to program the AXI DMA Engines, to communicate with the Interrupt Controller (IRQ CTRL) module, to program the Cameras through the I^{2}C module, to receive time informations from the Timer module, and to communicate with the UART module, and thus with the PC. In addition, the microblaze processor communicates with all video processing cores through software-accessible registers which are connected to the AXI4 Lite bus. Therefore, the AXI4 Lite is mainly used for control purposes of the Microblaze processor. The AXI4 is the most busy bus of the system. There are three masters on the AXI4 bus. These are the microblaze, the AXI DMA Engine of the AXI Stream FIFO and the AXI DMA Engine of the Ethernet Controller (ETH Mac). The AXI4 bus is used by these three masters to access the DDR3 memory through the DDR3 controller module. The microblaze utilizes the AXI4 bus, since the program and data memory of the processor is stored in the DDR3 controller. The AXI DMA Engine of the AXI Stream FIFO writes the output images of the selected video processing core to the DDR3. The AXI DMA Engine of the ETH MAC reads images from the DDR3 to transfer them to the PC using a raw-ethernet communication protocol.

The ChipScope module allows the user to observe internal signals in the FPGA system. This module is used to verify the correct functionality of the video processing cores. It is a useful module to debug hardware in real-time, without interrupting the software and the video processing cores.

The internal PLL of the FPGA is used to generate clock signals of the processors and all the peripherals. Additionally, the source clocks of the cameras are generated by the PLL module. Three image sensors utilizes the same source clock generated by the PLL module. The image sensors also have an oscillator on their board, so they can be used without sending a clock signal from the FPGA. However, three osciallators in three cameras generates clocks with different frequencies, which prevents camera synchronization. Therefore, a common camera clock signal is sent to image sensors from the FPGA. The image sensors can operate with a clock frequency upto 48MHz. In the proposed setup, the cameras are used at a 48MHz clock frequency to obtain their fastest available performance. The perfect synchronization of the three cameras also requires syncornization of I^2C communication. The perfect synchronization of I^2C communication is obtained by sharing a single I^2C module between the cameras. The I^2C multiplexing hardware module is explained Chapter 4.

The UART interface is used for transferring console outputs from the FPGA to the host PC and to transmit control data from the host PC to the system. The UART interface is used for debugging and providing information about the displaying the status of the system. The PC transfers all the commands and system parameters to the video processing cores through the UART Interface of the GUI. This interface is also used to manually adjust the camera settings from the PC, such as the color-gain and shutter-width parameters of the image sensors.

The Timer module is used for realizing auto color-gain and auto shutter-width corrections for the cameras. The auto color-gain and shutter-width corrections are iterative operations. While the image sensor parameters are modified through the I^2C, after a given time, the response of the cameras to new parameters should be observed before applying new parameters. The time between each iteration is controlled by the Timer module of the system.

The Interrupt Controller is a hub for all interrupt request lines. The interrupt controller waits for an interrupt signal and relays it to the microblaze. The two AXI DMA Engines, the UART module, Timer module, the UART module and Ethernet MAC IP can send interrupts to the processor. Then, the Microblaze contacts the interrupt controller via the AXI4 Lite interface to find out which of the peripherals requested the interrupt, and applies the required task of the interrupt.

The Ethernet MAC IP is used to transfer the outputs of video processing cores to the PC. Although there are other communication protocol options, Raw Gigabit Ethernet is chosen for the implementation. Since the system is implemented on a VC707 Virtex-7 Evaluation board, the choice of high-speed interfaces was limited to USB 2.0, Gigabit Ethernet, PCI-e and HDMI. HDMI is not suitable for RGB+D data transfer and to develop applications for a PC. PCI-e would allow to reach a very fast data transfers, however the mobility of the system would be drastically reduced since PCI-e requires direct attaching of FPGA to the desctop PC with very short cables or boards. USB 2.0 offers very limited transmission speed to transfer raw video without compression. Gigabit Ethernet allows high transmission speed even with long cables, and only requires the proper setup of the PC's Ethernet connection, which is suitable for the targeted high-performance real-time depth map estimation system. The Ethernet IP of the Xilinx supports the TCP, UDP and IP protocols. TCP and UDP are computationally heavy protocols for the low frequency microblaze processor due to the error detection and recovery scheme of these protocols. TCP and UDP are mainly useful if there are multiple devices in the long distance network and if high-frequency processors are used at both sides of the communication. Since the Microblaze and AXI bus do not operate at very high clock frequencies of the PC, TCP and UDP communications do not provide high bandwidth using FPGA. Raw-IP is selected for real-time verification of the system since the image transmission is over a point-to-point connection from the FPGA to the PC through a relatively short ethernet cable. Therefore, the chances of transmission errors are much lower than in a regular network environment, and thus the recovery scheme on the network is not targeted. Although Raw-Ethernet IP does not support error recovery, it supports checksum-based error detection. The checksum scheme is utilized in the proposed system to detect the packets with transmission errors. The PC controls checksum of all the packets, and the images that include erroneous packet are skipped from the processing of applications and not displayed. According to the tests, on the average, less than 2 images are dropped in a transmitted flow of 1000 images due to error detection with checksum verification.

On the average, a 435 Mb/s bandwidth is obtained using the Raw Ethernet IP. Although the DE system can process 55 fps at a 768×1024 XGA video resolution and this speed performance is functionally verified in real-time using Chipscope, the bandwidth of the implemented Raw-Ethernet interface allows to transfer maximum 18 fps. The 55 fps performance of the hardware will be fully exploitable using USB3 in future versions.

In order to send the image data and disparity maps via Ethernet packets, the images have to be written into a DDR3 memory. Xilinx provides an Ethernet MAC module for controlling DDR3 memory. The DDR3 memory should be connected to the microblaze processor through the AXI Bus in order to be used as an instruction and data memory. Therefore, an interface between the video processing hardware modules and the AXI Bus is created. All the video processing hardware modules of the system provide their video output in the same format consisting of the pixel value, and its respective enable signal, row number and column number. Since the video processing cores send data in the same format, the same interface can be used for all kinds of transmissions.

The interface between the video processing cores and the DDR3 Memory Controller is presented in Fig. 6.5. The interface is composed of an AXI4 Stream FIFO, an AXI DMA engine and an Output Selector. Two image-sized ping-pong buffer scheme is used in the DDR3 memory. While an image in the memory is sent to the PC via Ethernet, the next image is written into the other buffer. The Output Selector selects the target image format from the outputs of the video processing cores according to user command provided through the GUI, and fills the AXI4 Stream FIFO to transfer those images. The DMA Engine is always programmed before the first pixel of an image arrives in the AXI4 Stream FIFO. As soon as the first pixels arrive, DMA Engine starts copying pixels into the DDR3 memory until it encounters the last pixel which is marked with a *last bit* flag. As soon as the last pixel is transferred, the DMA Engine triggers an interrupt to inform the Microblaze that writing the image to the DDR3 is completed. This interrupt causes Microblaze to switch the destination address of the DMA transfer to the other buffer. The Microblaze re-programs the DMA Engine before the next image starts. Since the camera itself provides blank time between sending every images, this blank time exists for all the output images of the video processing cores. This blank time is sufficiently long to re-program the DMA Engine before starting to send the first pixel of the next image.

All the video processing cores have their own video outputs. Synchronously transferring all of these videos to the PC or simultaneously writing all of them to the DDR3 is not feasible due to bandwidth limitations. Therefore, the system is implemented to select the output of one of the video processing cores to transfer to the PC. This multiplexing process is not trivial since the output images of the video processing cores are not synchronized to each other in terms of clock frequency and scheduling. All cameras are driven by the same 48MHz clock due to the requirement of perfect synchronization, however there is no guarantee that the *pixel clocks* that cameras send back to the FPGA have the same phase. Therefore three *pixel clocks* of the cameras should be separately used for sampling and transferring pixels. The DE hardware operates with a clock of 190MHz frequency but provides its outputs with the clock of center camera. The rendering hardware synthesizes images with the clock of the center camera. In addition, the outputs of the Camera Interface, DE Hardware and the Rendering hardware are not synchronized to each other since they work in pipeline manner, thus they provide the first pixel of the images all in different times. Therefore, a multiplexing mechanism is implemented to safely switch the data and clock between different possible outputs of the video processing cores.

The embedded system presented in Fig. 6.2 is designed to transfer 1024×768 resolution images to the PC in real-time and to allow different possible output combinations with 32-bit pixel format. The format of the 32-bit pixel is determined by the user through GUI. The Output Selection module receives the all possible images from the Camera Interfaces, DE hardware and rendering hardware, and applies multiplexing to provide 32-bit output pixels values. The selected 32-bit pixel format images are dumped to the AXI FIFO to be transferred to the PC via Ethernet. The Output Selection module receives 24-bit RGB pixels of the Left, Center and Right cameras from the Camera Interface, 24-bit RGB and 24-bit YCbCr pixels of the Left, Center and Right cameras from DE hardware, 8-bit disparity value from DE hardware, 24-bit RGB depth representation value from the Depth Histogram hardware, and 24-bit RGB synthesized image of the rendering hardware. The Output Selection module is configurable to support different output formats. The most profitable format for many software-based video processing application consists of combining the synchronized 24-bit RGB pixel of the Center image and 8-bit disparity into 32-bits. Therefore, the system can provide an RGB+D output for the center camera with this 32-bit format. The output selection hardware is able to transfer RGB or YCbCr images of the any selected camera. In addition, transferring synchronized snapshots of the Left, Center and Right cameras is important to compute high-quality external camera calibration parameters. The Output Selection module is configurable to combine 8-bit Y values of the Left, Center and Right images and 8-bit disparity result in 32-bit format. Therefore, the Output Selection module is able to send gray images of the three cameras in parallel which are used in the calibration software of the GUI. Moreover, the Output Selection module can be configured to display disparity images either in RGB or Gray. If the 24-bit RGB depth display mode is selected to provide better visual histogram, the system does not transfer the original image pixel together with the disparity value. Therefore, the Output Selection module utilizes an 8-bit gray disparity value instead of a 24-bit RGB-based depth representation, if the user selects a configuration to receive the image pixel and depth together as in RGB+D format.

Figure 6.6 presents the processing scheme of the Output Selection module that allows to switch between the outputs of the video processing cores. The Output Selection module receives 12 videos as its inputs. In order to simplify the figure, the multiplexing scheme for 3 videos of video processing cores are presented. In the presented case, the Output Selection module receives the original left image and original right image from the Camera Interface modules and the disparity image from the DE module. The Output Selection module needs to

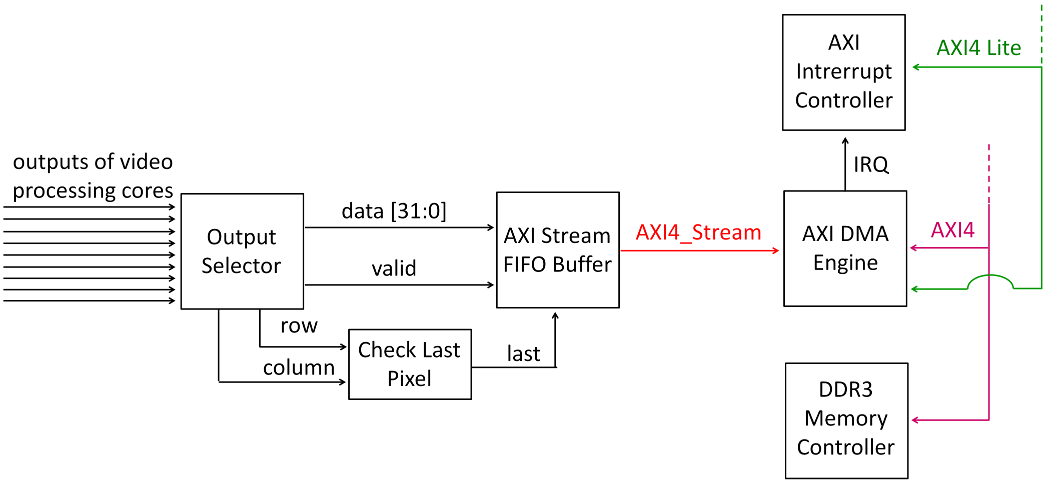


Figure 6.5: Block-diagram of the interface between the disparity estimation hardware and the DDR3 memory.

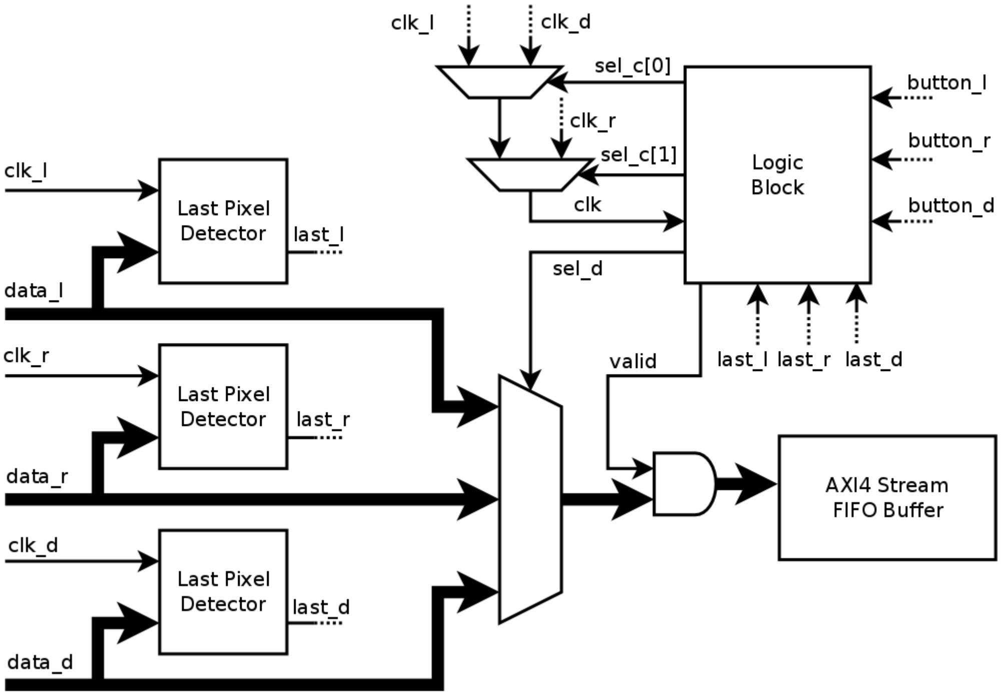


Figure 6.6: Block-diagram of the Output Selection circuit for three possible outputs.

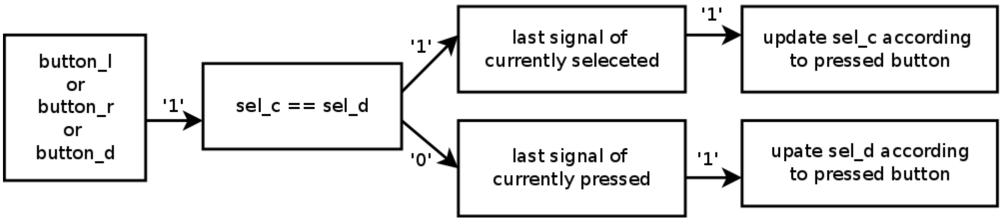

button_l
or
button_r
or
button_d
'1'
sel_c == sel_d
'1'
'0'
last signal of currently seleceted
last signal of currently pressed
'1'
'1'
update sel_c according to pressed button
upate sel_d according to pressed button

Figure 6.7: Flowchart representation of the logic block functionality.

multiplex not only data, but also the clock to fill the AXI FIFO. In addition, it should be able to dynamically switch between the output image of different video processing cores. Immediate switching between different videos is not trivial since the disparity video and original images are not synchronous to each other. Two clock multiplexers select a clock of a logic block using *sel_c* signal. On the left side of the figure, the three different clock inputs as well as the three data buses are presented. These buses include the pixel data, the pixel coordinates with row and column numbers, and the pixel valid signals. A hardware block named Last Pixel Detector generates a signal that is '1' for 12 clock cycles after the last pixel of the image is on the bus. These signals are named *last_l*, *last_r* and *last_d* in the figure. This information is used by the Logic Block that generates the select signals for the data multiplexer (*sel_d*) and for the clock multiplexers *sel_c*. The function of this block is to ensure safe switching between the different clock and data sources, without interrupting image transfers. The logic block also has inputs named *button*, which originate from the software accessible registers of the microblaze, to switch between the depth results, the left camera or the right camera. The command signals are named *button* since they transfer the selected functionality pressed on the PC by the user. The microblaze processor applies the command given by the GUI by accessing software accessible registers.

Figure 6.7 shows the functionality of the Logic Block. This block ensures that there is no leftover data in the FIFO buffer while switching between sources. In addition, Logic Block controls that the new selected source only starts writing data into the buffer when a new frame starts. If one of the three source is selected, it first checks whether or not the values of sel_clk and sel_data are equal. If this is true, the Logic Block should wait for the current frame to finish, before changing the select signal of the clock multiplexer. If sel_clk and sel_data are not equal, the block has to wait for current frame of the new source to finish, before changing the select system of the data multiplexer. During this transition period, when sel_clk and sel_data have different values, the output of the data multiplexer should not be stored in the FIFO Buffer. This is achieved by forcing the valid signal of the multiplexer output to 0 using the valid output of the block.

6.4 Embedded Software

The program that is operating on the MicroBlaze processor has four main functions, the system setup, controlling the video transfer operations from FPGA to PC, interfacing between user commands and video processing cores, and operating auto color-gain and auto shutter-width correction operations.

The first functionality of the microblaze processor is seting-up the hardware while the system boots up. The microblaze configures the cameras using I^2C with the initial parameters. Then it sets up the networking interface and establishes a connection with the host PC. The DMA that copies the image data from the FIFO buffer to the DDR3 memory is programmed to copy the full image. The Output Selection module and the DMA of the Ethernet MAC are configured. While the DMA of the FIFO buffer is waiting the first image to write to the DDR3, the microblaze sends *start* command to the video processing cores through software-accessible registers. The sampling of the pixels by the Camera Interface starts with the *start* command, which completes the boot-up process.

The second functionality of the microblaze processor consists of operating the video transfer. The program allocates two image sized buffers which are used alternately. The AXI DMA recognizes when the FIFO buffer contains any data using the *ready* signal of the AXI4 Stream interface. It automatically waits for more data to arrive and can therefore copy a full image without needing any reprogramming, even with varying transmission speeds. As soon as the DMA Engine transmits the last pixel of the image, it sends an interrupt request to the interrupt controller. In the interrupt service routine, the DMA Engine is once again programmed to transmit a full image, but this time to the other buffer. After the service routine has finished, the program starts sending the complete picture via Ethernet to the host PC. A small header message is sent to the PC to inform the PC that a new frame is starting. Packages of two rows of pixels are assembled and sent. Two rows contain 2048 pixels which means a total package size of 8192 Bytes. This packet size is close to the maximum of 9000 Bytes which facilitates data handling when receiving images on the PC.

The third functionality of the microblaze processor is interfacing between user commands and video processing cores. While the microblaze is operating the video transfer, it waits for the interrupt from the UART to receive the user commands. Therefore, user is able to dynamically write or read system parameters using GUI through the UART. The communication between the microblaze processor and the GUI is implemented with 5 byte packets. The first byte of the packet identifies the type of the request, the next 4 bytes are used to contain data. The type of the request can be writing or reading the parameters of image sensor through I^2C, or accessing the parameters of video processing cores through software accessible registers, or auto color-gain and auto shutter-width correction.

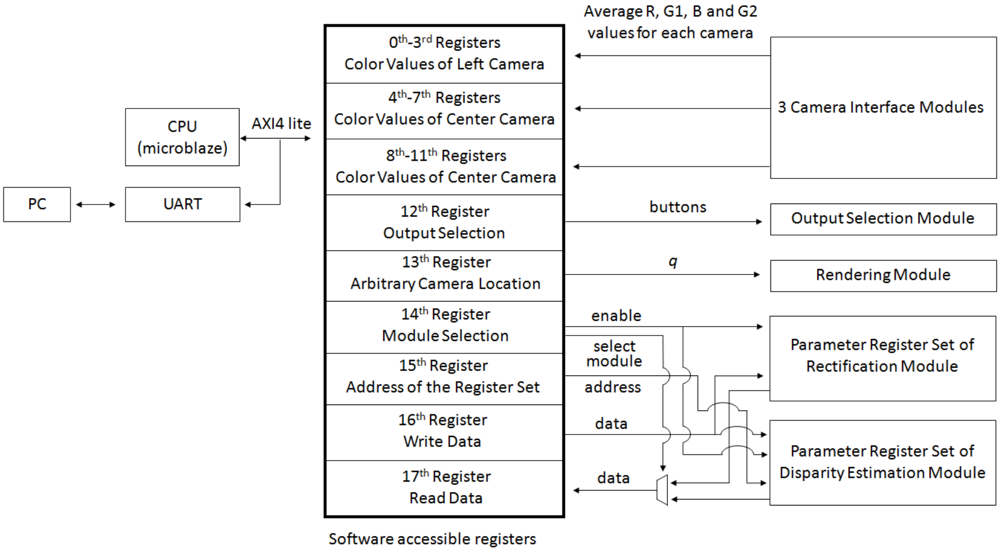

PC
UART
CPU (microblaze)
AXI4 lite
0th-3rd Registers Color Values of Left Camera
4th-7th Registers Color Values of Center Camera
8th-11th Registers Color Values of Center Camera
12th Register Output Selection
13th Register Arbitrary Camera Location
14th Register Module Selection
15th Register Address of the Register Set
16th Register Write Data
17th Register Read Data
Software accessible registers
Average R, G1, B and G2 values for each camera
3 Camera Interface Modules
buttons
Output Selection Module
q
Rendering Module
enable
select module
address
data
data
Parameter Register Set of Rectification Module
Parameter Register Set of Disparity Estimation Module

Figure 6.8: Top-level schematic of software accessible registers.

The top-level schematic of the software-accessible registers is presented in Fig. 6.8. The AXI4 Lite bus is connected to 18 32-bit registers to communicate with video processing cores. Each of these registers has a different address on the AXI4 Lite bus. The Camera Interface Modules write the average red, green1, blue and green2 values of the images from the three cameras to the 0^{th}-11^{th} registers to be used for the auto color-gain and shutter-width correction operations. The registers 12^{th}-16^{th} can be only written by the microblaze processor. The 12^{th} register is used to control Output Selection module. The 13^{th} register is used to provide the artificial camera location q of the rendering hardware. The registers 14^{th}-17^{th} are used to access any parameter register of the rectification hardware or DE hardware. The 2-bits of the 14^{th} register are used as *enable* signals to identify whether the microblaze writes parameter register of rectification hardware or DE hardware. The 1-bit of the 14^{th} register is used as a *select module* signal to identify whether the microblaze reads parameter register related to the rectification hardware or DE hardware. The 15^{th} register identifies the address of the internal register-set of rectification or DE hardware. The 16^{th} register identifies the data to be written to the internal register-set of rectification or DE hardware modules. The 17^{th} register identifies the data to be read from the internal register-set of rectification or DE hardware modules. Therefore using four 32-bit software accessible registers (14^{th}-17^{th}), the microblaze is able to access any address of the register-set of rectification or DE hardware modules for read or write purposes. The functionality of internal configuration register-set of the DE hardware is presented in Chapter 8.

The fourth functionality of the microblaze processor consists of operating auto color-gain and auto shutter-width correction operations. The auto color-gain and auto-shutter width corrections are useful to obtain almost identical color values for the same object from different cameras, which is one of the important requirements of high-quality disparity estimation. Additionally, these corrections are important to obtain better visual quality from the images. Auto color-gain correction is useful for white balance since in this case, the average of red, green and blue values should be expected to be equal. Auto color-gain correction is also used to slightly increase or decrease the brightness of the image. Auto shutter-width correction is useful to adjust the exposure time of the cameras to adapt different light conditions. Hence, if the environment is dark or very bright, the exposure time can be automatically adjusted to obtain satisfactory disparity estimation results.

The user provides the expected average red, green and blue values to the GUI. Then the GUI transfers these expected color values to the microblaze. The microblaze compares the average color values read from the Camera Interface modules with the user expectations, then iteratively increases or decreases the analog color-gain and shutter-width parameters of the image sensors through I^2C depending on the comparison results. The microblaze first arranges shutter-width to roughly reach the target values and subsequently changes analog-color gains to make sensitive adjustment. Changing the shutter-width parameter, i.e. exposure time, increases or decreases the frame rate of the camera if it is utilized out of a certain range. The user is also able to provide a command from the GUI to fix or unfix the frame rate using the Keep Current FPS button. If the fix frame rate option is selected from the GUI, the shutter-width is adjusted until the limit of the certain range to keep the initial frame-rate, then corrections are mainly made on analog color-gains.

7 Hardware and Software based Applications of Disparity Estimation

Depth map estimation can be used in a wide range of image and video processing applications. This thesis presents several real-time hardware and software based applications of disparity estimation. The implemented applications conceptually prove that the high-quality and high-performance RGB+D outputs of the proposed real-time disparity estimation hardware can be used for enhanced 3D based video processing applications. Firstly, the free view sythesis hardware that utilizes trinocular disparity estimation is presented. Secondly, the real-time software applications of disparity estimation operating at PC are presented.

7.1 Free View Synthesis Hardware Using Trinocular Disparity Estimation

The recent development of high-quality free viewpoint synthesis algorithms and their implementations allows to realize glasses-free 3D perception. Although many algorithms are developed for this application, the real-time hardware realization of a free viewpoint synthesis for real-world images is challenging due to its high computational load and memory bandwidth requirements. In this Chapter, the first real-time high-resolution free viewpoint synthesis hardware utilizing the proposed three-camera disparity estimation is presented. The proposed hardware generates high-quality free viewpoint video at 55 frames per second using a Virtex-7 FPGA at a 1024 × 768 XGA video resolution for any horizontally aligned arbitrary camera positioned between the leftmost and rightmost physical cameras.

7.1.1 Hardware-Oriented Three-Camera Free Viewpoint Synthesis Algorithm

The simplified classical concept of three-camera horizontally aligned free viewpoint synthesis is illustrated in Fig. 7.1. Assuming that an object is viewed as a single pixel in the x coordinate

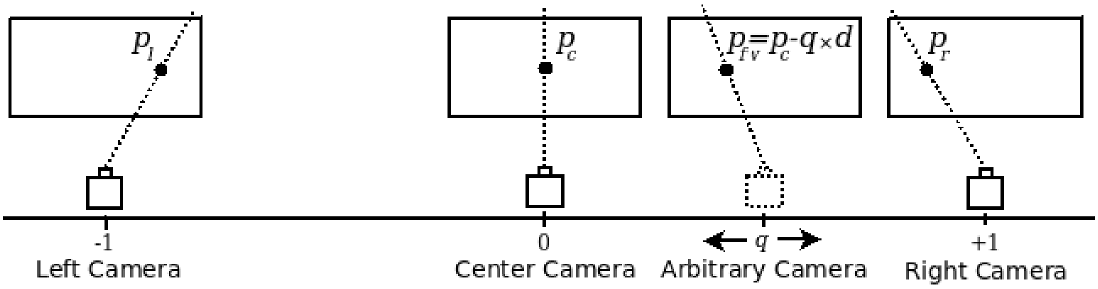


Figure 7.1: Definition of the three-camera free viewpoint synthesis setup.

p_c of the center image, p_l of the left image and p_r of the right image, then the disparity of a pixel at p_c is $d = p_l - p_c = p_c - p_r$. If the free view image is generated for an arbitrary camera at location q ($-1 \leq q \leq 1$, the normalized location of the left camera is at -1, and the right camera is at 1), 3D trigonometry dictates that the view coordinate of the same object in the generated image can be computed as $p_{fv} = p_c - q \times d$. Hence, the pixel at p_c of the center image can be picked and mapped to coordinate p_{fv} of the synthesized image. The free view synthesis is a complex problem using real-world images considering that the computation of the correct d for every pixel is very challenging especially for object boundaries, low textured regions and occluded parts of the images. In addition, the real-time implementation of high-quality free viewpoint synthesis algorithm causes significant challenges due to high computational load and memory bandwidth requirements, especially for high-resolution video.

The proposed hardware-oriented three-camera adaptive weight free viewpoint synthesis (TAW-FVS) algorithm generates the free viewpoint images of any single horizontally aligned arbitrary camera positioned between the leftmost and rightmost cameras. The TAW-FVS algorithm consists of three steps. First, lens distortions and camera misalignments are corrected using camera calibration and rectification. Second, the disparity values of every pixel of the center image are computed using the proposed trinocular adaptive weight disparity estimation algorithm. Third, free views are synthesized by a low computational cost rendering algorithm using the images of the three cameras, the disparity values of the center image and the position of the arbitrary camera (q).

Internal and external calibration values of the cameras are computed off-line using the Open-CV multiple camera calibration toolbox [70]. The Caltech rectification algorithm [50] is used to horizontally align the images captured from the three cameras. The proposed trinocular DE and rendering algorithms are developed to support efficient parallel operations, to consume low hardware resources and to avoid the requirement of the external memory while providing high quality free viewpoint synthesis results.

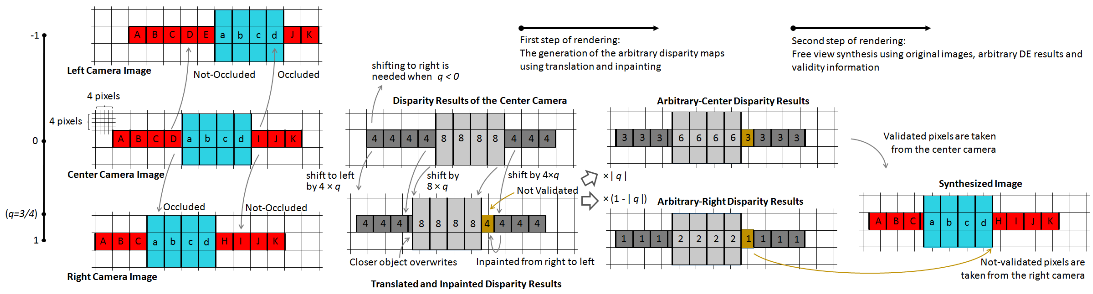

-1
0
(q=3/4)
1
Left Camera Image
Center Camera Image
Right Camera Image
4 pixels
4 pixels
Not-Occluded
Occluded
Occluded
Not-Occluded
shifting to right is needed when q < 0
Disparity Results of the Center Camera
shift to left by 4 × q
shift by 8 × q
shift by 4×q
Not Validated
Closer object overwrites
Inpainted from right to left
Translated and Inpainted Disparity Results
First step of rendering:
The generation of the arbitrary disparity maps using translation and inpainting
Arbitrary-Center Disparity Results
× | q |
× (1 - | q |)
Arbitrary-Right Disparity Results
Second step of rendering:
Free view synthesis using original images, arbitrary DE results and validity information
Validated pixels are taken from the center camera
Synthesized Image
Not-validated pixels are taken from the right camera

Figure 7.2: The illustration of free viewpoint synthesis algorithm that utilizes trinocular disparity estimation. (Each square grid represents 4×4 pixels. Only one out of four column/row grids are represented to improve the clarity of the representation).

The quality of the free viewpoint images essentially depends on the quality of the disparity estimation. Two-camera DE causes wrong disparity estimation values for the occluded regions. The three-camera DE solves most of the occlusion issues and provides much better DE results thanks to its double-checking scheme, i.e., using DE results of center-left and center right image pairs. Moreover, the three-camera system extends the horizontal range of the arbitrary camera location compared to the two-camera version.

In order to obtain high-quality and real-time DE, the trinocular DE algorithm that is presented in Chapter 4 is used. The proposed T-AWDE algorithm provides significantly better DE quality than the binocular AWDE by exploiting the fusion of the DE results of the center-left and center-right pairs. During the DE process of every pixel of the center image, the candidate disparities on the right side are searched for the center-left pair, and the candidate disparities on the left side are searched for the center-right pair. The T-AWDE compares the hybrid cost values of the center-left and center-right pairs for every disparity to select the one that exhibits a minimum cost as a disparity value. As an addition to the T-AWDE algorithm, an adaptive penalty is used in the DE step of the TAW-FVS algorithm. An adaptive penalty is utilized depending on the location of the free viewpoint while comparing the cost values computed by center-right and center-left pairs. If $q < 0$, the disparity selection is conducted to select the disparity value assigned by the center-left pair, and if $q \geq 0$, the disparity selection is conducted to select the disparity value computed by the center-right pair by adding $|q| \times 150$ to the matching cost of one pair. Adverse effects of sensitive rectification errors and wrong disparity selections are significantly removed using adaptive penalty. Subsequently, the T-AWDE iteratively smoothens the computed disparity map using the brightness values of the neighboring pixels following the refinement scheme prescribed by AWDE algorithm [9].

The rendering algorithm is applied upon completion of the DE process of the center image. The rendering process involves two steps as presented in Fig. 7.2. First, the arbitrary disparity maps of the arbitrary camera are synthesized by translating the disparity map of the center camera to the location of the arbitrary camera. If $q < 0$, the arbitrary disparity maps of the arbitrary-center and arbitrary-left pairs are generated. If $q \geq 0$, the arbitrary disparity maps of the arbitrary-center and arbitrary-right pairs are generated. In the next step, the free viewpoint is synthesized using the arbitrary disparity maps and the existing camera pictures.

As presented in Fig. 7.2, more than one pixel of the center camera's disparity image can map to the same pixel of the translated disparity map image. In this case, the higher disparity value belongs to a closer object, thus the higher disparity overwrites the lower disparity. Any pixel that is not mapped during the translation process identifies an occlusion; this location is marked as a *not-validated* pixel location. In order to obtain valid depth values for occluded regions, those pixels are filled by an inpainting method. When the disparity of a pixel is not validated, the disparity data from the last valid value is repeated. This inpainting process is applied from left-to-right when $q < 0$, and right-to-left when $q \geq 0$. The translated and inpainted depth map is then multiplied by $| q |$, $(1 - | q |)$ and again $(1 - | q |)$ to create the arbitrary-center, arbitrary-right and arbitrary-left disparity values, respectively. Those disparity values and the validity information are used to reconstruct the arbitrary image. The pixels of non-occluded, i.e. *validated*, areas are taken from the center image. The pixels of occluded, i.e. *not-validated*, areas are taken from the left image if $q < 0$, or from the right image if $q \geq 0$.

7.1.2 Real-Time Free Viewpoint Synthesis Hardware

The block diagram of the full system that implements the TAW-FVS algorithm is presented in Fig. 7.3. A Virtex-7 FPGA included in the VC707 Evaluation Board is used to prototype the developed hardware. The cameras are perfectly synchronized using the method explained in Chapter 4. All the real time video processing computations are implemented in hardware. The PC is used as a display, to control the system and to obtain camera calibration parameters. The resulting free views are transferred to the PC using 1Gb raw Ethernet. A standard 2D TV is connected to the PC using HDMI, which offers a better display and emulates a future glass-free 3D TV. A MicroBlaze softcore is used to initialize cameras through I^2C, to control Ethernet and to communicate with the PC. A DDR3 memory is only used for Ethernet buffering. The camera interface, rectification, disparity estimation and rendering hardware blocks are designed to avoid utilizing DDR3 memory; thus they receive a stream input and provide a stream output. Therefore, the video processing core can be easily converted to a single ASIC. The system is able to send either the 32-bit pixel RGB+Depth video of the center camera, or the RGB video of any physical camera, or the synthesized RGB video of a free view.

The implemented Caltech rectification hardware utilizes 64 on-chip BRAMs for each of the cameras. Each BRAM is used to buffer one row of the image. The rectification hardware processes the images of three cameras in parallel, and synchronously transfers rectified YCbCr images to the disparity estimation module. The details of the Caltech rectification hardware is presented in Chapter 5.

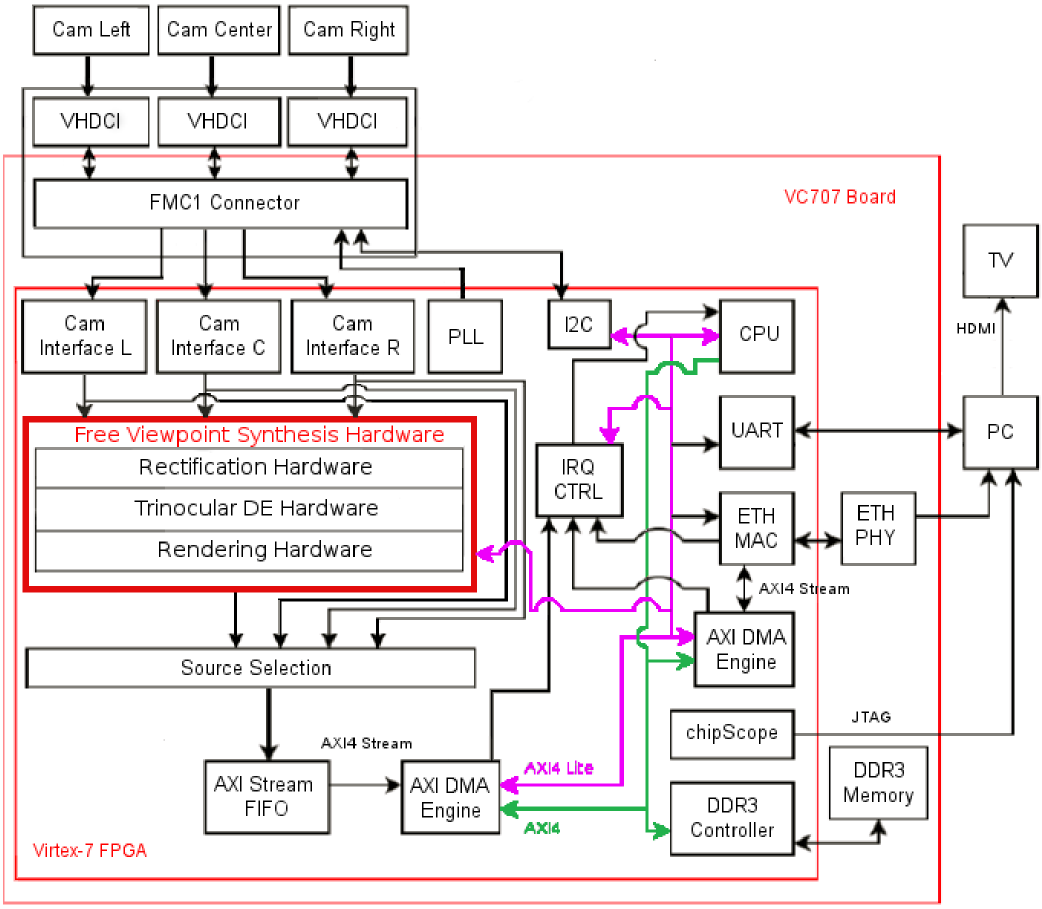


Figure 7.3: Block diagram of the free viewpoint synthesis system.

The DE hardware of the TAW-FVS system is based on the T-AWDE hardware presented in Chapter 4. The T-AWDE hardware is composed of two high-performance and high-quality binocular disparity estimators presented in Chapter 3. The outputs of the two estimators are two maps, one associated to the center-left pair of cameras and the other associated to the center-right pair. A fusion block selects which disparity value of these two maps will compose the final three-camera disparity map by comparing the matching costs computed by the two pairs. The fusion block of the DE hardware of TAW-FVS system adds adaptive penalties to the matching costs depending on the position of the arbitrary camera. Finally, three-camera DE hardware iteratively refines the DE values to smoothen the computed disparity map using the brightness values of the neighboring pixels. The disparity estimation hardware buffers the input pixel values using 39 BRAMs for each camera to realize a window based matching scheme. The 8-bit disparity values and the 24-bit pixels of the left, center and right cameras are additionally buffered and synchronized using 16 BRAMs for each channel before transferring these data to the rendering hardware.

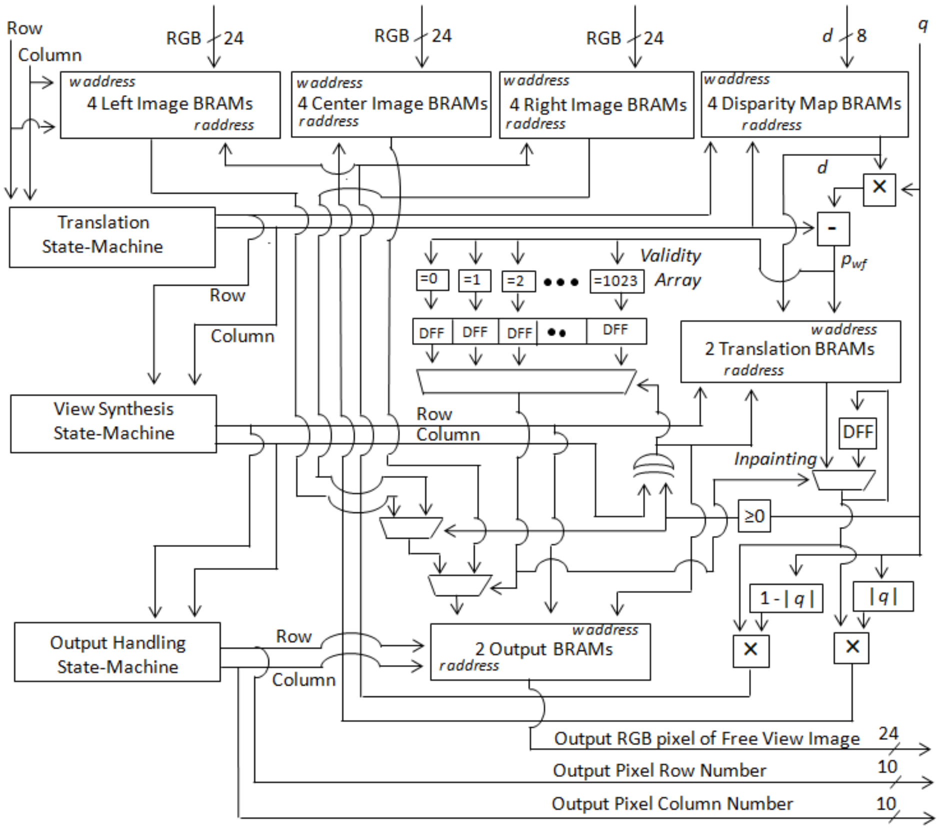


Figure 7.4: Rendering hardware (pipeline stages are not shown).

The rendering hardware is presented in Fig. 7.4. The rendering hardware is blind to the DE process. It receives the synchronized left, center and right RGB camera images and the disparity image of center camera as its inputs, row by row. The synthesized image is the output. The arbitrary camera location q is dynamically sent from the GUI. q is defined as a 5-bit signed fixed-point value to allow 32 arbitrary locations where '00000' represents the center camera location.

4 BRAMs are used to buffer 4 rows of each input channel. The rendering process is handled row by row. The least significant bits of the row number are used to select the BRAM. The column number is used as an address. The *translation state machine* generates row numbers and column numbers to read back the disparity map BRAMs pixel by pixel. Concurrently, p_{fv} values are computed using fixed-point multiplication and subtraction hardware, then rounded to the closest integer. p_{fv} is used as write address of the *Translation BRAMs* where the shifted disparity values are buffered. A 2×1024 size flip-flop array is used to buffer the validity bits. The *validity array* keeps track of the *not-validated* pixel locations which require inpainting. Whenever a disparity value is written to the p_{fv} address of the *Translation BRAMs*,

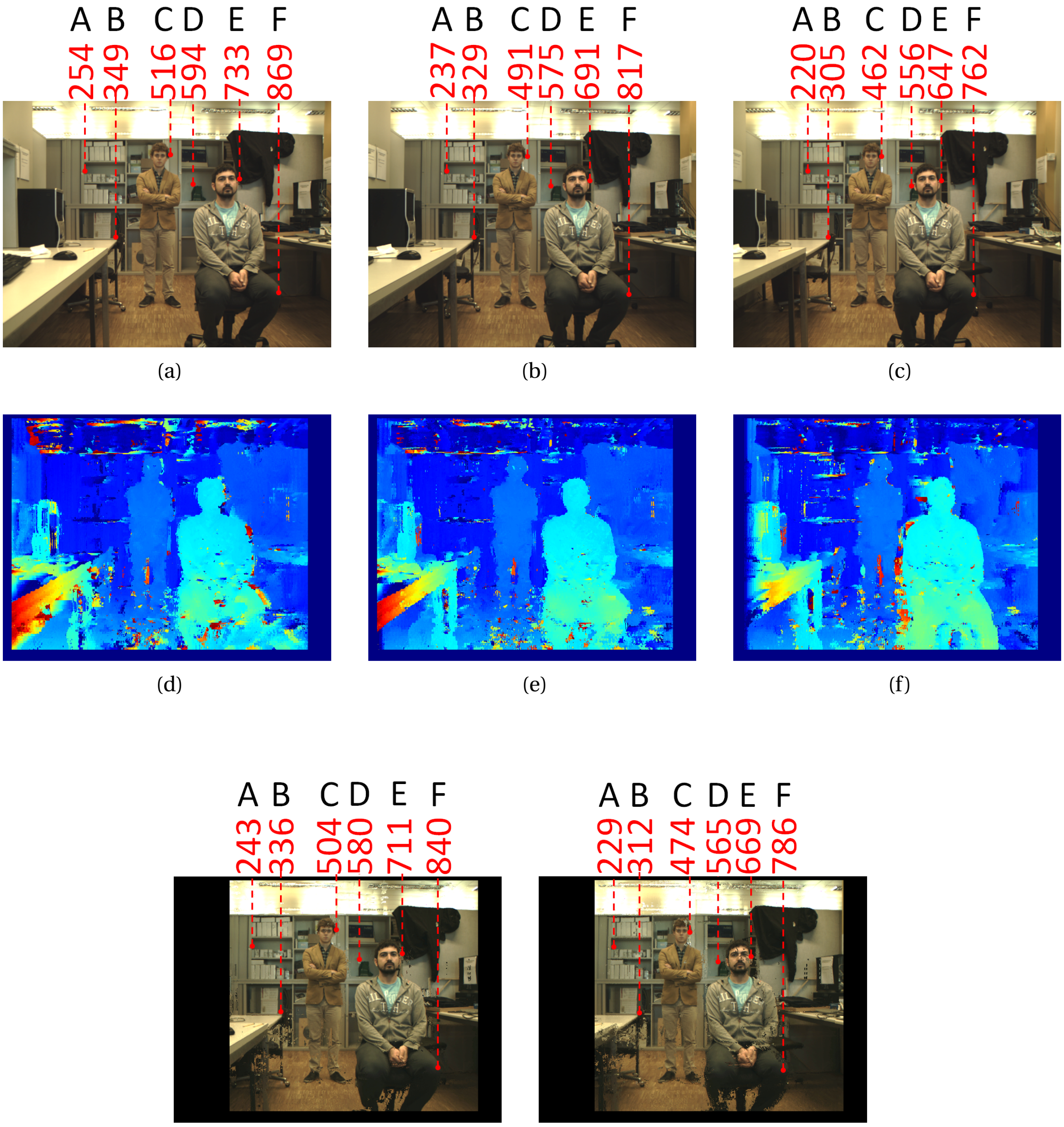
the column location p_{fv} is marked as valid by comparing p_{fv} with 1024 possible column addresses using 1024 comparators. The pixels in a row that are not validated remains as *not-validated* pixels since the *validity array* of a row is reset after the process of row synthesis.

The *view synthesis state machine* generates row numbers and column numbers to read back the *Translation BRAMs* and *validity array* pixel by pixel. The reading scan direction of a column is set from 1023 to 0 if $q < 0$, and 0 to 1023 if $q \geq 0$ since the inpainting direction changes according to the arbitrary camera location. The *validity bit* is used as a select signal of a multiplexer. If the pixel is not validated, the disparity value previously stored in the flip-flop replaces it, i.e. paints the read disparity value. The translated and inpainted disparity value is multiplied by $|q|$ and $(1-|q|)$ to compute the arbitrary disparity values of the arbitrary-center, arbitrary-left and arbitrary-right camera pairs. Arbitrary disparity values are used as a read address for the input channel BRAMs. The selection of the source pixel is handled by multiplexers. If the pixel is validated, the pixel of the center image is used as the synthesized pixel. If $q < 0$ and the pixel is not validated, the pixel of the left image is used as the synthesized pixel. If $q \geq 0$ and the pixel is not validated, the pixel of the right image is used as the synthesized image pixel. Synthesized image pixels are buffered in two output BRAMs. The *output handling state machine* reads back the output BRAMs pixel by pixel, and sends outputs of the rendering hardware as the final output of the free viewpoint synthesis hardware.

7.1.3 Implementation Results

The proposed real-time three-camera free viewpoint synthesis hardware is implemented using Verilog HDL, and verified using Modelsim 10.1d. The Verilog RTL models are mapped to a Virtex-7 XC7VX485T FPGA comprising 607k Look-Up-Tables (LUT), 303k DFFs and 1030 BRAMs. The rectification hardware consumes 13% of the LUTs, 6% of the DFF and 19% of the BRAM resources of the FPGA. The trinocular DE hardware consumes 25% of the LUTs, 11% of the DFF and 16% of the BRAM resources of the FPGA. The rendering hardware consumes 1% of the LUTs, 1% of the DFF and 2% of the BRAM resources of the FPGA.

The system is functionally verified in real-time. The speed performance of the free viewpoint synthesis hardware core is limited by the working frequency and parallelization of trinocular DE hardware. The hardware core operates at 175 MHz after place & route. The trinocular DE hardware and thus the complete TAW-FVS system can process up to 55 fps at a 1024 × 768 XGA video resolution for a 128 pixels disparity range. Although this speed performance is verified using Chipscope, the current display output of the system is 18 fps due to bandwidth limitation of raw Ethernet output. The 55 fps performance of the hardware will be fully exploitable using USB3 or HDMI in the future.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Figure 7.5: Real-time snapshots captured by the proposed system. (a) Left image (b) Center image (c) Right image (d)IR-AWDE for center-left (e) T-AWDE (f)IR-AWDE for center-right (g) synthesized free viewpoint image for an arbitrary camera located at q=(-0.5) (h) synthesized free viewpoint image for an arbitrary camera located at q=(0.5) (images best viewed in high resolution from the pdf files)

Real-time snapshots captured by the proposed system are presented in Fig. 7.5. While capturing multiple consecutive snapshots, models stand stable in front of the system. The horizontal locations of some specific locations are marked off-line with dashed lines in original left, center and right images in Fig. 7.5a, Fig. 7.5b, and Fig. 7.5c, respectively. The center-left, trinocular and center-right DE results provided for the center camera image are presented in Fig. 7.5d, Fig. 7.5e, and Fig. 7.5f, respectively. The trinocular DE solves most of the occlusion and wrong estimation errors by the fusion of the DE results of the two pairs which is essential to synthesize high-quality free view images. The synthesized free viewpoint images of two artificial viewpoints q=(-0.5) and q=0.5 are presented in Fig. 7.5g and Fig. 7.5h, respectively. The efficiency of the presented system is evidenced by horizontal pixel coordinates of the marked location in the synthesized image. Moreover, correct 3D appearance and occlusion effects in the synthesized free viewpoint images can be identified by observing the background objects near the models in Fig. 7.5.

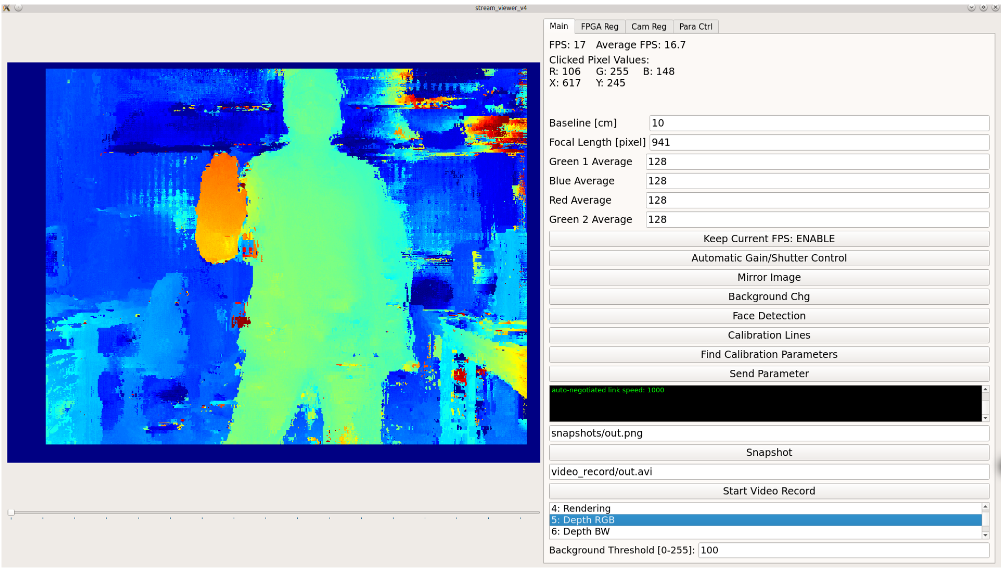
7.2 System GUI and Software Based Real-Time Applications

In this section, the graphical user interface of the system running in PC is explained. The software of the GUI delivers three fundamental tasks: the capture and display of the video output of the FPGA, allowing user to send commands to the system, and developing software based real-time applications. Several software applications are developed using the disparity estimation hardware. The implemented software applications are: Speed and distance measurement, depth based image thresholding, head-hands-shoulders tracking, virtual mouse using hand tracking, face tracking integrated with free viewpoint synthesis, and stereoscopic 3D display. The implementation details of these applications are presented in this section. The GUI, the visual results of video processing cores and the software based application running in PC are demonstrated in a video in Appendix A.

7.2.1 Capture and the display of the video

In order to capture and display the images on a PC platform, dedicated software is developed using Qt. Qt is very suitable to make the GUI work on different operating systems such as Windows, Mac OS and Linux. Since the data is sent to the PC using raw-ethernet, receiving the packets would usually require low-level access in the TCP/IP stack. The packet capture (pcap) library is used for Linux and Mac OS to access to packet data, and the winpcap library is used for windows operating system. Following to the real-time tests, Linux provides the fastest data transfer rate for raw-ethernet, thus the GUI is developed and verified mainly on Linux OS.

As described in the Chapter 6, the incoming image is sent by consecutively transferring of packets that include two rows of the image. Each of the row includes 1024 32-bit pixels. Therefore, 368 packets should be send to complete the transfer of a 768×1024 resolution image. Additionally, at the beginning of every image, a pre-defined *header packet* is sent from



FPS: 17 Average FPS: 16.7
Clicked Pixel Values:
R: 106 G: 255 B: 148
X: 617 Y: 245

Baseline [cm]: 10
Focal Length [pixel]: 941
Green 1 Average: 128
Blue Average: 128
Red Average: 128
Green 2 Average: 128

```
auto-negotiated link speed: 1000
```

snapshots/out.png
video_record/out.avi
5: Depth RGB
Background Threshold [0-255]: 100

Figure 7.6: Screen shot of the stream viewer software running on Linux.

the FPGA. The GUI counts the number of the packets between two headers, to determine if the full image is successfully transfered. If the transfer is successfully completed, the image is displayed and processed, otherwise it is skipped. If the user prefers to display 8-bit disparity results, the most significant 8-bits of the 32-bit pixels is equally used in the R, G and B channels to visualize gray images. Otherwise, the 24-bit least significant bits of the 32-bit pixels are used in the R, G and B channels to be displayed.

7.2.2 Front-end of the GUI

In this subsection, the user-friendly front-end, i.e. presentation, layer of the GUI is presented. The front-end of the GUI is design to easily send the commands of the user to the system.

The snapshot of the GUI is presented in Fig. 7.6. The GUI include display monitor in its left side, and the 4 control tabs in its right side. The four tabs are named as Main, FPGA Reg, Cam Reg and Para Ctrl. The Main tab allows the user to switch between the output data format of the FPGA and the implemented 3D-based video processing applications. In addition, auto color-gain and shutter-width corrections are initialized from the Main tab. The FPGA Reg tab allows the user to reach any software-accessible register of the microblaze processor. The parameter registers of the video processing cores can be controlled by using the FPGA Reg tab. The registers of the Aptina Cameras can be read or changed using the Cam Reg tab. The Para Ctrl tab is used to define the expected approximate body sizes of the user to be used for head-shoulders-hands tracking. The details of the front-end of the GUI are visually presented in Appendix A.

7.2.3 Software based real-time applications

The user is able to select the display format and software application through the GUI. Several software applications are developed using the disparity estimation system. The implemented software applications are: Speed and distance measurement, depth based image thresholding, head-hands-shoulders tracking, virtual mouse using hand tracking, face tracking integrated with free viewpoint synthesis, and stereoscopic 3D display. The real-time performance of the applications are presented in Appendix A. The software applications are implemented to evidence that the high-quality RGB+D outputs of the proposed system can be used to develop advanced 3D-based video processing applications in real-time, though improving the theory of the software applications itself is not targeted. Future possible improvements of the software applications and their prospective advanced utilizations are explained. [1]

Distance and Speed Measurements

The system sends a disparity map to the PC. The depth of the pixel or the distance of the object from the camera can be calculated by using this disparity value by the GUI. The eq. 2.2 that is presented in Chapter 2 formulates the depth z as a function of the baseline, the focal length, and the measured disparity. The baseline chosen for the initial setup is $B = 10cm$ and the focal length of the lenses is $f = 6mm$. Since the unit of the disparity is the number of the pixels, the f value should be also represented using the number of pixels. The f value with the unit of pixels is obtained as *941* from the camera calibration. The f and baseline values are written to the GUI by the user, a single time at the system power-on. If different type of optics are used or if the distance between the cameras are changed, the user is required to change the new fixed values in the Main tab of the GUI. The GUI is able to convert the disparity d to a distance z for every pixel using eq. 2.2. Since the computational load of eq. 2.2 is very low, and it can easily run in real-time, the conversion from disparity to depth is not implemented in hardware.

Since the distance can be computed for every pixel, computing the speed of the pixel into the direction z is also straightforward. The GUI computes the speed of the pixel in terms of cm/s using the frame rate and the change of distance z of a pixel, from image to image. The expression of the computation of speed (*spd* cm/s) is provided in eq. 7.1, where z_n and z_{n+1} identify the distance of the pixel with the unit of cm in two consecutive images n and $n+1$, respectively. *fps* represents the instantaneous frame rate of the camera. Currently, the speed and distance value can be numerically visualized for every clicked pixel from the GUI. The distance image is visualized in the monitor for all pixels. The high-resolution speed image can be visualized in the future, which can be useful to measure and display the speed of the vehicles and sportsmen.

---

[1] The software applications head-hands-shoulders tracking, virtual mouse using hand tracking, and face tracking integrated with free viewpoint synthesis, are developed in EPFL, as a collaboration with Youngjoo Seo, a phd candidate at KAIST, Korea.

$$spd = (z_{n+1} - z_n) \times fps \tag{7.1}$$

Depth Based Image Thresholding

A depth-based thresholding application is developed to be used as an alternative method of the green-wall technique used in the cinematography. In cinematography, green walls are used to replace the background with another image or video. Since the proposed system provides high-quality RGB+D outputs for every pixel, a similar technique can be implemented without using green walls.

In this application, first, the snapshot image of the static background is taken. The threshold disparity value is provided by the GUI. If measured disparity (d) of any RGB pixel is higher than the given threshold, it is regarded as the frontground and the RGB value is displayed. Otherwise, the pixel of the background image that is captured at system initialization is displayed. Therefore, if the object is far enough, it disappears from the display, and it is displayed if it is close enough.

In the current application, the system operates with the background image of the same environment. The background image can be changed with the picture of another environment in the future. In addition, instead of static background, a video of the another environment can be used as a background in the future. Thereby, any person or object can integrated in a video of another environment in real-time, which can be useful for the journalists for the presentation of live-tv news and sport events.

Head, Hands and Shoulders Tracking Using Skeltrack Software

The head-hands-shoulders tracking application is implemented using the open source Skeltrack software [90]. Skeltrack is originally implemented to benefit from the depth estimation results of Kinect. It utilizes a data-driven approach presented in [91] for real-time human skeleton detection. It does not use any database of pre-recorded body positions. In replacement of Kinect's depth result, the input to the software is obtained from the depth results of the proposed system. The Skeltrack software efficiently performs in real-time to track the head, hands and shoulders. The skeleton detection can be used to play computer games. In addition, the skeleton detection can be integrated with human pose estimation and gesture recognition techniques in the future. Since the presented system can operate outdoors, the human behavior can be detected outdoors which can be used in surveillance applications or in sports events. Occlusion is an important problem in efficient human pose estimation. Several proposed systems can be used to monitor the same environment without interference. Therefore, the user can be surrounded by multiple of proposed depth estimation systems to provide efficient real-time pose estimation results without occlusion problem.

Virtual Mouse Using Hand Tracking

The virtual mouse application is implemented by tracking hands and fingers using the algorithm presented in [92]. Convex and concave points in the hand are found using the depth image. The concave locations present the finger tips. Less than two concave points on the hand are a sign of a closed palm, i.e. a fist pose. The fist pose of the hand is utilized to identify clicking. If the hand is open, the user is able unclick and freely move the mouse. The virtual mouse application can be used to play computer games or implement touch-less device control. A similar finger tracking and identification technique can be used to implement touch-less keyboard application in the future.

Face Tracking Integrated with Free Viewpoint Synthesis

The free viewpoint synthesis requires to receive an arbitrary camera location q from the GUI. Two different methods are implemented to dynamically change the arbitrary camera location. In the first method, the cursor on the GUI is utilized to switch between different viewpoints in real-time. In the second method, the location of the user with respect to the monitor is used to dynamically change the q values. The face tracking algorithm presented in [93] is implemented in the GUI to detect the location of the user. The RGB camera of the PC is used for the face tracking. The integration of the face tracking in to the GUI to interact with the free viewpoint synthesis hardware allows the user to freely move its observation point between the cameras. This implementation can be used to allow the users to watch sports events in real-time from any viewpoint that they select, in the future.

Stereoscopic 3D Display

The system is able to switch to stereoscopic 3D Display mode to visualize Green and Blue values obtained from one camera and the Red value obtained from the other camera. Currently the system can be used to visualize 3D using color anaglyph-based 3D glasses.

The 3D visual quality can be increased using active-glasses based 3D display in the future. In addition, since the system is able to provide a disparity result for every pixel, it can be used for implementing advanced augmented-reality applications. Any 3D object can be artificially placed at any distance, and the user can interact with these arbitrary objects.

8 ASIC Implementation of Binocular Disparity Estimation

The ASIC implementation details of the binocular DE hardware is presented in this chapter. The ASIC is designed to be used as an accelerator for any system that requires stereoscopic depth computation. The ASIC solution for the developed hardware offers crucial advantages to the system compared to its FPGA implementation, such as consuming less power, faster performance and cost effectiveness. On the other hand, the ASIC implementation suffers some additional constraints compared to the FPGA implementation. These restrictions are mainly related with the pin count, area and the usage of multiple clock domains. Therefore, several modifications are applied to the binocular DE hardware that is presented in Chapter 3. In this chapter, first, the specifications of the ASIC implementation is presented. Secondly, the modifications applied on the binocular DE hardware are presented. Lastly, the ASIC design and its possible utilization scheme are presented.

8.1 Disparity Estimation ASIC Specifications

The binocular DE hardware core is implemented in an ASIC using TSMC 40nm bulk CMOS technology. DE module is chosen for the ASIC implementation since it is the main computationally intensive module of the complete two camera depth estimation system presented in Fig. 8.1.

The DE ASIC is designed to receive a stream input of images, and to provide a stream output of RGB+D or YCbCr+D data. It does not require external memory for the computation of disparity results. The ASIC is configurable through SPI interface to support different features. The binocular DE ASIC is designed to support trinocular DE by using two ASIC chips.

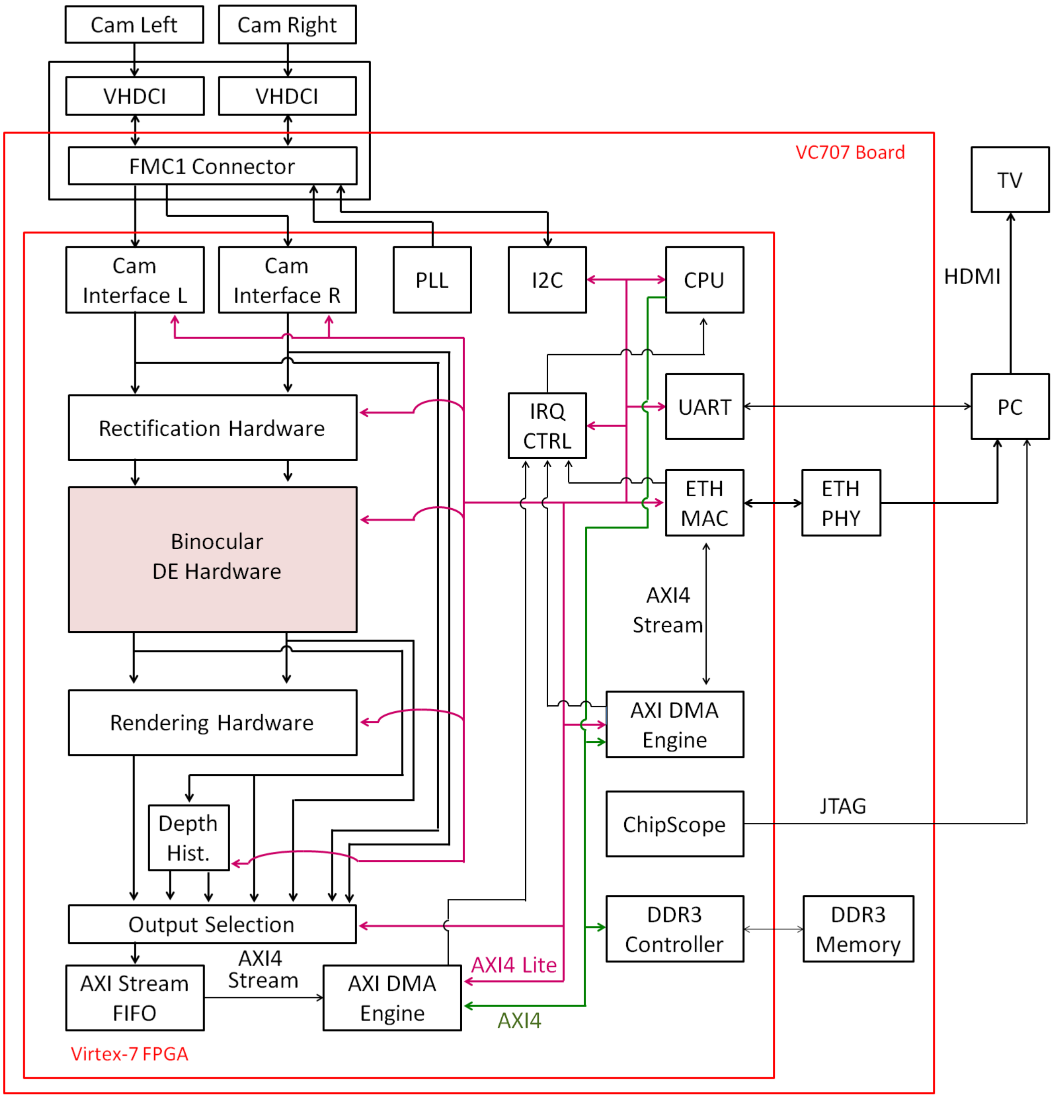


Figure 8.1: Depth estimation system that is utilizing two cameras

The ASIC implementation of the DE hardware dictates some additional constraints to the DE hardware verified on FPGA. DE hardware is an internal module in FPGA. The number of the pins is not a constraint for the DE hardware core of FPGA implementation, since the inputs and outputs of the DE module are internal signals of the complete system of the FPGA. However, the IOs of the DE module are the main pins of ASIC implementation, and a very high number of pin utilization can drastically increase the total chip area. The hardware resource consumption is an important but not a limiting restriction for FPGA implementation, since a Virtex-7 FPGA already provides a high amount of hardware resources. However, the low area implementation of ASIC may significantly reduce the cost of the chip. Therefore, hardware resources should be efficiently used for the ASIC implementation. The power consumption is not a significant constraint for the FPGA implementation of the DE hardware since the FPGA board already consumes high amount of static power due to the high amount of components on the board and the high static leakage of the LUT resources of the FPGA even in idle mode. Therefore, FPGA boards are typically used with power plugs. On the contrary, the power consumption of the ASIC can be significantly reduced by efficient utilization of the hardware resources, and it can be used in portable vision systems for which power consumption can be essentially important. Briefly, important design constraints have to be met for the ASIC design such as the chip area, the pin number and the power.

Three different versions of the ASIC are implemented considering three different specifications. The first one belongs to the ASIC design of the non-optimized FPGA implementation. The second one belongs to the optimized version of the first design which can be used together with any external forward mapping based rectification. The third one,i.e. final ASIC design, do not support forward mapping based rectification, but it can support external backward mapping based rectification and the implementation of trinocular DE using two chips. This section presents the main characteristics of the final DE ASIC. The analyses of the different design choices in order to obtain an efficient ASIC are presented.

The top-level block diagram of the final version of the ASIC is presented in Fig. 8.2. The descriptions of Controller, RDA, RCM, ADS and IR modules are presented in Chapter 3. The modifications to the binocular DE hardware required for ASIC compatibility are presented in Section 8.2. The top-level inputs and outputs of the ASIC are visualized in Fig. 8.3.

The ASIC is designed to receive the input images directly from cameras or an external device that processes backward mapping based rectification. The ASIC includes its own camera interface module, but does not include the rectification hardware. The rectification hardware is not included in the ASIC implementation since it requires a high number of BRAMs (a BRAM can be named SRAM in an ASIC). The inputs of the ASIC can be configured as either a 10-bit Bayer or 24-bit RGB or 24-bit YCbCr format. The *hsync* and *vsync* synchronization signals should be sent to ASIC for the capture of the pixels. The implementation details of the configurability of input format selection are explained in Section 8.2.

24-bit synchronized RGB or YCbCr image pixel are provided as an output of the ASIC in addition to 8-bit disparity values. Therefore, the final version of the ASIC is designed to act as RGB+D or YCbCr+D camera. Usually, cameras provide their outputs as *pixel data*, *pixel clock*, *hsync*, and *vsync* signals. The ASIC implementation provides its outputs in the signal format of typical cameras to enable its straightforward integration to various already existing systems and applications. The DE hardware is constrained to operate at 200 MHz clock frequency to compute disparity results in real-time. The outputs of the ASIC are provided with the *pixel clock* of the input camera (which is typically much lower than 200 MHz) to allow easy and low-noise capture of its RGB+D or YCbCr+D results by the external device.

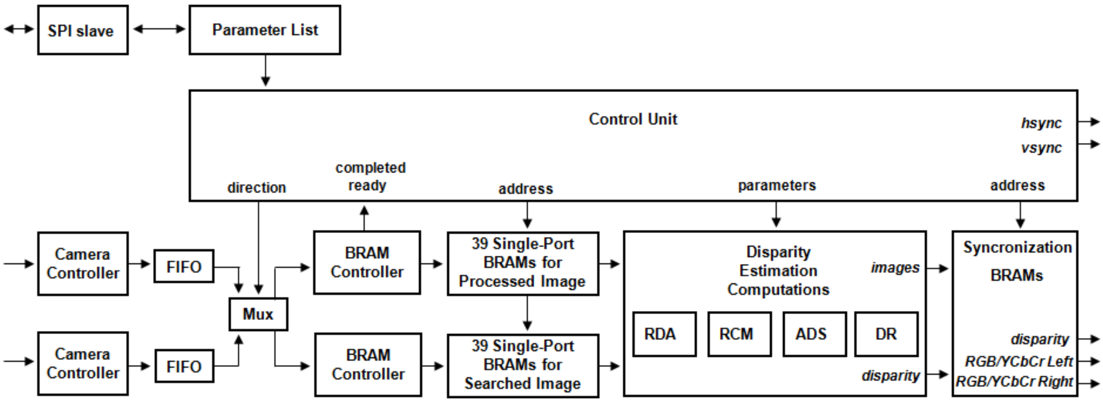


Figure 8.2: Modules present in the Disparity Estimation ASIC

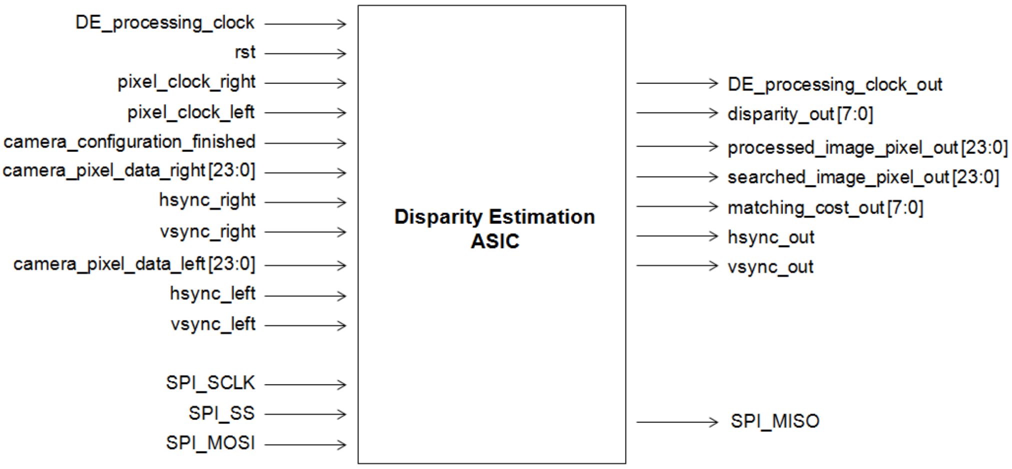


Figure 8.3: top level inputs and outputs of the disparity estimation final ASIC.

One of the most important constraints for the implementation of the ASIC relates to the utilization of BRAMs. The BRAM of the Xilinx FPGA contains 1024 addresses. 36-bit data can be stored in each address of these BRAMs. These BRAMs can be configured as a single-port BRAM or dual-port BRAM. Therefore, the utilization amount of BRAMs does not change depending on the selection of single-port or dual-port BRAM. The utilization of dual-port BRAMs is more convenient for most of the video processing applications, since these BRAMs allow to read and write concurrently. However, the selection of the dual-port or single-port BRAMs, significantly affects the total area of the ASIC implementation. The ASIC area comparison of BRAMs for the single and dual-port BRAMs is presented in Table 8.1. Dual-port BRAMs approximately consume two times the area of single-port BRAM. Therefore, single-port BRAMs should be

Table 8.1: ARM SRAMs Area Configuration Comparison (in µm^2) .

Configuration	**Single-port**	**True Dual-Port**
Bitcel	0.299	0.589
24 bits × 1024	24239	62998
8 bits × 1024	10012	21846

Table 8.2: Approximate power consumption for the different SRAM types of the ASIC design (in mW).

SRAM type	Internal power	Switching power	Leakage power	Total power
SP 24-bits × 1024	2.231	0.016	0.178	2.426
DP 8-bits × 1024	0.784	0.009	0.416	1.210
DP 24-bits × 1024	1.473	0.023	0.927	2.424

used as most as possible to reduce the area of the ASIC. An additional restriction about the BRAMs utilization stems from high power consumptions of the SRAMs. The approximate static power dissipated by one SRAM block for three different types are presented in Table 8.2. The power analysis is made for a case where an activity factor is 0.4, clock frequency of the DE module is 200 MHz, and the operating voltage is 0.8 V. A significant power consumption difference between single-port and dual-port 24-bit × 1024 memories is not observed. However both 2.426mW and 2.424mW are high power consumption values. Therefore, the hardware should be designed to utilize a small number of BRAMs to reduce the power consumption. The adaptation of the DE hardware to the ASIC implementation considering the BRAM type selection and the reduction of the number of BRAMs is presented in Section 8.2.

The hardware includes a programmable register list to be configured by the user. These registers allows user to configure the ASIC according to targeted applications. The ASIC chip includes SPI-slave interface to be programmed by an external device. The parameters, their default values and addresses are presented in Table 8.3. The description of the parameters are presented in Table 9.4.

Table 8.3: Register list and default values.
1 = always 1; 0 = always 0; d = programmable; – = read only

Register #	Register Description	Data Format (binary)	Default Value (decimal)
0	Chip Version	–	1
1	Row Size	–	767
2	Column Size	–	1023
3	Disparity Range	0000 0000 dddd dddd	255
4	Disparity Start	0000 0000 dddd dddd	0
5	Disparity End	0000 0000 dddd dddd	255
6	Disparity Estimation Start Column	0000 00dd dddd dddd	173
7	Disparity Estimation End Column	0000 00dd dddd dddd	949
8	Disparity Display Start Column	0000 00dd dddd dddd	173
9	Disparity Display End Column	0000 00dd dddd dddd	949
10	Disparity Refinement Level	0000 0000 0000 dddd	7
11	Parameter $tr_{7\times7}$	0000 0000 0000 0ddd	5
12	Parameter $tr_{13\times13}$	0000 0000 0000 0ddd	2
13	Parameter $ap_{7\times7}$	0000 0000 0000 0ddd	5
14	Parameter $ap_{13\times13}$	0000 0000 0000 0ddd	4
15	Parameter $ap_{25\times25}$	0000 0000 0000 0ddd	2
16	Threshold w	0000 0000 dddd dddd	8
17	YCbCr	0000 0000 0000 00dd	0
18	Total Block Cycles	–	118
19	Input Pixel Format	0000 0000 0000 00dd	0
20	Output Pixel Format	0000 0000 0000 000d	0
21	Search Direction	0000 0000 0000 000d	0
22	Searched and Processed Images	0000 0000 0000 000d	0
23	System Pause	0000 0000 0000 000d	0
24	Reserved	–	0
25	Reserved	–	0
26	Reserved	–	0
27	Reserved	–	0
28	Reserved	–	0
29	Reserved	–	0
30	Reserved	–	0
31	Reserved	–	0

Table 8.4: Core Registers descriptions

Regs. #	**Bits**	**Default**	**Name**
R0	2:0	1	*Chip Version* (RO)
	Version of the current hardware.		
R1	9:0	767	*Row Size* (RO)
	Number of rows		
R2	9:0	1023	*Column Size* (RO)
	Number of columns		
R3	7:0	127	*Disparity Range* (RW)
	Computed disparity range. Must be in the interval $[92;255]$.		
R4	7:0	0	*Disparity Start* (RW)
	The disparity estimation algorithm rejects disparity candidates found below this value.		
R5	7:0	127	*Disparity End* (RW)
	The disparity estimation algorithm rejects disparity candidates found above this value.		
R6	9:0	173	*Disparity Estimation Start Column* (RW)
	Column at which the DE starts to compute a disparity value.		
R7	9:0	949	*Disparity Estimation End Column* (RW)
	Column at which the DE ends to compute a disparity value.		
R8	9:0	173	*Disparity Display Start Column* (RW)
	Column at which the DE starts to display the computed disparity values. Below it, returned values are zero.		
R9	9:0	949	*Disparity Display End Column* (RW)
	Column at which the DE ends to display the computed disparity values. Above it, returned values are zero.		
R10	2:0	3	*Disparity Refinement Level* (RW)
	Disparity Refinement level, not yet implemented		
R11	2:0	5	*Parameter* $tr_{7\times7}$ (RW)
	Parameter $tr_{7\times7}$. See Chapter 3, equation 3.2.		
R12	2:0	2	*Parameter* $tr_{13\times13}$ (RW)
	Parameter $tr_{13\times13}$. See Chapter 3, equation 3.2.		
R13	2:0	5	*Parameter* $ap_{7\times7}$ (RW)
	Parameter $ap_{7\times7} = 2^{\mathrm{R13}}$. See Chapter 3, equation 3.6.		

R14	2:0	4	*Parameter* $ap_{13\times13}$ (RW)
	Parameter $ap_{13\times13} = 2^{\mathrm{R14}}$. See Chapter 3, equation 3.6.		
R15	2:0	2	*Parameter* $ap_{25\times25}$ (RW)
	Parameter $ap_{25\times25} = 2^{\mathrm{R15}}$. See Chapter 3, equation 3.6.		
R16	7:0	8	$Threshold_w$ (RW)
	Parameter $Threshold_w$. See Chapter 3, equation 3.3.		
R17	1:0	0	*YCbCr* (RW)
	Select Y, Cb or Cr for DE computations. $Y = 0$, $Cb = 1$, $Cr = 2$.		
R18	7:0	0	*Total Block Cycles* (RO)
	The number of 7×7 blocks in a row that are used for disparity estimation calculation		
R19	1:0	0	*Input Pixel Format* (RW)
	Controls the input pixel color format $YCbCr = 0$, $RGB = 1$, $Bayer = 2$.		
R20	0:0	0	*Output Pixel Format* (RW)
	Controls the output pixel color format $YCbCr = 0$, $RGB = 1$.		
R21	0:0	0	*Search Direction* (RW)
	The search direction of processed pixel *Search the left side = 0, Search the right side = 1.*		
R22	0:0	0	*Searched and Processed Images* (RW)
	Determining Processed and Searched Images as either left or right images *Processed image is the left image=0, Processed image is the right image=1.*		
R23	0:0	0	*System Pause* (RW)
	System run and pause *Run=0, Pause=1.*		

8.2 Adaptation of Disparity Estimation Hardware for ASIC

Several modifications are applied to the binocular DE hardware presented in Chapter 3 considering the ASIC design specification presented in Section 8.1. In this Section, the modifications to the binocular DE hardware are presented.

DE hardware module implemented on FPGA receives row and column numbers together with the pixel values. Row and column numbers require 10-bit for each of these signals. Therefore 40-bits should be used for two cameras only to identify the location of the pixel. Since these 40-bits were internal signals inside the FPGA hardware, this process does not cause any additional cost for the FPGA implementation. However, this high number of pins may unnecessarily increase the total area of the ASIC since the pitch of the PADs, i.e. distance between the center of the two adjacent PADs, should be large enough to do not limit the capacity of wire bonding technology.

The ASIC implementation is designed to receive inputs either directly from cameras or from another device that realizes backward mapping based rectification. In both cases, input pixels should be regularly provided to ASIC together with *hsync* and *vsync* signals. A camera interface module is integrated with the DE hardware to allow utilization of *hsync* and *vsync* for pixel addressing instead of row and column values. Thereby, the required pin count of the DE module for pixel addressing is reduced from 40 to 4 pins for the ASIC implementation. The same procedure is also applied to the outputs of the ASIC. The RGB+D or YCbCr+D outputs of the ASIC are provided with *hsync* and *vsync* singals instead of row and column values to reduce the number of pins.

The rectification hardware of the DE system that was provided between camera interface and DE module is removed for the ASIC implementation. Removing rectification hardware is important to reduce the on-chip memory area of the ASIC. As presented in Chapter 5, the Caltech rectification hardware utilizes 128 BRAMs for the rectification of two camera images in parallel. E-CLUTR hardware utilizes 111 BRAMs for storing the look-up-tables. This amount of BRAM utilizations would significantly increase the total area of the chip. Therefore rectification hardware is not integrated in the ASIC design, still using an external rectification device is possible.

The camera interface module of the FPGA implementation only receives 10-bit Bayer pixels. However, an external device operating rectification may provide its outputs as either 24-bit YCbCr or 24-bit RGB. In order to support inputs from both a rectification device and a camera, the input pixel data width is set to 24-bits. The multiplexing scheme is implemented for the camera interface. If the cameras are carefully aligned, and thus, the rectification process is not mandatory, the user can program the ASIC to receive its input in Bayer format. In this case, the camera interface captures the least significant 10-bits of the 24-bit data channel as 10-bit Bayer pixels, counts row and column numbers using *hsync* and *vsync* signals, and applies Bayer to RGB and YCbCr conversions. If the inputs of the ASIC are programmed for RGB format, camera interface of ASIC applies necessary RGB to YCbCr conversion. This multiplexing scheme not only allows the utilization of an external rectification device, but also allows the utilization of a camera with RGB or YCbCr format.

After the capture of images by the camera interface, camera interface sends the YCbCr pixel data to the BRAM controller together with its respective row and column numbers. The BRAM controller module writes the pixels into the addresses of the target BRAMs. During the write process, the DE module needs to read back the pixels from multiple BRAMs in parallel. The DE module needs to read pixels from 31 BRAMs in parallel to achieve the high bandwidth requirement of the DE process. Concurrent read and write processes could be handled using a minimum of 31 dual-port BRAMs if an external memory is used with the system. However, at least 7 additional dual-port BRAMs are needed since the pixels are captured row by row, and the DE module processes 7 rows in parallel. Moreover, the number of dual-port BRAMs should be more than 38.

The final ASIC design utilizes 39 single-port BRAMs for buffering each input image. The DE module can work with 39 single-port BRAMs for each image, since it does not need to reach the data of the next 7 rows while it reads pixels from 31 rows. Although the buffering of 38 rows is enough for the operation of DE hardware, buffering one additional row is allowed to provide additional margin for a possible camera dis-synchronization problem.

The clock frequencies of the DE module and camera are different. Dual-port BRAMs can be read and written by two clocks with different frequencies. However, utilizing the same scheme is not possible for a single-port BRAM since it only includes one clock port. Two different solutions are tested to solve multiple clock domain problem. The first solution consists of using clock multiplexers at the clock port of the single-port BRAMs. During the write process of a BRAM, reading is not required due to the utilization of additional 8 buffer BRAMs. Therefore, the BRAM controller can multiplex the clock input of a BRAM during its write process. This solution efficiently works for both FPGA realization and ASIC simulations. However, driving 39×2=78 single-port BRAMs with two clocks and applying the dynamic multiplexing method causes significant challenge to meet timing constraints. Therefore, an alternate second solution is proposed.

In the second solution, the outputs of the Camera Interface are written in a FIFO using the *pixel clock*. This data is read back by the BRAM controller using the fast clock of the DE module. These FIFOs are visualized in Fig. 8.2. Since the clock of the DE module is much faster than *pixel clock*, the FIFO can not be full. Therefore, the FIFO is designed to include only 3 words.

Using the BRAMs of the ASIC instead of the BRAMs of the FPGA offers an additional advantage in terms of on-chip memory size by removing redundant memory utilization. BRAM of the FPGA has a fixed size of 1024 addresses and each address includes 36-bits. However, the size of the BRAMs of the ASIC can be arranged by the designer. Since the pixels and disparities are stored in the BRAMs of the FPGA with 24-bit or 8-bit formats, there were redundant memory utilizations. This redundancy of on-chip memory utilization is removed for the ASIC implementation by generating SRAMs only at required sizes.

The DE ASIC is designed to support trinocular DE using two ASICs. Binocular AWDE implementation of FPGA searches the pixels of the left image in the right image. It is not able to search the right image pixel in the left image, since this process needs to change the searching direction. Trinocular DE hardware that is presented in Chapter 4 supports searching center image pixels in opposite directions in the left and right images. However, its search direction is fixed (right to left) in the left image and fixed (left to right) in the right image. Instead, the DE ASIC is designed to switch between the images to be able to process DE for the left image or right image. This process is handled by applying two modifications on the binocular DE hardware. The address generation scheme of the controller is modified, and image multiplexers are inserted between the input FIFOs and BRAM controllers.

The processed and searched images can be switched by image multiplexers. In Chapter 3, the input buffer BRAMs are named as Left Image BRAMs and Right Image BRAMs. In Chapter 4, the input buffer BRAMs are named as Left Image BRAMs, Center Image BRAMs and Right Image BRAMs. However, the input BRAMs of the ASIC implementation are named as Processed Image BRAMs and Searched Image BRAMs, as presented in Fig. 8.2. The disparity values are computed for processed image by searching their pixels in candidate disparities of the search image. In the ASIC implementation, the target BRAMs of the input images can be switched by input image multiplexers. The input image multiplexer is controlled by the user using parameters of the ASIC. In addition, the search direction is also parameterized. If the user programs the ASIC to process the left image, the Controller utilizes increasing order of address for Searched BRAMs. If the user programs the ASIC to process right image, the controller utilizes decreasing order of address for Searched BRAMs.

Since the DE can be processed in any of the right and left image, trinocular DE can be implemented using two ASICs. In this case, the center image should be connected to two ASICs. Two ASICs should be programmed for searching opposite directions and to use opposite BRAM multiplexing behavior. The external device should be connected to two ASICs to obtain two disparity maps. For realizing efficient trinocular DE, not only two disparity maps but also minimum matching cost (C) values are needed. Therefore, the ASIC design also synchronizes 8-bit matching cost value together with the RGB+D values. Therefore, the final ASIC design acts as a RGB+D+C camera. The hybrid cost value of the binocular AWDE algorithm can be between 3072 to 0, for which 0 identies perfect match. Therefore, 12-bits are needed to define the matching cost. In order to decrease the pin count, the 7 least significant bit of the 12-bit cost are used as the 7 least significant of the 8-bit cost output. If the matching cost is higher than 127, the most significant bit of 8-bit C output is assigned as 1. Therefore, for the cost comparison, if the most significant bit of the 8-bit cost is 1, the pixel value can be considered as not-trustable. Otherwise, the disparity with smaller value obtained from any pair can be used in final trinocular disparity map. The switching ability of the input images is not only important for allowing trinocular DE, but also binocular DE results can be enhanced with a double-checking scheme using two ASICs. In this case, the left and right images should be processed in parallel using two ASICs.

Another modification of the DE hardware relates to the write operation into the register list of the parameters. In the FPGA implementation, the write operation into the register list does not need any specific IO protocol since addresses and data can be generated by internal parallel wires. However, parallel access to the 32-bit write data, 32-bit read data, 5-bit address and 1-bit enable would consume 70 additional pins of the ASIC. Therefore, a slave SPI interface is implemented in the ASIC. The SPI interface requires 4 pins for its Serial Clock (SCLK), Slave Select (SS), Master Out Slave In (MOSI) and Master In Slave Out (MISO) signals.

The adaptations for the ASIC implementation are verified in real-time using FPGA emulation. The possible utilization model of the ASIC in a complete system is presented in Section 8.4.

8.3 ASIC Design

This chapter covers the ASIC design of the DE hardware from the synthesis to the placement and routing steps. A CMOS TSMC 40nm technology is used to design the ASIC. Three different versions of the DE hardware are synthesized for the ASIC, each includes different capabilities. Important results obtained from these syntheses are presented. Then, the place and route process of these three synthesized hardwares are performed. Three different place and route version of the DE hardware are presented and compared. Finally, the possible utilization model of the ASIC in a complete system is presented.

8.3.1 Synthesis

The synthesis of the DE hardware is performed with the Design Vision and Design Compiler tools from Synopsis. The design constraints focus on the timings and area. The hardware is synthesized for a DE system frequency of 250 MHz and camera clock frequency of 70 MHz to provide sufficient margins enabling utilization with a wide range of systems and camera sensors.

The synthesis results of the netlists corresponding to three different configurations are reported on Table 8.5. As explained in Section 8.2, only dual-port memories were part of the system at the beginning of the project. The synthesis results using only dual-port BRAM are presented in 8.5 under the column Configuration-I. The Configuration-I utilizes 64 dual-port BRAMs for buffering input pixels of each cameras. During the development, the second configuration (Configuration-II) involving single-port memories has been implemented. The Configuration-II utilizes 64 single-port BRAMs for buffering input pixels of each camera. The Configuration-I and Configuration-II allow the ASIC to capture its inputs from backward or forward mapping based rectification, but not directly from cameras. Configuration-I support more challenging camera misalignment than Configuration-II for the forward rectification since it utilizes dual-port BRAMs. The Configuration-I and Configuration-II provide RGB+D outputs but they do not deliver synchronized matching costs. Moreover these two configurations do not allow switching between the left and right images to change the processed image for disparity computations.

The third configuration (Configuration-III) covers all the specifications that are presented in Section 8.1. The Configuration-III utilizes 39 single-port BRAMs for buffering input pixels of each camera. It allows to directly capture inputs from cameras or from backward mapping based rectification. Reducing the number of BRAMs from 64 to 39 is not only important to reduce the area of the ASIC, but also to reduce power consumption. Configuration-III provides synchronized cost values together with RGB+D or YCbCr+D outputs. Moreover, it allows switching between the left and right images to change the processed image for disparity computations. Therefore, Configuration-III supports trinocular DE system if two ASICs are used. As presented in Table 8.5, memory blocks consumes most of the area of the ASIC design. Configuration-III is the most efficient solution in terms of area since it utilizes less BRAM than the other configurations and benefits from single-port BRAMs.

These three configurations are implemented and synthesized. Their gate-level netlist and Standard Delay Format (SDF) files containing the timing delays are generated. The ASIC designs are verified with post-synthesis simulations using Modelsim. The place and route results of these three configurations are presented in the next section.

Table 8.5: Area report of the three synthesized hardware.

	Configuration I	Configuration II	Configuration III
Number of 8×1024 DP SRAMs	16	16	32
Number of 24×1024 DP SRAMs	160	32	32
Number of 24×1024 SP SRAMs	-	128	78
Number of ports	230	230	128
Number of nets	20989	20864	24919
Combinational area (mm^2)	0.196155	0.357315	0.358311
Buf/Inv area (mm^2)	0.019642	0.040513	0.030437
Noncombinational area (mm^2)	0.237605	0.697702	0.654184
Memory blocks area (mm^2)	10.429225	5.468099	4.605677
Total area (mm^2)	10.882627	6.563631	5.618173

8.3.2 Placement and Routing

The Place & Route of the synthesized gate-level netlists are processed by using the Encounter Digital Implementation (EDI) tool from Cadence Design Systems. TSMC 40nm CMOS technology is used for the design. Three different place & route designs are implemented, each includes one of the three configurations presented in Table 8.5. Post-P&R simulations are performed with Modelsim to verify the ASIC designs.

The most important decision for the place and route process relates to the placement of the SRAM blocks in the floorplan. SRAMs occupy the major area of the ASIC. Therefore, they must be carefully placed without creating timing or geometry violations.

First Design

The first design utilizes Configuration-I. The core is placed in the center and is surrounded by the SRAM blocks. This arrangement reduces the length of the wires that connect the memories. The memory blocks are grouped according to their respective function. For example, as can be see on Fig. 8.4, the 64 SRAMs that store the pixels from left image are placed together on the left of the core, and the 64 SRAMs that store the pixels from right image are placed together on the right of the core. This design scheme reduces the wire density since some of the nets of the grouped SRAMs are shared. Fig. 8.7a presents the designed ASIC. The dimensions of the floorplan of second design is 3804 × 3928 µm with a core size of 249 × 3542 µm with density of 50.045%. The total area is 14.94 mm^2.

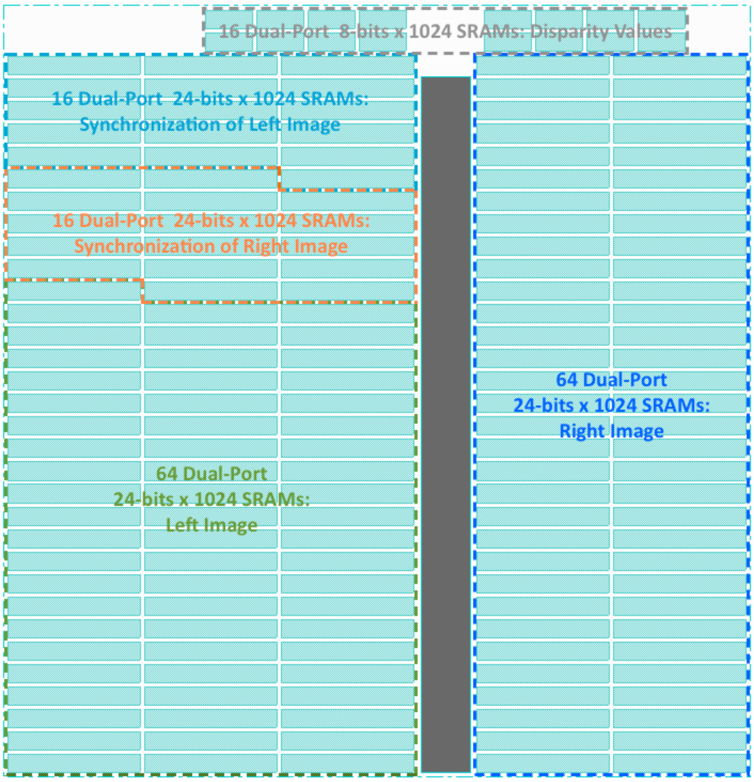

16 Dual-Port 8-bits x 1024 SRAMs: Disparity Values
16 Dual-Port 24-bits x 1024 SRAMs: Synchronization of Left Image
16 Dual-Port 24-bits x 1024 SRAMs: Synchronization of Right Image
64 Dual-Port 24-bits x 1024 SRAMs: Right Image
64 Dual-Port 24-bits x 1024 SRAMs: Left Image

Figure 8.4: Floorplan of the first design.

Second Design

The second design utilizes Configuration-II. It presents a smaller die area than the first design since the 128 dual-port 24-bits SRAMs are replaced with two times smaller single-port SRAMs. In addition, a second metal layer is added to the power rings of the second design to have a better margin for the current flow. The floorplan arrangement is nearly the same as the first design with a central core as presented in 8.5. The dimensions of the floorplan are 3304 × 2990 µm and the core is 399 × 2936 µm with a density of 90.876%. The total area of the second design is 9.883 mm^2. Therefore the total area is reduced by 33.8% compared to the first design.

Third Design

The third, i.e. final, design utilizes Configuration-III. The design realizes all the specifications presented in Section 8.1. The dimensions of the floorplan are 3000 × 2659 µm and the core is 456 × 2605 µm with a density of 94.01%. The total area is 7.977 mm^2. It presents a smaller die area than the second design since 39 BRAMs are used instead 64 BRAMs for buffering pixels of each camera. Therefore, the total area is reduced by 19.2% compared to the second design, and by 46.6% compared to the first design. Further memory size reduction is possible if the ASIC is designed to capture and store only Y data instead of YCbCr, and to output only

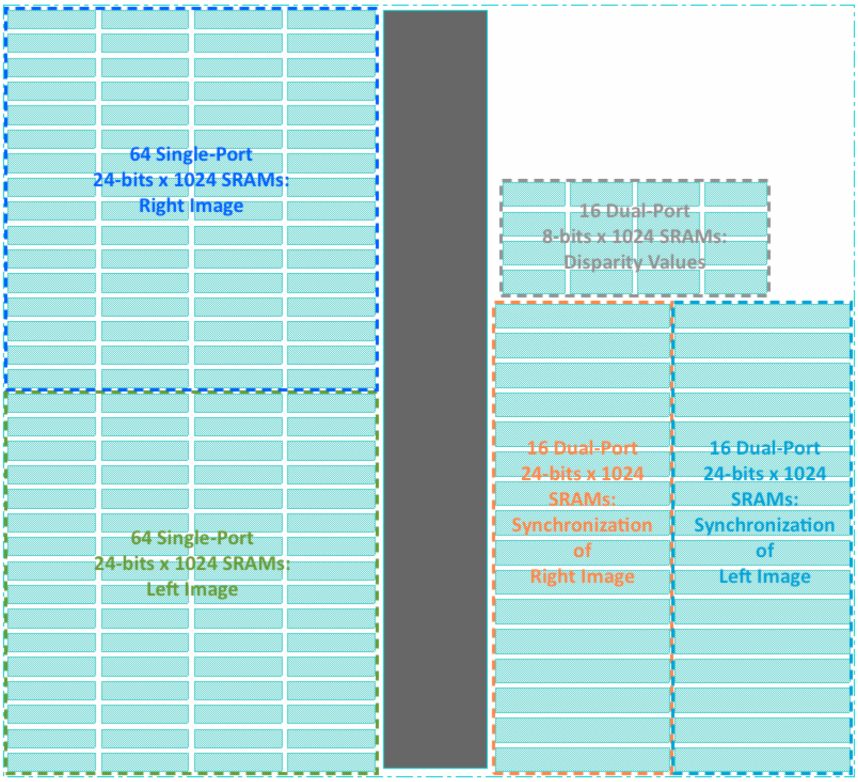


Figure 8.5: Floorplan of the second design.

8-bit disparity results but not the synchronized RGB values. However, only providing disparity value as an output may limit the ASIC to be used in RGB+D based applications.

This design supports a clock frequency up to 200 MHz clock. This clock frequency maintains a 63 fps performance for 1024×768 resolution DE for a 128 pixels disparity range, which is slightly higher than the maximum possible performance of the FPGA implementation. Expectedly, ASIC implementations can target faster performances compared to FPGA implementation, but it should be noted that the Virtex-7 FPGA utilizes a 28nm technology whereas the ASIC is designed using a 40nm technology.

The total power dissipated by the ASIC chip of the second and third designs are compared in Table 8.6. The power analysis is made using the Power Analysis tool of Encounter considering an activity factor is 0.4 and an utilized clock frequency of the DE module at 200 MHz. 0.81 V supply is used for the three power rings VDDCE, VDDPE and VDD.

As presented in Table 8.6, the second and third designs consume 647 mW and 557 mW of power, respectively. The largest amount of power is consumed by the SRAM blocks. According to the number of the utilized BRAMs given in Configuration II in Table 8.5 and the power

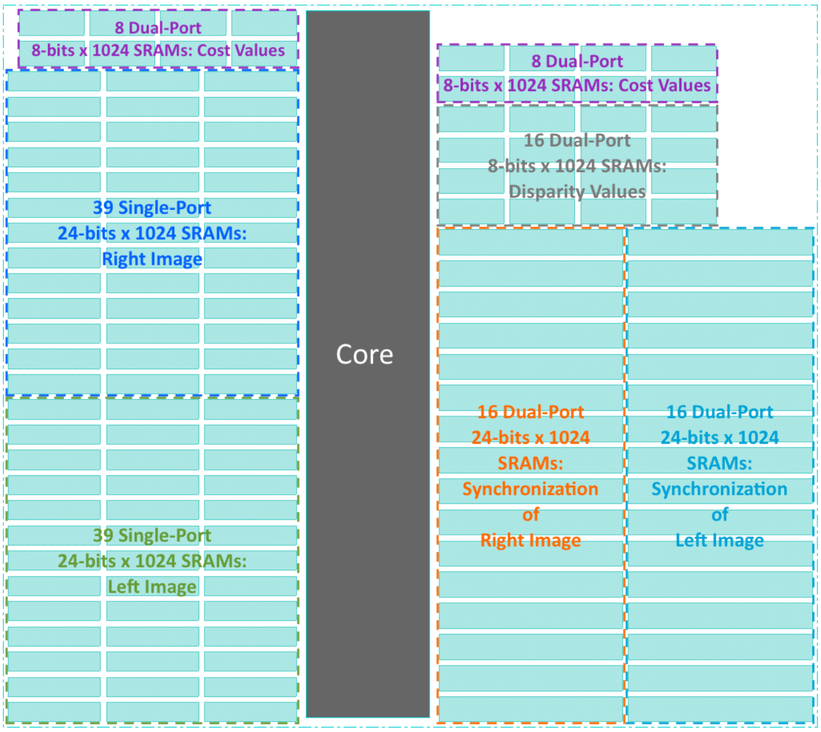


Figure 8.6: Floorplan of the third design.

consumption of SRAM blocks in Table 8.2, the second design consumes 408 mW of power, only for SRAMs. The third design utilizes Configuration III, therefore it approximately consumes 306 mW of power, only for SRAMs. Therefore, the power consumption of the second design is reduced by decreasing the number of BRAMs, although some new functionalities are included into the third design such as the support of configurable search directions and enabling direct connection to the camera.

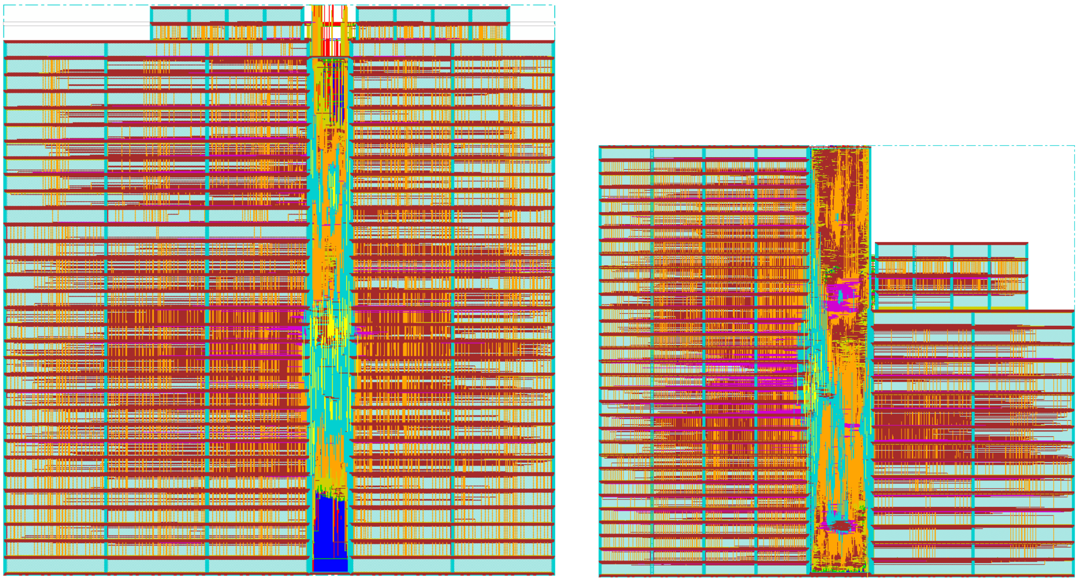
Table 8.6: Total power of the DE ASIC (in mW).

P&R	**Total Internal Power**	**Total Switching Power**	**Total Leakage Power**	**Total Power**
Second Design	463.8 71.7%	112.1 17.3%	70.8 19.9%	646.5 100%
Third Design	388.9 69.8%	99.7 17.9%	68.1 12.2%	556.7 100%

Figure 8.7 presents the three ASIC designs and Table 8.7 summarizes their respective sizes and areas. The third design is more efficient in terms of area. Fig. 8.8 presents the ASIC after placing the IO pads at the periphery of the floorplan. In addition, filler cells are added to fill the remaining holes of the die. The dimensions of the final ASIC after placing pads are 3450 × 3080 µm. The total area is 10.626 mm^2.

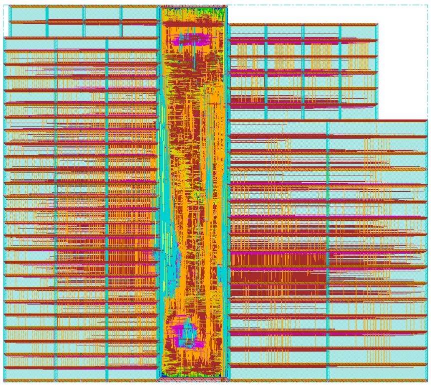
Table 8.7: Dimensions of ASIC designs

	First Design	Second Design	Third Design
Floorplan (µm)	3804 × 3928	3304 × 2990	3000 × 2659
Floorplan Area (mm^2)	14.94	9.883	7.977
Core (µm)	249 × 3542	399 × 2936	406 × 2605



(a) First design

(b) Second design



(c) Third design

Figure 8.7: The three disparity estimation ASIC designs after P&R. For visual comparison, the proportions have been kept. (a)First design (b)Second design (c) Third design

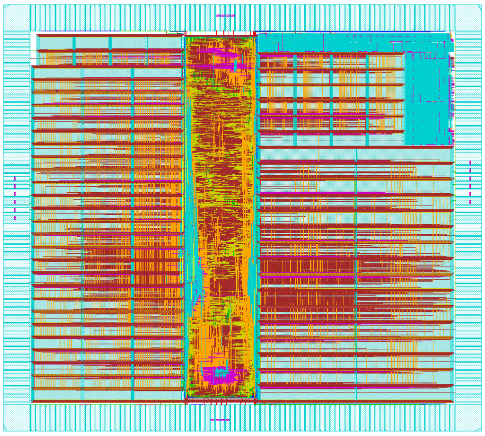


Figure 8.8: The final ASIC design after P&R and placing the IO pads

8.4 Test System

After the manufacturing of the ASIC, post-fabrication verification should be performed. In order to verify the behavior of the system, a test environment is proposed. This section presents the functions of this system.

Figure 8.9 shows the proposed complete system for the verification of ASIC. The proposed test system utilizes VC707 evaluation board. Another board with much less hardware resource can be used for the test environment, but changing the FPGA type may require additional design time. After the fabrication of the ASIC, it will be placed on a test-board. This board will be connected to the FMC connector of the FPGA. The ASIC and FPGA will be communicated through the FMC connector. The hardware on FPGA will transfer either the rectified image pixels to the ASIC or it will by-pass the camera signals. The ASIC will be configured through the SPI interface. The final RGB+D results of the ASIC will be captured by FPGA with a ASIC Interface module. This module is similar to the camera interface module, since the ASIC provides its outputs with the format of the camera. The outputs of the ASIC will be transferred to the PC using ethernet. The high resolution disparity estimation ASIC will remove the hardware complexity of DE from the FPGA based complete system. Consequently, the implemented ASIC may allow the development of cheap depth estimation systems.

The emulation of the proposed system is presented in Fig. 8.10. In this setup, two FPGAs are connected to each other using FMC to FMC cable. The FPGA above emulates the ASIC implementation which involves DE hardware. The FPGA below involves microblaze processor, and SPI master, Ethernet and DDR3 interfaces. The FPGA below is connected to the cameras, which bypass the pixels to the ASIC emulator and controls the register set of DE hardware using SPI interface. ASIC emulator provides RGB+D results to FPGA below. FPGA below transmits the RGB+D results to PC using Ethernet. The emulation set-up verifies the standalone functionality of the DE hardware. The FPGA above will be replaced with the ASIC implementation as presented in Fig. 8.9.

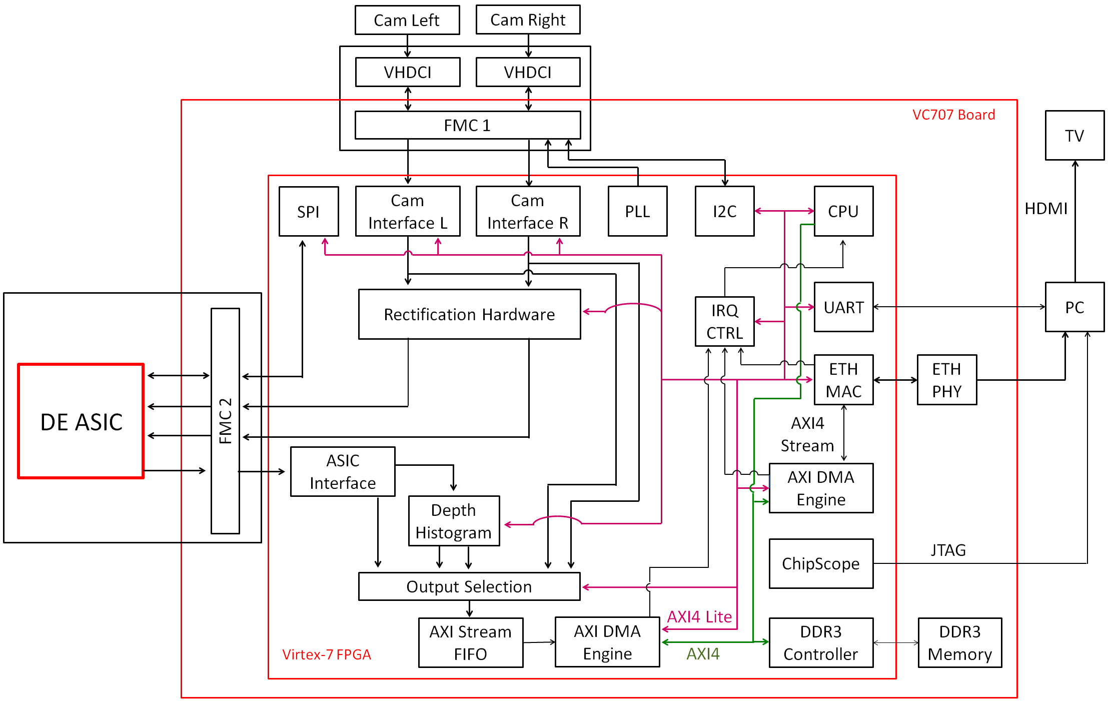


Figure 8.9: Top-level diagram of the test system of ASIC. One VC707 board is connected to the test-board containing the ASIC package.

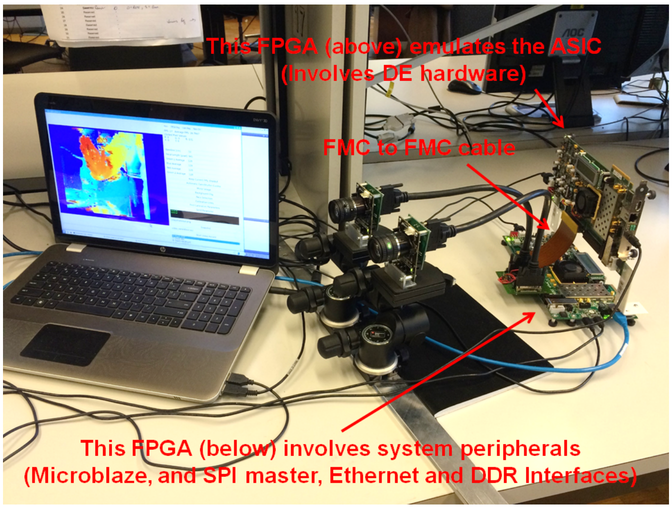

This FPGA (above) emulates the ASIC (involves DE hardware)
FMC to FMC cable
This FPGA (below) involves system peripherals (Microblaze, and SPI master, Ethernet and DDR Interfaces)

Figure 8.10: The emulation and verification set-up of the ASIC.

9 High-Quality Omnidirectional Multi-Camera Systems

This chapter presents two implementations that are proposed to improve the existing real-time 360° × 90° omnidirectional cameras. First, the equal area distribution method is presented [57]. This method provides homogeneous resolution over the entire reconstruction area. The equal area distribution method is implemented in the real-time system presented in [19] to improve the realistic aspects of its omnidirectional video outputs. Still, the omnidirectional video quality of this system is limited by the 352 × 288 resolution of the image sensors. Therefore, secondly, a new system is implemented using high-resolution cameras [58, 59]. An ultra high resolution omnidirectional video record system called Giga-Eye is implemented to generate very high quality 360° video with an off-line processing. In this system, 44 5MP Cameras are used to generate 21.6 MP omnidirectional video at 30 fps and 81.3 MP omnidirectional video at 9.5 fps.

9.1 Enhanced Omnidirectional Image Reconstruction Algorithm and Its Real-Time Hardware Implementation

The equal area distribution method and its hardware implementation are proposed to enhance realistic aspects of omnidirectional camera. In this section, the details of the proposed enhanced omnidirectional image reconstruction algorithm (EOIR) that utilizes the equal area distribution method, its flexible real-time hardware architecture, and its implementation results are presented.

9.1.1 Equal Area Distribution Algorithm

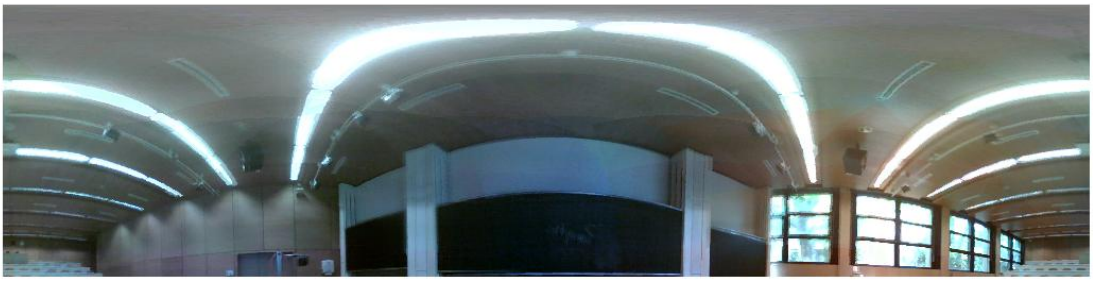
The OIR algorithm [19] presented in Chapter 2 provides more importance to top view than the side cameras in the constructed omnidirectional image. This unequal partitioning causes slightly unrealistic omnidirectional images. The picture in Fig. 9.1a is captured from the omnidirectional camera presented in [19] which utilizes OIR algorithm. In addition, images captured by the top camera and one of the side cameras are shown in Fig. 9.2. As observed in Fig. 9.1a where linear pixelization is applied as the pixelization method, the details on the sides of the image, exemplified by desks located on the right side, occupy less area compared to the contribution originating from the top camera, considering ceiling and lamps in this case. Conversely, since the human eye naturally observes along the horizontal axis naturally fallowed by the majority of conventional imaging applications, data located in the horizontal plane is more important than data issued from the top plane.

The 'On hemisphere' line in Table 9.1 indicates the ratio of the covered surface area over total hemispherical area for various θ angles, assuming that the surrounding space is observed from the center of the hemisphere in Fig. 2.5a. The other entries in the table present an elliptical distribution (an alternate way of pixelization method based on elliptical surface area distribution), and the existing linear distribution method of OIR. As expected due to the basic 3D geometry, the Equal Area Distribution of EOIR results completely match with the On Hemisphere calculations, resulting in equal resolution distribution on the whole hemispherical surface. In contrast, when the linear pixelization proposed in [19] (Linear Distribution in Table 9.1) is applied, the resolution of the reconstructed image decreases from the top to bottom, and the contribution from top cameras is overestimated, while data from side cameras is undervalued. The picture taken from top cameras utilize more pixels on the omnidirectional image than the side view cameras as observed in Fig. 9.1a and Fig. 9.2 which results in providing less resolution for side cameras than the top cameras. Therefore, a novel pixelization method is proposed based on equal area distribution. The anticipated method allocates N_θ latitude pixels with respect to the surface area of hemispheres covered at particular θ angles. An example is presented in Fig. 9.3, where the total surface area considered at $\theta = \pi/6$ is shown by spherical cap S. The contribution at each θ angle is calculated using equations (9.1) and (9.2), derived using 3D geometry. In contrast to linear pixelization, this method results in varying latitude pixel steps, which produces a non-linear relation between the N_θ and θ angles, which is presented in Fig. 9.4.

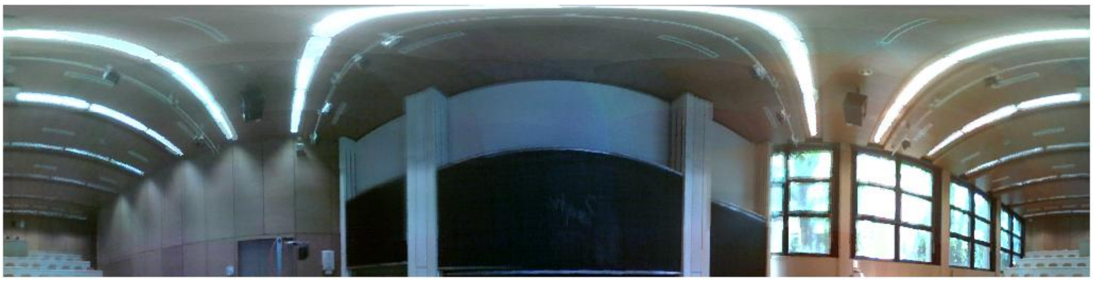
Table 9.1: Surface Area Coverage (%) with respect to θ angles (radians) for Possible Pixelization Methods.

	$\pi/2$	$\pi/3$	$\pi/4$	$\pi/6$	$\pi/12$
On hemisphere (reference)	100	50	29	13	3.4
Equal Area Distribution	100	50	29	13	3.4
Elliptical Distribution	100	25	13	6	2
Linear Distribution	100	66	50	33	17

$$\frac{\int_0^{2\pi}\int_0^{\theta} r^2 \sin\theta \, d\theta \, d\varphi}{2\pi r^2} = \frac{N_\theta}{256} \tag{9.1}$$

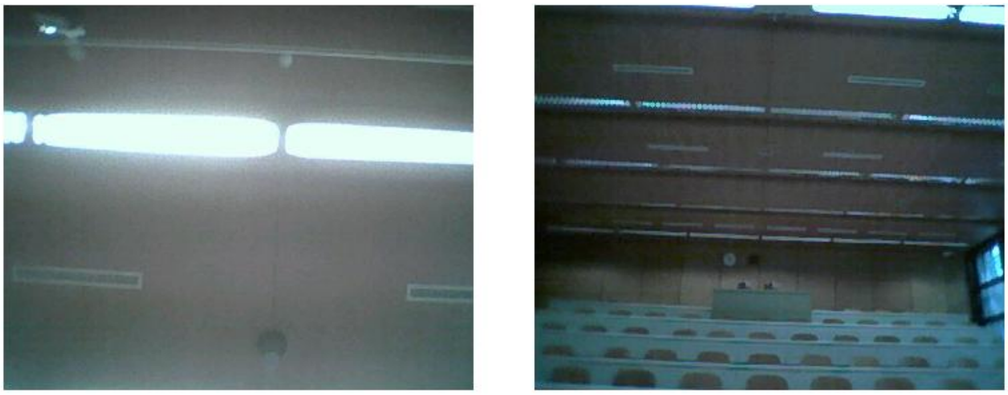


(a)



(b)

Figure 9.1: 256 × 1024 resolution omnidirectional 2D reconstructions using a) *Linear pixelization* b) *Equal area distribution*



(a) (b)

Figure 9.2: Captured images from two different cameras a) top camera b) side camera

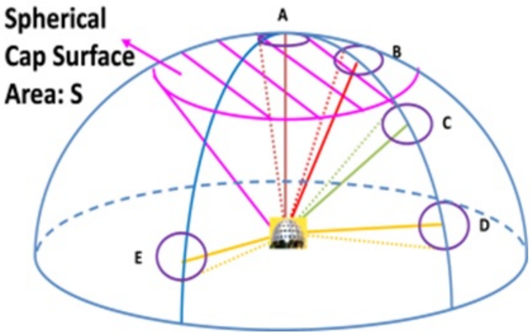


Figure 9.3: Hemispheric structure for resolution calculation in terms of ppaa

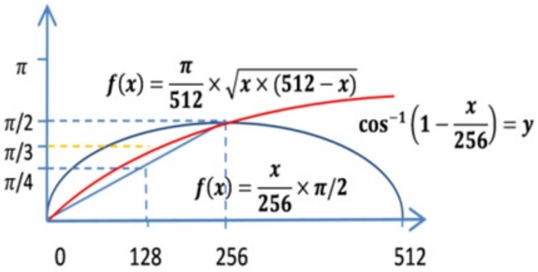


Figure 9.4: Relationship between N and - angle for different methods of pixelization

$$\arccos\left(1 - \frac{N_\theta}{256}\right) = \theta \tag{9.2}$$

The universally accepted unit of measurement of 2D image resolution is line pairs per millimeter (*lp*). Since all the longitudes merge on the pole of hemisphere, this unit is not an appropriate measurement for defining hemispherical image resolution. Therefore, we provide a new suitable unit of measurement which is the Pixels Per unit Angular Area (*ppaa*) metric. The unit angular area is defined in this context as the surface area of a spherical cap with exactly 1° θ angle which provides constant surface area on hemisphere due to the fixed radius for all possible θ and φ. Resolutions at different θ angles which are presented in Fig.

Table 9.2: Resolution values [$ppaa$] for different θ angles

θ Angle	**Corresponding Unit Angular Area**	**Linear**	**Equal Area Distribution**
0°	A	2913	39.92
10°	B	146.3	39.92
39.6390°	C	39.92	39.92
80°	D	26.81	39.92
80°	E	26.81	39.92

9.3 are compared in Table 9.2 in terms of their *ppaa*. When the linear pixelization scheme is adopted, small θ angles (top view) yield a very high resolution such as 2913 *ppaa* for $\theta = 0°$, gradually decreasing for higher θ angles. For instance, $\theta = 80°$ produces a *ppaa* value 26.81, which is approximately 100 times smaller than *ppaa* of $\theta = 0°$. Since the human eye captures constant resolution while looking at different angles, linear pixelization does not provide realistic enough hemispherical images. On the other hand, the equal area pixelization scheme provides a homogeneous resolution distribution over the entire omnidirectional image by distributing 39.92 *ppaa*. Hence, as shown in Fig. 9.1b, the equal area distribution scheme realizes realistic pictures by homogenously distributing equal number of pixels to the top view and side view cameras.

Equal Area Distribution is not only important for the visual quality but also it is significant for interconnected network of multiple cameras and FPGAs while generating real-time panorama. The EOIR algorithm equalizes the usage of the pixels of the cameras for the panorama reconstruction. Therefore, it equally distributes the network traffic between multiple FPGAs. This advantage of EOIR is proved by omnidirectional camera system presented in [83] which utilizes 49 cameras and 7 FPGAs. [83] benefits from the EOIR algorithm to equalize network traffic between 7 FPGAs. In this thesis, the hardware implementation details of EOIR is only explained for the system presented in [19]. The benefit of EOIR for the interconnected network of FPGAs for the panorama construction is detailed in [83].

The block-diagrams of the OIR and EOIR algorithms are shown in Fig. 9.5. The main computationally intensive part of the OIR and EOIR algorithms is the projection computations, i.e. pixel position generation. This module is same for both OIR and EOIR. Since the algorithmic enhancements are on θ angle generation and interpolation, the proposed enhancements are easily adapted to the hardware presented in [19]. The final implementation allows user to switch between OIR and EOIR.

9.1.2 Hardware Implementation of Equal Area Distribution Algorithm

The hardware platform which has been presented in [19] is used to implement the EOIR algorithm in real-time. Since the proposed EOIR algorithm is a hardware-oriented algorithm, its adaptation into the system presented in [19] is straightforward. As shown in Fig. 2.3, the platform consists of two concentrator FPGAs handling local image capturing with 20 imagers in parallel, a central FPGA to control concentrator FPGAs and to transmit final omnidirectional images to the PC, and four 40MB SRAM memories buffering captured images. Each single camera provides 352 × 288 resolution images at 25 fps. As shown in Fig. 9.6, the main processing of the algorithm is handled in the Concentrator FPGA and it involves six sub-blocks. The details of the Cameras Control, Camera Input Channels, Data Transmit Multiplexer, Data and Control Unit, Memory Controller modules and efficient memory organization are presented in [19]. In this subsection, the hardware implementation of the EOIR algorithm is presented for the Image Processing and Application Unit of the

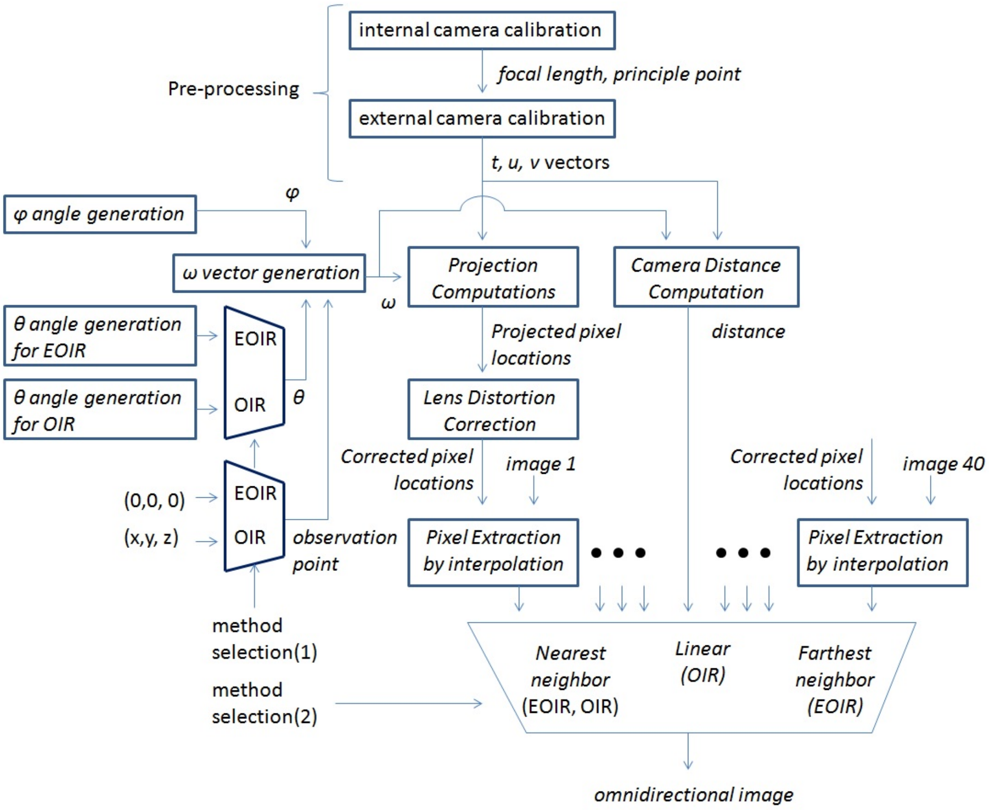

Pre-processing
internal camera calibration
focal length, principle point
external camera calibration
t, u, v vectors
φ
φ angle generation
ω vector generation
ω
Projection Computations
Camera Distance Computation
θ angle generation for EOIR
θ angle generation for OIR
EOIR
OIR
θ
Projected pixel locations
distance
Lens Distortion Correction
Corrected pixel locations
image 1
Corrected pixel locations
image 40
(0,0, 0)
(x,y, z)
EOIR
OIR
observation point
Pixel Extraction by interpolation
Pixel Extraction by interpolation
method selection(1)
method selection(2)
Nearest neighbor (EOIR, OIR)
Linear (OIR)
Farthest neighbor (EOIR)
omnidirectional image

Figure 9.5: Block diagram of omnidirectional image reconstruction algorithms

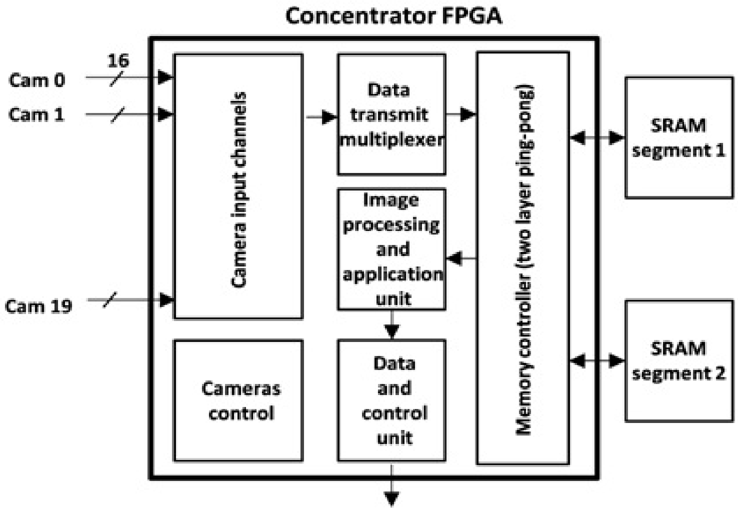


Figure 9.6: Block diagram of the Concentrator FPGA

concentrator FPGA.

The proposed hardware implementation of the omnidirectional image reconstruction is flexible to realize the OIR and EOIR algorithms with configurable resolutions, AOV and camera selection options. Options are selectable by the embedded soft controller (microblaze processor) and the GUI.

The system-level architecture of the EOIR algorithm consists of five sub-blocks depicted in Fig. 9.7. The angle generation module shown in Fig. 9.8 creates the spherical coordinates of pixel direction ω=(θ_ω, φ_ω) using two accumulators. To generate all possible combinations of θ_ω and φ_ω, one accumulator is assigned to increment, while the other completes its full-range cycle. Here, θ and φ have separate linear incrementing indexes K_θ and K_φ as well as minimum and maximum values defined for both angles, which are all parameters supplied to the reconstruction system. Defining initial and final value parameters of θ and φ provides flexibility to visualize either the whole reconstruction of 360° × 90° image or some portion of the vision of Panoptic camera. The parameterized incrementing indexes K_θ and K_φ define the step amount, thereby the size of omnidirectional image.

A 13-bit look-up-table in Fig. 9.8 is used for implementing equation (9.2) in order to add the new pixelization scheme based on the equal resolution distribution. Arccos operation and divisions for θ processing are pre-computed for all different values of N_θ and placed in a lookup table (LUT) to save the hardware resources.

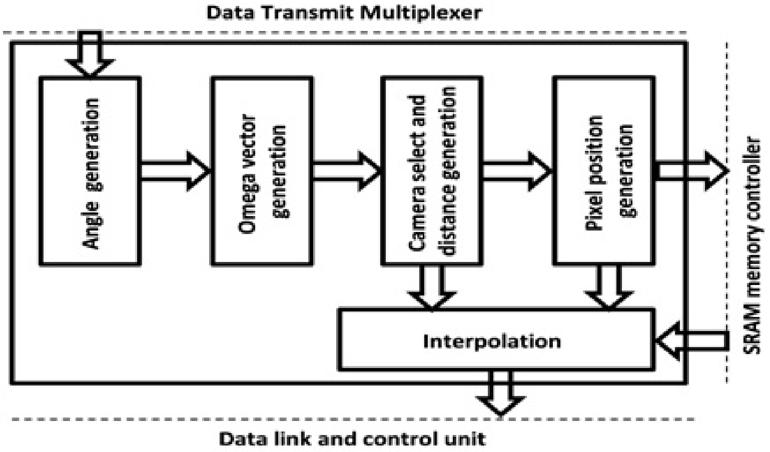

Data Transmit Multiplexer
Angle generation
Omega vector generation
Camera select and distance generation
Pixel position generation
Interpolation
SRAM memory controller
Data link and control unit

Figure 9.7: Block diagram of the omnidirectional vision reconstruction unit inside the image processing and application block

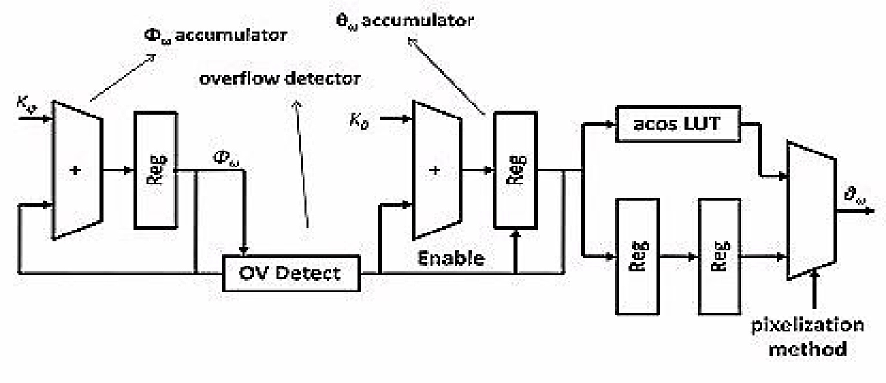


Figure 9.8: θ and φ angle generation module architecture for the OIR and EOIR implementations

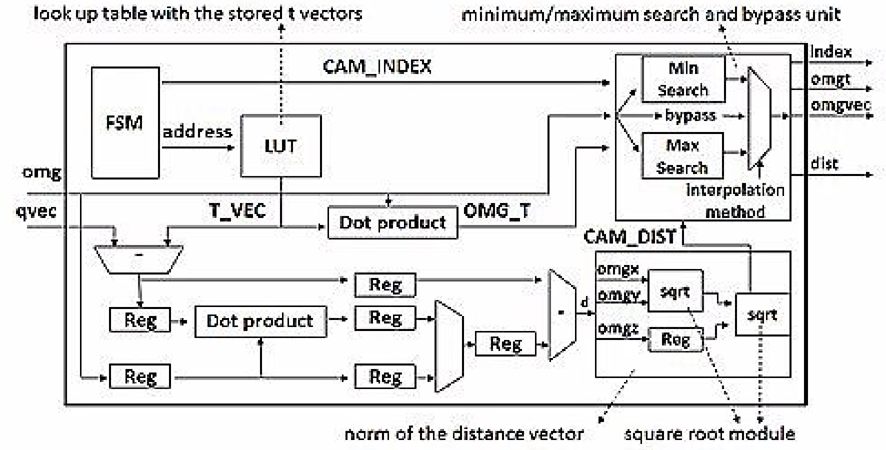


Figure 9.9: Architecture of the camera select and distance generation sub-block for the EOIR algorithm

In order to supply the user with the ability to switch between the OIR and EOIR algorithms in real-time processing, the output of the OIR algorithm is delayed by two clock cycles using two pipelined registers as shown in Fig. 9.8. This additional pipelining has no adverse effect on the throughput of the OIR algorithm.

Following the computation of θ_ω and φ_ω, the ω vector is generated in the next sub-block named *ω vector generation* in Fig. 9.7. This sub-block is used as it is presented in [19]. The *camera select and distance generation* module in Fig. 9.6 identifies cameras that contribute into pixel direction ω. This block is also in charge of calculating weights for interpolation. If the linear interpolation method is chosen, the weights of contributing cameras and their corresponding indexes are sent to the interpolation sub-block. On the other hand, if the nearest neighbor scheme is chosen, a maximum search of the weights of the contributing cameras is carried out, and the index of the camera with the maximum weight is passed to the pixel position generation sub-block, as depicted in Fig. 9.9.

The algorithmic studies in [84] presents that distances to objects can be processed from omnidirectional images of the scene captured from distinct reference centers of panoptic camera by processing the residual of these omnidirectional images. The farthest neighbor interpolation method is proposed to provide significant amount of residual compared to nearest neighbor interpolation for the possible 360° depth map estimation without changing reference center. A maximum-distance search algorithm is supplied to the omnidirectional image reconstruction hardware for the implementation of the farthest neighbor interpolation method, in order to retrieve the light intensity value from the farthest camera. This operation is handled as the exact inverse operation of the nearest neighbor method that is presented above. The block-diagram of minimum-maximum search and bypass unit is also shown in Fig. 9.9. Effective residual generation results obtained using this method is presented in [57], but the depth map generation using this residual has not been studied. The algorithm and hardware implementation of 360° depth map estimation using this method can be studied in future.

After determining which cameras correspond to the projection of each ω vector, the pixel position module retrieves the contributing pixel value on the image frame of the contributing cameras from the SRAM. Subsequently, this retrieved value is sent to the interpolation sub-block in order to implement linear interpolation using the pixels received from the contributing cameras. Finally, this sub-block supplies the calculated intensity value to the data link and control unit.

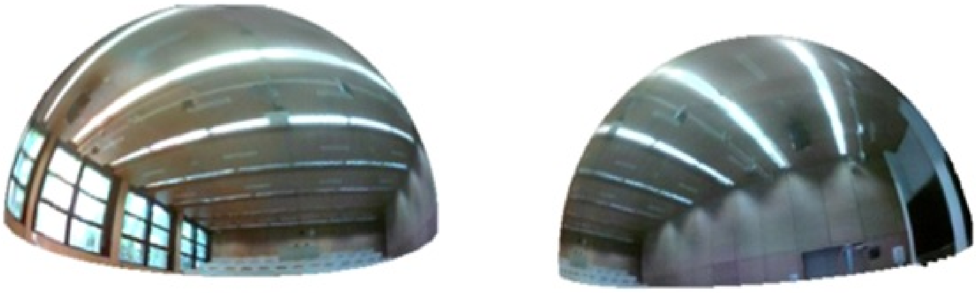


Figure 9.10: Reconstructed 360° views

9.1.3 Implementation Results

The proposed hardware architectures are implemented using VHDL. The VHDL RTL codes are synthesized and mapped to Xilinx XC5VLX50-1FF1153 FPGAs using Xilinx ISE 11.5. The hardware implementations are verified with post place and route simulations using Mentor Graphics Modelsim 6.2b and the full verification is carried out on the Panoptic camera platform.

Table 9.3: Implementation Results

Resources & Parameters	**Unit**	**OIR [19]**	**EOIR**
Occupied Slices	Occupied Utilization	4127 57%	4105 57%
Slice LUTs	Occupied Utilization	9343 32%	9351 32%
Slice registers	Occupied Utilization	10538 36%	10621 36%
BRAM/FIFO	Occupied Utilization	22 45%	24 50%
Maximum Clock Frequency	MHz	212	212
Latency	cycles	116	118

The resource allocation of omnidirectional image reconstruction hardware on the selected platform is presented in Table 9.3, in which EOIR indicates the configurable hardware that is able to implement both OIR and EOIR algorithms, providing a selection for the user to choose either of them. The results given in Table 9.3 indicate the required hardware resources for a single FPGA that performs operations for 40 cameras. The hardware resource consumption is divided by two using two concentrator FPGAs each performs operations for 20 cameras. Hardware resource consumption details about the central FPGA is provided in [19].

The total pipeline latency of the OIR implementation is equal to 116 clock cycles. This latency increases in the EOIR implementation to 118 clock cycles, without causing an effect on the throughput of the pipelined system. Arithmetic operations are implemented in 16 bit fixed-point precision format, which provides good tradeoff between image quality, hardware area and clock frequency.

The post place and route maximum operating frequency is measured by the XILINX ISE timing analyzer tool at 212 MHz, which is identical for both OIR and EOIR implementations. As observed in Table 9.3, the hardware implementation of the EOIR algorithm does not cause a significant increase of resources over the OIR's implementation. The area and the memory usages approximately match, while the resolution of the 3D reconstructed image and the quality of the residual are visibly improved.

Finally, the 256 × 1024 resolution omnidirectional images that are reconstructed at 25 fps are transmitted to the PC through a USB channel. Subsequently, 2D omnidirectional images are converted to 3D using OpenGL C++ and the Qt library, as a real-time video display process. Fig. 9.10 shows two different views of a real-time 360° × 90° image produced after the construction of the real-time omnidirectional image on the computer. The EOIR implementation is used in a system that utilizes Panoptic Camera and Oculus Rift [85] to artificially make the viewer feel himself being at the place of the omnidirectional camera. In addition, the presented system may be used with prospective future 360° hemispheric displays. Thereby, improvements presented in this section of thesis participates in the effort of providing virtual reality.

9.2 Giga-Eye Camera

The quality of the real-time omnidirectional images obtained by the system presented in the previous section is limited by the low resolution of the image sensors. Therefore, very high quality omnidirectional video quality is targeted using high-resolution cameras. In order to reach this goal, a high quality omnidirectional video recording system is implemented using 5MP cameras. The target of the research presented in this section is not real-time omnidirectional video reconstruction, but triggers research in real-time ultra-high resolution omnidirectional video processing and its applications.

In this work, cameras are positioned in a way to guarantee that every target location in the horizontal plane is covered by at least two cameras. Covering target locations by at least two cameras enables a smooth transition between the images while constructing omnidirectional images, and enables using the device in applications that requires depth map estimation [84]. In this system, the coverage analysis methodology presented in [86] is used to measure the required angles between the camera layers and horizontal plane θ and the angles between the horizontally neighboring cameras φ.

Very-high AOV lenses cause image distortion while low AOV lenses necessitate a large number of cameras to guarantee a large effective AOV. A 5MP image sensor with a 6mm lens is selected for the construction of the system since it offers an efficient trade-off between large AOV and distortion. The selected camera provides 53° and 43° for the horizontal and vertical axis AOVs respectively, and the full resolution of each camera is 2592×1926 pixels. Following the methodology presented in [86], 44 cameras are positioned on four levels where level-4 represents top camera and level-1 represents cameras in bottom layer. From the top to the bottom layers, 1, 6, 15 and 22 cameras are distributed, respectively. The top camera is perpendicular to the ground plane. The θ angles of the four layers from top to bottom are 0°, 39.6°, 59.3° and 80.8°. The φ angles between the cameras for the layers 1, 2, 3 are 16.36°, 24° and 60°, respectively. The resulting coverage analysis is given in sub-section 9.2.3.

Table 9.4: Properties of the omnidirectional imaging system.

Individual Camera Parameters					Obtained Video				Requirements	
Option	Observed Resolution	Skipping-Binning	Bit/Pixel	Clock (MHz)	AOV h×v	Output Resolution	Frame Rate	Record duration (min)	Memory Size (GB)	Bandwidth (MB/s)
1	2592 × 1944	0-0	12	96	$53^{o} \times 43^{o}$	2592 × 1944	14	30	177	101
2	2592 × 1944	0-0	12	66	$53^{o} \times 43^{o}$	2592 × 1944	9.5	14	56	68
3	2592 × 1944	1-1	12	86	$53^{o} \times 43^{o}$	1296 × 972	30	30	95	54
4	2592 × 1944	1-0	12	66	$53^{o} \times 43^{o}$	1296 × 972	30	30	95	54
5	2592 × 1944	1-0	12	66	$53^{o} \times 43^{o}$	1296 × 972	30	17	54	54
6	2592 × 1944	1-0	8	66	$53^{o} \times 43^{o}$	1296 × 972	30	30	63	36
7	2592 × 1944	2-0	8	36	$53^{o} \times 43^{o}$	864 × 648	30	30	28	16
8	1024 × 768	0-0	12	46	$21^{o} \times 16^{o}$	1024 × 768	30	30	40	23

9.2.1 System Parameters and Requirements

After the type, number and positions of individual cameras are fixed, the main two aspects that define the performance of the image recording system are the image resolution and frame-rate of individual sensors. The image sensor chosen in the construction of this system has programmable parameters which affect these two main aspects. Table 9.4 presents some example configurations of camera parameters and their respective memory size, bandwidth and operating frequency requirements with respect to the image capture duration.

The selected image sensor delivers 12-bit raw Bayer data output. The bit precision is not reduced, in order to improve the histogram when the camera is used in extreme dark or bright conditions. Moreover, most of the data compression techniques applied in consumer electronic video recording cameras reduce the image quality and light field capturing capability of the imaging system, thereby creating an incompatibility with high quality post processing. Therefore, a 12 bit raw Bayer recording format is maintained, which considerably impacts the memory requirement. Adjusting the frame-rate of the image sensor is possible by tuning the operating frequency of the sensor and frame size. At the fastest frame rate achieved at 96 MHz, the memory requirement for a 30 minute video recording is equal to 177 GB and the memory bandwidth requirement is 101 MB/s for a single camera. Considering 44 cameras, the memory requirement increases to 5.2 TB for the first implementation option presented in Table 9.4. Thus, the memory bandwidth and size requirements for the selected frame-rate and frame size determine the constraints for the selection of the storage device.

The chosen image sensor offers two options for subsampling the frame prior to delivering it, which are known as skipping mode and sub-window selection. The skipping mode allows keeping the highest AOV while reducing the resolution of the sensor, whereas selecting sub-window as exemplified in eighth implementation option of Table 9.4 reduces the AOV. Overlapping the wide AOVs of individual image sensors is crucial for better sampling of the light field and light field based image processing applications. As shown in the third and fourth analysis cases of Table 9.4, operating in skipping mode reduces the image size to one quarter of the original 5MP, 1296×972, while maintaining a constant angle of view (AOV). The binning mode aggregates the Bayer data of the skipped pixels when it is combined with skipping mode. Using the binning mode together with skipping requires a 86MHz clock to provide 30fps video. Providing the same resolution without using the binning mode decreases the required pixel clock frequency to 66 MHz. Recording 30 minutes using one of these options requires 95 GB of memory and a 54 MB/s bandwidth for each camera.

The chosen image sensor does not provide differential output signals, and thus, additional limitations pertaining to signal integrity arise for building the complete system. Constructing a complete system on a dome that has a radius of 20 cm requires data cables with a minimum length of 40 cm. Transmission of a 96 MHz clock over 40 cm cables is prone to noise due to signal reflection, even using very-high-density cable interconnect (VHDCI). Nevertheless, any noise in the image vanishes when the clock frequency is decreased below 80 MHz. Consequently, selecting the appropriate camera operation frequency is a determinant system-level constraint.

The system requires large-capacity and large-bandwidth storage devices. The three main candidate storage devices are i) the Compact Flash, ii) the Hard Disk Drive (HDD) and iii) the Solid State Drive (SSD). Compact Flash devices are very expensive compared to HDD and SSDs, when their considered capacity is larger than 64 GB. HDDs are the most cost-efficient storage devices. However, their mechanical structure has a crucial impact on their effective recording bandwidth under environmental vibration conditions. Vibration robustness tests carried out with several different HDDs have demonstrated that the effective data bandwidth can be reduced down to 20MB/s. Storage devices are expected to support constant bandwidth under vibration since the complete imaging system may not remain stationary, and move during image recording. Yet, offering a constant recording bandwidth is an important constraint of the system, which only SSD technology can provide in presence of external vibrations. Therefore SSDs are selected as the primary, i.e., real-time, storage devices.

For SSD communication interface, Serial ATA (SATA) 2.0 standard is chosen, in order to fulfill the BW requirements in burst write mode and due to their availability on the market. SSD devices with SATA 2.0 usually sustain a bandwidth of approximately 150 MB/s, which is higher than the 101 MB/s bandwidth requirement of a full resolution system operating at 14 fps. Constructing a system that supports a resolution of 1296×972 pixel require a 54 MB/s bandwidth. Hence, one SSD can act as the storage device for two cameras simultaneously. 128 GB SSDs can continuously record for 17 minutes from two cameras that operate at 30 fps and with a 1296×972 pixel resolution. There is also need for intermediate buffer memory in the architecture of the targeted embedded system in order to make the burst writes to the SSD possible. Buffering a single frame in full resolution, and 1296×972 pixel resolution approximately requires a memory size of 8 MB and 2 MB, respectively. DDR2 SDRAM memories typically have a data size of 256 MB or more, and an approximate bandwidth of 5 GB/s. DDR2 are thus selected as the efficient devices to use in conjunction with the SATA protocol for buffering images. Moreover, benefiting from the maximum resolution of the cameras is possible, but in this case the frame rate of the cameras should be reduced due to bandwidth limitations. 128 GB SSDs can continuously record for 14 minutes from two 2592×1944 pixel resolution cameras that operate at 9.5 fps.

The developed system is planned to sustain constraints and target features summarized in Table 9.5 and Table 9.6. A maximum flexibility in terms of resolution and frame rate is aimed. When the system is configured to utilize 1296×972 resolution settings of the cameras with the system constraints presented in Table 9.5, the systems is able to provide 21.6 MP omnidirectional video at 30 fps. When the system is configured to utilize 2592×1944 resolution settings of the cameras with the system constraints presented in Table 9.6, the systems is able to provide 82.3 MP omnidirectional video at 9.5 fps.

Table 9.5: System Constraints to generate 30 fps 21.6 MP Omnidirectional Video.

#Cameras	44	Pixel Clock	66 MHz
Resolution per Camera	1296×972	Record Duration	17 min
Skipping Mode	1	Buffer Memory	DDR2
Binning Mode	0	Storage Device	SSD
#Cameras/#SSDs	2	Back Up Device	HDD
Pixel Resolution	12-bit Bayer	Storage Protocol	SATA II

Table 9.6: System Constraints to generate 9.5 fps 82.3 MP Omnidirectional Video.

#Cameras	44	Pixel Clock	66 MHz
Resolution per Camera	2592×1944	Record Duration	14 min
Skipping Mode	0	Buffer Memory	DDR2
Binning Mode	0	Storage Device	SSD
#Cameras/#SSDs	2	Back Up Device	HDD
Pixel Resolution	12-bit Bayer	Storage Protocol	SATA II

9.2.2 System Architecture

In order to support the constraints explained in sub-section 9.2.1, the hardware platform should be flexible and configurable. Moreover, it should be expandable for future developments in terms of handling critical parts of the image processing on the captured light field. Thus, an FPGA based embedded system is the most suitable hardware platform for the mentioned purposes. The choice for the hardware platform is the Xilinx XUPV5-LX110T FPGA Board. The chosen FPGA board has 2 SATA connections, 64 available external pins for connecting two cameras, 128 MB DDR2 memory, a UART, digital video interface (DVI) and SubMiniature version A (SMA) interfaces. 128 GB Kingston SATA 2.0 SSDs are selected as storage devices, each supporting two cameras. In summary, 22 FPGA boards and 22 SSDs are used for recording 17 minutes video captured from 44 cameras at 1296×972 resolution or 14 minutes video captured from 44 cameras at 2592×1944 resolution.

One of the 22 FPGAs serves as Master FPGA (M-FPGA), which is responsible of receiving commands from a host PC system. The commands are for covering the tasks such as start/stop, snapshot, and changing the image sensor parameters by writing to their responsible registers. The M-FPGA receives commands via RS232 interface and broadcasts to the slave FPGAs (S-FPGA). A Graphical User Interface (GUI) designed to handle the communication between user and imaging system. The system architecture of the M-FPGA is shown in Fig. 9.11. The S-FPGA architecture is identical to the one in Fig. 9.11 except the clock scheme, which is further detailed later in this sub-section. As presented in Fig. 9.11, the hardware architecture of a single FPGA includes a Microblaze Soft Processor, Processor Local Bus (PLB), Image Capture modules, DVI Displayer modules, SATA Interface modules, and System Interconnection modules.

The Image Capture modules are used to capture images from two cameras and to buffer them into a DDR2 Memory. Additionally, an automatic color gain controller (AGC) and automatic shutter width controller (ASW) are implemented.

The Microblaze initializes the cameras through an I^2C bus, and one single I^2C bus controls two cameras, consistently. The Camera Controller module samples the pixels using camera control signals hsync and vsync. The Automatic Gain Control is an important feature pertaining to the white balance. The Automatic Shutter Width enables to adjust the exposure time in extreme dark and bright conditions. The AGC+ASW sub module in Fig. 9.11 computes the average red, green, and blue values in the image, and transfers these values to the Microblaze processor using software-accessible registers. Depending on the application or light conditions, the software hosted by the Microblaze provides flexibility to the user for manual and automatic adjustment of the color gains and shutter width. Two native port interface (NPI) of the multi port memory controller (MPMC) module are used to simultaneously write two images into the DDR2 memory. Row Buffers are placed as a FIFO between the Camera Controller, Burst Converter and NPI Controller sub modules to correctly exchange data while they are operating with different clock rates. These buffers provide flexibility to the user to capture and save images with different resolutions and frame-rates.

The DVI Displayer module reads the images captured by one camera, converts the Bayer images to RGB format, and displays the images on a DVI monitor. The Bayer to RGB hardware is only implemented for display purposes. The NPI controller of the DVI module is able to switch between the DDR2 memory pointers to display images of the selected camera. The DVI connection is initialized by the I^2C module for 1024×768 resolution at 60 fps to display sub-window of the obtained images during the recording process. The DVI interface is operated with a 216 MHz differential clock.

The Host SATA IP can be connected to two storage devices. Although the SATA IP is able to switch between the SATA ports, it is not able to dump data into two storage devices, consistently. Therefore, a 128 GB SDD is connected to one of its two SATA ports. The other port of the IP is used for a backup purpose. Connecting one 2TB HDD for backup support increases the record duration to more than 3.5 hours. After every continuous record of 14 minutes at a 2592×1944 video resolution or 17 minutes at a 1296×972 video resolution, the system is able to make a data backup from the SSD to the HDD. Upon backup completion, the system makes itself ready to record the next continuous video on the SSD. The SATA IP is connected to the DDR2 through MPMC using two Direct Memory Access (DMA) controllers.

The System GUI includes start and stop options to enable the capture of short videos, and enables capturing single and time-delayed repetitive shots for photographic applications. Moreover, the last used memory address is also saved into the SSD. Consequently, the system GUI provides options to resume from the last memory address or overwriting the previous records, after a system power off-on cycle.

Camera Synchronization is one of the most important technical issues in multiple-camera systems. For perfect synchronization, the number of images taken from different cameras

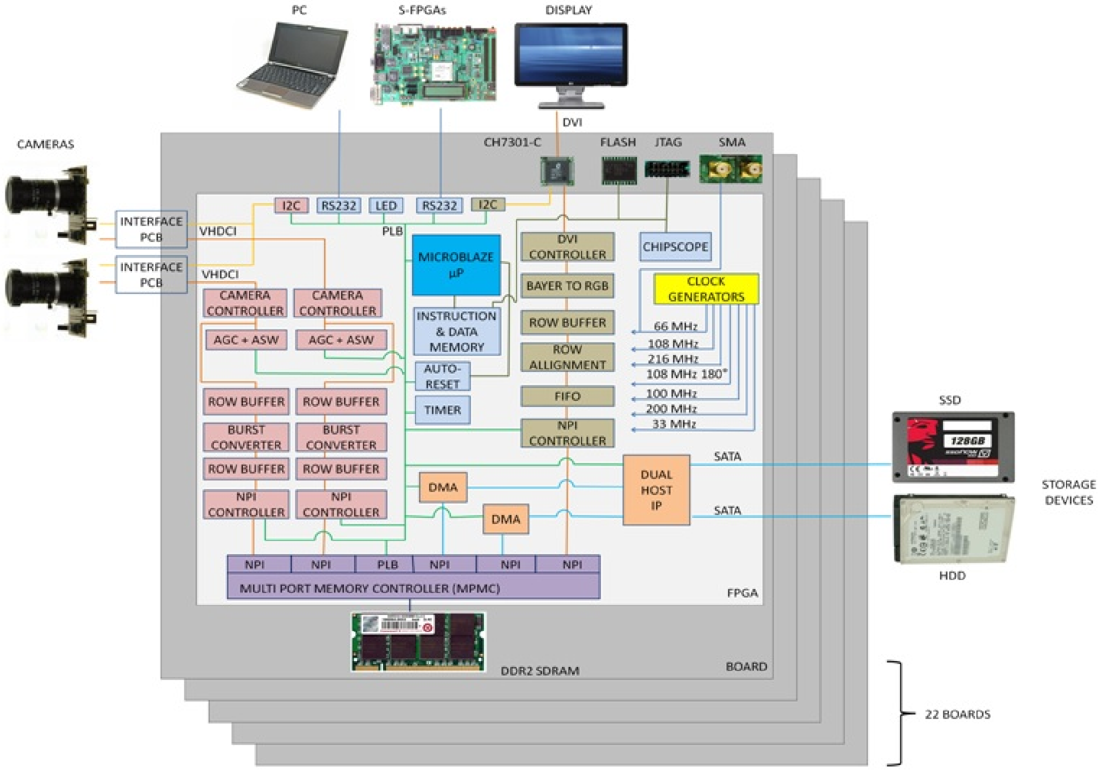


Figure 9.11: Top-level block diagram of the system hardware architecture

should be identical during the capture of a video, and every image originating from the 44 cameras should be shot at the same moment.

In order to force all cameras to capture an identical number of frames, the frame-rates of the cameras must exactly match. All cameras are programmed to the same resolution and exposure time. In addition, the main factor guaranteeing setting the frame rate is the pixel clock frequency. The clock crystal mounted on the XUPV5-LX110T board provides 25MHz with a ±0.0004% tolerance, which means ±100 clock cycles. If each of the 22 separate crystal oscillators is used as main clock source for each camera pair via the internal PLLs of FPGAs, the ±0.0004% deviation causes different frame rates, and prevents synchronization. In order to guarantee clock synchronization, a clock chain is built up that provides an identical clock rate to all of the cameras in the system. This shared clock is generated from the crystal oscillator of the M-FPGA. Each FPGA receives its clock from its neighbor, and transmits it to the FPGA located next within the chain, using coaxial SMA cables and connectors. All the clock inputs and outputs of the SMA connection are buffered to increase the clock drive strength.

In addition to generating an exact frame rate, an additional constraint for the camera synchronization resides in the timing for the acquisition of the images. The image snapshots should be acquired almost at the same moment for a successful synchronization. To this aim and as shown in Fig. 9.11, one I^2C module of the PLB bus is used for two cameras. Since the camera chips have the same I^2C address, two cameras connected to the same I^2C module can be configured at the same moment, which enables a perfect synchronization. The shutting time difference between the cameras connected to the same FPGA is 0 or 1 clock cycle at a 66 MHz pixel clock.

Two serial communication chains are implemented in the system. The first serial communication chain is implemented between the PC host system, M-FPGA and S-FPGAs. This chain shares the user commands and system orders of the M-FPGA over all connected FPGAs for starting and stopping the record on different FPGA boards synchronously. All FPGAs initialize their own connected two cameras at the same time. The posing time difference between the cameras connected to different FPGAs is measured to be less than 400 clock cycles, i.e., less than 5 µs. While a single camera operates at 30 fps, the posing time delay between two consecutive frames of the same camera is approximately 33 ms. Hence, the 5 µs delay is negligible. Thus, the synchronization of the cameras that are connected to the different FPGAs can be considered almost perfect.

The second RS232 chain is implemented to provide information about the recording status of the S-FPGAs to the M-FPGA. This feature is mainly important for the SSD to HDD backup process since the backup process is not synchronized due to variable performance of HDD. Each S-FPGA informs its neighboring board via second serial chain as soon as it finishes the current backup process. When all the S-FPGAs finish their backup process, the M-FPGA configures all the S-FPGAs for the next record to the SSD using the first serial chain.

9.2.3 Implementation Results

The hardware architecture of the proposed video recording system is implemented using ISE 12.4 and XPS. The presented system is constructed using 44 Aptina cameras, 22 XUPV5-LX110T Virtex-5 FPGAs and 22 Kingston 128 GB SSDs. The backup process is verified using 2TB Hitachi HDDs. The XUPV5-LX110T FPGA includes 69k Look-Up Tables (LUT), 69k DFFs and 148 Block RAMs (BRAM). The proposed hardware consumes 31% of the LUTs, 25% of the DFF resources and 48% of the BRAM resources. The Microblaze microprocessor and the SATA IP are operated at 100 MHz and 200 MHz, respectively. The system is able to record 17 minutes of continuous video at a resolution of 1296×972 and a raw Bayer data format at 30 fps, or 14 minutes of continuous video at a resolution of 2592×1944 at 9.5 fps. The proposed backup system enables increasing the record duration to more than 3.5 hours with discrete video records of 14 min or 17 min using a 2TB HDD for each FPGA board. The system is perfectly synchronized for the two cameras connected to the same FPGA, and the time delay between

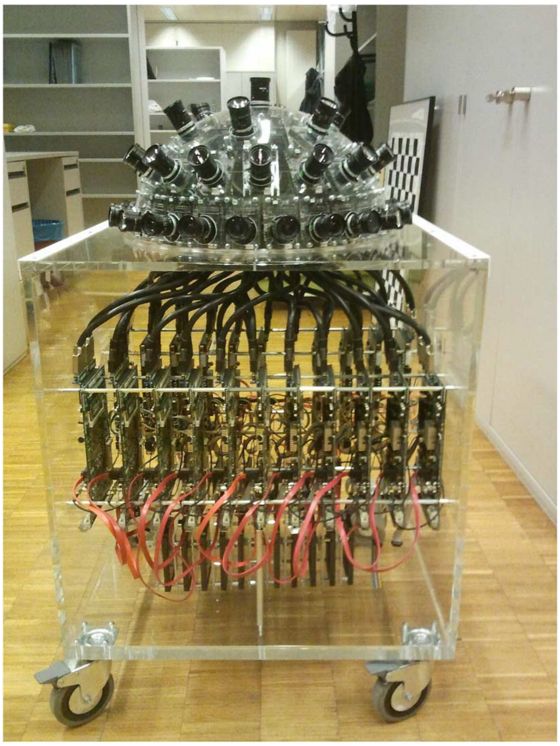


Figure 9.12: The complete omnidirectional imaging and recording system (Giga-Eye), overall system dimensions are 56x48x78 cm

the cameras connected to different FPGAs is equal to 5 μs which is considered negligible.

The final constructed Giga-Eye system is shown in Fig. 9.12, which has a size of 56×48×73 cm and a weight of 55kg without the carrier. The system is powered by a single power supply with a 5V output voltage. Total system consumes 92A, and thus 460W of power.

The video frames captured by Giga-Eye are converted into omnidirectional video sequences by offline processing using the Autopano-sift [71], which is a commercial stitching software. Calibration parameters that are estimated by Autopano are used for the reconstruction. Continuous omnidirectional video rendering is obtained by merging these stitched images in time-domain using Matlab. High-resolution video results of the Giga-Eye are demonstrated in the Appendix A. One of the 21.6MP omnidirectional picture obtained using the presented Giga-Eye system is shown in Fig. 9.13. In addition, a 81.3 MP omnidirectional image result of Giga-Eye is presented in Fig. 9.14. As presented in Fig. 9.13 and Fig. 9.14, although the static and dynamic objects that are shown with sub-windows are quite far, Giga-Eye is able to visualize these objects while providing 360° omnidirectional image.

The obtained coverage map after building the prototype and calibration of the cameras is

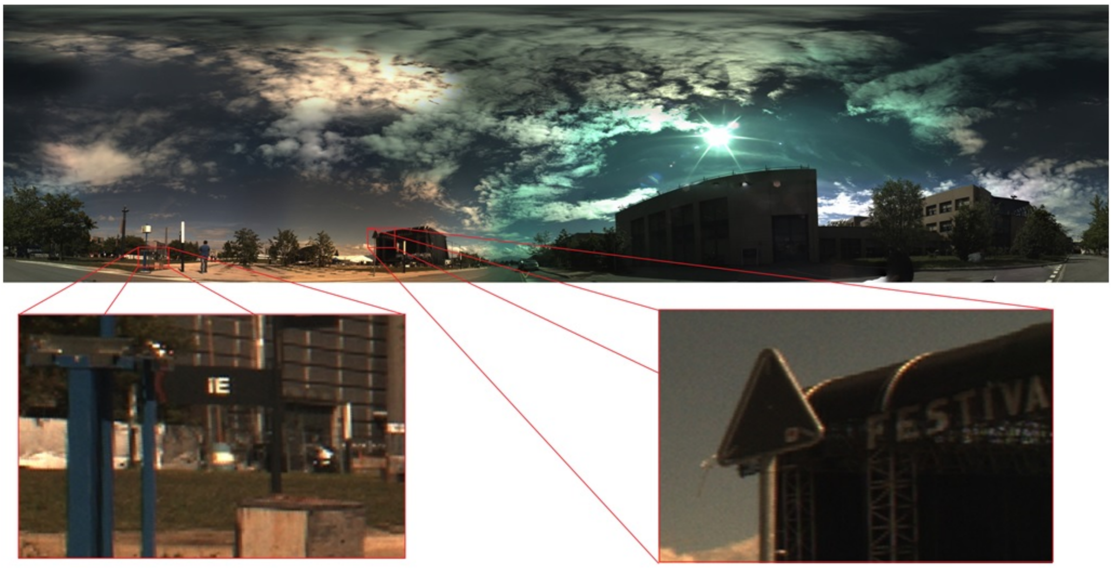


Figure 9.13: Omnidirectional image obtained with the Giga-Eye system at 21.6 MP resolution showing the central campus square of EPFL, and two selected details (sub-regions) in this image. This omnidirectional image corresponds to one single frame of the 30 fps video obtained by the system.

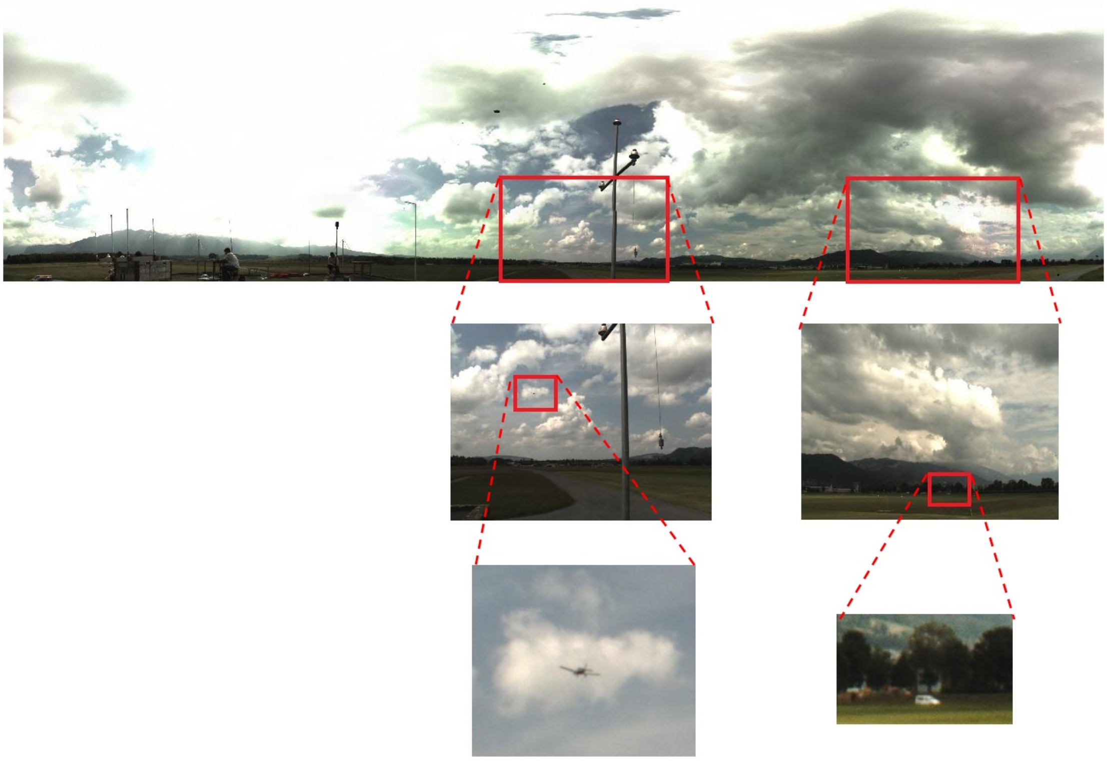


Figure 9.14: Omnidirectional image obtained with the Giga-Eye system at 82.3 MP resolution. This omnidirectional image corresponds to one single frame of the 9.5 fps video obtained by the system. Flying plane and the moving car are shown in sub-windows

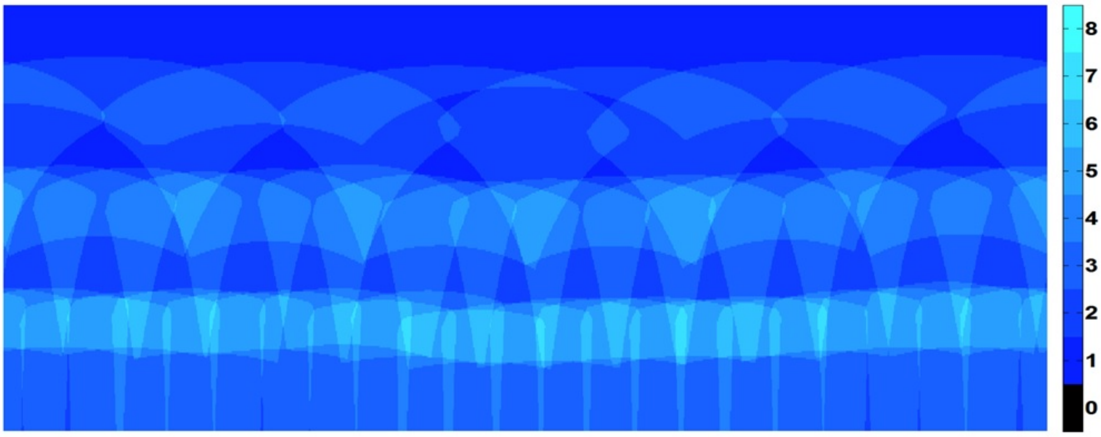


Figure 9.15: Measured coverage map of the omnidirectional imaging system showing a high pixel redundancy especially close to the equator. The color labels indicate the number of the overlapping individual camera AOVs

presented in Fig. 9.15. At most 7 cameras, and at least 2 cameras are capable of capturing every direction, provided that θ angle of the observed direction is below 60°. Therefore, the proposed system does not only provide panorama, but also its efficient coverage enables using the device in light field based image processing applications such as refocusing and 3D rendering.

In Table 9.7, the proposed hemispherical video record system is compared with existing large angle of view video capture systems. The frame rates below 25 fps are not typically considered as video, but they are also compared with the presented work in the table. AOVs of the compared systems and their final resolutions are provided in the comparison. Currently, the proposed Giga-Eye system is the highest resolution 360° omnidirectional camera that provides standard frame-rate video output by its 21.6 MP video output capability at 30 fps. Moreover, Giga-Eye is the highest resolution 360° omnidirectional camera with its 82.3 MP output capability at 9.5 fps. The resolution of the Giga-Eye system can be further increased by omnidirectional image based super-resolution techniques [87] or using higher resolution sensors.

Table 9.7: Comparison of the Giga-Eye with existing high-resolution omnidirectional camera systems.

System	Total Sampling Pixel Amount (#Cameras× Camera Resolution)	Record Frame Rate (fps)	AOV	Resolution
Ladybug3 [16]	11 MP	15	360° × 150°	2048 × 4096
Google [17]	75 MP	0.4	360° × 160°	4096 × 8192
Panoptic [86]	4 MP	25	360° × 100°	256 × 1024
Giga-Eye	55 MP	30	360° × 100°	2400 × 9000
Giga-Eye	220 MP	9.5	360° × 100°	4650 × 17700

10 Conclusion

In this thesis, novel, efficient and high-performance multiple-camera systems for large AOV image capture and depth map estimation are presented. The presented systems can be used for the development of advanced virtual reality applications and 3D-based video processing systems. In this chapter, the major results presented in this thesis are reviewed. In addition, some prospective research fields related with the individual chapters are stated.

Binocular Adaptive Window Size Disparity Estimation Algorithm and Its Hardware Implementation

A binocular hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time high-resolution reconfigurable hardware implementation are presented. The algorithm is designed to be efficiently parallelized and to require minimal on-chip memory size. The proposed hardware provides dynamic and static configurability to have satisfactory disparity estimation quality for the images with different contents. It provides dynamic reconfigurability to switch between window sizes of 7×7, 13×13 and 25×25 pixels in run-time to adapt to the texture of the image. The proposed AWDE hardware combines the strengths of the Census Transform and the Binary Window SAD (BW-SAD) methods. The proposed AWDE implementation utilizes a pixel intensity based refinement step to remove faulty disparity computations. The disparity estimation quality of the AWDE algorithm is improved using an iterative disparity refinement process. The proposed enhanced AWDE algorithm that utilizes Iterative Refinement (AWDE-IR) is implemented in hardware and its implementation details are presented. The proposed hardware architectures for AWDE and AWDE-IR provides 60 frames per second at a 1024×768 XGA video resolution for a 128 pixel disparity range. The proposed AWDE and AWDE-IR implementations provide better DE results than existing real-time high-resolution DE hardware implementations [43–45] for the tested HR Middleburry benchmarks. The proposed high-quality and high-resolution real-time disparity estimation hardware can be used in advanced 3D-based video processing applications where the depth computation is required.

The cost aggregation method can be integrated into the proposed AWDE implementation as a prospective research. The cost aggregation step is a computationally intensive process and requires significant amount of hardware resources. Nevertheless, the adaptation of cost aggregation method to the proposed AWDE hardware can remove most of the incorrect estimations in the disparity map. In addition, the real-time disparity estimation for full HD or 4K resolution video can be targetted in the future. The parallelization of the proposed hardware should be increased to achieve real-time for full HD or 4K disparity map, which may require a significant increase in hardware resource consumption. Higher resolution disparity maps can provide better accuracy for depth measurement. In addition, increased object definitions in disparity maps can be attractive to many advanced 3D based video processing applications such as 3D reconstruction for 3D printers and finger tracking. Furthermore, disparity estimation can be performed using stereo infrared sensors and infrared projection instead of utilizing standard RGB cameras. In this case, very high resolution disparity estimation may not be targeted since the resolution of the infrared sensors is currently limited to VGA resolution. However, the prospective system would provide high quality results even if it is used in a very dark environment.

Trinocular Adaptive Window Size Disparity Estimation Algorithm and Its Hardware Implementation

A trinocular hardware-oriented adaptive window size disparity estimation (T-AWDE) algorithm and its hardware implementation are proposed to improve the disparity estimation quality of the binocular AWDE and AWDE-IR implementations. The T-AWDE hardware generates a very high-quality depth map by merging two depth maps obtained from the center-left and center-right camera pairs. The T-AWDE hardware enhances disparity results by applying a double checking scheme which solves most of the occlusion problems existing in the AWDE-IR implementation while providing correct disparity results even for objects located in the left or right edge of the center image. The implemented T-AWDE hardware is the first hardware implementation that succeeds to provide real-time trinocular DE for HR video. The T-AWDE hardware can process 55 fps at a 768×1024 XGA video resolution for a 128 pixels disparity range. The proposed trinocular DE hardware can be used if removing erroneous disparity estimation results in the occluded region is a significant necessity of a 3D-based video processing application.

A trinocular disparity estimation system which supports unequal distances for center-left and center-right pairs can be studied as prospective research. If the distance between the stereo cameras is large, the depth estimation accuracy for far objects is high. However, a large distance between the cameras reduces the overlapping region between stereo images which prohibits depth computation for close objects. Therefore, the distance between the cameras should be low to measure the distance of close objects. A hybrid solution can be utilized in the future that combines the strengths of large and small distances between cameras. A large distance can be utilized for one pair of trinocular disparity estimation, and a small distance can be utilized for another pair. Then the two disparity maps can be combined to obtain accurate disparity map for both close and far objects.

Compressed Look-Up-Table Based Rectification Algorithms and Their Hardware Implementations

A novel algorithm which compresses the rectification information to fit the look-up-table into the on-chip memory of a Virtex-5 FPGA is presented. The proposed compressed look-up-table based rectification algorithm (CLUTR) can be used to rectify stereo images if the lens distortion is not extreme and the cameras are not excessively misaligned. In addition, in order to solve difficult camera alignment and distortion issues while maintaining the low complexity architecture, an enhanced version of the compressed look-up-table based rectification algorithm (E-CLUTR) and its real-time hardware are presented. The low-complexity de-compression processes of CLUTR and E-CLUTR require a negligible amount of hardware resources to support their real-time implementation and do not require the existence of external memory to store the look-up-tables. The capacity of CLUTR and E-CLUTR to fit the look-up-tables into the on-chip memory of the Virtex-5 FPGA is approximately six times and two times more efficient than [56], respectively, as a benefit of their efficient compression scheme.

Furthermore, the Caltech rectification algorithm [50] which does not benefit from look-up-tables is implemented in hardware, and its hardware resource consumption results are presented to improve the hardware comparison and to evidence the efficiency of CLUTR and E-CLUTR in an appropriate way. Hardware implementations of CLUTR and E-CLUTR require much less hardware resource than the hardware implementation of Caltech rectification while providing almost identical rectification results with very high PSNR results.

The CLUTR and E-CLUTR implementations utilize forward mapping based rectification. Applying the similar compression scheme for backward mapping based rectification can be studied as prospective research. Backward mapping based rectification increases the memory requirement for input pixels buffering; however it can reduce the size of the compressed tables since backward mapping does not cause voids in the rectified images, which avoids the necessity of coding double-target locations. In addition, the compressed rectification method can be verified using trinocular disparity estimation hardware in the future.

Embedded System for Depth Map Estimation

Most of the disparity estimation hardware architectures in the state-of-the-art present the main disparity estimation video processing cores but they do not reveal significant information related to important peripherals of the complete embedded system such as softcore processor, its data bus, camera interface, display interface, external memory interface, DMA modules etc, and most of them claim verification only according to behavioral simulations. In this thesis, the full depth estimation embedded system that verifies the real-time functionality of the disparity estimation hardware is explained. The efficient communication and data exchange scheme of the system peripherals to transfer RGB+D output to a PC are presented. The proposed embedded system can be used to guide the developers of state-of-the-art disparity estimation hardware implementations to allow efficient real-time realization of their disparity estimation hardware.

Hardware and Software based Applications of Disparity Estimation

This thesis presents several real-time hardware and software based applications of disparity estimation. The implemented applications conceptually prove that the high-quality and high-performance RGB+D outputs of the proposed real-time disparity estimation hardware can be used for enhanced 3D based video processing applications.

In this thesis, the first real-time high resolution free viewpoint synthesis system that utilizes three-camera disparity estimation hardware is presented. The proposed hardware generates high-quality free viewpoint video at 55 frames per second using a Virtex-7 FPGA at a 1024×768 XGA video resolution for any horizontally aligned arbitrary camera positioned between the leftmost and rightmost physical cameras. The proposed hardware can be used in glass-free 3D TVs to synthesize free view images in real-time.

The implemented free viewpoint synthesis hardware supports the location of the arbitrary camera only on the horizontal axis of the cameras. The arbitrary camera location can be changed in the axis of depth (z) in the future. While the arbitrary camera is moving in z locations, closer object should be re-sized considering a larger ratio than further object. The autostereoscopic 3D-TV technology does not require to generate arbitrary veiwpoint for a camera that is changing its position in z axis. However, this process is important to deceive the human brain into perceiving 3D by utilizing standard 2D display. In order to obtain better quality for this purpose, the arbitrary image should be also synthesized for a camera moving in the z axis. The implementation of this type of real-time free viewpoint synthesis hardware may require complex rendering hardware and large memory size and bandwidth.

In this thesis, the user-friendly graphical user interface (GUI) of the complete system and the implemented depth estimation based real-time software applications executed in a PC are presented. The implemented software applications consists of speed and distance measurement, depth based image thresholding, head-hands-shoulders tracking, virtual mouse using hand tracking, and face tracking integrated with free viewpoint synthesis. The implemented software applications demonstrate that the proposed system can be used for advanced video processing applications where the depth computation is required.

The proposed disparity estimation system can be utilized in many other 3D based video processing applications in the future. The system can be used to implement touchless 10 fingers keyboard typing application for TVs and PCs. The system can be integrated to smart watches and glasses to control these devices using only the hand gestures without touching the devices. In addition, it can be used for augmented reality applications such as generating cartoon character over existing objects or surfaces in the image. Indeed, most of the applications and games that are implemented using Kinect can be integrated to the proposed depth estimation system. The proposed system provides better resolution than Kinect, but its most important advantage compared to Kinect is that it can operate outdoors under sunlight. Therefore, integrating the applications to the proposed system which may also need to operate outside environment can be very beneficial. For example, several interactive 3D video processing based games can be implemented and utilized in public places such as bus stops, parks and squares.

ASIC Implementation of the Binocular Disparity Estimation

The proposed binocular disparity estimation hardware is implemented in an ASIC using a CMOS TSMC 40nm technology. The ASIC implementation details of the binocular DE hardware are presented. The dimensions of the final ASIC after placing pads are 3450 × 3080 µm. Its total area is 10.626 mm^2. The ASIC is designed to be used as an accelerator for any system that requires stereoscopic depth computation. The binocular DE ASIC supports trinocular DE by using two ASIC chips. The DE ASIC is designed to receive a stream input of images, and to provide a stream output of RGB+D or YCbCr+D data. It does not require external memory for the computation of disparity results. The proposed ASIC solution for the developed hardware offers crucial advantages to the system compared to its FPGA implementation, such as smaller power consumption, faster performance and cost effectiveness. These advantages are important if massive production of the depth estimation system for consumer electronics is targetted.

A single ASIC which includes disparity estimation, rectification, USB3 interface and softcore processor can be implemented in the future. This system-on-chip can result in a cheaper and less power consuming depth estimation system than the system that utilizes presented

ASIC as an accelerator. Therefore, the prospective ASIC can be more beneficial to be used in consumer electronic products where depth estimation is required.

High-Quality Omnidirectional Multi-Camera Systems

Two implementations are proposed to improve the existing real-time $360° \times 90°$ omnidirectional cameras. First, an enhanced version of the omnidirectional image reconstruction algorithm (EOIR) and its real-time hardware implementation are presented. The EOIR algorithm provides homogeneous resolution over the entire reconstruction area. The proposed EOIR algorithm increases the realistic aspect of omnidirectional images captured by the Panoptic camera. The entire system provides the high bandwidth required to simultaneously process data originating from 40 cameras, and reconstruct omnidirectional images of 256×1024 pixels at 25 frames per second.

A novel very high-resolution multiple-camera omnidirectional video recording system called Giga-Eye is implemented using 5 MP cameras. Giga-Eye records high-resolution images to reconstruct very-high resolution omnidirectional images in off-line processing. The proposed Giga-Eye system is the highest resolution 360° omnidirectional camera that provides standard frame-rate video output (more than 25 fps) by its 21.6 MP video output capability at 30 fps. Moreover, Giga-Eye is the highest resolution 360° omnidirectional camera with its 82.3 MP output capability at 9.5 fps. Giga-Eye is able to view the target locations close to horizontal plane with more than two cameras, which enables using this device for ultra-high resolution omnidirectional depth map estimation in the future.

A very high-resolution $360° \times 90°$ AOV disparity estimation algorithm and its hardware implementation can be studied as a prospective research field. This process imposes significant challenges for algorithm and hardware development. The 44 image sensors of the Giga-Eye are not positioned in a parallel configuration due to the hemispherical structure; thus the overlapping region of the images of two adjacent cameras is low compared to the planar alignment. Sensitive calibration and rectification requirement of the disparity estimation would be challenging for very high number of cameras on a hemispherical surface. Moreover, high number of camera pairs should be processed to complete 360° disparity map, which can require a significant amount of hardware resources. Utilization of multiple disparity estimation ASICs presented in this thesis may significantly reduce the cost of this prospective system. Alternative method to obtain very high-resolution 360° disparity map may consist of simultaneously utilizing two horizontally aligned Giga-Eye cameras, and performing disparity estimation directly from their potential real-time omnidirectional image outputs. In this case, the disparity candidates of a processed pixel may not be in a vertical or horizontal epipolar line, but in a curve or in a two-dimensional region of the omnidirectional image. Searching disparity candidates in a curve or two-dimensional region may challenge the data-reuse and parallel processing for real-time implementation. Although the implementation of very-high resolution 360° disparity map is very challenging, this device can be very beneficial for the vision system of self-driving cars and robots. In addition, very high quality stitching of the omnidirectional multiple-camera images requires depth measurement to remove the parallax errors that is caused by depth discontinuity. Combining depth estimation system together with omnidirectional image reconstruction system can be used to implement very high quality 360° × 90° AOV real-time visualization systems in the future. The 360° × 90° AOV RGB+D camera can be very useful for cinema industry, gaming and generation of street level maps.

Final Remarks

High-resolution and high-quality depth estimation and 360° × 90° AOV hemispherical multiple-camera systems are important requirements of advanced 3D-based video processing and virtual reality applications. The high-performance systems and novel efficient ideas that are presented in this thesis allow their utilization in a wide range of applications.

Developing the perfect quality omnidirectional image reconstruction system using multiple cameras without any stitching error and obtaining perfect accuracy depth estimation are ill-posed problems of video processing. Although there is not any perfect algorithmic answer to these issues, the real-time hardware implementation of those prospective algorithms appears as another complex problem. The achievements presented in this thesis may trigger the development of new algorithms and digital hardware design perspectives to approach the solutions of these severe problems.

A High-Resolution Visual Results of Proposed Systems

High-resolution visual results of the depth estimation system and Giga-Eye multiple-camera system are demonstrated in this Appendix. The videos and the pictures accesses from the links below are best viewed in downloaded version. In addition, they are available in the printed version of the thesis in an attached DVD. Watching some of the videos may require to use VLC player.

https://zenodo.org/record/16544
http://dx.doi.org/10.5281/zenodo.16544

A.1 Visual Results of the Depth Estimation System

The GUI of the proposed depth estimation system, the visual results of video processing cores and the software based application operated in PC are demonstrated in the video files "DepthEstimationSystemVideoExample1.mov" and "DepthEstimationSystemVideoExample2.mp4". The videos evidence that the proposed depth estimation system provides accurate depth estimation results, and it can be used for a wide range of 3D-based video processing applications.

A.2 Visual Results of the Giga-Eye Multiple-Camera System

17730×4654 (82.5 MP) resolution image results of GigaEye omnidirectional multiple camera system are presented. The ultra-high resolution picture presented in "GigaEyeImageExample1.jpg" allows zooming into any target location to see more details. In addition, the video sequence of the presented image and two additional videos are presented in the files "GigaEyeVideoExample1.asf", "GigaEyeVideoExample2.wmv" and

"GigaEyeVideoExample3.wmv". The videos do not include ultra-high resolution images since the video encoder does not support ultra-high resolution video. Ultra-high resolution video encoder is supposed to be available in the future. The system can be used in advanced video processing applications where large AOV, and ultra high resolution video capture are required.

Bibliography

[1] A Neil. Autostereoscopic 3d displays. *Computer*, 8:32–36, 2005.

[2] Johnny Chung Lee. Hacking the nintendo wii remote. *Pervasive Computing, IEEE*, 7(3):39–45, 2008.

[3] Joshua Gluckman, Shree K Nayar, and Keith J Thoresz. Real-time omnidirectional and panoramic stereo. In *Proc. of Image Understanding Workshop*, volume 1, pages 299–303. Citeseer, 1998.

[4] Lidong Chen, Maojun Zhang, Bin Wang, Zhihui Xiong, and Gang Cheng. Real-time fpga-based panoramic unrolling of high-resolution catadioptric omnidirectional images. In *Measuring Technology and Mechatronics Automation, 2009. ICMTMA'09. International Conference on*, volume 1, pages 502–505. IEEE, 2009.

[5] Jacques W Duparre, Peter Schreiber, Peter Dannberg, Toralf Scharf, Petri Pelli, Reinhard Völkel, Hans-Peter Herzig, and Andreas Bräuer. Artificial compound eyes: different concepts and their application for ultraflat image acquisition sensors. In *Micromachining and Microfabrication*, pages 89–100. International Society for Optics and Photonics, 2004.

[6] Els Moens, Youri Meuret, Heidi Ottevaere, Mukul Sarkar, David San Segundo Bello, Patrick Merken, and Hugo Thienpont. An insect eye-based image sensor with very large field of view. In *SPIE Photonics Europe*, pages 77162D–77162D. International Society for Optics and Photonics, 2010.

[7] Shinsaku Hiura, Ankit Mohan, and Ramesh Raskar. Krill-eye: Superposition compound eye for wide-angle imaging via grin lenses. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2204–2211. IEEE, 2009.

[8] Kenjiro Hamanaka and Hiroshi Koshi. An artificial compound eye using a microlens array and its application to scale-invariant processing. *Optical Review*, 3(4):264–268, 1996.

[9] Peter Rander, PJ Narayanan, and Takeo Kanade. Virtualized reality: constructing time-varying virtual worlds from real world events. In *Proceedings of the 8th conference on Visualization'97*, pages 277–ff. IEEE Computer Society Press, 1997.

[10] Shiloh L Dockstader and A Murat Tekalp. Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE*, 89(10):1441–1455, 2001.

[11] Jong-Eun Ha and I-Sak Choi. Simple method for calibrating omnidirectional stereo with multiple cameras. *Optical Engineering*, 50(4):043608–043608, 2011.

[12] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):267–282, 2008.

[13] Guilan Feng, Weijian Tian, Changqing Huang, Tao Liu, and Shuqin Zhang. Wide field of view ccd camera based on multi-sensors image mosaics. In *Image and Signal Processing, 2008. CISP'08. Congress on*, volume 2, pages 432–435. IEEE, 2008.

[14] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 765–776. ACM, 2005.

[15] Wai-Kwan Tang, Tien-Tsin Wong, and Pheng-Ann Heng. A system for real-time panorama generation and display in tele-immersive applications. *Multimedia, IEEE Transactions on*, 7(2):280–292, 2005.

[16] Pointgrey. ladybug. [Online]. Available: http://www.ptgrey.com/products/spherical.asp.

[17] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.

[18] H. Afshari, L. Jacques, L. Bagnato, A. Schmid, P. Vandergheynst, and Y. Leblebici. Hardware implementation of an omnidirectional camerawith real-time 3d imaging capability. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, 2011.

[19] Hossein Afshari, Laurent Jacques, Luigi Bagnato, Alexandre Schmid, Pierre Vandergheynst, and Yusuf Leblebici. The panoptic camera: A plenoptic sensor with real-time omnidirectional capability. *Journal of Signal Processing Systems*, 70(3):305–328, 2013.

[20] Federico Tombari, Stefano Mattoccia, and Luigi Di Stefano. Stereo for robots: quantitative evaluation of efficient and low-memory dense stereo algorithms. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 1231–1238. IEEE, 2010.

[21] Shucheng Yang, Guoman Huang, Zheng Zhao, and Ningna Wang. Extraction of topographic map elements with sar stereoscopic measurement. In *Image and Data Fusion (ISIDF), 2011 International Symposium on*, pages 1–4. IEEE, 2011.

[22] Tony KS Cheung and KT Woo. Human tracking in crowded environment with stereo cameras. In *Digital Signal Processing (DSP), 2011 17th International Conference on*, pages 1–6. IEEE, 2011.

[23] Matthew Field, Duncan Clarke, Stephen Strup, and W Brent Seales. Stereo endoscopy as a 3-d measurement tool. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 5748–5751. IEEE, 2009.

[24] G Yahav, GJ Iddan, and D Mandelboum. 3d imaging camera for gaming application. In *Consumer Electronics, 2007. ICCE 2007. Digest of Technical Papers. International Conference on*, pages 1–2. IEEE, 2007.

[25] Marcus Grosse, Johannes Buehl, Holger Babovsky, Armin Kiessling, and Richard Kowarschik. 3d shape measurement of macroscopic objects in digital off-axis holography using structured illumination. *Optics letters*, 35(8):1233–1235, 2010.

[26] Dongbo Min, Donghyun Kim, SangUn Yun, and Kwanghoon Sohn. 2d/3d freeview video generation for 3dtv system. *Signal Processing: Image Communication*, 24(1):31–48, 2009.

[27] Philipp Merkle, Yannick Morvan, Aljoscha Smolic, Dirk Farin, Karsten Mueller, PHN de With, and Thomas Wiegand. The effects of multiview depth video compression on multiview rendering. *Signal Processing: Image Communication*, 24(1):73–88, 2009.

[28] Yuji Mori, Norishige Fukushima, Tomohiro Yendo, Toshiaki Fujii, and Masayuki Tanimoto. View generation with 3d warping using depth information for ftv. *Signal Processing: Image Communication*, 24(1):65–72, 2009.

[29] Cheon Lee, Hyok Song, Byeongho Choi, and Yo-Sung Ho. 3d scene capturing using stereoscopic cameras and a time-of-flight camera. *Consumer Electronics, IEEE Transactions on*, 57(3):1370–1376, 2011.

[30] Velodyne. Hdl-g4e. [Online]. Available: http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx.

[31] Merrill I Skolnik. Introduction to radar. *Radar Handbook*, page 2, 1962.

[32] Microsoft. Kinect. [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/.

[33] Occipital. Structure sensor. [Online]. Available: http://structure.io/.

[34] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008.

[35] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[36] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 467–474. IEEE, 2011.

[37] Zeng-Fu Wang and Zhi-Gang Zheng. A region based stereo matching algorithm using cooperative optimization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[38] Andreas Klaus, Mario Sormann, and Konrad Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 15–18. IEEE, 2006.

[39] NY-C Chang, Tsung-Hsien Tsai, Bo-Hsiung Hsu, Yi-Chun Chen, and Tian-Sheuan Chang. Algorithm and architecture of disparity estimation with mini-census adaptive support weight. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(6):792–805, 2010.

[40] Yosuke Miyajima and Tsutomu Maruyama. A real-time stereo vision system with fpga. In Peter Y. K. Cheung and GeorgeA. Constantinides, editors, *Field Programmable Logic and Application*, volume 2778 of *Lecture Notes in Computer Science*, pages 448–457. Springer Berlin Heidelberg, 2003.

[41] Seunghun Jin, Junguk Cho, Xuan Dai Pham, Kyoung Mu Lee, Sung-Kee Park, Munsang Kim, and Jae Wook Jeon. Fpga design and implementation of a real-time stereo vision system. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(1):15–26, 2010.

[42] Sang Hwa Lee and Siddharth Sharma. Real-time disparity estimation algorithm for stereo camera systems. *Consumer Electronics, IEEE Transactions on*, 57(3):1018–1026, 2011.

[43] Christos Georgoulas and Ioannis Andreadis. A real-time occlusion aware hardware structure for disparity map computation. In *Image Analysis and Processing–ICIAP 2009*, pages 721–730. Springer, 2009.

[44] Christos Ttofis, Stavros Hadjitheophanous, A Georghiades, and Theocharis Theocharides. Edge-directed hardware architecture for real-time disparity map computation. 2012.

[45] Pierre Greisen, Simon Heinzle, Markus Gross, and Andreas P Burg. An fpga-based processing pipeline for high-definition stereo video. *EURASIP Journal on Image and Video Processing*, 2011(1):1–13, 2011.

[46] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *Computer Vision—ECCV'94*, pages 151–158. Springer, 1994.

[47] Mikhail Mozerov, Jordi Gonzàlez, Xavier Roca, and Juan José Villanueva. Trinocular stereo matching with composite disparity space image. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2089–2092. IEEE, 2009.

[48] Andy Motten, Luc Claesen, and Yun Pan. Trinocular disparity processor using a hierarchic classification structure. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 247–250. IEEE, 2012.

[49] Lei Chen, Yunde Jia, and Mingxiang Li. An fpga-based rgbd imager. *Machine Vision and Applications*, 23(3):513–525, 2012.

[50] Jean-Yves Bouguet. Camera calibration toolbox for matlab, 2004. [Online] Available: http://www.vision.caltech.edu/bouguetj/.

[51] Abdulkadir Akin, Ipek Baz, Baris Atakan, Irem Boybat, Alexandre Schmid, and Yusuf Leblebici. A hardware-oriented dynamically adaptive disparity estimation algorithm and its real-time hardware. In *Proceedings of the 23rd ACM International Conference on Great Lakes Symposium on VLSI*, GLSVLSI '13, pages 155–160, New York, NY, USA, 2013. ACM.

[52] D Scharstein and R Szeliski. Middlebury stereo evaluation-version 2, 2011.

[53] Hyeon-Sik Son, Kyeong-ryeol Bae, Seung-Ho Ok, Yong-Hwan Lee, and Byungin Moon. A rectification hardware architecture for an adaptive multiple-baseline stereo vision system. In *Communication and Networking*, pages 147–155. Springer, 2012.

[54] Cristian Vancea and Sergiu Nedevschi. Lut-based image rectification module implemented in fpga. In *Intelligent Computer Communication and Processing, 2007 IEEE International Conference on*, pages 147–154. IEEE, 2007.

[55] KT Gribbon, CT Johnston, and DG Bailey. A real-time fpga implementation of a barrel distortion correction algorithm with bilinear interpolation. In *Image and Vision Computing New Zealand*, pages 408–413, 2003.

[56] Deuk Hyun Park, Hyoung Seok Ko, Jae Gon Kim, and Jun Dong Cho. Real time rectification using differentially encoded lookup table. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, page 47. ACM, 2011.

[57] Abdulkadir Akin, Elif Erdede, Hossein Afshari, Alexandre Schmid, and Yusuf Leblebici. Enhanced omnidirectional image reconstruction algorithm and its real-time hardware. In *Digital System Design (DSD), 2012 15th Euromicro Conference on*, pages 907–914. IEEE, 2012.

[58] Abdulkadir Akin, Omer Cogal, Kerem Seyid, Hossein Afshari, Alexandre Schmid, and Yusuf Leblebici. Hemispherical multiple camera system for high resolution omni-directional light field imaging. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 3(2):137–144, 2013.

[59] Omer Cogal, Abdulkadir Akin, Kerem Seyid, Vladan Popovic, Alexandre Schmid, Beat Ott, Peter Wellig, and Yusuf Leblebici. A new omni-directional multi-camera system for high resolution surveillance. In *SPIE Sensing Technology+ Applications*, pages 91200N–91200N. International Society for Optics and Photonics, 2014.

[60] Andy Motten and Luc Claesen. A binary adaptable window soc architecture for a stereo vision based depth field processor. In *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, pages 25–30. IEEE, 2010.

[61] Abdulkadir Akin, Ipek Baz, Alexandre Schmid, and Yusuf Leblebici. Dynamically adaptive real-time disparity estimation hardware using iterative refinement. *Integration, the {VLSI} Journal*, 47(3):365 – 376, 2014. Special issue: {VLSI} for the new era.

[62] Abdulkadir Akin, Raffaele Capoccia, Jonathan Narinx, Ipek Baz, Alexandre Schmid, and Yusuf Leblebici. Trinocular adaptive window size disparity estimation algorithm and its real-time hardware. In *VLSI-DAT, 2015*. IEEE, 2015.

[63] Abdulkadir Akin, Ipek Baz, Luis Manuel, Alexandre Schmid, and Yusuf Leblebici. Compressed look-up-table based real-time rectification algorithm and its hardware. In *Proceedings of the IFIP/IEEE International Conference on VLSI-SOC*, 2013.

[64] Abdulkadir Akin, Luis Manuel Gaemperle, Halima Najibi, Alexandre Schmid, and Yusuf Leblebici. Enhanced compressed look-up-table based real-time rectification hardware. *Springer book chapter, VLSI-SoC: At the Crossroads of Emerging Trends*, April 2015.

[65] Jong Dae Oh, Siwei Ma, and C-CJ Kuo. Disparity estimation and virtual view synthesis from stereo video. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 993–996. IEEE, 2007.

[66] Aljoscha Smolic, Karsten Mueller, Philipp Merkle, Christoph Fehn, Peter Kauff, Peter Eisert, and Thomas Wiegand. 3d video and free viewpoint video-technologies, applications and mpeg standards. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 2161–2164. IEEE, 2006.

[67] Luat Do, Germán Bravo, Svitlana Zinger, and PHN de With. Real-time free-viewpoint dibr on gpus for large base-line multi-view 3dtv videos. In *Visual Communications and Image Processing (VCIP), 2011 IEEE*, pages 1–4. IEEE, 2011.

[68] Egor Bondarev, Sveta Zinger, and PHN De With. Performance-efficient architecture for free-viewpoint 3dtv receiver. In *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*, pages 65–66. IEEE, 2010.

[69] Abdulkadir Akin, Raffaele Capoccia, Jonathan Narinx, Jonathan Masur, Alexandre Schmid, and Yusuf Leblebici. Real-time free viewpoint synthesis using three-camera disparity estimation hardware. In *Circuits and Systems, 2015. ISCAS 2015. IEEE International Symposium on*. IEEE, 2015.

[70] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Incorporated, 2008.

[71] Sebastian Nowozin. Autopano-sift, making panoramas fun, 2006. [Online] Available: http://user.cs.tu-berlin.de/nowozin/autopano-sift/.

[72] Hachem HALAWANA. Partial demosaicing of cfa images for stereo matching.

[73] Gary R Bradski. Intel open source computer vision library overview. *Intel Labs, Intel Corporation*, 2002.

[74] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[75] Xiaoyan Hu and Philippos Mordohai. Evaluation of stereo confidence indoors and outdoors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1466–1473. IEEE, 2010.

[76] Andy Motten, Luc Claesen, and Yun Pan. Binary confidence evaluation for a stereo vision based depth field processor soc. In *Pattern Recognition (ACPR), 2011 First Asian Conference on*, pages 456–460. IEEE, 2011.

[77] XU Zhen-Ying, LI Wen-Bin, WANG Yun, LUO Chun, GAO Shu-Yuan, and CAO Dan-Dan. Trinocular calibration method based on binocular calibration. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 10(6):1439–1444, 2012.

[78] Liuxin Zhang, Bin Li, and Yunde Jia. A practical calibration method for multiple cameras. In *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pages 45–50. IEEE, 2007.

[79] Gregorij Kurillo, Zeyu Li, and Ruzena Bajcsy. Framework for hierarchical calibration of multi-camera systems for teleimmersion. In *Proceedings of the 2nd International Conference on Immersive Telecommunications*, page 1. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[80] Robert Laganière and Florian Kangni. Projective rectification of image triplets. *Signal, image and video processing*, 4(4):389–397, 2010.

[81] Vincent Nozick. Multiple view image rectification. In *Access Spaces (ISAS), 2011 1st International Symposium on*, pages 277–282. IEEE, 2011.

[82] Young Ki Baik, Jonghyun Choi, and Kyoung Mu Lee. An efficient trinocular rectification method for stereo vision. *Proc. Frontiers of Computer Vision (FCV)*, 2007.

[83] K. Seyid, V. Popovic, O. Cogal, A. Akin, H. Afshari, A. Schmid, and Y. Leblebici. A real-time multi-aperture omnidirectional visual sensor based on interconnected network of smart cameras. *Circuits and Systems for Video Technology, IEEE Transactions on*, PP(99):1–1, 2014.

[84] Luigi Bagnato. *Omnidirectional light field analysis and reconstruction.* PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2012.

[85] Luis Gaemperle, Kerem Seyid, Vladan Popovic, and Yusuf Leblebici. An immersive telepresence system using a real-time omnidirectional camera and a virtual reality head-mounted display. In *will be published in Multimedia (ISM), 2013 IEEE International Symposium on.* IEEE, 2015.

[86] H. Afshari, V. Popovic, T. Tasci, A. Schmid, and Y. Leblebici. A spherical multi-camera system with real-time omnidirectional video acquisition capability. *Consumer Electronics, IEEE Transactions on*, 58(4):1110–1118, November 2012.

[87] Luigi Bagnato, Yannick Boursier, Pascal Frossard, and Pierre Vandergheynst. Plenoptic based super-resolution for omnidirectional image sequences. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2829–2832. IEEE, 2010.

[88] Abdulkadir Akin, Raffaele Capoccia, Jonathan Narinx, Alexandre Schmid, and Yusuf Leblebici. Trinocular adaptive window size disparity estimation algorithm and its real-time hardware. In *VLSI Design, Automation and Test (VLSI-DAT), International Symposium on*, pages 1–4. IEEE, 2015.

[89] Edward L Hauck. Data compression using run length encoding and statistical encoding, December 2 1986. US Patent 4,626,829.

[90] Joaquim Jocha. Skeltrack : A free software library for skeleton tracking, 2012. [Online] Available: https://github.com/joaquimrocha/Skeltrack.

[91] Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Consumer Depth Cameras for Computer Vision*, pages 71–98. Springer, 2013.

[92] Chan Wah Ng and Surendra Ranganath. Real-time gesture recognition system and application. *Image and Vision computing*, 20(13):993–1007, 2002.

[93] Bongjin Jun and Daijin Kim. Robust face detection using local gradient patterns and evidence accumulation. *Pattern Recognition*, 45(9):3304–3316, 2012.

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AD	Absolute Difference
ADS	Adaptive Disparity Selection
AGC	Automatic Color Gain Controller
AOV	Angle of View
ALUT	Adaptive Look-Up-Table
AP	Adaptive Penalty
APCO	Artificial Apposition Compound Eyes
ASIC	Application-Specific Integrated Circuit
ASW	Automatic Shutter Width Controller
AWDE	Adaptive Window Size Disparity Estimation
AWDE-HC	High Computational Cost version of AWDE
AXI	Advanced Extensible Interface
B-DT	Below Double Target
BB-DT	Below-Backward Double Target
BF-DT	Below-Forward Double Target
BRAM	Block Random-Access Memory
BW	Binary Window
C&A	Compare and Accumulate
C&S	Compare and Select
CCD	Charge Coupled Device
CIF	Common Intermediate Format
CLUTR	Compressed Loop-Up-Table based Rectification
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit

DB	Desibel
DDR	Double Data Rate
DE	Disparity Estimation
DFF	D Flip-Flop
DMA	Direct Memory Access
DR	Disparity Refinement
DSP	Digital Signal Processor
DT	Double Target
DVI	Digital Visual Interface
E-CLUTR	Enhanced Compressed Loop-Up-Table based Rectification
EOIR	Enhanced Omnidirectional Image Reconstruction
ETH	Ethernet
FIFO	First In First Out
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
FVT	Free Viewpoint Television
GB	Gigabyte
GPO	General Purpose Output
GUI	Graphical User Interface
HC	Hybrid Cost
HD	High Definition
HDD	Hard Disk Drive
HDMI	High Definition Multimedia Interface
HMD	Head-Mounted Display
HR	High Resolution
I^{2}C	Inter-Integrated Circuit
IO	Input Output
IP	Intellectual Property
IR	Iterative Refinement

LIDAR	Light Detection And Ranging
LR	Low Resolution
LUT	Look-Up-Table
M-FPGA	Master FPGA
MAC	Media Access Control
MAD	Mean Absolute Difference
MB	Megabyte
MHZ	Megahertz
MISO	Master In Slave Out
MOSI	Master Out Slave In
MP	Megapixel
MPMC	Multi-Port Memory Controller
NPI	Native Port Interface
P&R	Place And Route
PC	Personal Computer
PE	Processing Element
PLB	Processor Local Bus
PLL	Phase-Locked Loop
PPAA	Pixels Per Unit Angular Area
PSNR	Peak Signal to Noise Ratio
RAM	Random-Access Memory
RCM	Reconfigurable Computation of Metrics
RDA	Reconfigurable Data Allocation
RGB	Red Green Blue
RO	Read Only
ROM	Read-Only Memory
RTL	Register Transfer Level
RW	Read Write
S-FPGA	Slave FPGA
SAD	Sum of Absolute Difference
SATA	Serial Advanced Technology Attachment
SCLK	Serial Clock
SD	Squared Differences
SMA	Sub-Miniature Version A

SRAM	Static Random-Access Memory
SS	Slave Select
SSD	Sum of Square Differences, Solid State Drive
T-AWDE	Trinocular Adaptive Window Size Disparity Estimation
TB	Terabyte
TCP	Transmission Control Protocol
TOF	Time of Flight
TR	Threshold
TSMC	Taiwan Semiconductor Manufacturing Company
TV	Television
OIR	Omnidirectional Image Reconstruction
UART	Universal Asynchronous Receiver/Transmitter
UB-DT	Upper-Backward Double Target
UDP	User Datagram Protocol
UF-DT	Upper-Forward Double Target
USB	Universal Serial Bus
XGA	Extended Graphics Array
VGA	Video Graphics Array
VHDCI	Very High Density Cable Interconnect
VHDL	Verilog Hardware Description Language
WTA	Winner Take All
YCbCr	Luminance Chrominance-Blue Chrominance-Red
ZBT	Zero-Bus Turnaround

ABDULKADIR AKIN

Date of Birth: 16.06.1985
Contact Address: Chemin du Bugnon 29 1024 Ecublens Vaud/Switzerland
GSM: +41787180560
E-Mail: abdulkadir.akin@epfl.ch
Website: http://people.epfl.ch/abdulkadir.akin

RESEARCH INTERESTS

Low-cost and low-power hardware design for real-time video processing applications using FPGA and ASIC.
Embedded system development for FPGA.
Developing video enhancement and video compression algorithms.
Hardware design for cryptographic algorithms.

EDUCATION

03/2011-present: Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne/Switzerland
PhD, Electrical Engineering (EDEE)
Expected Graduation: April 2015

09/2008-06/2010: Sabanci University, Istanbul/Turkey
Master of Science, Electronics Engineering (MS-EE)
(I am rewarded with the Gursel Sonmez Research Award in the graduation ceremony because of coming into prominence with outstanding research during my MS study)

09/2003-06/2008: Sabanci University
Bachelor of Science, Electronics Engineering (BS-EE), and Mathematics Minor

WORK & INTERNSHIP

03/2011-present:

- Member of Microelectronic Systems Laboratory (LSM) in EPFL.
- Research assistant in the project titled as *Real-Time High-Resolution Multiple-Camera Depth Map Estimation Hardware and its Applications*.
- Assistant of Test of VLSI Systems (EE-530) course.
- Supervising 5 master theses, 2 master level semester projects, 7 internship projects.

07/2010-03/2011: Internship student in ESL and LTS2 Labs of EPFL in the project titled as *Accelerating the Implementation of Chambolle Algorithm using FPGA*.

09/2008-06/2010:

- Member of System-on-Chip Design & Test Laboratory in Sabanci University.
- Research assistant in Sabanci University in the *Low Complexity Motion Estimation Techniques and Their SoC Implementation* project supported by TUBITAK (The Scientific and Technological Research Council of Turkey) and KRF (Korea Research Foundation).
- Assistant of ENS 203 (Circuit I) and EL310 (Hardware Description Languages) courses.
- Supervising BS degree graduation projects of 8 students.

09/2007-09/2008: Part time (3 days in a week) work in the video enhancement department of VESTEK (R&D Company of VESTEL) on *Frame Rate Up Conversion Hardware* project.

08/2007-09/2007: Summer internship at ETA-IC Design Center on *Analog to Digital Converter* project.

06/2007-08/2007: Summer internship at VESTEK on *Sharpness Hardware* project. The hardware is used in the 2nd version 1366*768 LCD PIXELLENCE televisions of VESTEL.

06/2006-11/2006: Internship at Sabanci University in the *Cardiovascular Protein Sensor Design* project supported by TUBITAK.

PATENT

A. Akin, Y. Leblebici, A. Schmid, I. Baz, I. Boybat, H. B. Atakan, "A hardware-oriented dynamically adaptive disparity estimation algorithm and its real-time hardware", US Patent Application n° 14/267,140 filed May 1st, 2014.

PUBLICATIONS

Journal Papers:

[1] K. Seyid, V. Popovic, Ö. Cogal, **A. Akin**, H. Afshari, A. Schmid and Y. Leblebici, "A Real-time Multi-aperture Omnidirectional Visual Sensor Based on Interconnected Network of Smart Cameras", *IEEE Transactions on Circuits and Systems for Video Technology*, Sept. 2014.

[2] **A. Akin**, I. Baz, A. Schmid and Y. Leblebici, "Dynamically adaptive real-time disparity estimation hardware using iterative refinement", *Elsevier, Integration the VLSI Journal*, June 2014.

[3] V. Popovic, K. Seyid, **A. Akin**, Ö. Cogal, H. Afshari, A. Schmid, and Y. Leblebici, "Image Blending in a High Frame Rate FPGA-based Multi-Camera System", *Springer Journal of Signal Processing Systems for Signal, Image and Video Technology*, Nov. 2013.

[4] **A. Akin**, Ö. Cogal, K. Seyid, H. Afshari, A. Schmid, and Y. Leblebici, "Hemispherical Multiple Camera System for High Resolution Omni-Directional Light Field Imaging", *IEEE Journal of Emerging and Selected Topics in Circuits and Systems*, April 2013.

[5] **A. Akin**, M. Cetin, Z. Ozcan, B. Erbagci, and I. Hamzaoglu, "An Adaptive Bilateral Motion Estimation Algorithm and its Hardware Architecture", *IEEE Journal, Tran. on Consumer Electronics*, May 2012.

[6] **A. Akin**, G. Sayilar, and I. Hamzaoglu, "High Performance Hardware Architecture for One Bit Transform Based Single and Multiple Reference Frame Motion Estimation", *IEEE Journal, Tran. on Consumer Electronics*, May 2010.

[7] O. Tasdizen, **A. Akin**, H. Kukner, and I. Hamzaoglu, "Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation", *IEEE Journal, Tran. on Consumer Electronics*, Aug. 2009.

[8] **A. Akin**, Y. Dogan, and I. Hamzaoglu, "High Performance Hardware Architectures for One Bit Transform Based Motion Estimation Algorithms", *IEEE Journal, Tran. on Consumer Electronics*, May 2009.

Book Chapter:

[1] **A. Akin**, L. M. Gaemperle, H. Najibi, A. Schmid, and Y. Leblebici, "Enhanced Compressed Look-up-Table based Real-Time Rectification Hardware", *Springer, VLSI-SoC: At the Crossroads of Emerging Trends*, 2015. (accepted for publication)

International Conference Papers:

[1] **A. Akin**, R. Capoccia, J. Narinx, J. Masur, A. Schmid, and Y. Leblebici, "Real-Time Free Viewpoint Synthesis Using Three-Camera Disparity Estimation Hardware", *ISCAS Conference,* Lisbon, Portugal, 2015. (accepted)

[2] **A. Akin**, R. Capoccia, J. Narinx, A. Schmid, and Y. Leblebici, "Trinocular Adaptive Window Size Disparity Estimation Algorithm and Its Real-Time Hardware", *VLSI-DAT Conference,* Hsinchu, Taiwan, 2015. (accepted)

[3] Ö. Cogal, **A. Akin**, K. Seyid, V. Popovic, A. Schmid, and Y. Leblebici, "A New Omni-Directional Multi-Camera System for High Resolution Surveillance", *SPIE Defense and Security Symposium,* Baltimore, United States, 2014.

[4] **A. Akin**, I. Baz, L. M. Gaemperle, A. Schmid, and Y. Leblebici, "Compressed Look-up-Table based Real-Time rectification hardware", *VLSI-SoC Conference*, Istanbul, Turkey, 2013. **(Nominated for Best Paper Award)**

[5] **A. Akin**, I. Baz, B. Atakan, I. Boybat, A. Schmid, and Y. Leblebici, "A Hardware-Oriented Dynamically Adaptive Disparity Estimation Algorithm and its Real-Time Hardware", *International Great Lakes Symposium on VLSI (GLSVLSI)*, Paris, France, 2013.

[6] H. Afshari, **A. Akin**, V. Popovic, A. Schmid, and Y. Leblebici, "Real-Time FPGA Implementation of Linear Blending Vision Reconstruction Algorithm using a Spherical Light Field Camera", *SiP Symposium*, Québec City, Canada, 2012. **(Received Best Paper Award)**

[7] **A. Akin**, E. Erdede, H. Afshari, A. Schmid, and Y. Leblebici, "Enhanced Omnidirectional Image Reconstruction Algorithm and its Real-Time Hardware", *Euromicro Conference on Digital System Design*, Cesme, Turkey, 2012.

[8] **A. Akin**, I. Beretta, A. A. Nacci, V. Rana, M. D. Santambrogio, and D. Atienza, "A High-Performance Parallel Implementation of Chambolle Algorithm", DATE Conference, Grenoble, France, 2011.

[9] **A. Akin**, O. C. Ulusel, T. Z. Ozcan, G. Sayilar, and I. Hamzaoglu, "A Novel Power Reduction Technique for Block Matching Motion Estimation Hardware", FPL Conference, Crete, Greece, 2011.

[10] **A. Akin**, M. Cetin, B. Erbagci, O. Karakaya, and I. Hamzaoglu, An Adaptive Bilateral Motion Estimation Algorithm and its Hardware Architecture, *VLSI-SoC Conference*, Madrid, Spain, 2010.

[11] **A. Akin**, A. Aysu, O. Ulusel, and E. Savas, “Efficient Hardware Implementations of High Throughput SHA-3 Candidates Keccak, Luffa and Blue Midnight Wish for Single- and Multi-Message Hashing”, *SIN Conference*, Taganrog, Russia, 2010.
[12] **A. Akin**, G. Sayilar, and I. Hamzaoglu, “A Reconfigurable Hardware for One Bit Transform Based Multiple Reference Frame Motion Estimation”, *DATE Conference*, Dresden, Germany, 2010.
[13] **A. Akin**, Y. Dogan, and I. Hamzaoglu, “A High Performance Hardware Architecture for One Bit Transform Based Motion Estimation Algorithms”, *Euromicro Conference on Digital System Design*, Patras, Greece, 2009.
[14] O. Tasdizen, H. Kukner, **A. Akin**, and I. Hamzaoglu, “A High Performance Reconfigurable Motion Estimation Hardware Architecture”, *DATE Conference*, Nice, France, 2009.
[15] O. Tasdizen, **A. Akin**, H. Kukner, I. Hamzaoglu, and H. F. Ugurdag, “High Performance Hardware Architectures for a Hexagon-Based Motion Estimation Algorithm”, *VLSI-SoC Conference*, Rhodes Island, Greece, 2008.

Paper Review Experience:
IEEE Journals: Transactions on Circuits and Systems for Video Technology, Transactions on Very Large Scale Integration Systems.
Elsevier Journal: Digital Signal Processing.
Conferences: FPL, VLSI-SoC, CHES, ICECS.

SKILLS

Languages:	Turkish (native), English (advanced), French (intermediate)
Programming Skills:	Verilog HDL, VHDL, MATLAB, C/C++, Assembly
Work Environment:	Modelsim, Xilinx ISE XST EDK, MS Visual Studio, Cadence

SOCIAL PROJECTS, INTERESTS AND ACTIVITIES

Dance: Argentine Tango (advanced), Disco-Fox
Sports: Rollerblade, ice skating, skiing, swimming, boxing, tennis, squash, cycling, windsurf, football, basketball
Others: Playing Turkish Musical Instrument Baglama, Chess, Cooking
Social Projects: Helping children in orphanage for their lessons (Sabanci University civic involvement project)

REFERENCES

Yusuf Leblebici	Prof. at EPFL Microsystems and Microelectronic Department Director of my PhD thesis	yusuf.leblebici@epfl.ch
Alexandre Schmid	Dr. MER at EPFL Microsystems and Microelectronic Department Co-director of my PhD thesis	alexandre.schmid@epfl.ch
İlker Hamzaoğlu	Assoc. Prof. at Sabanci University Electronics Engineering (EE) Department Director of my MS thesis	hamzaoglu@sabanciuniv.edu