AMERICAN UNIVERSITY OF BEIRUT

ENHANCED STEREO MATCHING USING IMPROVED CLASSIFICATION

by
MOHAMMED HUSSEIN BAYDOUN

A dissertation
submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
September 2014

AMERICAN UNIVERSITY OF BEIRUT

ENHANCED STEREO MATCHING USING IMPROVED CLASSIFICATION

by
MOHAMMED HUSSEIN BAYDOUN

Approved by:

[illegible]

Dr. Mohammed Adnan Al-Alaoui, Professor — Advisor
Electrical and Computer Engineering

[illegible]

Dr. Nesreene Ghaddar, Professor — Chairman of Committee
Mechanical Engineering

Ali Chehab

Dr. Ali Chehab, Associate Professor — Member of Committee
Electrical and Computer Engineering

[illegible]

Dr. Fadi Karameh, Associate Professor — Member of Committee
Electrical and Computer Engineering

[illegible]

Dr. Daniel Asmar, Associate Professor — Member of Committee
Mechanical Engineering

[illegible]

Dr. Lina Karam, Professor — Member of Committee
Electrical and Computer Engineering, Arizona State University, USA

[illegible]

Dr. Haidar Harmanani, Professor — Member of Committee
Computer Science, Lebanese American University, Lebanon

[illegible]

Dr. Maha El Choubassi, Doctor — Member of Committee
Electrical and Computer Engineering, Intel, USA

[illegible]

Dr. Rony Ferzli, Doctor — Member of Committee
Electrical and Computer Engineering, Intel, USA

Date of thesis defense: August 28, 2014

ACKNOWLEDGMENTS

In the name of ALLAH, the most Gracious, the most Merciful

My first and great thanks go to the One and only Almighty ALLAH because everything in this work has been performed based on his will and guidance.

Second, one must dedicate his thanks to those who have always helped, supervised, encouraged and guided him during every step. Professor Al-Alaoui is my advisor and mentor. He has always provided me with support and creative ideas. He helped mold my abilities and I am sure I have become a better person and a better scholar under his supervision.

One cannot but admit that the committee members have been very patient during the period of my work. They provided valuable ideas for improving the work and my skills. So, I greatly thank them for being very helpful and supportive.

My parents and family are always there to help and support me in every possible manner, I have to thank them and no matter what I say or do I will never do them justice. Thank you.

In addition, all my friends whether away or at AUB were supportive especially Mohammad Dawi, and every Doctor I took a course with or assisted. So, thank you all.

Last but not least, it is a pleasure to acknowledge The Lebanese National Council for Scientific Research for its critical support by providing this work with their essential grants. Without their help, this work might not have been possible. Additionally, Nvidia's hardware and software grants helped to launch The CUDA Training center at AUB and the course ECE696 Applied Parallel Programming at the ECE department. The above resources introduced me to GPU and CUDA and provided me with the main resources for employing GPU in performing parallel processing in my research and PhD dissertation.

AN ABSTRACT OF THE DISSERTATION OF

Mohammed Hussein Baydoun for Doctorate of Philosophy
Major: Electrical and Computer Engineering

Title: Enhanced Stereo Matching Using Improved Classification

This work aims at providing an accurate and fast stereo matching system. Stereo matching targets determining the depths of the pixels in an image(s) obtained through a stereo setup. Stereo matching is one of the most addressed problems of image processing.

The purpose here is achieved using varying notions that might seem disconnected, but they will be developed and related throughout the whole work. The different ideas mainly depend on using different image processing techniques and classification.

At first, the stereo images are treated using histogram information to reduce color differences. Afterwards, initial stereo matching is considered using various approaches with emphasis on basic ones that utilize measures such as the Sum of Absolute Distances (SAD) and Normalized Cross Correlation (NCC). Furthermore, edge based segmentation is proposed to improve on stereo matching, where the edge detection uses digital differentiator approximation.

Besides, different classification approaches are used to enhance stereo matching using a set of proposed features that are depicted from the stereo pair and the preliminary stereo matching. The classification methods, not exclusive to others, include boosting and others. And in order to advance boosting and similar methods, some modifications are proposed based on training and selecting classifiers. These propositions are experimentally proven to generally enhance the accuracy and performance of classification whether regarding stereo matching or otherwise.

The stereo images are primarily selected from the Middlebury stereo database to show the validity of the various propositions in accordance with previous literature.

A basic time analysis is provided for many of the ideas especially that speed is a major factor in stereo matching. Compute Unified Device Architecture (CUDA) on Nvidia Graphics Processing Units (GPUs) is used to show the real time performance of several of the suggested notions.

Moreover, several possibilities are discussed to provide directions towards future work.

CONTENTS

Page

ACKNOWLEDGMENTS vi

ABSTRACT vii

LIST OF ILLUSTRATIONS xii

LIST OF TABLES xv

Chapter

1. INTRODUCTION 1

    1.1. Overview of Related Ideas 1

    1.2. Methodology 6

        1.2.1. Evaluation 9

    1.3. Main Contributions 10

    1.4. Thesis Outline 11

2. REVIEW AND BACKGROUND 13

    2.1. Image and Stereo Matching 13

    2.2. Edge Detection 26

    2.3. Edge Based Segmentation 31

    2.4. Classification 33

    2.5. Stereo Matching and Classification 39

    2.6. 3D Reconstruction 41

    2.7. Parallel Computing and CUDA with the Related Problems 43

3. EDGE DETECTION AND SEGMENTATION 47

3.1. Edge Detection 47

3.2. Edge Based Segmentation 51

3.3. Discussion about Edge Based Segmentation 58

4. STEREO MATCHING 61

4.1. Initial Processing 61

4.2. Proposed Approach 66

4.2.1. Left to Right Cross Checking (LRC) 72
4.2.2. Sub Pixel Accuracy 73

4.3. Stereo and Segmentation 74

4.3.1. Direct Stereo Matching from Segmentation 74
4.3.2. Completed Stereo Matching Using Segmentation 78
4.3.3. Segmentation with adaptive support weights stereo matching 80

5. CLASSIFICATION...85

5.1. The Proposed Method 85

5.1.1. Modified Training 86
5.1.2. Combining Classifiers Search 87
5.1.3. Separating Classifiers Search 90
5.1.4. Discussion and Combining the Approaches 93
5.1.5. Simulated Annealing Based Approach 94

5.2. Classifiers or Learners 95

5.3. Real Life Scenario 96

6. STEREO MATCHING WITH CLASSIFICATION................99

6.1. Stereo Matching Using Classification 99

6.2. Proposed Features 101

6.2.1. Feature 1: Mean Intensity Difference 102
6.2.2. Feature 2: Mean Edge Disparity Value... 102
6.2.3. Features 3 and 4:Similar Disparities & Mean Disparity Difference. 103

6.2.4. Feature 5: Disparity Value ... 104
6.2.5. Feature 6: Candidate Accurate Disparity ... 104
6.2.6. Feature 7: Histogram Disparity Value Difference ... 105
6.2.7. Features 8 and 9: Regional Histogram Disparity Value Differences 105
6.2.8. Features 10 and 11: Local Histogram Disparity Value Differences . 106
6.2.9. Feature 12: Segmentation Feature ... 106

6.3. Completion Stage ... 107

7. 3D RECONSTRUCTION ... 110

7.1. Proposed Method ... 111

8. CUDA IMPLEMENTATIONS ... 118

8.1. Notion about CUDA ... 118

8.2. Edge Detection using CUDA ... 119

8.3. Stereo Matching using CUDA ... 121

9. RESULTS ... 126

9.1. Results on Classification ... 126

9.1.1. Datasets ... 128
9.1.2. AdaboostM1 with Selection Results ... 129
9.1.3. GentleBoost with Selecting Results ... 130
9.1.4. LogitBoost with Selecting Results ... 130
9.1.5. Bagging with Selecting Results ... 131
9.1.6. Discussion and Insight ... 132

9.2. Results on Stereo Matching ... 134

9.2.1. Discussion about Features ... 143
9.2.2. Discussion about Results ... 144
9.2.3. Advantages and Limitations ... 145

9.3. CUDA Related Results ... 147

9.3.1. CUDA Edge Detection Results ... 147
9.3.2. CUDA Stereo Matching Results ... 149

10. CONCLUSION ... 152

REFERENCES..154

ILLUSTRATIONS

Figure Page

1.1. Flow chart presenting the work 8

1.2. Main Tested Stereo Image Pairs 9

2.1. Stereo Camera Basic Setup 16

2.2. Image Disparity Example 17

2.3. Example of Left and Right Stereo Images with their True Disparities from Middlebury [1] 18

2.4. Example of Obtained Disparity Map Using SSD with Radius=5 21

2.5. Examples of the best Found Disparities from [1, 14] 22

2.6. Magnitude response of the FIR Al-Alaoui differentiator: Comparison with the IIR Al-Alaoui differentiator and other FIR approximations obtained by windowing techniques. [60] 29

2.7. Example of Edge Detection using Sobel, Al-Alaoui and Canny 30

3.1. Absolute magnitude error of the Optimized Al-Alaoui FIR and IIR filters 49

3.2. Edge Detection Obtained Images using Prewitt, Sobel, Al-Alaoui-O-1 and Al-Alaoui-O-2 51

3.3. Example of Canny Edge Detector Image 52

3.4. Example of using Bresenham’s algorithm for connecting objects for a size 5 kernel 53

3.5. Example of the Connected Edge Images 54

3.6. Example of the Updated Connected Edge Images 55

3.7. Example of a Segmented Image 56

3.8. Example of the Updated Segmented Image 57

3.9. Example of Updated Segmented Image without edges 57

3.10. Example of Downsized Segmented Image 58

3.11. Example of Directly Edge Segmented Image 59

3.12. Example of segmented object utilization 60

4.1. Histograms of Gray Scale Teddy [1] Images showing the different regions 63

4.2. SAD Stereo Matching with and without initial processing.................................. 65

4.3. BT Stereo Matching with and without initial processing..................................... 65

4.4. Example of a kernel and its 4 parts with Radius = 5.. 67

4.5. SAD, BT and NCC Remaining Confident Pixels 69

4.6. Examples of Voted Disparity Maps 70

4.7. SAD and NCC best distance amongst the four possible disparities...................... 71

4.8. Example of SAD completed disparity map (left and right) 72

4.9. Example of Right and Left SAD validated disparity maps.................................. 73

4.10. Example of Obtained Disparity Maps through SAD Based Segmentation........... 76

4.11. Example of Obtained Disparity Maps through SAD Based Segmentation with smaller objects... 77

4.12. Example of Obtained Disparity Maps through SAD Based Segmentation after filling the objects... 78

4.13. Example of Obtained Disparity Maps through the different stages (left and right)... 79

4.14. (a) Initial disparity map as by [19], (b) Initial map through gradient image, (c) Combined validated map, (d) Filled map through nearest to median value, (e) Validated map, (f) LRC validated map 84

5.1. Flowchart of Combining Classifiers Search 88

5.2. Example of the Combining Approach for T=7 and X=3 89

5.3. Flowchart of Separating Classifiers Search 91

5.4. Example of the Separating Approach for (T=7) and (X=3).................................. 92

6.1. Best Possible Map through checking neighboring disparities and the Correct Obtained Pixels 100

6.2. Initial Disparity Map and Obtained Disparity Map 101

6.3. Example of the Edge Image of the Disparity Map.. 103

6.4. Example of empty regions where each (gray level) pixel with a colored border being in the vicinity of the empty (white) region.. 108

6.5. Obtained Disparity Map through SAD+AdaboostM1+Completion 109

7.1. Stereo Pair from [155] with the corresponding disparity maps 110

7.2. Point 3D Reconstructed Scene 112

7.3. Basic Proposed Points Interpolation 113

7.4. Interpolated 3D Reconstructed Scene 114

7.5. Interpolated Disparity Maps.. 116

7.6. Reconstructed Images through Interpolated Disparity Maps 116

8.1. Explanation of the Algorithm Used for Filtering Images in CUDA for a (3 x 3) mask 120

9.1. Disparity Maps using SAD (Top Row) and through SAD+Proposed Approach (Bottom Row)... 136

9.2. Disparity Maps using NCC (Top Row) and through NCC+Proposed Approach (Bottom Row)... 136

9.3. Disparity Maps using [29] (Top Row) and through [29]+Proposed Approach (Bottom Row).. 137

9.4. Disparity Maps using [30] (Top Row) and through [30]+Proposed Method(Bottom Row) 138

9.5. Disparity Maps using [44] (Top Row) and through [44]+Proposed Approach (Bottom Row).. 138

9.6. Disparity Maps using [36] (Top Row) and through [36]+Proposed Approach (Bottom Row).. 139

9.7. Disparity Maps using segmentation with ASW 141

9.8. Disparity Maps using segmentation with ASW+ AdaboostM1 142

9.9. Disparity Maps using segmentation with ASW+ AdaboostM1 with feature 12... 142

TABLES

Table Page

1.1. Input and Output of Related Fields of Research 2

9.1. Datasets Basic Information 128

9.2. AdaboostM1 with Selection Error Results 129

9.3. GentleBoost with Selection Error Results 130

9.4. LogitBoost with Selection Error Results 131

9.5. Bagging with Selection Error Results 131

9.6. Results on the Middlebury Test Images Measured by Percentage Error Threshold Difference for Proposed Approach 139

9.7. Occlusion Handling Results on the Middlebury Test Images Measured by Percentage Error Threshold Difference 141

9.8. Results on the Middlebury Test Images Measured by Percentage Error Threshold Difference For The Proposed Segmentation with ASW 142

9.9. Example Parameters for CUDA Implementation 148

9.10. Portion In Overall Time Of Each Component For The Teddy MAP Computation Using SAD+AdaboostM1 150

9.11. Execution Time (ms) Comparison With Other Stereo Matching Methods 151

CHAPTER 1

INTRODUCTION

The work aims at constructing a considerably fast and accurate stereo matching system. Stereo matching is one of the most targeted problems in computer vision and image processing [1]. It has many applications that range from security applications that include face detection and recognition, Robotics, Sensing, Medical Technology, 3D reconstruction, and entertainment applications that include 3D systems, enhanced user interaction and many others.

Stereo matching is a branch of image matching that handles two corresponding stereo images obtained through a stereo camera or a stereo setup, and tries to find the disparity which is the horizontal difference in pixel positions, along the same row, between matching pixels in the stereo pair. This leads to determine the depth information of the image after using the camera or setup parameters.

There are many methods used for solving this problem, and this work tries to improve on the speed and the accuracy through different information that include the edges of the image. Besides, the work here implements new methods of classification with stereo matching. Although the ideas might not always seem connected due to the diversity of the methods used, this document will explain the different notions and connect them, but this requires the patience of the reader.

1.1 Overview of Related Ideas

Image processing has many definitions. Wikipedia [2] defines Image Processing as a form of processing data where the input is any image whether from a photograph or a video with the output being any required form of data whether an image or not. Another definition considers Image Processing as the enhancement or treatment of an image to achieve another image [3]. It is worth taking into account other fields that have great relations with Image Processing; these include Computer Vision, Computer Graphics, and Artificial Intelligence. Table 1.1 shows the inputs and outputs and outputs of these different fields.

Table 1.1. Input and Output of Related Fields of Research

Field Name	**Input**	**Output**
Image Processing	Image	Image
Computer Vision	Image	Description
Computer Graphics	Description	Image
Artificial Intelligence	Description	Description

An image is a two dimensional signal and thus, many of its processing techniques are mainly used in a similar manner to one dimensional signal processing which include frequency domain methods such as Fast Fourier Transforms (FFTs). Image processing is a constantly growing and developing research field with great interrelations with the previously noted fields.

The applications of this research field are infinite. They include basic photographic applications such as printing, enhancing images' resolution, coloring, to a wide range of medical imaging applications that use X-Rays, Magnetic Resonance Images (MRIs), or Microscopic images like identifying different viruses, diseases and conditions. Also, the applications include face detection and recognition with its own different applications in addition to object and motion detection. So, clearly, there are never-ending possibilities of applications and in many cases some famous applications are fields upon themselves.

Edge detection is one of the main and early parts of image processing [2], since edge information often encompasses highly relevant data relevant to the required application. Most edge detection methods use masks that are correlated to the image. Further processing involves filtering and picking the better edge candidates.

Edges are used for many purposes. The applications include Quality Control where the silhouette of a certain object is found to check for errors. Also, they can be used for video and image compression. Additionally, they are used for object detection, image segmentation, and as an input to many classification tasks.

Another domain of image processing that is highly important to underline here is Image Matching. Image matching or registration is one of the challenging problems in image processing. Given two comparable images, taken, for example, at different times, in different conditions or perspectives, the goal is to determine a reasonable transformation even if for every pixel, such that a transformed version of the first image is as much similar as possible to the second one. The first image is considered as the reference and the other the target. Thus, it can be stated as finding the corresponding pixel or transformation such as translation for every pixel in the reference image.

There are many areas which benefit from registration, including biology, physics, chemistry, criminology, genetics, art, astrophysics, and basically any area involving imaging techniques. More specific examples include remote sensing (e.g. generating a global picture from different partial views), security (e.g. comparing current images with a data base), robotics (e.g. tracking of objects), and, in particular, medicine like in radiation therapy, MRIs, or treatment verification.

A particular part of this field includes Stereo Image Matching, which is the main interest here, where the matching process is done between two images taken by a stereo camera. These images are matched to determine the depth of every point or pixel in the two images. The pixels or picture elements in this case belong to the same row which renders the problem easier than the original problem of registration where the pixels move in all directions. This is a research area on its own with many applications. A direct application of stereo matching is 3D reconstruction and modeling.

3D Reconstruction in the field of image processing, as its name indicates, aims at rebuilding the imaged scene based on the different information retrieved from the image(s) being worked upon [2]. There are many applications here and they include entertainment such as modeling objects or faces for games, 3D printing, object recognition including face recognition which leads to face modeling, and other applications such as those used in medical fields or augmented reality, etc…

At a different level, classification is a huge domain on its own. It is directly related to machine intelligence and is simply defined as the process of determining the type of a certain input based on the input's properties or features amongst the different possible types. Classification can be supervised and unsupervised, where the former deals with a finite set of known types or classes with training samples of known classification, and the latter tries to infer the data similarities with the feature values and define the types and classes themselves through clustering. In this work, we concentrate on supervised classification with the supervised term directly associated with the learning or training stage, where the training data features and classes are known and used to create the classifier and any new input or feature vector needs to be classified. Of course, this field is very wide and constantly updated with the implementations used in many fields including Image Processing, Computer and Machine Vision and many others.

On another level, parallel computing or programming is another field of research. It involves creating a parallel version of the required algorithm and implementing it on a capable hardware [2]. Whilst a serial or regular program runs serially or step by step on the host hardware, the parallel program should exploit independencies in the algorithm to speed-up the implementation according to the limits of the hardware. For example, instead of summing two arrays in a “for loop”, each element is summed on its own according to the parallel architecture limitations, and thus the speedup would be the size of the array. Parallel computing is one of the recent fields of study for any application or algorithm especially that multi-core and many-core programming is currently a growing area of research aiming to speed up processing.

There are many applications for parallel computing since there are various candidate parallelizable algorithms. It is worth noting that this field is constantly updated nowadays especially due to the availability of parallel architectures even on the Central Processing Unit (CPU). This is mainly due to current power and energy constraints in transistors and hardware development.

Of course, many methods or architectures exist for the purpose of parallel computing. These include programming on the Field Programmable Gate Arrays (FPGA), Graphics Processing Units (GPU), Multi-Core CPUs, Networks or Clusters, Super computers, and many others developed for this purpose. It is worth noting that there are many languages that are used for parallel programming and the two major ones that rely on CPU usage are Message Passing Interface (MPI) and Open Multi Processor (OpenMP). In this work, we use the GPU and thus we concentrate on it.

In the GPU field, two major companies exist. These are Advanced Micro Devices (AMD) which owns Array Technologies Incorporated (ATI) on one hand and Nvidia on the other. And as any industrial field, the competition between these two companies is great and this includes pleasing the parallel programming users. After the emergence and popularity of General Purpose GPU programming (GPGPU) that used the GPU to perform parallel computations with the main languages being OpenGL and DirectX, the companies tried to exploit the capabilities of their hardware by introducing a suitable parallel programming API. Initially, Apple introduced Open Computing Language (OpenCL) which was able to utilize the parallel processing abilities of the GPU and they even collaborated with ATI and Nvidia. Afterwards, Nvidia developed CUDA especially that there was rather little progress in the relatively complicated OpenCL.

CUDA stands for Compute Unified Device Architecture. It is a parallel architecture/software developed for the GPUs solely made by Nvidia in order to boost the parallel capabilities of the GPU previously known as Visual Graphics Adapter (VGA) after the rise of GPGPU. It is worth noting that CUDA translates into PThreads before directly running on the GPU and in this, it is similar to OpenCL.

Although these different ideas might seem hard to relate, the different fields are greatly related in practice and they will be explained in the coming chapters with their relations.

1.2 Methodology

The previous section presented different topics related to the addressed ideas of this work. As already emphasized, considering all the ideas together might be overwhelming, but dividing them and showing the contribution of each will help present a better view.

The system is composed of different stages that together aim to achieve a fast and accurate stereo matching system. The main stages of the proposed work are shown in Figure 1.1.

At first, the input is handled. This is mainly the stereo image pair that can be a couple of stereo images or videos in addition to other parameters such as the disparity range or the maximum value that a pixel can be displaced. Then the images are preprocessed to enhance the possibility of stereo matching based on histogram information. Afterwards, an initial disparity map is computed, which can be achieved through any stereo matching method as later explained. Also, edge based segmentation is used to enhance stereo matching. The segmentation approach is not necessarily a part of the system especially that it requires additional time, but it presents an improved approach for targeting stereo matching. Then, a feature vector for every pixel is built using the different obtained information. The information mainly consists of the input data and the computed disparity map with various data derived from them. The feature vector is used in a Classification stage which leads to obtaining a new enhanced disparity map. This disparity map can be used for 3D reconstruction or other implementations. The different stages are composed of various algorithms that are candidates for parallel processing. So, some of the presented ideas are parallelized using CUDA.

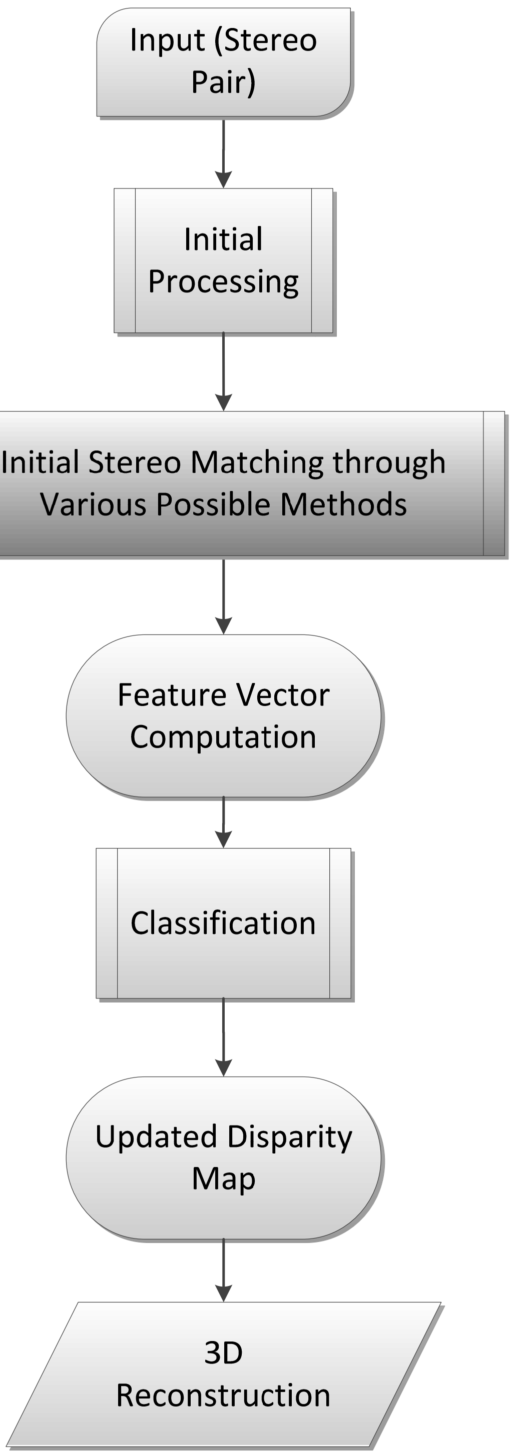


Fig.1.1 Flow chart presenting the work

1.2.1 Evaluation

In order to evaluate the performance of the proposed ideas especially in regards to stereo matching, the work concentrates on images selected from [1] which is the most cited reference related to stereo matching and in particular four of the images depicted from there are used. These include the teddy, cones, tsukuba and venus shown in Figure 1.2.

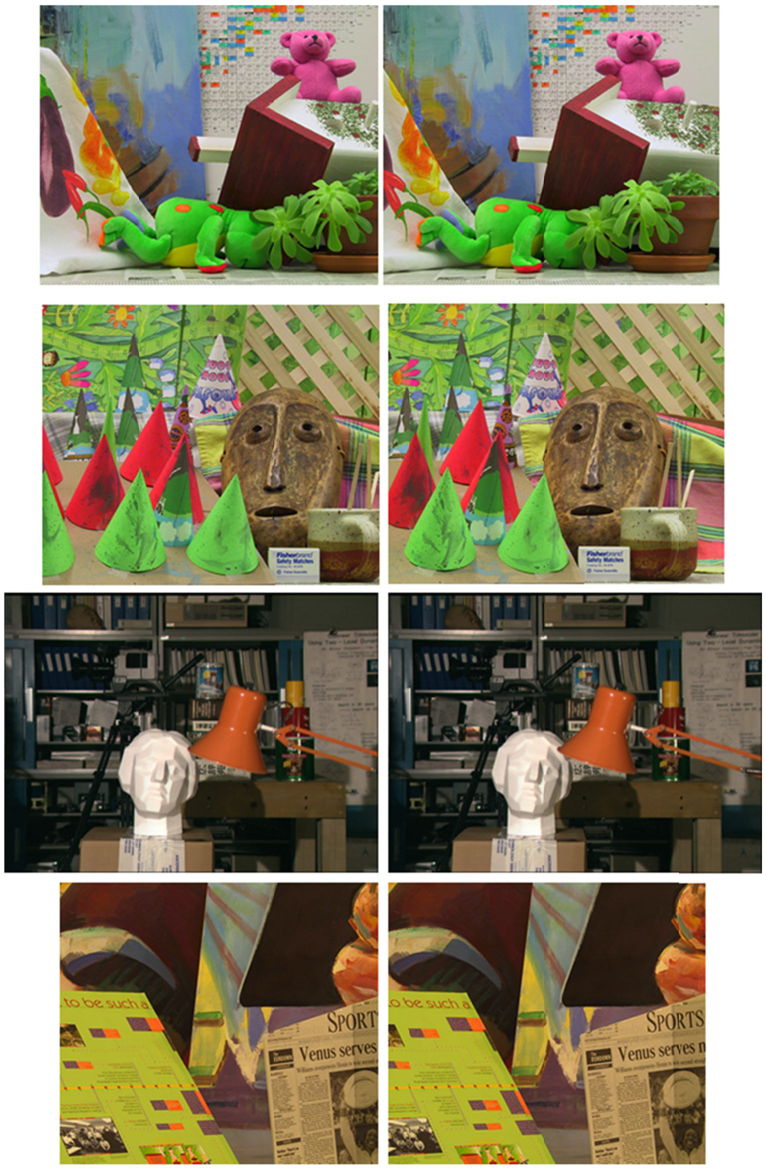


Fig.1.2 Main Tested Stereo Image Pairs

Of course, it is further possible to test any of the other available pairs but this is not done in this document. Besides, although the colored versions are shown, this work concentrates on using the gray scale figures since it should be enough in best case scenarios and the same principles can be used for the color versions. Furthermore, the work considers the low resolution versions of the images since these are generally the harder ones and at the same time these would be the faster ones to compute and verify.

It is important to mention that the literature, especially aiming at high accuracy, generally uses the high resolution colored versions of the images or their derivatives as in modified color spaces like YCbCr instead of RGB. Of course, this means that such scenarios would lead to better results, but all of this is at the expense of time, which is the second major factor, besides accuracy, in Stereo Matching. And in this work the emphasis is on both accuracy and speed, so the work generally concentrates on getting the best out of the gray scale low resolution images which is both easier and harder at the same time. This is clearly harder since there is less information but it is also easier since fewer choices generally limit the problem and render the possibilities easier to handle.

1.3 Main Contributions

After discussing the main ideas, it is essential to highlight the different contributions especially due to the diversity of the targeted notions.

Stereo matching is initially preprocessed through a new method that uses the histograms of the stereo images to compute a measure and after that implements histogram matching.

Furthermore, a new edge based segmentation method is proposed. This utilizes a new edge filtering mask. Segmentation has many applications, but is mainly used in this work to present an improved approach for obtaining relatively accurate stereo matching.

Moreover, this work presents a new generic approach that uses classification as a post processing stage to enhance stereo matching by remedying the erroneous disparities.

Various classification methods are used in this work with emphasis on boosting. So, and as another contribution, a new additional phase is proposed to improve the accuracy of boosting and similar ensemble learning methods.

Besides, and since we consider obtaining a fast system, several parts of this work are implemented on parallel CUDA devices.

1.4 Thesis Outline

The main objective here and as noted throughout the work is to obtain a capable stereo matching system using different image processing techniques and through classification. And the next chapters should further elaborate this work and clarify the ambiguities.

The remaining chapters present the different ideas noted in this work. In Chapter 2, a literature review is considered for the different fields and ideas addressed. Chapter 3 deals with edge detection and edge based segmentation. Then, Chapter 4 concentrates on stereo matching and discusses its different possible approaches in addition to the main image processing techniques used in this work. Afterwards, Chapter 5 deals with classification on its own and shows the work done in regards to this field. Next, Chapter 6 presents the adopted approach that aims to enhance stereo matching through classification in addition to the proposed feature vector. After that, Chapter 7 presents a brief notion about 3D Reconstruction. Chapter 8 tries to discuss the parallel implementation of some of the main procedures based on using the Nvidia GPU CUDA architecture. Then, Chapter 9 discusses the different results for the diverse ideas handled here. Finally, Chapter 10 presents the conclusions and some ideas for future work.

CHAPTER 2

REVIEW AND BACKGROUND

The main objective here is solving the stereo matching problem in an efficient, accurate and fast manner. And for this purpose, several areas of research were addressed as already noted in the previous chapter.

Each research area is initially treated separately for review purposes and then the main relations between the different ideas are handled in the coming chapters. In any case, the handled fields are interrelated and this work is not necessarily the first to use them together. Besides stereo matching, the fields include different image processing problems that include edge detection, and segmentation. The fields include classification which is a massive domain on its own. Also, parallel computing, which is constantly updated to provide faster algorithms and procedures, is a field of interest in this work to increase the speed of computations. Moreover, 3D reconstruction which is a part of Computer Graphics is briefly considered to show an example of applying the stereo matching problem.

Thus, this chapter tries to present a general review of the different areas handled in this work.

2.1 Image and Stereo Matching

Image matching or registration for 2D or 3D images is a wide area of research with varying implementations as the literature indicates and the following review is not intended to be complete but to describe some of the important and relevant works in this active field of research.

This problem is considered essential in the fields of Image Processing and Computer Vision [4]. The different applications include medical imaging like Single-Photon Emission Computed Tomography (SPECT) Brain images [5], Cardiac Magnetic Resonance images [6] and other types. In addition to that, topography is an important application [7]. Additionally, this field is related to 3D and security applications that include 3D reconstruction from multiple images [8].

Many methods exist for solving this problem which is usually divided into two categories. The first is rigid or affine due to the nature of its transformations [9]. This type usually involves a set of known transformations which include translation, rotation, and scaling that are applied to the input image in order to match it with the reference image. This is performed in a matrix form which means that the optimization is done for the image as a whole for a certain set of parameters. The other type is often referred to as non-rigid [10] or deformable [11] registration. This usually includes finding the translation vector for every pixel one by one which is more demanding computationally than the previous kind. In [12], the difference between rigid and deformable registrations is presented.

In this work, we concentrate on stereo matching which is a particular case of image matching [2]. The aim of this greatly targeted problem, and as previously mentioned, is to compute the disparity or depth map.

Of course, stereo matching is not the only method to find the depth since there are many that rely on using different sensors and data. These mainly include laser scanning [13], structured light [1, 13], and using binocular or stereo vision [1, 13] which is our main issue of concern. In addition to these, there are other generally more complex methods, such as depth from motion using a single camera, stereo from video and sensors, or using multiple cameras or even a combination of the previous methods.

The different noted methods vary in terms of different aspects that include different requirements such as setup, accuracy, simplicity and cost. The laser scanning and structured light methods require an enhanced setup and cannot be constantly used. Thus, in the different literature, stereo matching is the least expensive, and rather user friendly, but definitely the least accurate [1, 13]. Therefore, much research is constantly proposed in order to improve the two aspects of stereo matching which are accuracy and computational speed. The accuracy of stereo matching is considered always behind those of the other methods, which are in many cases used to compare and determine the viability of the proposed algorithms of stereo matching [1, 14]. It is worth mentioning, in regards to speed, that obtaining real-time stereo matching is an important target since it is of key importance in the different applications. Thus, there is constant work on improving these ideas.

In particular, stereo matching is directly related to human stereo vision. Since humans have two eyes, they help them better visualize depth. It is notable that with just one of the eyes open, depending about the brain nature of the person, it could be harder to note the depth of objects in comparison with having both eyes open.

Stereo matching usually utilizes a stereo camera, which is a camera with two or more lenses producing images that simulate the binocular human vision and estimate 3D positions [2]. There are many possibilities for the lenses, as in the optical axes which can be parallel or converging, or the distance between the lenses which is usually similar to the distance between the human eyes, etc…

An example of the basic stereo matching setup is shown in Figure 2.1.

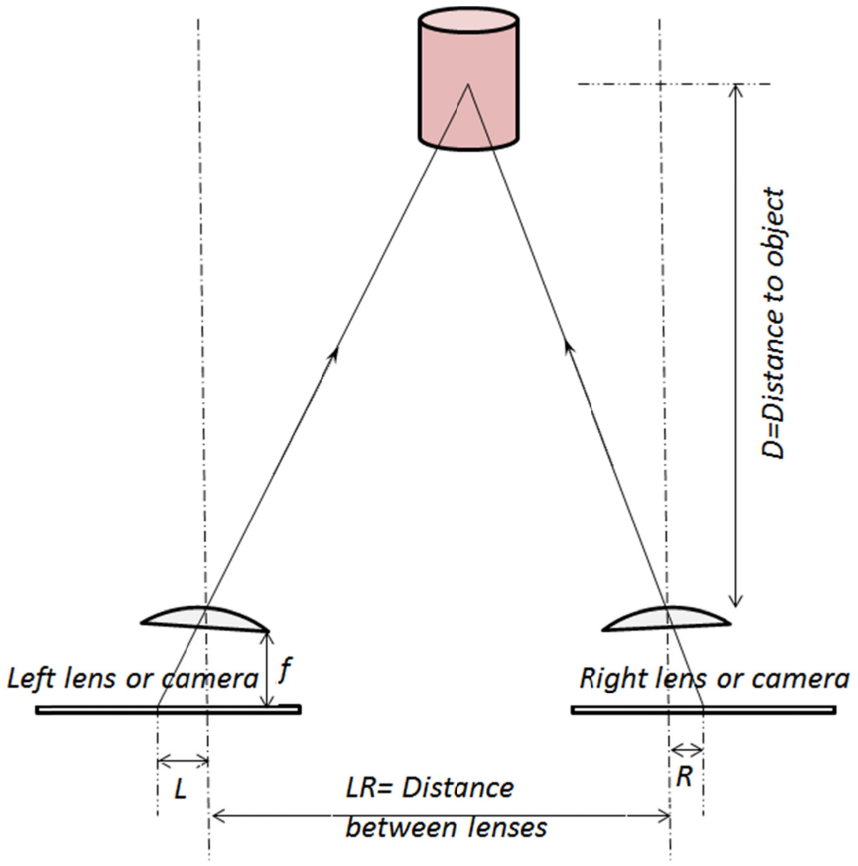


Fig.2.1. Stereo Camera Basic Setup

The following terms are defined according to Figure 2.1:

D: Distance to object

LR: Baseline distance between the stereo lenses

f: focal length of the lens which should be similar for both left and right

d: difference in pixel positions or the Disparity (difference between L and R)

By relying on the pinhole camera model and using similar triangles' theory and parallel theorems, it is directly verified that:

$$D = \frac{LR * f}{d} \tag{2.1}$$

In simple terms, stereo matching is an Image Processing task that is basically aimed at finding the matching pairs of points between the left and right images of the same scene. This is initially done by finding the corresponding pairs of pixels between the images. And by doing so, one determines the disparity and thus the 3D position of each pixel in each of the images. This can be noted in Figure 2.2.

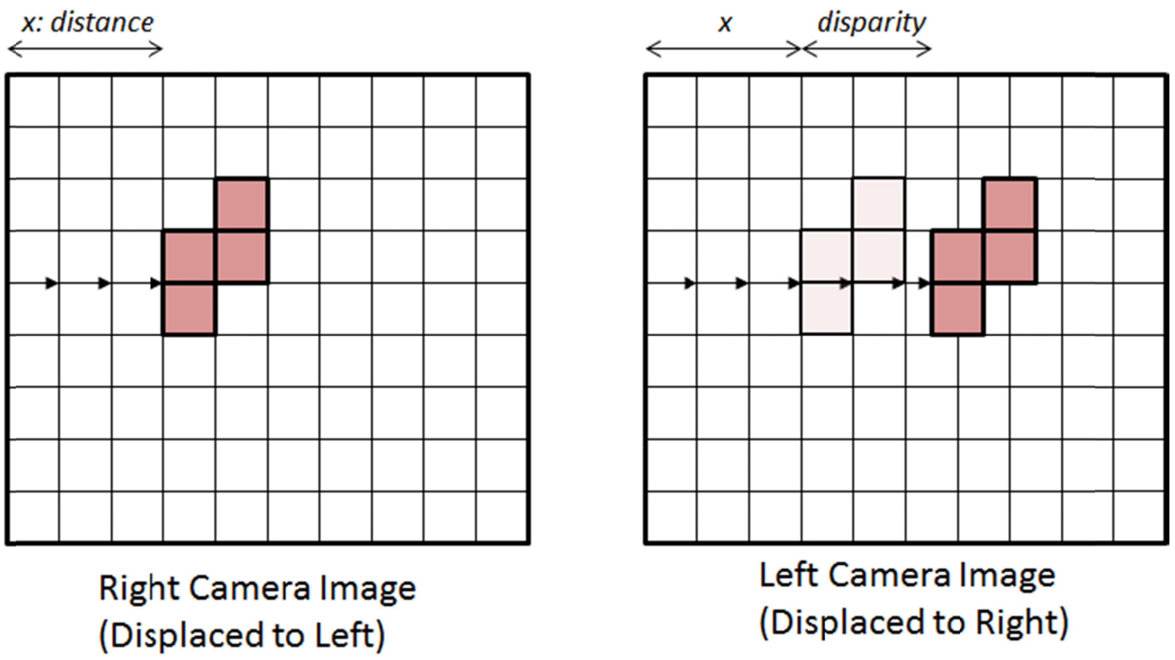

x: distance
x
disparity
Right Camera Image (Displaced to Left)
Left Camera Image (Displaced to Right)

Fig.2.2. Image Disparity Example

So, we need to compute *d* or the disparity from the images and afterwards we can use the camera parameters and determine *D* which corresponds for the depth dimension.

Thus, knowing *d*, *LR* and *f* one can, in many cases, find the third dimension or *D* according to equation (2.1) based on the pinhole camera model and considering the stereo camera setup.

It is worth noting that the higher the disparity the closer the object is to us. This can be seen in the true disparity images shown in Figure 2.3 or the computed disparity maps shown later such as in Figure 2.4.

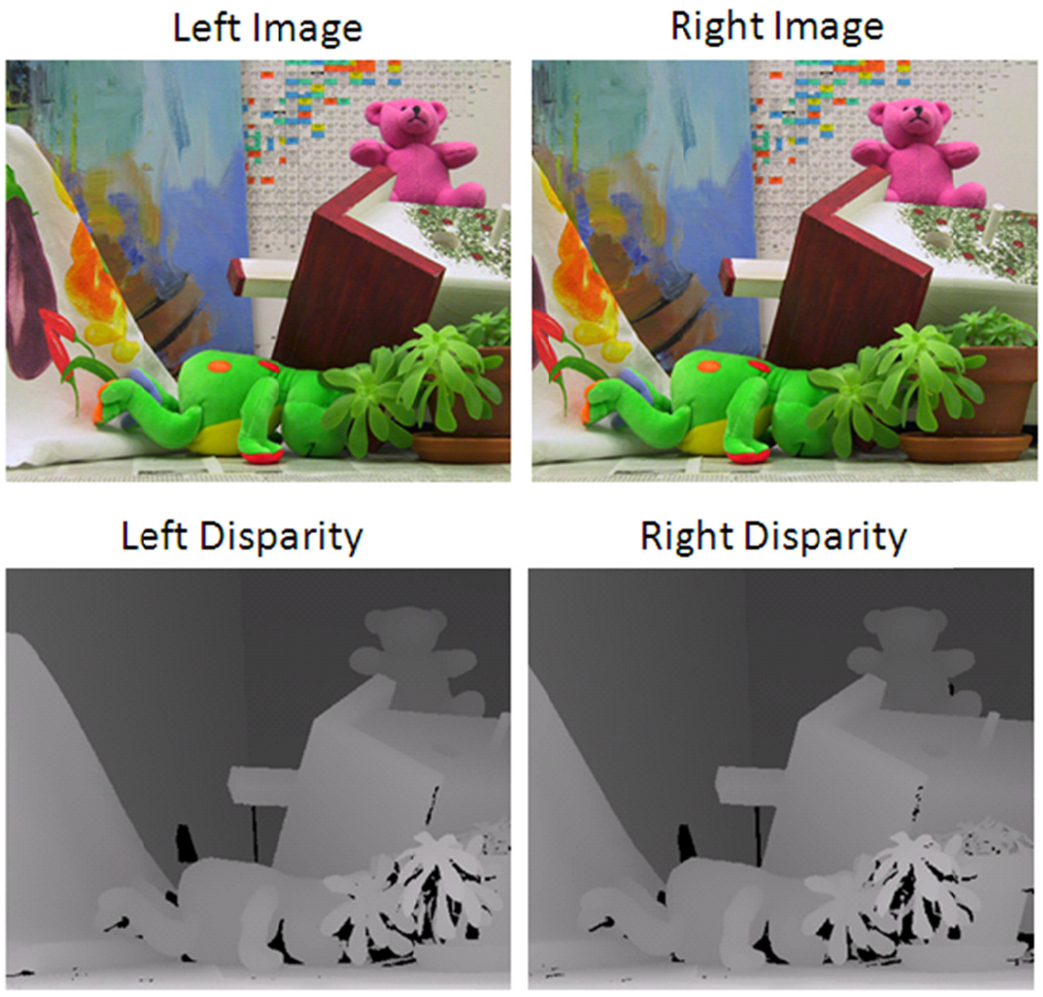


Fig. 2.3. Example of Left and Right Stereo Images with their True Disparities from Middlebury [1]

There are many methods to tackle this famous matching problem. The different methods used to solve this problem are divided into two main types that include local and global matching methods. Also, there are other methods that are termed semi-global as they use a combination of both methods.

Concerning the local methods, these usually implement a measure of the difference between the pixels. These include the main three measures of Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), and Normalized Cross Correlation (NCC).

$$SAD_{x,y} = \sum_{i=x-R}^{x+R} \sum_{y-R}^{y+R} \left| L_{(i,j)} - R_{(i,j)} \right| \tag{2.2}$$

$$SSD_{x,y} = \sum_{i=x-R}^{x+R} \sum_{y-R}^{y+R} (L_{(i,j)} - R_{(i,j)})^2 \tag{2.3}$$

$$NCC = \frac{1}{n-1} \sum_{i=x-R}^{x+R} \sum_{y-R}^{y+R} \frac{(L_{(i,j)} - \bar{L})(R_{(i,j)} - \bar{R})^2}{\sigma_L \sigma_R} \tag{2.4}$$

The different terms in these equations correspond to the following:

L is the left image

R is the right image

n is the number of pixels

$I_{(i,j)}$ is a pixel in the image

$\bar{I}$ is the mean of image I

σ is the standard deviation

In addition, there are other difference measures that use more complicated and computationally demanding distances such as the census transform, mutual information, etc…

Besides these, there are other local methods that employ an improved distance based on the original ones such as the (BT) measure presented in [15] which is an improved version of (SAD) that is less sensitive to image sampling and noise.

Furthermore, many local methods use adaptive measurements like different windows for the (SAD) method [16]. Some of these are summarized in [25]. Moreover, in [18], the authors presented adaptive support weights (ASW) as opposed to windows with varying size which led to several related research. The weights are based on intensity and distance relations. As an example of related work, [19] used segmentation with adaptive weights while considering intensity differences. Their work proved important for stereo matching. Part of this proposed work related to segmentation and stereo matching relies on the work in [19] as the main related approach. Besides, several works used over-segmentation with stereo matching. For example, [20] used Belief propagation (BP) with over segmentation to achieve accurate stereo matching. In [21], the stereo map was computed using soft segmentation with B-splines as coherent stereo surfaces. The work in [21] led to relatively accurate results using stereo matching and segmentation while targeting both areas. In [21], each object was considered to have a uniform 3D disparity plane, color model and a 3D connectivity model.

In regards to global matching methods, there are definitely many ones, and these are mainly based on the implementation of different optimization techniques. These usually consume more time because they account for the pixel's disparity in addition to the neighboring disparities and this is the reason they are called global. Some of these rely on graph theory such as [22]. Others use cooperative optimization like [23, 24]. Furthermore, some use nonlinear optimization techniques like [25, 26]. Moreover, others use Markov Chain based methods like [27]. Also, there are many works that rely on BP such as [28], [29] and [30]. Besides these, others have used a combination of the different global approaches such as [31] which used BP and Markov methods; while others combined local and global methods as in [32, 33].

Additionally, there are semi-global approaches as in [34] in which Mutual Information was used. Others are based on minimizing different energy functions such as [35] where the authors used an energy function based on the (BT) cost. Also, the work in [36] implemented a new distinctive similarity measure that combined local and global properties.

In regards to combining edge detection and stereo matching, it was previously used in some works. The authors of [37] used Sobel masks in edge detection for colored images. In [38], edge detection was used to classify the image into levels and then use the levels for stereo matching. While in [39], edge detection was combined with segmentation for stereo matching.

As a very basic example, Figure 2.4 shows the corresponding right and left SSD disparity maps of the Teddy Images shown in Figure 2.3. From this, it is clear that there is some considerable difference between the maps and there is room for improving the results.

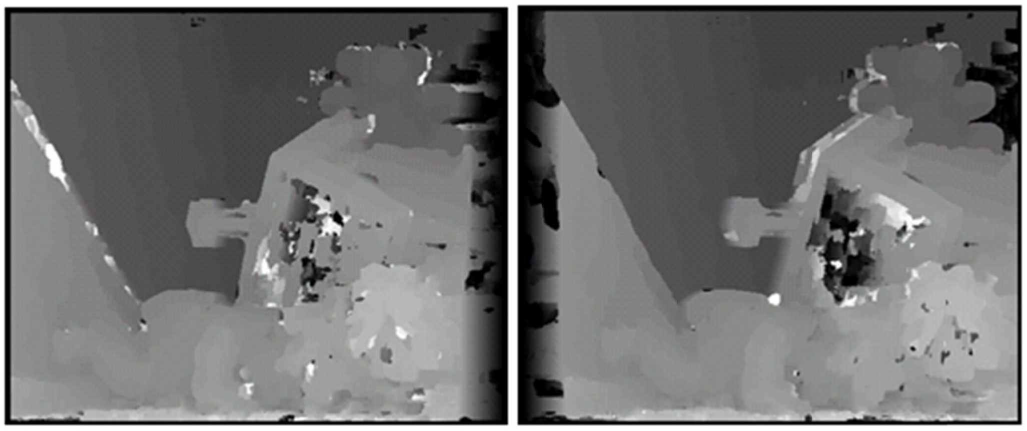


Fig.2.4.Example of Obtained Disparity Map Using SSD with Radius=5

It is worth noting some of the best methods used as per the Middlebury website, with some of the results shown in Figure 2.5.

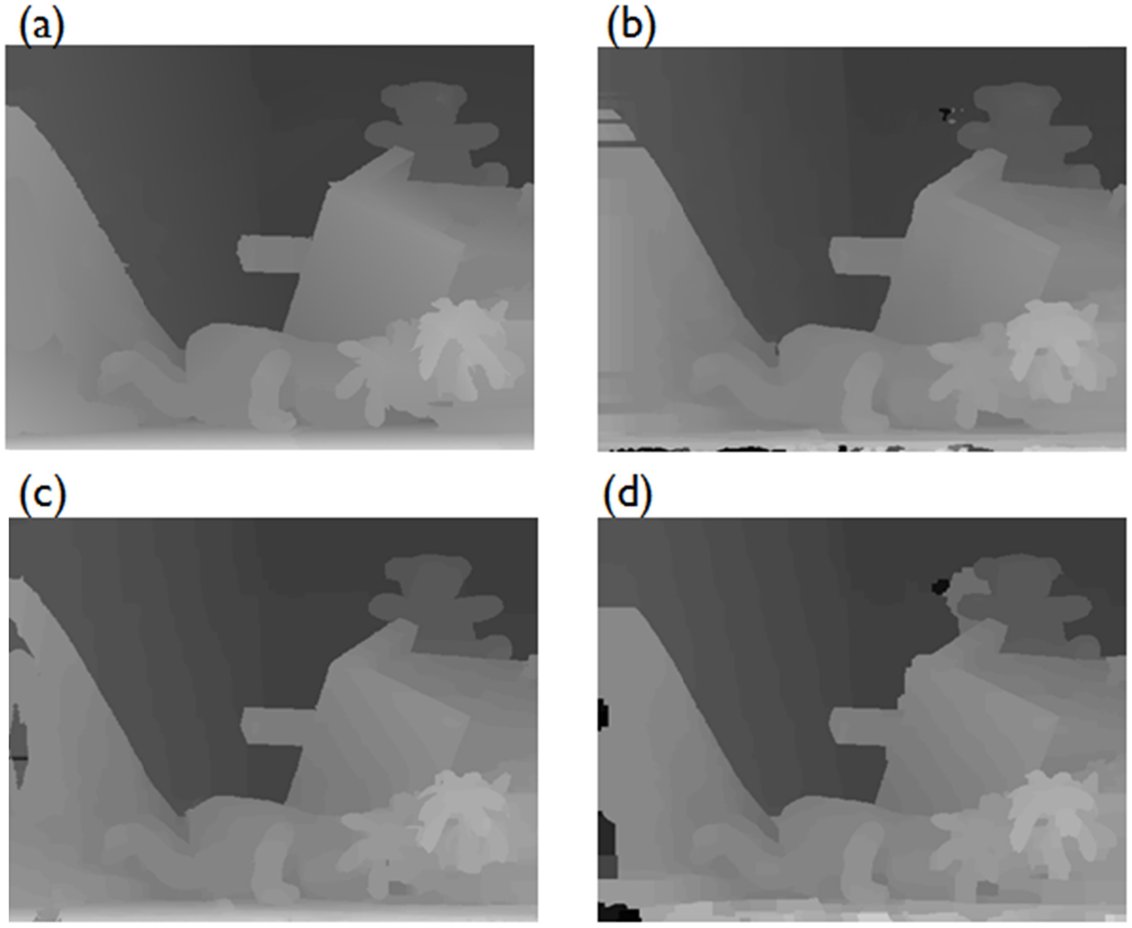


Fig.2.5.Examples of the best Found Disparities from [1, 14]

Through this explanation, it is emphasized that the two main properties of a capable stereo matching approach are accuracy and speed.

Concerning accuracy which is usually measured according to the Middlebury test images, the top performing methods are global ones. So, it is possible to use image segmentation. Furthermore, it is possible to combine information obtained from local methods to obtain an initial accurate set of points and these can be later used to obtain a better estimate of the remaining points. The accuracy can be further improved by considering the disparities of the edges which are usually the points where the disparity changes and this is another useful concept [36].

The main faced accuracy related problem in stereo matching is occlusion where some pixels appear in one image and disappear in the other due to varying depths. The occlusion related problems are mainly solved by computing all the disparity maps and validating the matching pixels in the images. This is termed left to right cross-checking (LRC). Some of the main ideas related to occlusion are summarized in [40]. Also, there are many works that concentrate on occlusion such as [33].

Another important issue is the resolution of the cameras that could be lower and unable to represent the imaged scene. Furthermore, the resolution of the different lenses could differ and any slight difference could be a problem. These kinds of problems all fall under illumination related issues, which is an important problem in image matching. Moreover, a problem lies in the discrete nature of images which does not allow for exact measurements which is directly related to sub-pixel accuracy. The illumination or color inconsistency problem is handled in this work through histograms as later explained.

Also, an issue lies in that very far objects have very small disparities which renders determining their disparities hard and reduces accuracy.

And although the different methods denoted so far do not depend on using the histograms of the stereo pair, histograms are widely used in stereo matching literature whether for local, global or semi-global approaches.

In [41], contrast context histogram was used with segmentation as part of an initial stage for stereo matching.

Also, in [42], co-segmentation which is basically segmenting similar parts of the image pairs was done and then matching the histograms of different parts was used for computing disparity.

In [43] a method that matches line segments between un-calibrated stereo pairs was done, and also used Color histogram matching as part of a scheme to match segments.

In addition, the work in [44] reformulated the cost aggregation problem of matching from the perspective of a histogram aiming at improving accuracy and complexity.

Also, and in regards to illumination and related color problems, much research tried to address these ideas with and without using histograms. In [45], the work performed a study on different matching costs, whether, local, semi-global or global to evaluate them under varying intensity changes that include lighting, vignetting and others like noise. Furthermore, in [46], interactive segmentation was used before using a Histogram-Based Matching Cost to enhance stereo matching even with different illumination conditions.

The work in [47] used an initial estimate for the disparity of SIFT points and built upon a histogram equalization method to have similar color stereo images that enhance disparity computations even for images with radiometric differences. In addition, [48] dealt with stereo matching through a histogram based perspective. Moreover, [49] mainly used histogram equalization followed by a tophat filter to reduce the effect of different illuminations. Of course, there are several other ideas that aim to obtain accurate stereo matching methods, but listing them is beyond the scope here.

In addition to being accurate, the stereo matching system needs to be fast and achieve real-time performance. The frame rate is usually 25 or 30 frames per second (fps) and thus a single frame consumes 1/25 seconds which is 40 milliseconds, or 1/30 seconds which is 33 ms. These numbers are indeed a hard target and are usually achieved in the literature with rather low accuracy.

Performing stereo operations on the CPU serially consumes a lot of time based on the programming platform even if done using C, C++ or Fortran. Furthermore, even using multi core programming on the CPU through MPI or OpenMP cannot lead to a performance that is more than *N* times the serial CPU version where *N* is the number of available CPU cores. Thus, achieving a high speed similar to the video frame rate is almost impossible with current CPU properties [49].

Since a simple implementation of the SSD on the CPU could require more than 0.5 seconds based on the frequency of the CPU, using a parallel platform is inescapable and as the literature indicated, CUDA stands out to be a suitable candidate. It is worth noting that the FPGA is considered as a major candidate for obtaining speedup in this problem, but it has some drawbacks which mainly include the data transfer problem between the FPGA and the camera system especially that FPGAs have limited capacity. Other problems include FPGA's availability and the rather hard implementation, although this is becoming easier with time due to the increasing popularity and the availability of new enhancing tools that utilize Matlab and Simulink amongst others.

In addition to choosing the appropriate architecture and programming language, the developed algorithm must be parallelizable and adequate for use on a parallel architecture, but thankfully and due to the nature of the problem, it is not hard to parallelize, or even pleasantly (embarrassingly) parallel, which is a term used for these kinds of problems that are usually available in the image processing field. This is mainly due to the image being an array of pixels and similar operations are performed on each pixel. Also, the algorithm should not be very complicated, involving many stages and serial parts, which would render the algorithm slower.

Moreover, and since videos usually do not have extremely varying pixel values from frame to frame, this could prove very useful based on the type of video, which would lead to a speedup of (space size)/(new space size). Of course, this is not always valid but can be considered true for a certain set of frames based on the type of the video. This issue is usually called temporal propagation of information. It is further important to mention that there is much research is this field especially that is similar to motion estimation.

2.2 Edge Detection

The main work done here in regards to edges is related to the review in this section. Edge detection is one of the main and early parts of image processing and computer vision [2, 3], since edge information often encompasses the relevant data. Most edge detection methods use masks like the Prewitt, Sobel, and Difference of Gaussian filters that are correlated to the image with further processing involving filtering and picking the better edge candidates as in the Canny [50] algorithm which can be used for obtaining accurate edges.

Edges are used for many purposes as previously argued for. The main approach used to determine edges is through differentiation as they indicate differences between pixels. The standard method used for edge detection passes an *n* by *n* filter which is a discrete approximation of the differentiator and convolves it with the image in two forms, first as it is and then with the mask transposed. The main used masks include the Prewitt and Sobel masks as noted in the equations

$$\mathrm{Pr}ewitt\_x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \ \& \ \mathrm{Pr}ewitt\_y = \left(\mathrm{Pr}ewitt\_x\right)^T \tag{2.5}$$

$$Sobel\_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \ \& \ Sobel\_y = \left(Sobel\_x\right)^T \tag{2.6}$$

Where *(T)* designates the transpose operation.

As already noted, Canny [50] edge detection is used for obtaining accurate edges. Canny uses the following four stages algorithm for finding edges.

At first a simple filtering operation for smoothing the image and reducing the noise is performed. The algorithm then applies a given mask which is usually the difference of Gaussian to filter the image. Non maximum suppression is then performed, in which the local maximum is chosen from the output of the previous stage according to the gradient angle found. For example, in case the gradient angle is 0, the edge is validated if the magnitude value is greater than other counter directions. Thus, and after this stage we obtain a rather thin edges image.

A thresholding stage is performed afterwards, where only larger intensity gradients are classified as edges. This is usually done using two thresholds (low and high) and is called Hysteresis Thresholding. Since in most cases, edges are connected, the larger threshold is applied first to mark the edges using the information obtained from the previous stages, and while going over an edge the smaller threshold is applied to find edges even if they are at a low value as long as they are connected to the edges. Thus, and after these different stages, the edge image is obtained.

Another edge detection method was created by Deriche [51] which is similar to [50], except in the initial stages since a special IIR filter is performed to the image instead of the mask.

On another hand, digital differentiators are useful components in many engineering applications such as communications, and biomedical applications as noted in [52-57]. Edge detection uses digital filters in the processing of the images. The filter used for this purpose can be an FIR or IIR digital filter, and since this work includes the implementation of a new filter, it is important to include a review of related ideas.

The Al-Alaoui integrator [58] has proven to be the best performing digital first order IIR integrator when compared to the analog integrator noted in (2.7).

$$H_{Int}(\omega) = \frac{1}{j\omega} \tag{2.7}$$

This was obtained by interpolating the trapezoidal and the rectangular integration methods leading, for a period value equal to 1, to the digital integrator in (2.8).

$$H_{Int}(z) = \frac{7(z+1/7)}{8(z-1)} \tag{2.8}$$

Inverting this filter led to obtaining the Al-Alaoui differentiator in (2.9).

$$H_{Diff}(z) = \frac{1.1429 - 1.1429z^{-1}}{1 + z^{-1}/7} \tag{2.9}$$

Al-Alaoui [59] approximated the Al-Alaoui IIR differentiator to an FIR filter to be used in edge detection. The usage of an IIR filter can and was used for edge detection as by Deriche [51], but since the mask operation is rather preferable for parallel computing the work here and as in [59] deals with FIR filters. And so, unlike usual edge detection which use masks based on FIR differentiators, the IIR filter was tested for edge detection, then transformed into an FIR filter shown in equation (2.10).

$$H_{Diff\_FIR}(z) = 0.36z^2 - 0.42z + 0.06 \tag{2.10}$$

This obtained filter is compared to the original IIR one in equation (2.8) with other FIR approximations of the differentiator obtained using several windowing techniques like the Hamming, Hanning, and Kaiser windows with the obtained filters presented in Figure 2.6.

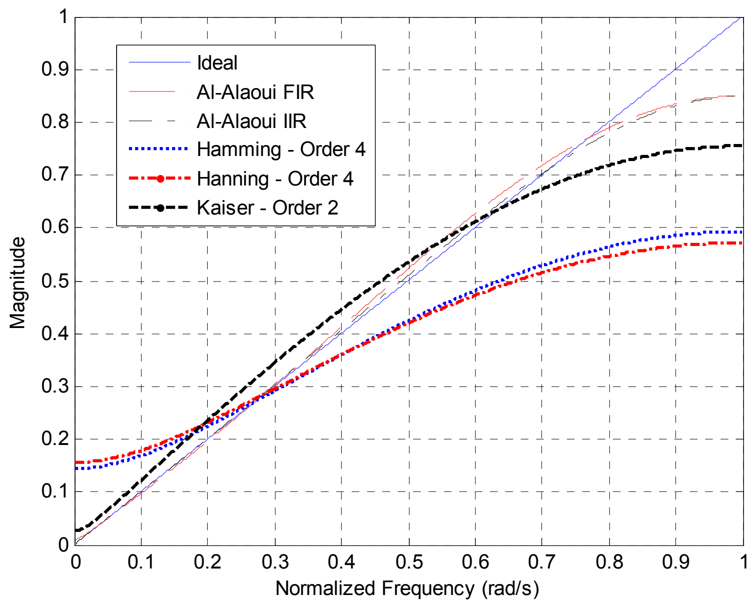


Fig.2.6. Magnitude response of the FIR Al-Alaoui differentiator: Comparison with the IIR Al-Alaoui differentiator and other FIR approximations obtained by windowing techniques. [60]

This FIR filter is then transformed into masks termed Al-Alaoui Edge Masks that are somewhat similar to the previously mentioned Prewitt and Sobel Masks. The different explanation of the method used to develop this is to be further developed later. These are represented by the following equations.

$$\text{Al-Alaoui-1\_x} = \begin{bmatrix} 6 & 6 & 6 \\ -7 & -7 & -7 \\ 1 & 1 & 1 \end{bmatrix} \;\&\; \text{Al-Alaoui-1\_y} = \left(\text{Al-Alaoui-1\_x}\right)^T \tag{2.11}$$

$$\text{Al-Alaoui-2\_x} = \begin{bmatrix} 6 & 12 & 6 \\ -7 & -14 & -7 \\ 1 & 2 & 1 \end{bmatrix} \;\&\; \text{Al-Alaoui-2\_y} = \left(\text{Al-Alaoui-2\_x}\right)^T \tag{2.12}$$

Like Canny [50] and Deriche [51] these can be used in the earlier stages of a Canny like algorithm.

An example of performing edge detection with some of the methods is shown in Figure 2.7 which is depicted from [3].

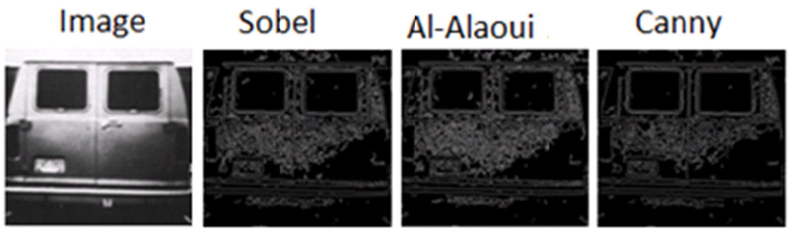


Fig.2.7. Example of Edge Detection using Sobel, Al-Alaoui and Canny

In this work, the well-established optimization technique simulated annealing (SA) is used in order to improve the digital differentiator and its matching to the ideal differentiator. Thus, it is important here to give a general idea about SA which is a rather statistical heuristic search or optimization method that aims at achieving the global optimal solution of a certain problem. It is based on heat annealing which uses varying heating and cooling especially through slow cooling metallic materials in order to harden and change the properties of the matter into a better atomic arrangement. In the optimization version, and similar to the metallic process, the SA method generates a random solution at each iteration according to certain initial limits. The solution is either accepted or rejected based on the obtained function value and the temperature variable

(*T*) which is analogously decreased as in the cooling process. The randomness of the solution is high when *T* is high in order to escape local optima and it decreases to near the optimal as *T* decreases [60]. It is worth noting that performing SA multiple times from the start enhances the possibility of nearing the global optimum. Of course, it is not the only method of its kind and there are even many methods that are based on it and try to achieve better results.

2.3 Edge Based Segmentation

Image segmentation is one of the major methods used in image processing and computer vision. It basically aims at labeling different pixels in an input image with similar pixels having common labels. This leads to having a set of different objects or segments constituting the image. This is very useful with lots of applications in various fields that include object detection and classification, medical imaging, security, machine intelligence and many others including entertainment and surveillance.

This field is in constant update with various methods that try to find a suitable solution according to the targeted application. The main methods include basic methods to highly complex ones. A common basic method usually includes thresholding as Otsu's method [61]. Other methods are histogram based and these usually rely on the peaks and valleys of the histogram as in [62].

Besides these, many methods rely on clustering such as k-means clustering [63] and mean shift segmentation whose origins are based on the works of [64, 65] and have been extensively used after the works of [66], [67] and others that are edge mean shift based approaches like [68].

Also, there are methods that are compression based since these consider that segmentation should lead to further compressing the image. Furthermore, many methods use different trainable classification techniques such as neural networks. Many other methods use graph theory and cuts with connectivity and similar concepts as in [69]. Other cut based methods are also worth mentioning like spectral min cut [70]. In addition to these, there are many methods that are edge based which are similar to the main concentration of this work. Of course, these are not the only ones as there are many famous approaches like watershed segmentation [71], Mumford Shah segmentation [72], model, and application based methods with many others. Other approaches include local variation based methods as in [73].

Concerning edge based approaches, it is important to present a brief review of these approaches. In [74], the work was constructed on a minimum description length (MDL) criterion which used similar approaches as the split and merge segmentation methods to build a method to segments edges for usage in parts-based object recognition. Also, the work of [75] performed a study regarding different edge detection approaches that are used for segmentation. The work explained techniques including standard methods as Prewitt, and Canny with newer methods such as fuzzy and neural networks based techniques. In [76], the work presented a study and compared some of the Edge Based Segmentation techniques concentrating on the Laplacian of Gaussian Operator and Canny. Also, in [77], the work considered image segmentation through edges and dealt with Edge Maximization Technique along with Sobel edge detection and others. Also, various works as [78], [79], [80], and [81] presented a study of basic edge techniques that use mask filters like Sobel, Prewitt and others for segmentation.

Concerning evaluating segmentation approaches, and although there are several efforts as in [82] and [83], this is still a continuously developing topic and in any case this is out of our scope since we concentrate on stereo matching.

2.4 Classification

Classification is a very active field of research that is directly related to machine intelligence. It is mainly defined as the process of determining the type of a certain input based on the input's properties or features amongst the possible types or classes. Classification can be supervised and unsupervised, where the former considers a finite set of known types or classes with training samples of known classification, and the latter tries to infer the data similarities with the feature values and define the types and classes themselves through clustering. In this work, we deal with supervised classification with the supervised term directly associated with the learning or training stage, where the training data features and classes are known and used to create the classifier and any new input or feature vector needs to be classified. Of course, this field is very wide and is constantly updated with the implementations used in many fields including Image Processing, Computer and Machine Vision and others.

Covering the different aspects of this field is beyond the scope of this work. In fact, if we only review boosting, a book was made on this field on its own [84]. In this work the concentration is on solving the classification problem in a similar manner to the work in [85-89].

Before directly dealing with the different ideas, it is important to define a weak learner, which is mainly and according to many references, as in [90], an algorithm that is able to correctly predict the true class of a certain sample in slightly more than 50%

of the cases, or that is slightly better than random choice in a 2 class differentiating algorithm. Of course, there are many algorithms that are considered as weak learners with the main one being decision trees of a specific number of nodes, of which one tree nodes are often referred to as stumps or even perceptrons for being similar to single neural network nodes. Other examples of a weak learner are naïve Bayes classifiers, lines of which stumps are a particular case, etc… Of course, this means that a strong classifier is basically better than the weak learner since it should be able to achieve much higher accuracies where the weak learner achieves the near 50% values of accuracy.

The previous work [85], [86], [87], [88], [89] mainly aimed at improving the mean square error or Pseudo Inverse method (MSE) through increasing the weight of erroneous samples in the training stage or through, cloning or replicating, in which the erroneous samples are repeated. This algorithm is not necessarily a strong learner and this directly depends on the dataset on which it is being implemented. The algorithm initially used linear classifiers but is easily adoptable for any kind of classifier, weak or strong. It was shown to improve the classification error rate according to the number of iterations carried out. Moreover, the algorithm was carried in batch and single pattern adaptation modes, where in the first case all the errors were added to modify the set after each iteration, while in the second case; a pattern is added to update the set as soon as an error is found. It is worth noting that although it is further possible to consider single, batch, or a combination of both versions for boosting, where only parts of the erroneous data could be added, but this is not carried out here. Moreover, such ideas would be related to online boosting as in [91] and could further help improve the different developed algorithms in this field. Further research of the ideas related to online boosting would be possible but is not in the scope of this work because it is considered a domain on its own [92].

In [87], the algorithm was used in multilayer neural networks, which correspond to the case of nonlinear methods and has proven to speed up the back-propagation method for the training stage and thus the whole training stage for this rather strong learner.

In this work, we present enhancements to ensemble learning techniques with emphasis on boosting.

Adaptive boosting (Adaboost) [93] is a machine learning algorithm that combines weak classifiers of the same type based on the error rate, thus boosting the weak learners into a strong one. Initially, it is based on Bagging [94, 95] and combines a set of weak classifiers by taking their mean. The method starts from a rather weak classifier, and continuously updates it leading into a new classifier, at the end of every iteration, by increasing the weight of each erroneous sample according to a certain criteria. This is directly similar to the Al-Alaoui algorithm since adapting the weights is a common stage in both methods. This relation was noted in [96], which highlighted that Al-Alaoui initially performed replication and weighting samples according to being in error or not, which was later used by Adaboost. The main difference between the two methods is that Adaboost further combines all the obtained weak classifier into a single one by assigning a different weight to each classifier based on its error rate, and the different weight is a major difference from Bootstrap Aggregation. Many modifications of Adaboost exist such as Linear Programming boosting and gentle boosting, etc…

There are different versions and types of boosting and in this work we review some of them. In terms of type, there are different ways to classify boosting techniques, two class or multi-class types, online or offline boosting where the latter is the standard and usually deals with a fixed set of training patterns, while the former deals with constantly updated training data, with other classifications for types of boosting. Thus, some examples of boosting algorithms include AdaboostM1 [93], AdaBoostM2 [93], LogitBoost [97], Gentle Boost [97], Linear Programming Boost [98], Least Squares Boost [98], and many others as in [100]. All of these ideas definitely indicate that this field is receptive to different fields of adaptation.

AdaboostM1 is one of the earliest methods and probably the most popular binary classification boosting algorithm since it is the least complex one although it does not yield the best results. The algorithm basically starts with a learner and with equal weighting for all the samples, trains it, obtains the error rate *e(i)* where *(i)* stands for the weak learner number. In this *e(i)* should be less than half, and achieving a value greater than that leads to halting the algorithm. And then the weighting of each training sample is modified based on being true or not. True found patterns change their weights by multiplying the previous weights with

$$\beta(i)=\frac{e(i)}{1-e(i)} \tag{2.13}$$

While the false classified patterns keep their previous weights, and afterwards all the weights are normalized. This modification for the weighting leads to obtaining a new classifier through training, which is a process that is carried out for a number of steps which is the number of weak learners. This leads to the creation of a strong classifier that achieves classification by weighted voting for all the weak learners. Each weak classifier has a weight which is analogous to the error rate, meaning the better the classifier the higher its weight according to the following equation.

$$W(learner\, i) = 0.5 \text{ x } \log\left(\frac{1}{\beta(i)}\right) \quad (2.14)$$

It is worth mentioning here that AdaboostM1 can be viewed as minimizing exponential loss according to (2.15)

$$\sum_{n=1}^{N} D_n \exp(-y_n \bullet f_n) \quad (2.15)$$

Where *N* stands for the number of training patterns, D_n stands for the weighting of the n-the sample, y_n is the class value which is either -1 or 1 for the 2 class case, and f_n is the output of the set of classifiers.

Although AdaboostM1 can be performed on multi class problems, it is not often the case, as an extended version called AdaboostM2 is used for such problems. The basic idea is the same except that each classifier outputs a vector of probabilities for each class, with the total sum being 1, and then the total probability for each class is considered with the highest one being the output, this is of course, a weighted pseudo loss and usually leads to better results, but at the expense of extra although minor processing and storage.

Another type of boosting is LogitBoost [97]. This is similar to AdaboostM1 except that it minimizes a logistic loss called binomial deviance in (2.16) relative to the exponential loss of AdaboostM1. Also LogitBoost places less emphasis or weighting on the very badly classified samples.

$$\sum_{n=1}^{N} D_n \log(1 + \exp(-y_n \bullet f_n)) \quad (2.16)$$

GentleBoost is a combination of AdaBoostM1 and LogitBoost. It mainly minimizes the exponential loss of Adaboost but its weighting is similar to LogitBoost. RobustBoost is another type of boosting which tries to avoid the problem of

AdaboostM1 concentrating on the very badly classified samples of training. Also, BrownBoost modifies the weighting strategy in Adaboost in a similar notion to Brownian motion [101], [102]. Other types that are worth mentioning include LSBoost which uses least squares to fit the weak learner in each stage with the purpose of minimizing the mean squared error.

Of course, these types are not the only ones, since there are many other such as MadaBoost [103], etc…It is important to note here that the latest versions of MATLAB present a good boosting and bagging library with the main references in the helping documents for some of the previously mentioned boosting algorithms [104]. Besides these ideas, [105] presents a review of the different boosting and bagging methods especially those related to class imbalances.

It is worth emphasizing that basic decision trees are considered the major weak learner for use with bagging or even boosting [93], [94], [95].

Concerning combining or pruning classifiers there are many works that deal with this problem according to various strategies and concepts whether regarding ensemble classifiers or not. Perhaps the most influential work regarding selecting classifiers with boosting was done in [106] which dealt with standard Adaboost only and discussed five selection strategies. The work in [106] concluded that the Kappa Pruning and the Reduced Error (RE) are the best selection or pruning strategies amongst the tested ones. The authors of [106] also concluded that pruning rarely performs better except if the new set of classifiers is very close in size to the original set in boosting. Also, the work in [106] and others, as in [107], show that pruning is better used in bagging approaches as opposed to boosting. The recent survey on pruning approaches for bagging [107] proves that RE is still one of the best performing methods. The RE method is used in this work for comparison.

Besides, many of the work related to pruning and selecting classifiers uses Genetics Programming (GP) such as the work in [108] which considered using Bayesian networks for selecting classifiers related to GP Ensembles. Other works used boosting to prune bagging [109]. Others concentrated on pruning neural networks as in [110]. In addition, [111] presented an information theoretic combination of pattern classifiers which included boosting.

Definitely, the literature has many examples of pruning but the work is generally related to bagging and neural networks as opposed to boosting which is more diverse in comparison with bagging [106].

From the different ideas and in order to highlight on of the main objectives here, the classification related work is initially performed to further improve any of the mentioned methods through applying an additional stage or stages that choose the best set of weak or strong classifiers, thus reducing the storage and increasing accuracy.

2.5 Stereo Matching and Classification

Since our main target is improving the stereo matching problem, we address this problem as a classification problem. Such work was not previously carried in the literature in the novel proposed manner. Of course, such an aim is not easy to achieve. If we present a simple example here, where the disparity of any pixel could be one of 60 values starting from 0 to 60, where a disparity of 5 means that the pixel has moved 5 pixels in the other image. Directly converting the classification into choosing one of 60 classes is possible but extremely hard, because choosing the feature vector is the main issue in this case. Another possibility is to minimize the number of classes or segment the disparity vector into, 4 parts for example, and this would make the problem easier but it should lead to just an initial estimate of disparity value that is not the required value. In this work, we do not take the disparity value as the class but rather a certain kernel amongst a set of kernels because we consider a good chosen kernel would usually have the same properties and this would render the problem of choosing the set of features an easier process. This is especially noted, since the literature has proven that a good kernel with good weights would lead to having the correct disparity [18]. This process would then lead to help in determining the disparity of the required pixel, which is the main objective of stereo matching. It is worth to define the kernel in this work, as a set of neighboring pixels around the pixel of interest. The shape of the kernel is usually a square or a rectangle.

Also, and concerning similar works, according to our knowledge, work that combines classification and stereo matching is relatively rare due to the nature of the problem of stereo matching that do not require classification in a regular direct approach. Still, there are related works. These mainly used classification and feature matching between pixel pairs to find the disparity which is not the approach adopted here. The work in [112] discussed integrating feature matching with disparity estimation and contour detection. Also, the works in [113, 114, 115] aimed at performing stereo matching through supervised learning, Support Vector Machines (SVM) and Hopfield neural networks. Additionally, the work in [116] used learning and feature selection for stereo matching, with [117] discussing the application of genetic algorithms to matching. Furthermore, the work in [118] implemented features based on Gabor filters and a neural model to estimate the disparity on the GPU. Also, [119] used biologically and psychologically inspired features in an ASW based algorithm.

Also, and which could seem related to this work, [120] differentiated between reliable and other pixels according to a set of measurements. This is different from our work since [120] did no classification or training. Also, [120] is constricted to the single method of matching used in [121] whereas the proposed work can be used with various techniques.

Of course and besides the mentioned literature, one may argue that using classification is not adequate to the problem, but it is the approach noted here that allows for this. Also, one can consider that some of the literature presents classification based stereo matching as in using Graph theory, or Belief Propagation or any other method like SSD where the disparity is the class to be found out of a set of possible classes which may be large according to the disparity range, but this is not the approach adopted and explained here.

2.6 3D Reconstruction

This issue is not emphasized here but since it is one of the main applications of stereo matching or even image registration, it is important to further dwell on Reconstruction especially in relation to stereo matching.

In [122], the authors aimed to improve 3D reconstruction by improving dense stereo matching. Their method is composed of the following four different stages.

- Extract the different feature points using Scale invariant feature transform (SIFT),
- Select some key points based on their description and properties.

- Use the key points as seed ones and they try to perform matching based on intensity correlations.
- Perform initial dense matching using area growing and then further refine it by removing some false matches using SED (Symmetrical Epipolar distance).

After the matching process, the disparity or depth map is structured with the 3D model created using Delaunay triangulation with texture mapping.

The main addition here in [122] was SED which improved the results by about 30%. It is worth noting that the authors used C++, Open Computer Vision (OpenCV) and Open Graphics Library (OpenGL). In [123], the main ideas behind the disparity is similar to other works, because they start from an initial value and iterate to improve the solution then the 3D reconstruction uses 3D Studio Max and achieves acceptable results. Furthermore, in [124], SIFT was used for detecting features in both images and that are then matched to determine the distance. Finally, those textures of triangles are used for creating the 3D surface.

Since 3D face reconstruction is of great importance in the field of reconstruction, [125] presented a basic method for the reconstruction of faces from different stereo frames and thus video. The method basically uses a set of 65 or 72 (The 72 are obtained by adding 7 obtained by symmetry) features that are then pointed on the face image(s). Then, the points are compared to those of a generic model and a matching is performed between the obtained face and that of the model. Additionally, in [126], the work presented a simple method for the construction of a 3D face model from 2 stereo images. The work first computes the disparity map using SSD and argues it is even more efficient for their application than some global methods. Then they find the model and argue that it is not good since they did not use epipolar correction so they perform it using certain points at the corners of their images. Indeed this improves the result. Thus, they obtain the disparity map using epipolar matching and then SSD, and then they describe a simple triangular method for texture mapping and 3D model creation.

It is worth mentioning that many works use 3D reconstruction from stereo in Face Recognition, such as [127, 128] and others.

2.7 Parallel Computing and CUDA with the Related Problems

Since we do not only aim at achieving an accurate system but a rather real time one, we try to develop the diverse ideas on a parallel hardware architecture which is mainly CUDA. CUDA [129] was developed by Nvidia in 2007 and has seen constant increase in implementation in different fields where there is a possibility of using parallel programming.

Work related to stereo matching and parallel hardware is not scarce at all. In [130], the authors compared GPU and FPGA for different applications including stereo matching; they explain each stage of this system and implement it on both the GPU and FPGA. The work has led to interesting conclusions that GPU is better in many cases than FPGA according to the different comparisons or the basis of their comparison, whether performance, power, etc…

In [131], the work presented a comparison between the FPGA and the GPU for a dynamic programming algorithm used for stereo disparity matching. It uses an FPGA and Nvidia GT 280 of the same price to show the results. The authors do not dwell on the algorithm which is called Symmetric Dynamic Programming Stereo (SDPS) arguing that this method is well suited for parallel processing especially that it is considered as a semi global algorithm. They use two stages one for rectification and the other for disparity calculations. The results are very close, except that the FPGA outperforms the GPU by a notch without much effect to the image acquisition stage.

Also, in [132], the work implemented stereo depth calculation on the GPU starting from a Birchfield Tomasi (BT) cost calculation and later using semi global cost matching optimization. The reported time is about 180 milliseconds.

Also, the work of [22] dealt with stereo matching using the graph cuts or maxflow / mincut on the GPU. It is used on Markov Random Fields (MRF) and it basically parallelizes the push re-label algorithm. It is not restricted to stereo vision use because it can be applied for image segmentation, restoration, etc… The speeding reported is 70-100 times.

Furthermore, in [133], the authors compare the usage of local and global methods on the GPU with the CPU. They use SAD for local and semi-global matching. They report about 5 times speedup.

Moreover, in [134], the work presented 3 algorithms that include: MultiView Stereo Matching, Feature Extraction, and jpeg2000 encoding. The algorithms were conducted on a CPU with OpenMP and a GPU. The authors present some guidelines for performing algorithms on the GPU.

Besides, in [34], the implementation of semi global matching on the GPU was carried out. The energy cost function is based on MI and histogram calculation in addition to the cost calculation. In [135], the authors use a method of their own design that tries to find for every pixel a different window of correspondence with the other image. They also add a penalizing term for the size of the region which helps achieve bigger regions. They also test the performance on a GPU. The time for 16 disparities takes about 60ms. They compare their approach with other methods listed on the Middlebury Database [1, 14]. Furthermore, in [39], they basically use Mumford Shah [72] along with Stereo Matching. They start from an initial disparity map using a local approach. They then use edge detection and segmentation for updating the disparity matrix. They minimize an energy function with regularization for obtaining a smooth disparity after a number of iterations. They also apply their method using the GPU for speedup.

Also, in [27], many of the local methods used for stereo matching were considered. The work is comparable to the standard Middlebury work [1, 14]. Also, the authors use median filtering after computing the disparity for refinement purposes. They utilize GPUs for checking real time performance of the proposed methods. In particular, they compare 6 new real time approaches for quality and speed. It is worth noting that the simplified adaptive-weight approach [18] seems to lead to very good results.

In addition, it is worth mentioning [136] which discussed a local algorithm based on bitwise voting and applied it on the GPU. Moreover, [137] presented a GPU implementation of exponential ASW and exponential step message propagation. Also, [138] proposed an iterative refinement method for ASW based matching. In any case, the CUDA related literature presented approaches that are slower than the SAD algorithm [139], especially at a small radius.

Most of the work highlights the speedups obtained by the GPU, but this is not always the case, since it is worth noting that in [140], the authors basically consider different algorithms and apply them on Multicore CPUs and GPUs. The algorithms include: SGEMM, saxpy, monte carlo, convolution, FFT, sort, etc… They conclude that enhancement is about 2.5x and not a lot more.

Most of the work mentioned here deals with stereo matching and thus it is worth noting that the image registration problem also used parallel hardware since many publications like [141] have tackled the problem using GPUs and in particular CUDA by Nvidia. The different authors reported different speedups and accuracies.

Many other works exist, and most of them promote CUDA as a rather suitable platform whether in terms of speedup or ease of implementation, and thus it was chosen as the main parallel architecture, besides being promoted and supported by Nvidia.

CHAPTER 3

EDGE DETECTION AND SEGMENTATION

The work uses edge detection and edge based segmentation as parts in its stereo matching approach with and without classification and since some innovations were introduced in these fields, it is of great importance to explain them in details.

3.1 Edge Detection

Edge detection is already highlighted, in the past chapter, as one of the earliest parts of image processing and computer vision [2, 3] with many applications in various fields as noted in [142] that include sharpening amongst others.

Concerning digital filters, and as previously noted, Al-Alaoui [143] used his own differentiator after inverting the found integrator. This differentiator has proven to be of useful implementation in analog to digital conversion in addition to other applications as noted in [143-147]. This IIR differentiator is then transformed into an FIR one as outlined in the work. Thus the following differentiator is obtainable.

$$H_{Diff\_FIR}(z) = 0.36z^2 - 0.42z + 0.06 \tag{3.1}$$

His work further transformed the filter into several masks in order to be simply implemented in image processing techniques and in particular edge detection since it is a differentiator. The approach used in the referenced work led to improving the detection in many cases.

In this work, and as previously mentioned, we use SA as an optimization method for obtaining a better digital differentiator. So, SA is implemented for multiple times to optimize the different coefficients of the IIR equation. Of course, using SA as an optimization technique in particular is not necessary, because there are other optimization methods and anyone should probably work since we further tested direct constrained optimization and it led to the same result. This would lead to having different IIR filters, amongst which the one with the most desirable properties is chosen and although some of the tested filters were found to be very slightly better, the IIR filter with exactly opposite values in the numerator is chosen since it is easier to implement. Thus, the following filter was selected.

$$H_{Diff}(z) = \frac{0.3618 - 0.3618z^{-1}}{1 + 0.1956z^{-1}} \tag{3.2}$$

After that, this filter is transformed into an FIR in the same method in the paper [59]. Thus, if the IIR filter has the equation, which was found through optimization, the FIR filter found through transforming the IIR version would have the following equation:

$$H_{Diff\_FIR}(z) = 0.3682z^2 - 0.4326z + 0.0846 \tag{3.3}$$

Afterwards, this filter is optimized in a similar manner as the initial IIR using SA multiple times with simple modifications to the initial solution in order to further explore the search space and then best solution is chosen ensuring that the sum of the elements of the FIR filter is zero. The optimization is considered by minimizing the error with respect to the ideal or analog output. Thus, the following FIR filter is found:

$$H_{Diff\_FIR}(z) = 0.3582z^2 - 0.4292z + 0.0710 \tag{3.4}$$

Figure 3.1 shows the error performance of the found filters and compares them with the original version.

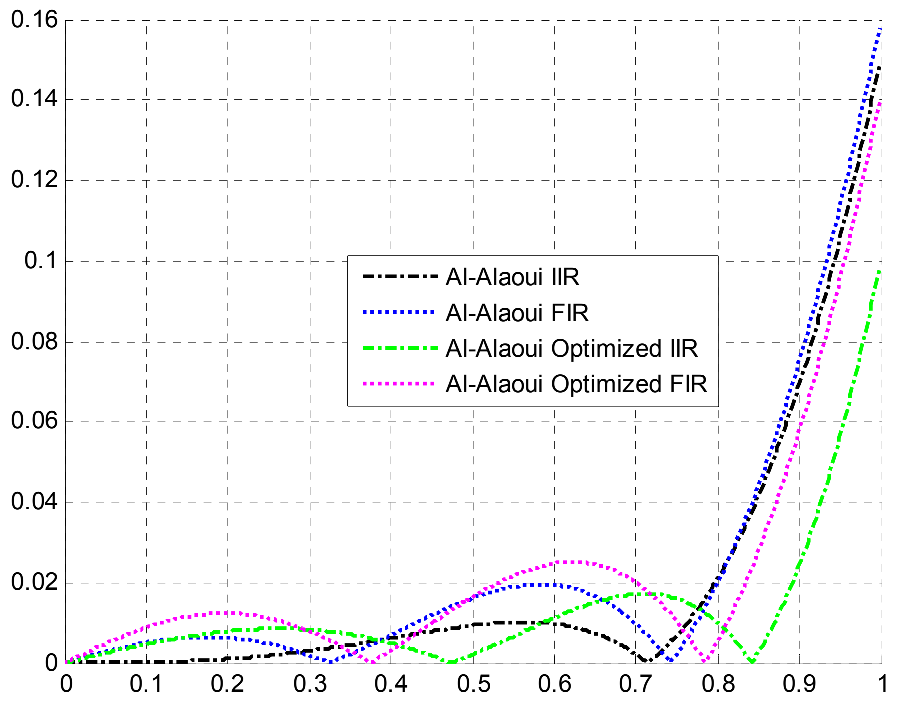


Fig.3.1. Absolute magnitude error of the Optimized Al-Alaoui FIR and IIR filters

Also, in a similar manner as in [59], the following possible sets of masks are obtained.

$$Al-Alaoui-O-1x=\begin{bmatrix}0.3582\\-0.4292\\0.071\end{bmatrix}\begin{bmatrix}1&1&1\end{bmatrix}\simeq\frac{1}{14}\begin{bmatrix}5&5&5\\-6&-6&-6\\1&1&1\end{bmatrix} \qquad (3.5)$$

This mask is considered as the Al-Alaoui-O-1x mask in the x direction, where O stands for optimized. Transposing this mask leads to Al-Alaoui-O-1y which is the mask in the y direction.

In a similar fashion, multiplying the mask ($[0.3582\ -0.4292\ 0.071]^T$) by [1 2 1], a Sobel-like smoothing mask, yields

$$Al-Alaoui-O-2x=\begin{bmatrix}0.3582\\-0.4292\\0.071\end{bmatrix}\begin{bmatrix}1&2&1\end{bmatrix}=\frac{1}{14}\begin{bmatrix}5&10&5\\-6&-12&-6\\1&2&1\end{bmatrix} \qquad (3.6)$$

Also, this is referred to as Al-Alaoui-O-2x in the x direction. And transposing this mask leads to Al-Alaoui-2y mask in the y direction.

Moreover, combining ([0.3582 -0.4292 0.071]^T) with [1 3 1], yields Al-Alaoui-O-3x mask in the x direction and the corresponding transpose Al-Alaoui-3y in the other direction.

$$Al-Alaoui-O-3x = \begin{bmatrix} 0.3582 \\ -0.4292 \\ 0.071 \end{bmatrix} \begin{bmatrix} 1 & 3 & 1 \end{bmatrix} = \frac{1}{14} \begin{bmatrix} 5 & 15 & 5 \\ -6 & -18 & -6 \\ 1 & 3 & 1 \end{bmatrix} \quad (3.7)$$

The different effects of the implementation of this filter are similar to those reported in [59].

An issue still remaining here is that the mask filter values have somewhat small values when compared to Sobel and Prewitt masks and in order to be able to compare them with other standard masks in a better, the values are approximated to be similar to Prewitt and Sobel Masks by setting the first row values similar to the other masks to 1 and updating the other coefficients according to that, and thus the following filters are obtained.

$$Al-Alaoui-O-1x \simeq \begin{bmatrix} 1 & 1 & 1 \\ -1.2 & -1.2 & -1.2 \\ 0.2 & 0.2 & 0.2 \end{bmatrix} \quad (3.8)$$

$$Al-Alaoui-O-2x = \begin{bmatrix} 1 & 2 & 1 \\ -1.2 & -2.4 & -1.2 \\ 0.2 & 0.4 & 0.2 \end{bmatrix} \quad (3.9)$$

$$Al-Alaoui-O-3x = \begin{bmatrix} 1 & 3 & 1 \\ -1.2 & -3.6 & -1.2 \\ 0.2 & 0.6 & 0.2 \end{bmatrix} \quad (3.10)$$

It is worth noting that the difference between the new and the previously found masks is somewhat minor but it does lead to improvement in the MSE with synthetic edges as shown in [59]. Another important note that is not addressed here is the possibility of implementing SA to enhance the mask itself and this could be done in the future.

Now, these different edge masks can be used in place of the Sobel masks or any other ones for different purposes. Also, they can be used as part of the Canny [50] edge detection algorithm. Figure 3.2 shows the edge detection results obtained using Prewitt, Sobel, the proposed masks Al-Alaoui-O-1 and Al-Alaoui-O-2 after these are applied to the Cones image [1]. Clearly, there is not much difference except in the level of illumination.

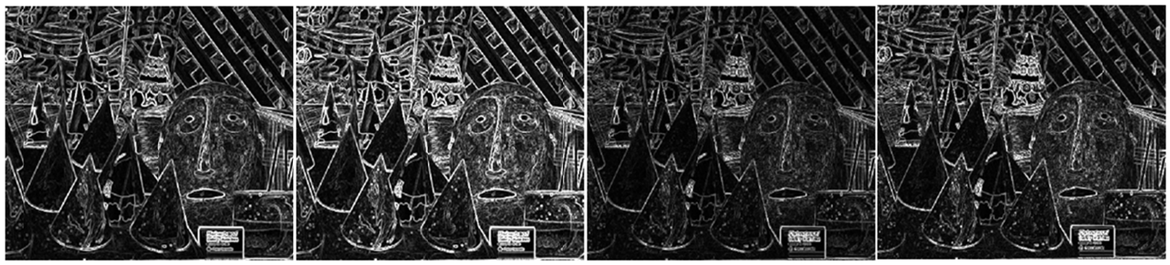


Fig.3.2. Edge Detection Obtained Images using Prewitt, Sobel, Al-Alaoui-O-1 and Al-Alaoui-O-2

It is worth noting here that in this work the Sobel like Al-Alaoui optimized mask is considered in the remainder of this work whether as part of the Canny algorithm or otherwise especially in regards to the edge based segmentation approach adopted here and explained next.

3.2 Edge Based Segmentation

As the name implies edge based segmentation relies on the found edges to segment the different objects of the image. The objects here are not meant to be whole objects like in finding a car in an image or other more complete objects but rather a set of pixels that combine to form a single object due to their closeness in intensity and color values.

The approach here is basically divided into a couple of stages. The first stage performed the Canny edge detection based on using the Optimized Al-Alaoui Sobel like mask which leads to having an edge image, but with some discontinuities. Afterwards, a connection stage is considered for reducing the discontinuities. Figure 3.3 shows an example of the Canny edge image using the colored version of the Cones image.

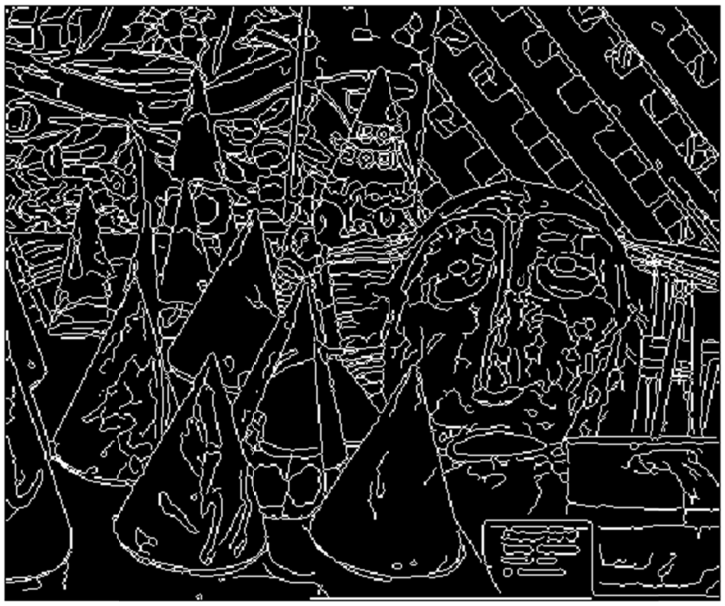
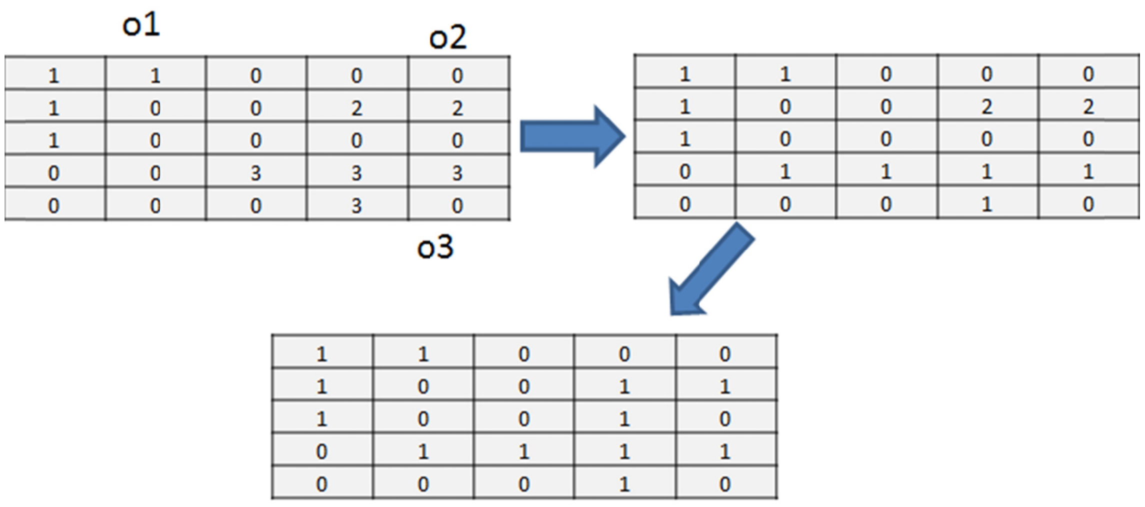


Fig.3.3. Example of Canny Edge Detector Image

The gaps are handled in a manner to turn the edge image into a linked set of edges and thus a set of segmented regions. The method is rather fast, as preferred, because we aim at delivering a fast system.

This is addressed by considering a kernel around each edge pixel and checking for different connected objects or sets of neighboring 1s. The different objects are then connected together starting from the bigger objects and downwards according to the size of the object. Figure 3.4 shows an example of a kernel having a radius of 2 pixels with its different parts. In this figure, there are three objects. Object 1 is made up of 4 pixels, object 2 has 2 pixels, and object 3 is made up of 3 pixels. So and since object 1 and 3 are the bigger ones, they are first connected together to become one object (object 1) and then object 3 is connected to object 1. It is important to note that the size of the kernel in each direction is important here and is usually small and less than 7 which correspond to a radius equal to 3 unless set by the user at a different value. In order to connect these parts, we use the standard algorithm for drawing lines. This is Bresenham's [148] algorithm which is greatly used in Computer Graphics for drawing connected components such as lines, circles, etc… since it does not require much computations. In simple terms, the algorithm selects a single pixel per each row and column by computing the slope and checking if the current pixel is close enough to the line or not.



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

1	1	0	0	0
1	0	0	2	2
1	0	0	0	0
0	1	1	1	1
0	0	0	1	0

1	1	0	0	0
1	0	0	1	1
1	0	0	1	0
0	1	1	1	1
0	0	0	1	0

Fig.3.4. Example of using Bresenham’s algorithm for connecting objects for a size 5 kernel

An example of the obtained continuous edge image is shown in the Figure 3.5.

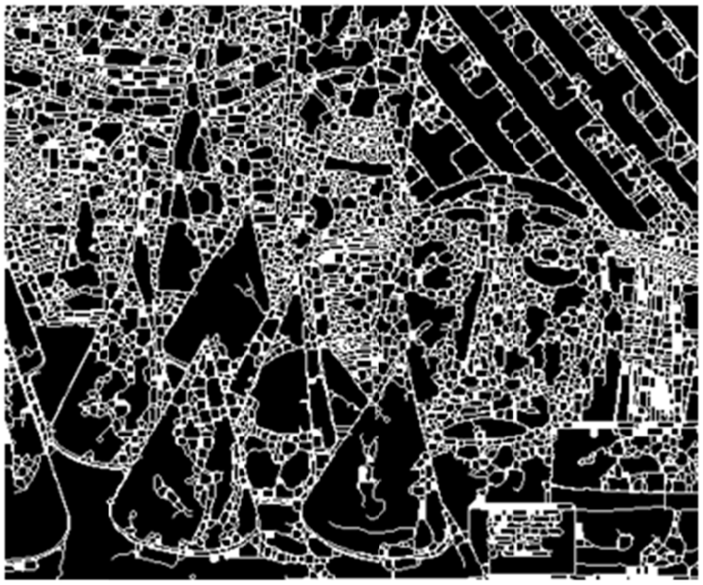


Fig.3.5. Example of the Connected Edge Images

Although this method connects the images in an acceptable manner, it tends to over-connect the objects and create unnecessarily small objects, so an approach is presented to reduce this problem. This is done by accounting for a larger kernel surrounding the worked on kernel. This kernel is checked for connections at first, because it is possible that the objects to be connected in the small kernel are already connected, so these are checked and only if there is no connection, the smaller kernel connections are applied according to the manner already described. A worthy note here is that the larger kernel's radius is greater by 2 or 3 and can be set to any other value, but much increasing it beyond these values tends to greatly reduce the connections between the objects. Figure 3.6 shows an example of the connected edge image using this explained approach.

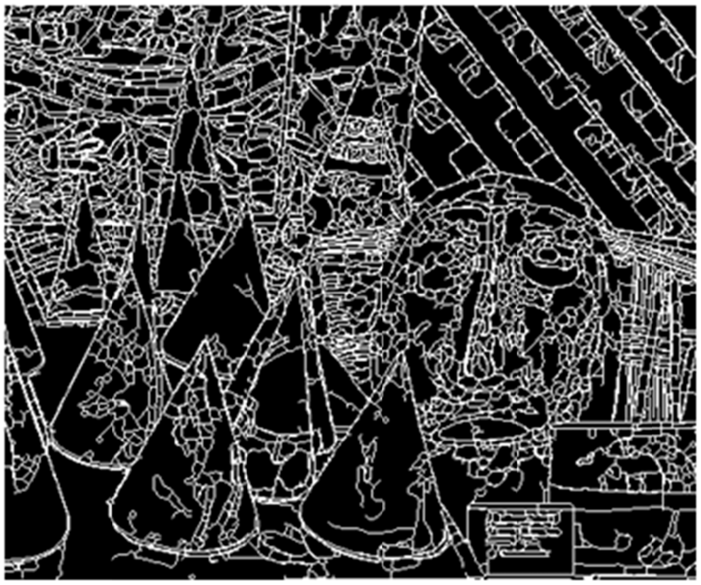


Fig.3.6. Example of the Updated Connected Edge Images

And using this image, the different objects are found by performing a filling operation similar to the one done in paint, (bucket operation). It is worth noting that this operation consumes very little time especially if compared to other methods for segmentation but of course most available algorithms are more general and comparison with them is beyond the scope of this work. The corresponding segmented objects are shown in the Figure 3.7 where each object is filled with its mean intensity value for display purposes and to show that it is possible to use this in the context of obtaining cartoony images.

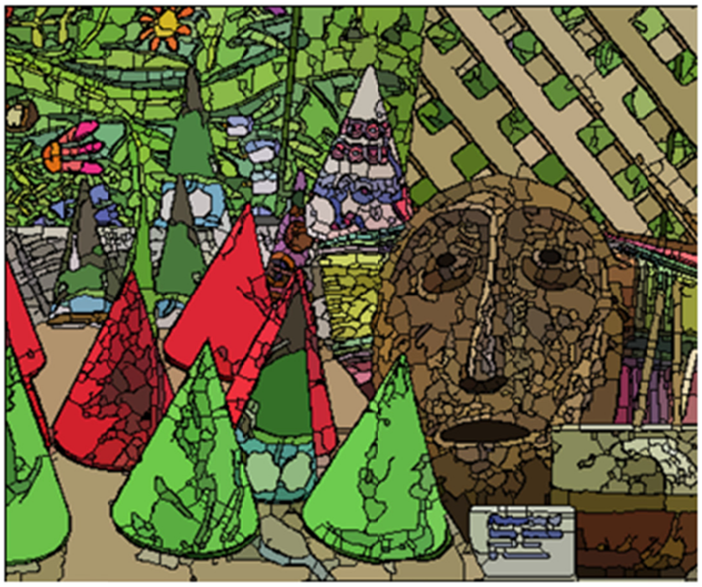


Fig.3.7. Example of a Segmented Image

However, the process of segmentation is not yet complete, since some objects are rather small. And so, the next stage considers the sizes of the different segmented parts for checking in comparison with a certain size threshold and if they fall below the minimum limit they are added to the nearest neighboring object with the most similar intensity properties if they are close enough in terms of intensity. Of course, this limit is set by the user which is like any other image processing procedure since there are different input values. A good example of this limit would be to remove objects whose size are less than 0.005% of the total size of the image which is the case used in this work. An example of the output obtained after applying this stage is shown in the next figure. Another approach would be to consider the total number of objects in the image and diffuse the smaller objects based on a maximum number of objects limit, but this was not performed here.

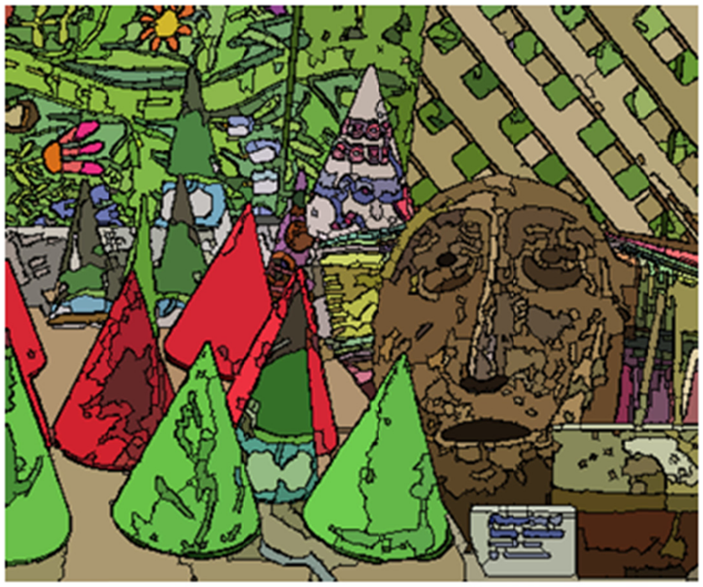


Fig.3.8. Example of the Updated Segmented Image

Afterwards, it is rather straightforward to assign each edge to an object by checking the object it is nearer to in terms of intensity to obtain an image similar to the one shown in Figure 3.9.

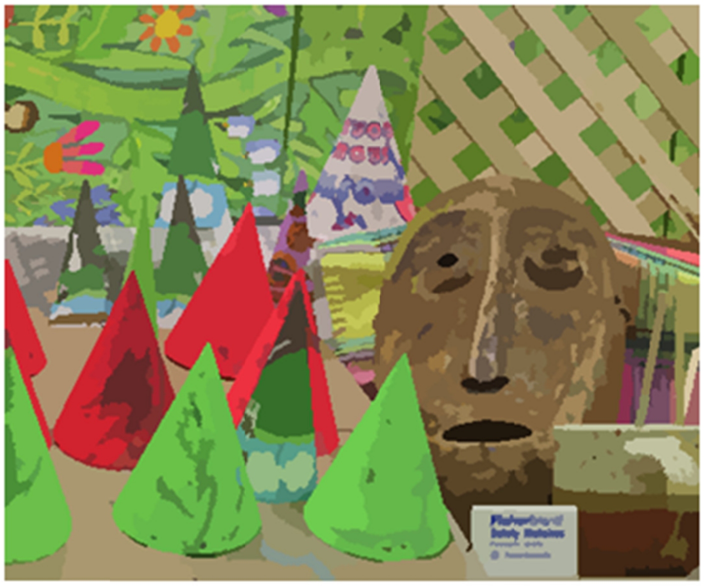


Fig.3.9. Example of Updated Segmented Image without edges

Thus, the number of objects obtained after the mentioned process is relatively acceptable, about a thousand in each image of our main tested images which are considered rather condensed since they have lots of varying colors and are composed of different parts.

3.3 Discussion about Edge Based Segmentation

It is worth mentioning here that there many possible improvements in order to reduce the number of objects, and a notable mention is to reduce the size of the image to an acceptable dimension, say in the neighborhood of [128,128] where the smaller size is transformed to 128. So if the original size was [s1,s2], it becomes [round(s1/min(s1,s2)*128) round(s2/min(s1,s2)*128)]. Thus, one can now perform the segmentation on the obtained smaller image which leads to a much less number of objects at the expense of lost details. An example of such approach is shown in the next figure.

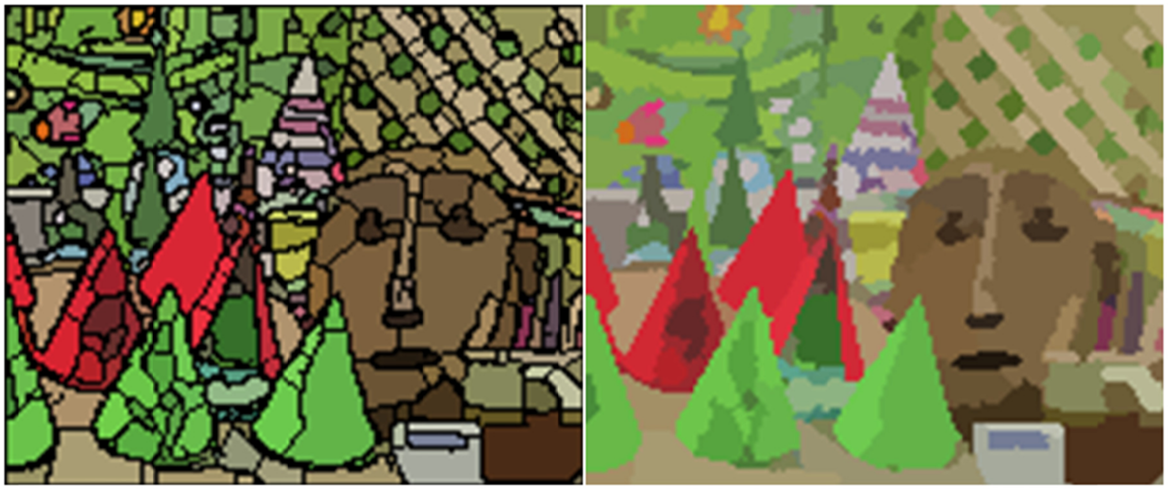


Fig.3.10. Example of Downsized Segmented Image

Besides these mentioned points, and although there are discontinuities in the edge image, one might note that directly using them and performing the filling operation, but this approach simply does not work for the gaps and example of the segmented images obtained after directly using edge detection without performing connections is shown in the following figure. It is worth mentioning here that although it is possible to use resized images in a similar manner as before, gaps generally still exist.

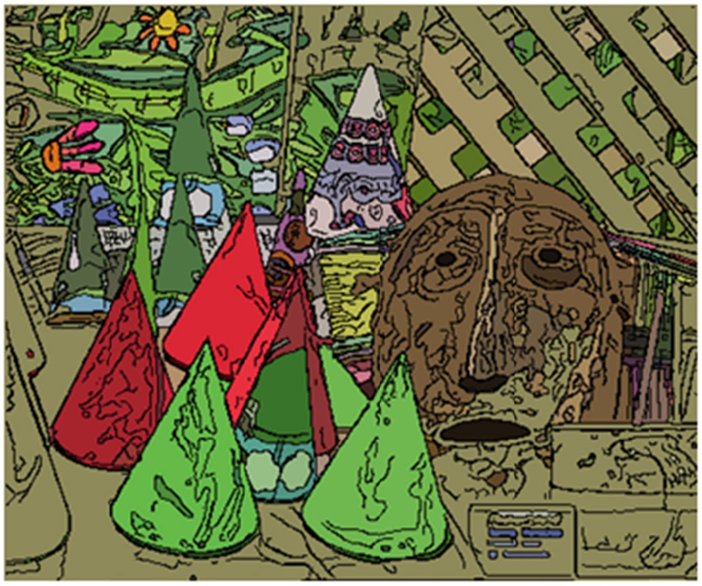


Fig.3.11. Example of Directly Edge Segmented Image

Thus, and through this explanation, the segmentation is performed. And although the usage here is concentrated on stereo matching, there are many possible implementations with a simple example shown in the next figure where an object is cut and placed in another image.

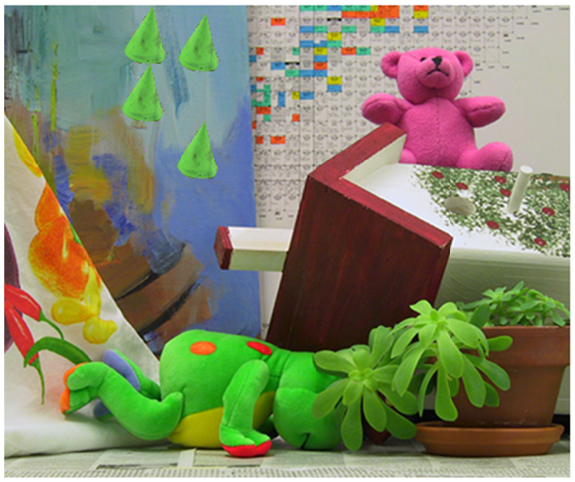


Fig.3.12. Example of segmented object utilization

CHAPTER 4
STEREO MATCHING

Stereo Matching is the main addressed problem in this work, with the aim of obtaining an accurate and fast system as clearly emphasized in the previous chapters. So, in this chapter, we further deal with the different issues related to this problem and present some of the contributions of this work. However, this is not the whole work with regards to stereo matching since the main ideas in this chapter are further addressed when dealing with classification and relating it to stereo matching in Chapter 6.

4.1 Initial Processing

The main problems previously highlighted in this work include illumination differences and occlusion between the stereo pair. Illumination differences whether small or big are solved through initial processing of the images. The first step done to the input images is median filtering, which is performed to the pair to reduce the effects of the noise, but this is not a necessary step unless the images were noisy. Also, the work here relies on histograms and in particular histogram specification to address the illumination differences. For this purpose, it is important to consider histogram equalization which is one of the most common method for contrast enhancement. Histogram matching builds upon histogram equalization and tries to match the histogram of an image with another histogram. As mentioned in the previous section, these methods have been widely used for preprocessing images in the problem of stereo matching as in [49] and others.

It is worth mentioning here that the implementation of histogram equalization and related ideas problems has led to many histogram based techniques such as [149] which presented a framework for contrast enhancement through varying histogram equalization.

The main idea here as published in [150], relies on trying to match the value of the similar pixels in the stereo pair. The pixels are rarely exactly the same unless they are obtained through a complex setup as in [1] due to many reasons that include digitization or photometric issues. In perfect conditions, the non-occluded pixels of the stereo pairs, or those that are present in both images should have the exact same pixel value on a 0-255 scale which is very common for standard digital images. Based on this premise, the histograms of the non-occluded pixels should be exact. Thus, if the perfect disparity map is available, it should be possible to match the histograms of the two images. In such cases, it is possible to determine the best possible performance of any stereo matching method whether local or not in best illumination and image conditions.

The method here uses the histograms of the stereo images for finding the illumination difference between the stereo pair and reducing the color inconsistency effect leading to enhancing any stereo matching method. This is done by finding the histograms of the stereo pair. Then the peaks of each histogram are found and the intensity difference between them is computed. If the intensities at the peaks are different, histogram specification can be directly used or a further validation is done. In this validation, the histogram is divided into three equal regions (Regions 1, 2 and 3). In addition to these regions, two other regions are considered. These two regions overlap the previous regions with Region 4 starting from the middle of Region 1 and ending at the middle Region 2. Region 5 is considered similarly starting from Region 2 and ending at Region 3. In each region, the peak of its histogram is easily found for both images. After that, the difference between them is computed whether being negative or positive. Thus, five values are found and since in some cases two of them could be overlapping, only one value is selected. Besides, the sign of all the values are checked and the governing sign is selected to remove the values of the different sign. Thus, a vector is obtained and the median value of this vector is found.

It is important to note that the peaks are used because they are easy to determine and since occluded pixels do not usually exceed 20% of the pixels, it is probable that the peaks coexist.

Figure 4.1 shows the histograms of the smallest sized gray scale Teddy image pair available through [1] without adding any illumination effect. In this case, Regions 2 and 4 and Regions 3 and 4 lead to the same value, since they are the same. The median value output in this case is found to be 2, and this does not mean it is always positive and it should be 0 in many cases where the images are almost perfect.

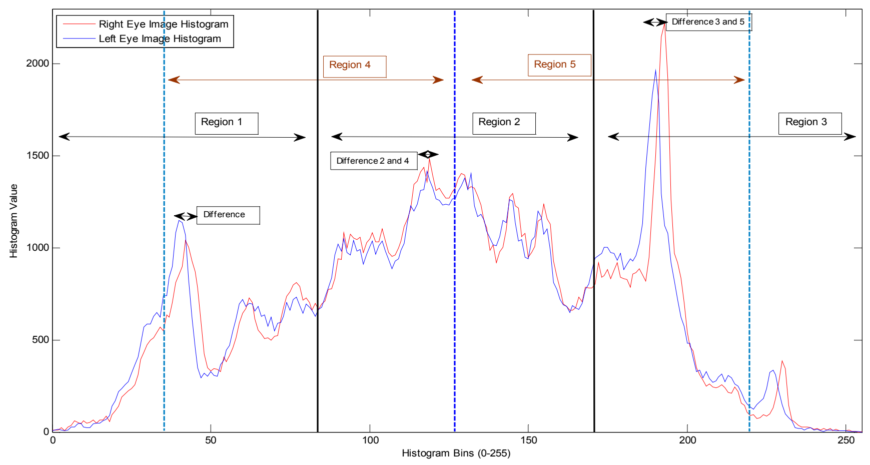


Fig.4.1. Histograms of Gray Scale Teddy [1] Images showing the different regions

After finding the required value, a general idea about the illumination differences is obtained. If this value is rather large, histogram specification of one of the images is performed based on the histogram of its counterpart. Thus, any of the local or global methods is performed using the histogram specified image. For example, (SAD) becomes (SADh) where (h) stands for histogram matching implementation.

$$SADh_{x,y} = \sum_{i=x-R}^{x+R} \sum_{j=y-R}^{y+R} \left| R_{(i,j)} - Lh_{(i,j)} \right| \tag{4.1}$$

Thus, it is easily possible to compute the disparity in a similar manner as the initial method. And since histogram specification creates a transformation (T) that assigns for each value a corresponding value. This transformation can be subtracted from its pair leading to (Td) which means that:

$$Lh_{(i,j)} = L_{(i,j)} + Td(L_{(i,j)}) \tag{4.2}$$

Besides this, it is possible to combine the regular and specified image by selecting the minimal value between them and considering it as the combined measure. Of course, this is more computationally demanding but later shown to be of acceptable effect and better accuracy. For example, the (SAD) equation becomes (SADc) where (c) is for combined and:

$$SADc_{x,y} = \min(SAD_{x,y}, SADh_{x,y}). \tag{4.3}$$

It is worth mentioning here, that the effect of the proposed approach is minimal in the (NCC) case, which is the least prone to linear illumination differences.

Figure 4.2 shows an example of the effect of the proposed preprocessing while using (SAD) for stereo matching.

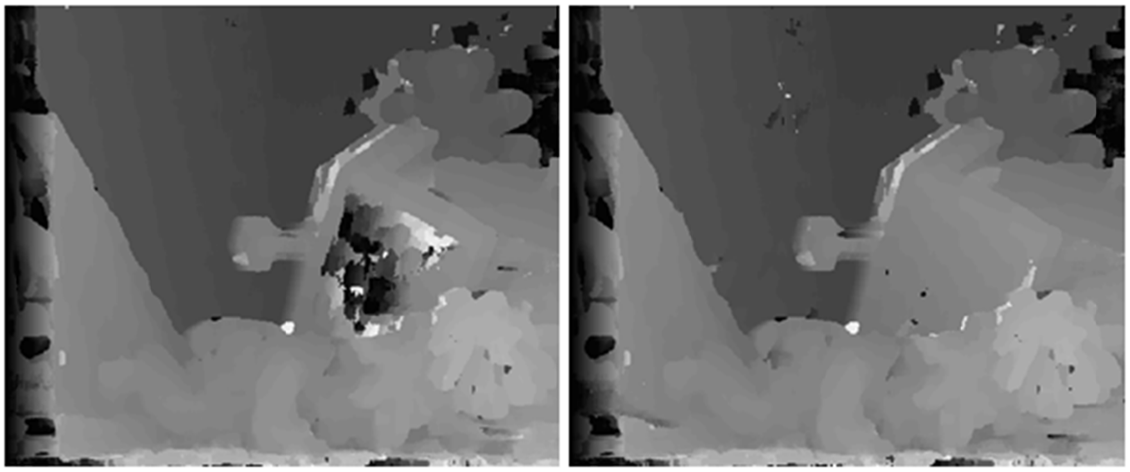


Fig.4.2. SAD Stereo Matching with and without initial processing

And Figure 4.3 shows another example of the proposed preprocessing while using (BT) for calculating disparity.

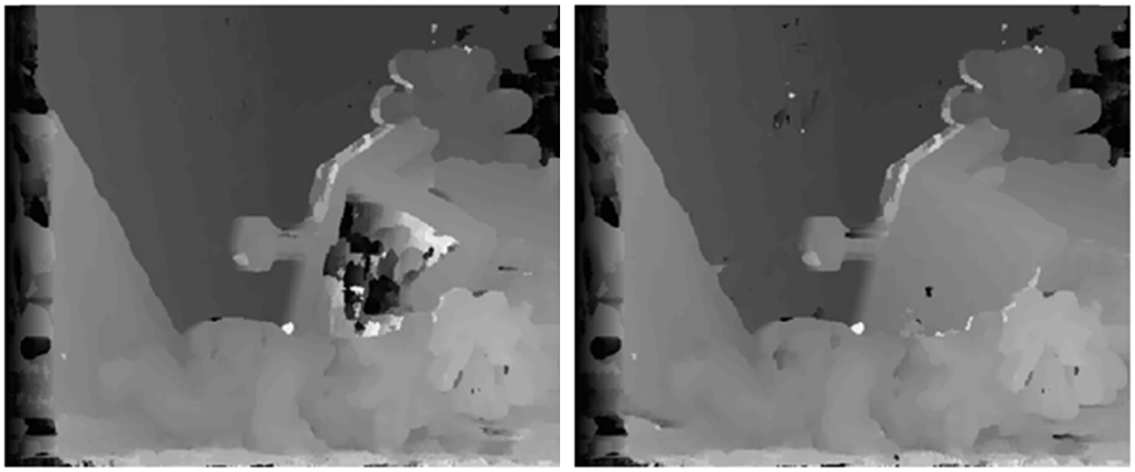


Fig.4.3. BT Stereo Matching with and without initial processing

This processing is done for gray scale images and the same processes are easily generalized for color images where this process is performed to each of the red, green and blue scales leading to a better accuracy.

4.2 Proposed Approach

As clearly emphasized in this document, the aim is in delivering an accurate and fast stereo matching system. And as already noted most of the procedures aiming at accuracy use global approaches or at least semi global ones, while if speed is the target, local approaches are preferred. Of course, combining both is not an easy task at all and would require testing different procedures. The main difference between local and global methods lies in that the local methods do not account as much as the global methods for the disparities of neighboring or similar pixels when computing the disparity of the current pixel while the global methods usually rely on that since they consider the image as a whole. Combining them leads to semi-global approaches and this category is the one under which this work falls since it starts with any method which could be local and improves it using different measurements like those obtained from neighboring disparities.

The proposed approach here can be used with any method whether local or global or semi global and mainly consists of stages that start with the chosen method to obtain an initial disparity map and aims at improving the accuracy of the found map using a relatively fast procedure. Of course, to have a totally fast method it makes more sense to start with a fast stereo matching method such as the local methods of SAD, which will be the initial method of concentration here due to its wide usage, speed and simplicity. The usage of (SAD) is not by any means exclusive since other methods like (NCC) or (BT) stereo matching has led to similar results as later noted. Of course, using more complex methods would be of varying effects or improvements as later discussed.

The main idea of our developed algorithm is to start and retrieve a set of tie or accurate points that are then used to complete the remaining points, and this is not the first time a similar method is implemented.

Before explaining the different parts of the implemented method, it is important to note that these ideas were found after testing and trying different possibilities to select the best possible solution. The concept here implements the initial disparity method like SSD or SAD in such a manner as to obtain a value that is considered true with relatively high confidence. This is initially done by dividing the original kernel, which is a square of neighboring pixels around the pixel under concentration of radius R, into four parts (top left, top right, bottom left and bottom right) as shown in the Figure 4.4. Thus, the four values at the corners of each square along with the center value constitute a vector of five values which is referred to as the possible vector from here onwards.

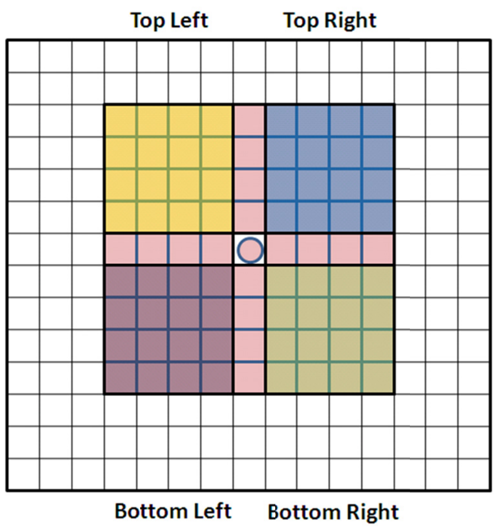


Fig. 4.4. Example of a kernel and its 4 parts with Radius = 5

The kernel size is considered to vary among selected values according to the required accuracy but mainly the chosen radius was set to 5, which means the size of the kernel would be about 11x11 so choosing a radius from 3 to 7 could be acceptable. It is worth mentioning that these values were used on the low resolution stereo pairs of [1], and could be slightly modified according to the images' size when accounting for other stereo pairs.

Afterwards, the main initial operation is done using regular SAD, BT, SSD, NCC or any other method on the stereo images and this is done for the four indicated parts and at the lower kernel size. And in the case of having close values at a certain pixel, this pixel is considered a confident pixel. "Close" means that the absolute difference between each pair is $\leq T_D$. If all the differences are $\leq T_D$, the disparity is kept, while otherwise removed. T_D is set to 2 for Cones and Teddy and 1 for Venus and Tsukuba.

Figure 4.5 shows the remaining disparities through the different stages when using SAD, BT, and NCC. All of these results are based on using the stereo pair after performing the initial processing discussed in the previous section. The obtained map here is termed Disparity Map A (D_A).

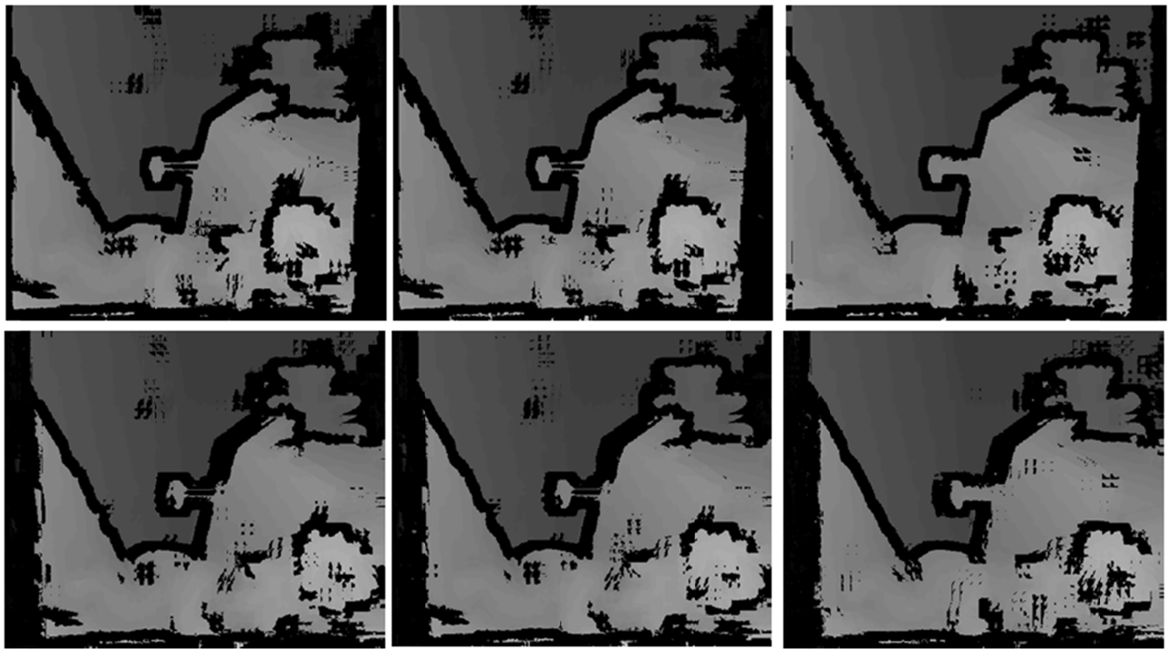


Fig.4.5. SAD, BT and NCC Remaining Confident Pixels

Definitely the remaining pixels still need to be found, and this should be done in a manner so as to minimize the error. There are many possibilities to solve this issue. A simple direct approach would be to perform voting when considering the four obtained disparities of each pixel which often leads to having a disparity of better accuracy but still far from being accurate as shown in Figure 4.6 for the SAD disparity case. Concerning the voting process it is not direct or there would be equal votes in most cases and this is due to the value of the outputs being slightly different. For example, it is possible to obtain 18,21,30,31 and in such a case it would be more meaningful to choose 30 or 31. Thus, the voting is performed in the following manner. At first a difference threshold is selected for the vector of possible values. This threshold is usually 1 or 2. Then each possible outcome *(v)* is replicated according to this threshold *(t)*, to lead to the interval of values from n-t to n-t. If *(n)* is 18 and *(t)* is set to 2, then the possible values would include 16,17,18,19, and 20. This is done for every value in the possible vector leading to a larger vector of *(2*t+1)*4* values. These values should have duplicates and the most occurring one or “mode” is chosen. And in case there were additional candidates, the lowest value or the median can be chosen. Also, it is possible to consider the original SAD disparity in such a case or even if the voted value is less than or equal to 2 as shown in the right of the next figure. Of course, there are other possibilities here that include weighted voting according to a certain distance and others, but the improvement obtained through these seemed negligible.

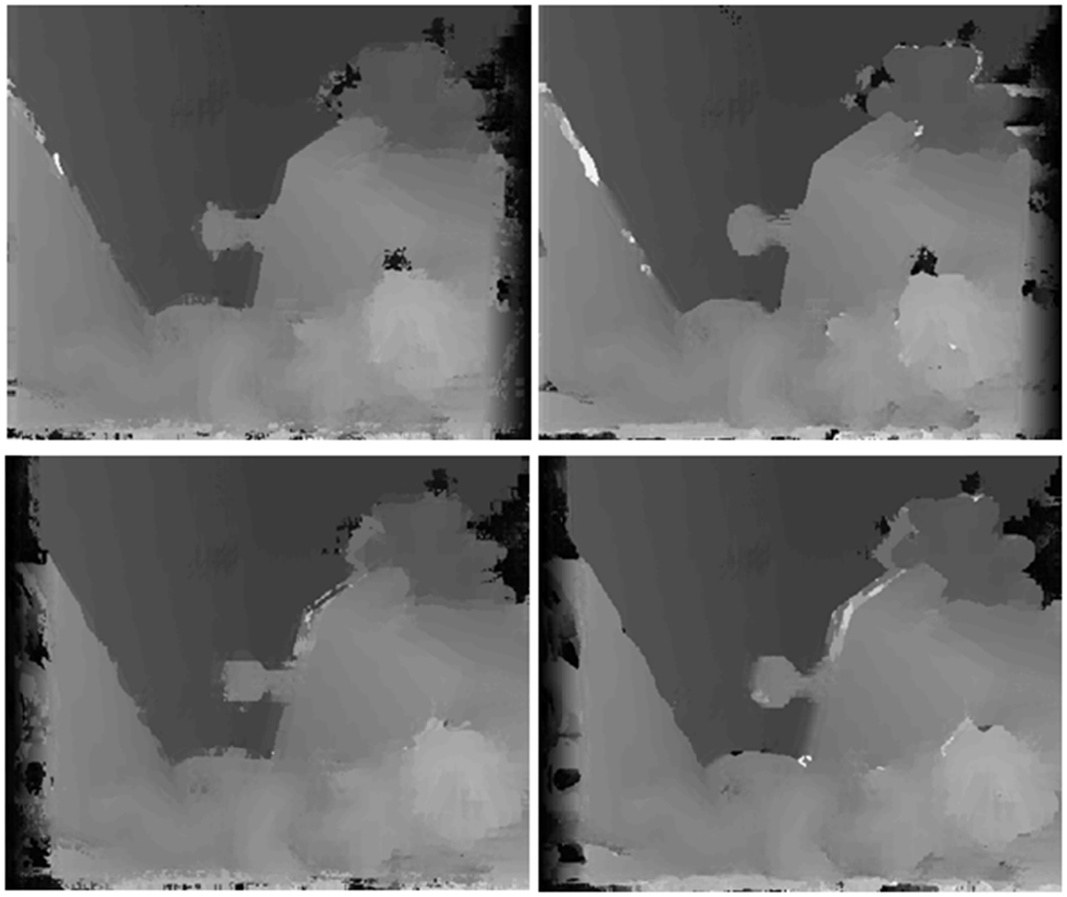


Fig.4.6. Examples of Voted Disparity Maps

Furthermore, there is a potential for picking the value with the maximum correlation or the least SAD distance which also improved the results, but they were still very far from being accurate as shown in Figure 4.7.

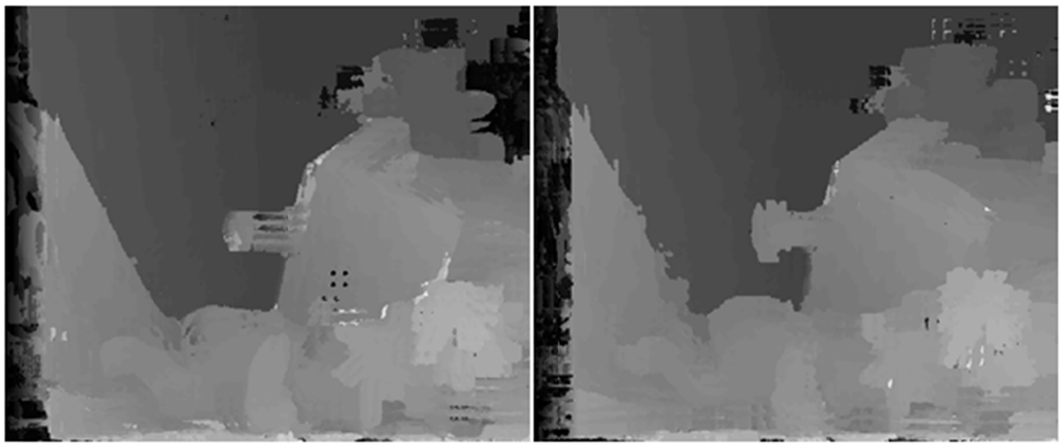


Fig.4.7. SAD and NCC best distance amongst the four possible disparities

Another possible approach to consider here is to use nearest neighbor filling for the unknown disparities. And in this case, it is possible to choose the nearest neighbor in terms of intensity and then choose the nearest value amongst the four or five possible values if considering the initial disparity value at the pixel. An example of an obtained disparity map using this approach is shown in Figure 4.8 and although this helps improve the results and better than the previous methods, it is still not very accurate.

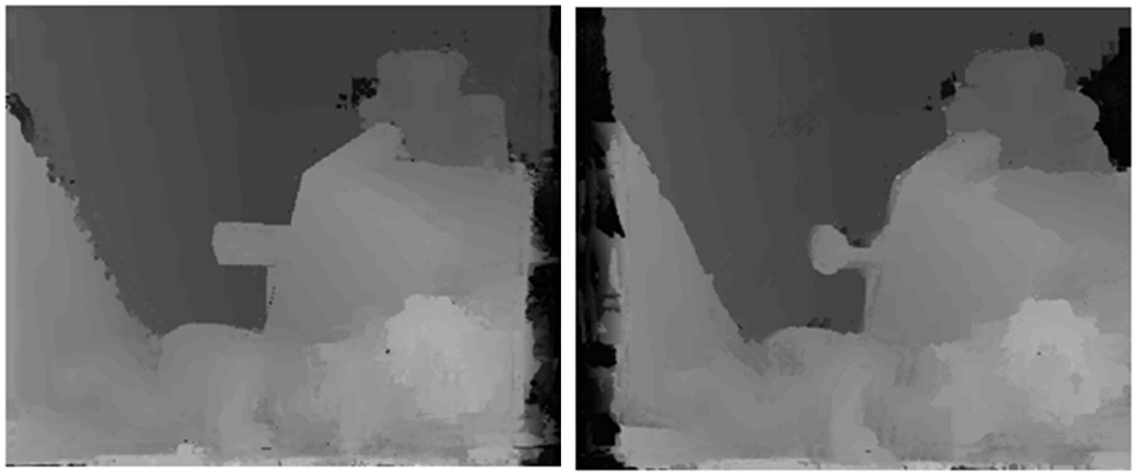


Fig.4.8. Example of SAD completed disparity map (left and right)

Besides these, there are many potential methods for completing the disparity map using Graph Methods or using any sort of grouping pixels together as through segmentation which is discussed in the next section. Moreover, there is a possibility of using classification and this is the main technique presented in this work, since the obtained map here constitutes the main basis for the features later described in Chapter 6.

In addition to the mentioned points, several ideas are discussed and these generally help improve on the results of stereo matching such as right and left validation which is used to reduce the occlusion problem already explained in Chapter 2. Another idea is sub-pixel accuracy which is further used to improve the accuracy of the found disparities. These ideas are briefly discussed next.

4.2.1 Left to Right Cross Checking (LRC)

Computing the left to right disparity means finding for the left pixels the best possible match on the right and performing the right to left means the opposite. Each one of these is a different disparity map, but they can be directly combined together.

This is done by checking the matching between the pair on each side and this leads to validating some of the found disparities since in perfect conditions, each pixel should be exactly matched on both sides. Figure 4.9 shows the left and right disparity maps after performing this process.

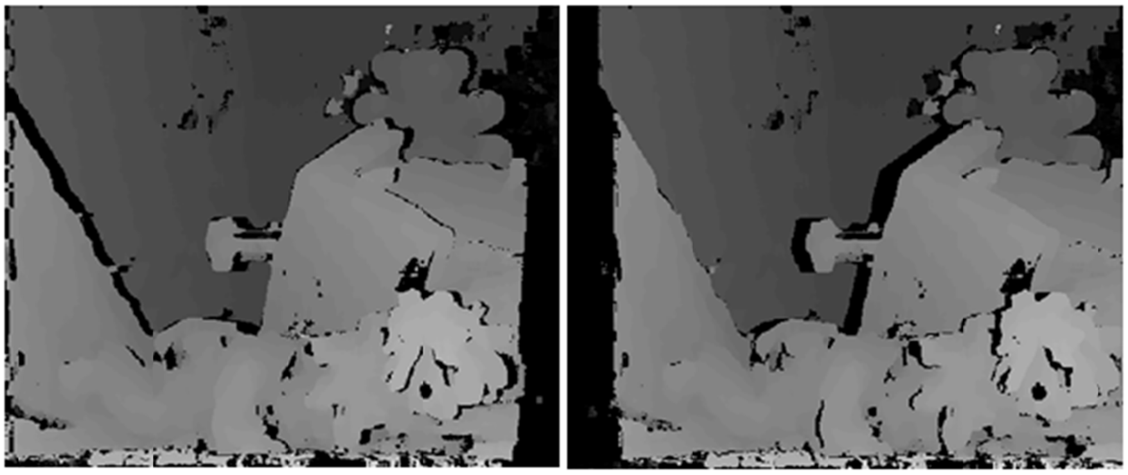


Fig.4.9. Example of Right and Left SAD validated disparity maps

4.2.2 Sub Pixel Accuracy

After completing the disparity values using the proposed ideas and as many other developed algorithms in this field, sub pixel disparities can be computed for every pixel, but this is not addressed here since the effect seemed of small enhancement. In any case, this can be done by considering on every pixel a similar disparity region with a relatively lower difference such as 1 or 2 from the disparity of the pixel according to the disparity range, and then using this region as a kernel that is matched on the opposite image within a very small interval that usually varies between disparity-1 and disparity+1, meaning that the output disparity value could be any value in this interval. In order to obtain this value, the SAD distance is considered at the decimal values of the interval which are usually three and then parabolic interpolation is used to find the point with the lowest value in the parabola is selected to be the sub pixel distance. Of course, this is not the most convenient way as there are many works that target sub pixel accuracy issues as in [151], but this work is restricted to the basic method [1].

Since the effect of performing this process on the disparity image improves accuracy by slightly varying the disparity, no visual differences are observed and therefore, no figure is displayed to show the effect here.

4.3 Stereo and Segmentation

Segmentation is a method that groups sets of pixels together according to their intensity values or other properties, and in this work we use edge based segmentation in the manner explained in Chapter 3. And since it is already noted in the previous section, that it is possible to use segmentation for completing the disparity values of unknown pixels this approach is first discussed here.

Moreover, and especially that ASW based methods are very popular for addressing stereo matching, we present a new improved approach that combines segmentation and adaptive weights to achieve relatively accurate disparity maps.

Besides the used approaches, there are many ways of using segmentation to solve the stereo matching process and we cannot explain them all.

4.3.1 Direct Stereo Matching from Segmentation

Since segmentation aims at grouping pixels into different blocks, it is directly possible to use the found blocks as kernels when performing a local stereo matching method such as SAD or NCC. Thus, the kernel is no longer a square neighborhood of pixels surrounding the pixel of interest, but rather the pixels that belong to the same segment. This directly leads to greatly reducing the number of checked kernels since instead of checking number of pixels kernels, number of objects kernels should be checked although these kernels are not rectangle or square shaped. There are some problems with this method since some objects are too large, so they are preferably decomposed into smaller ones, or some objects are too small, so they are better considered as parts of other larger ones. Another issue here is that all the pixels in the same object would have the same disparity which is rarely the true case, which means that it is better to perform smoothing for objects with similar disparities. It is important to note here that several combinations were tested to improve on the accuracy of directly using segmentation and these are not detailed in this work. Figure 4.10 shows an example of the obtained disparities using the SAD method for the segmented regions.

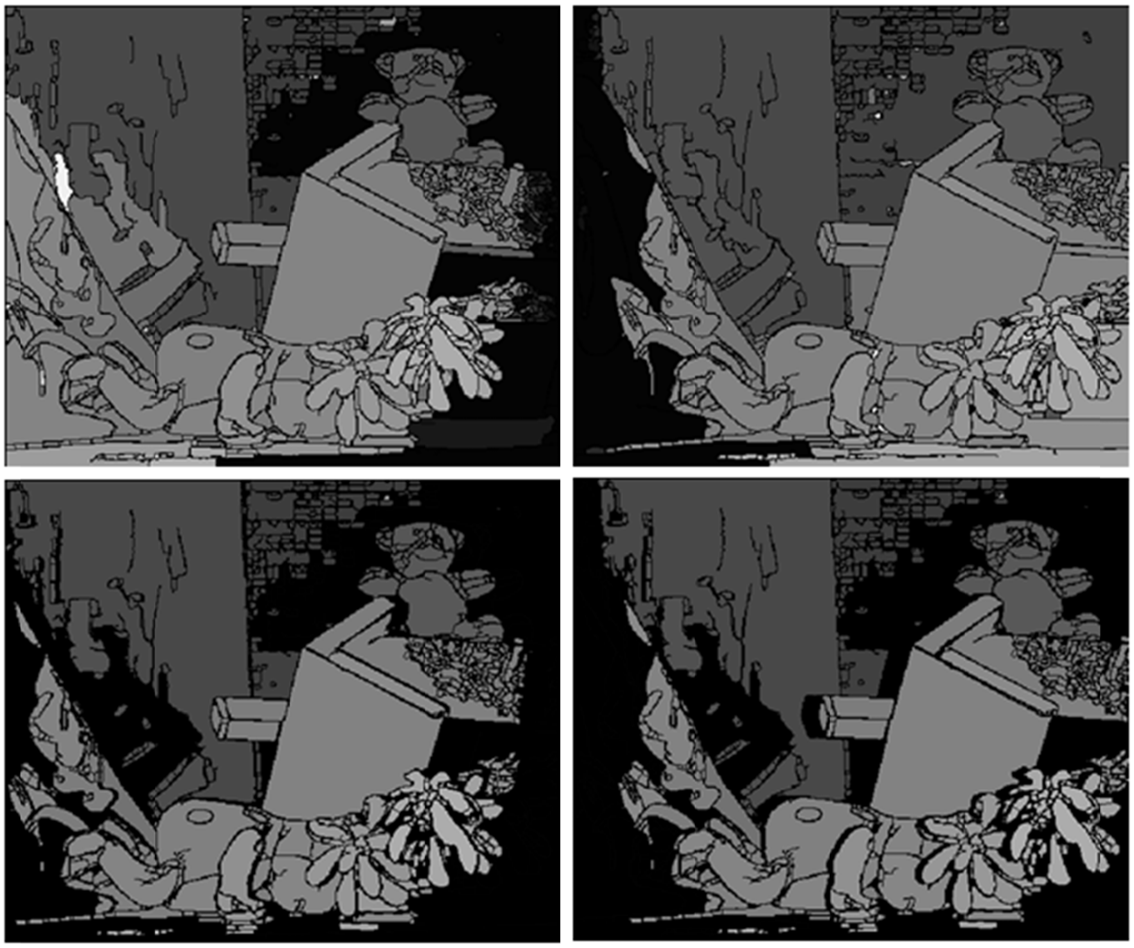


Fig.4.10. Example of Obtained Disparity Maps through SAD Based Segmentation

Figure 4.11 shows the same previous example but after dividing the rather large blocks into smaller ones. Of course, it is clear that there are some errors existing in this. So, right and left validation is performed which also shows that errors exist and leaves some unknown disparities.

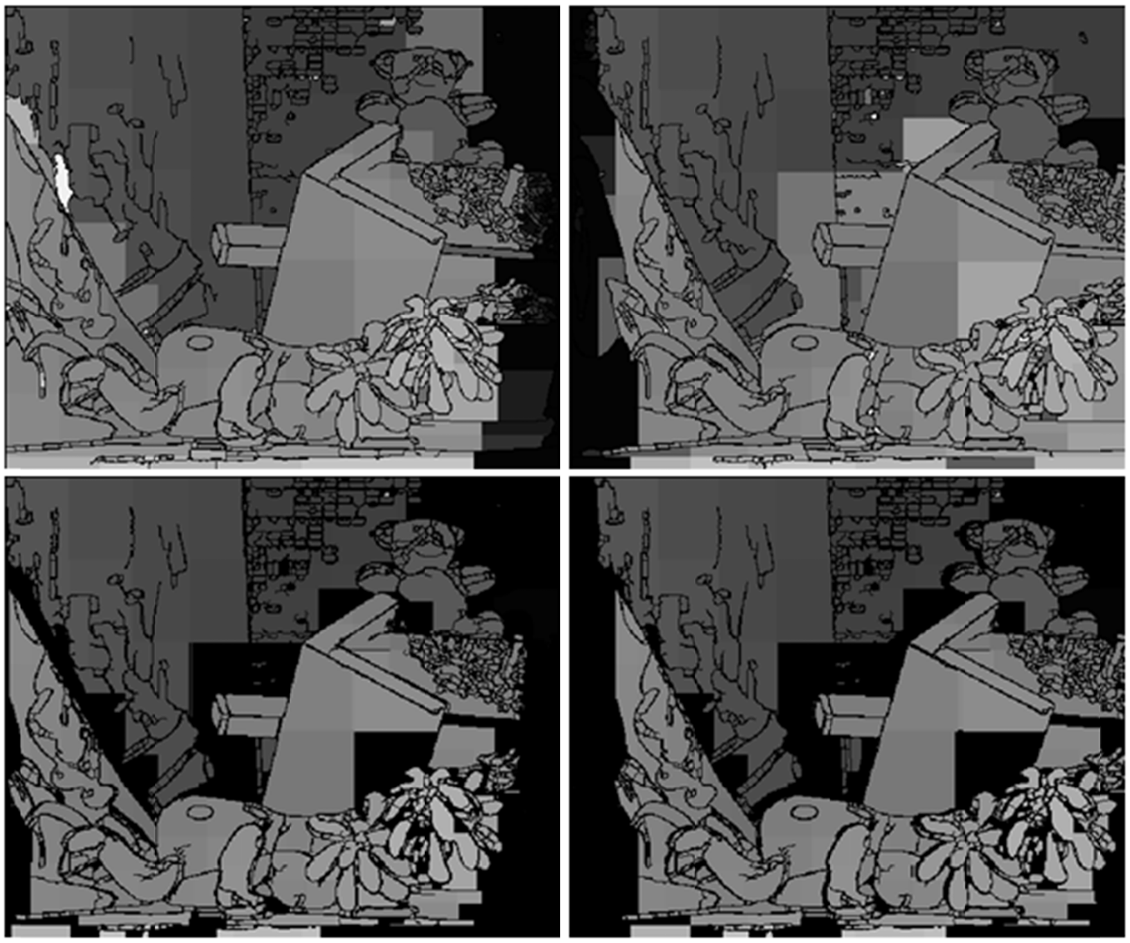


Fig.4.11. Example of Obtained Disparity Maps through SAD Based Segmentation with smaller objects

Afterwards, there are some possible modifications that include removing the problematic objects at the boundaries of the image which mean the right pixels in the image to the left and left pixels in the image to the right. Then, the relatively large objects that were divided into smaller parts can be combined and further left and right validated as shown in the next figure which is relatively of acceptable accuracy.

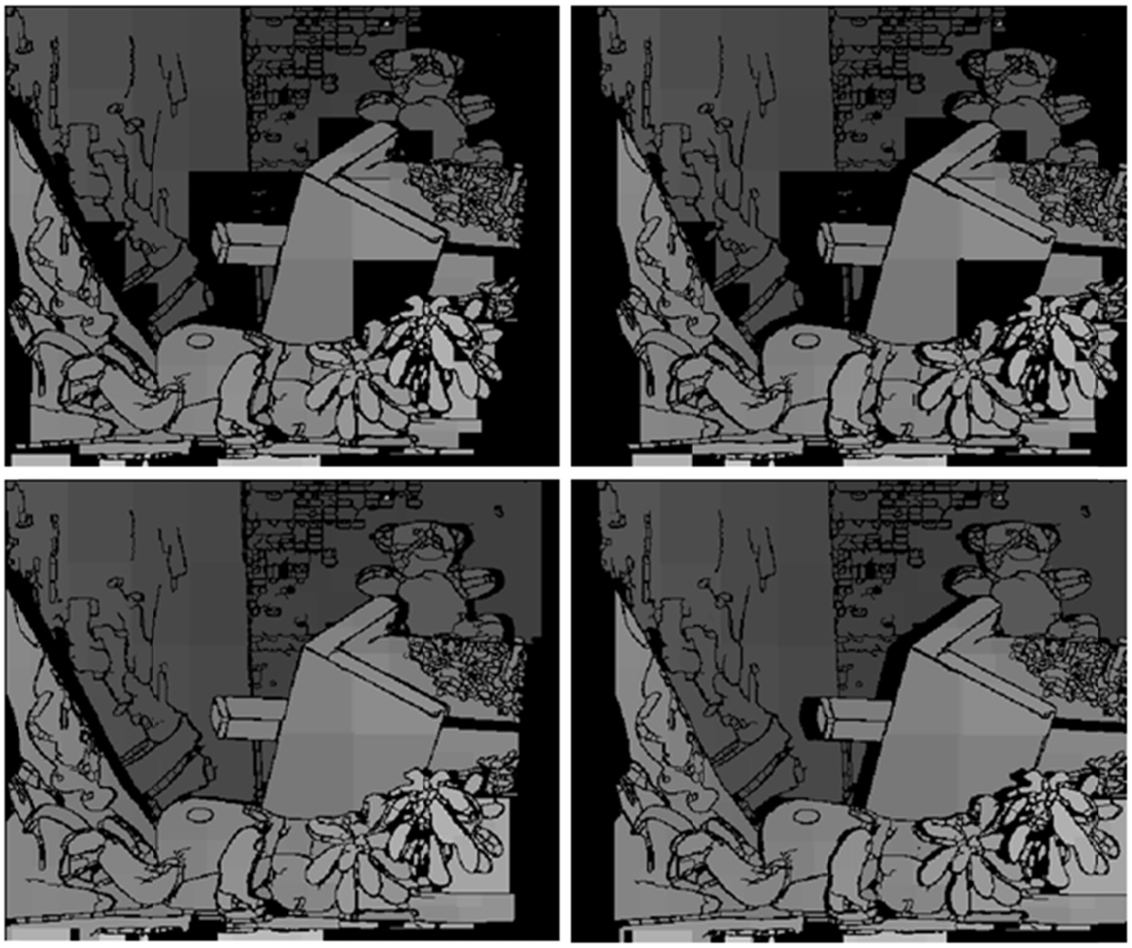


Fig.4.12. Example of Obtained Disparity Maps through SAD Based Segmentation after filling the objects

4.3.2 Completed Stereo Matching Using Segmentation

In Section 4.2, stereo matching was addressed in terms of finding a set of high confidence disparities that left many disparities unknown, and segmentation was mentioned as a candidate for completing the pixels of unknown disparities which is dealt with in this section.

At first, and starting from the disparity map composed from unknown and pixels with rather confident disparity values, any segment or object having a confident disparity is regarded to consist of pixels that have disparities that are near to the confident value. This is not direct since objects that have confident but very different disparities are not considered. This is reasonable because pixels in same objects should not greatly differ from each other. Thus, the median value of this object is found and all the pixels having close disparity values to the average of the object are regarded true with those that are not close enough regarded false and replaced with the median value.

Then, for every object, and for every pixel in the object, the disparity in the possible vector that is closest to the confident disparities in that object is chosen unless it is very different since in such a case this means none of the possible vector disparities are true further processing needs to be carried.

Of course, this does not solve the whole problem, since many objects have no confident disparities and some objects have pixels with unidentified disparities. So, the objects with known disparities are filled according to their neighbors. And afterwards, the remaining objects are also filled according to their nearest neighbors. An example of the obtained disparity map through the main stages is shown in the next figure.

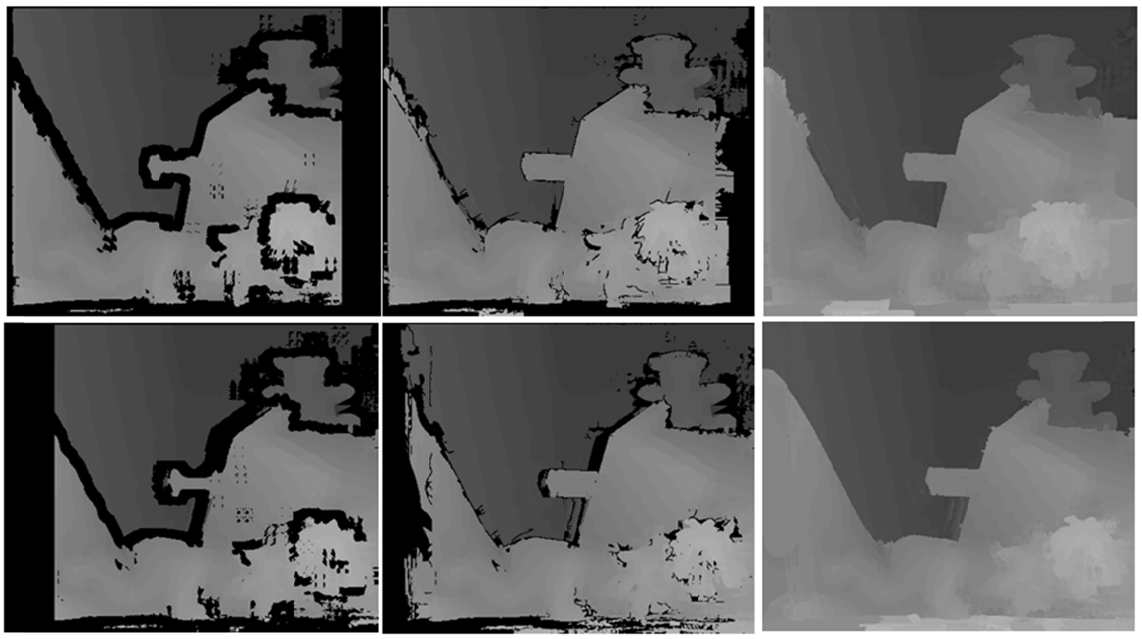


Fig.4.13. Example of Obtained Disparity Maps through the different stages (left and right)

It is important to emphasize that the black values indicate a blank disparity and these are filled according to the steps indicated. Also, it is necessary to indicate that some of the erroneous pixels can be fixed using left to right and right to left validation where the disparity maps are found for each image and then validated, with the remaining pixels filled according to the nearest edge pixels or the near known pixels in the same segmented region.

Afterwards, sub pixel disparity can be found. This can be done by accounting for similar disparity regions with a relatively lower difference such as 1 or 2, and then a smoothing operation can be performed along the edges of the different objects which proves the importance of edges.

4.3.3. Segmentation with adaptive support weights stereo matching

The proposed approach uses a method similar to [19]. The method exploits the segmentation information in an ordered manner to improve the accuracy of stereo matching although this consumes much time. The main assumption considers that pixels lying in the same segment have close disparities, which is generally meaningful if the segments are relatively small. This assumption is especially valid if the segmentation was well performed.

So, we consider a kernel window around each pixel ($p_{x,y}$), and for every pixel ($p_{i,j}$) in this window we need to assign a weight. The weight is determined according to:

$$w(p_{i,j}, p_{x,y}) = \begin{cases} 1 & \text{if } p_{i,j} \in \mathit{same\ segment\ as}\ p_{x,y} \\ \exp\left(\dfrac{-dist(I(i,j), I(x,y))}{\delta}\right) & \mathit{otherwise} \end{cases} \qquad (4.4)$$

This equation is depicted from [19]. In this equation $I(i,j)$ denotes the intensity value of ($p_{i,j}$) and δ is a weight factor that is set to 22.

Also, $dist(I(i,j), I(x,y))$ represents a measure of the difference between intensity values. This difference could be the basic absolute difference in gray intensity values for gray scale values or the summation of the absolute differences for all the color scales.

Since stereo matching aims at finding matching pairs in the two stereo images, it can be viewed as minimizing a cost function for every pixel to obtain the disparity map. The usual cost function could be the SAD or NCC as already discussed in Chapter 3, but in order to use the found weights, the following cost function is used:

$$C\left(L_x, R_{x_r}\right) = \frac{\sum\limits_{L_i \in W_l\ ,\ R_{i_r} \in W_r} w_l\left(L_i, L_x\right) \bullet w_r\left(R_{i_r}, R_{x_r}\right) \bullet TAD\left(L_i, R_{i_r}\right)}{\sum\limits_{L_i \in W_l\ ,\ R_{i_r} \in W_r} w_l\left(L_i, L_x\right) \bullet w_r\left(R_{i_r}, R_{x_r}\right)} \tag{4.5}$$

In this equation, L_x stands for the left pixel under consideration, and R_{x_r} stands for the candidate right pixel. Also, L_i stands for the left pixel in the left window W_L, and R_{i_r} stands for the right pixel in the right window W_R. The different (w s) stand for the different weights in the corresponding windows. Moreover, TAD stands for Truncated Absolute Difference, which is the same as the absolute difference (AD) except that it introduces an additional constraint according to $TAD(a,b) = \min(abs(a-b), v)$ where v is a preset maximum value. v is set to 80 in this work.

In equation (4.5), $TAD\left(L_i, R_{i_r}\right)$ could be the truncated absolute difference in gray intensity values or the summation of the truncated absolute differences for all the color scales.

Using this relatively time consuming approach one can achieve the required disparity map. Moreover, many works tend to combine the gradient edge information by choosing cost functions that further use the TAD for the gradient images as in equation (4.6).

$$C\left(L_x, R_{x_r}\right) = \alpha \frac{\sum_{L_i \in W_l,\ R_{i_r} \in W_r} w_l\left(L_i, L_x\right) \bullet w_r\left(R_{i_r}, R_{x_r}\right) \bullet TAD\left(L_i, R_{i_r}\right)}{\sum_{L_i \in W_l,\ R_{i_r} \in W_r} w_l\left(L_i, L_x\right) \bullet w_r\left(R_{i_r}, R_{x_r}\right)} + (1-\alpha) \frac{\sum_{L_i \in W_l,\ R_{i_r} \in W_r} w_l\left(L_i, L_x\right) \bullet w_r\left(R_{i_r}, R_{x_r}\right) \bullet TAD\left(GL_i, GR_{i_r}\right)}{\sum_{L_i \in W_l,\ R_{i_r} \in W_r} w_l\left(L_i, L_x\right) \bullet w_r\left(R_{i_r}, R_{x_r}\right)} \tag{4.6}$$

In this equation G is added to denote the gradient value of the corresponding image. Also, α stands for the percentage of effect due to the regular intensity image. It is important to mention here that the Gradient can use any preset mask, and in this work we relied on using the proposed Al-Alaoui mask in equation (3.9).

So, using equation (4.6) should further improve the results, but it was noticed that using the second term on its own leads to another disparity map which can be more useful, so this was performed in this work. This means that one now has two maps that can be used together according to certain conditions.

- The first step in the proposed approach here is to validate the similar disparity values in both maps, which leaves a set of invalid pixels.

- Next, we determine the median value, excluding invalid pixels, for each segmented region. So, and for every invalid pixel, we choose the nearest value amongst the two disparity map values to the computed median value, which yields an enhanced disparity map.
- After that, it is possible to re-compute the median value for each segmented region and validate that each disparity value does not fall far from the median value, which leaves the disparity map with unknown values that can be later filled.
- Afterwards, and since it is possible to compute the two corresponding disparity maps using this approach, one can perform LRC as already discussed in section 4.2, which leads to new disparity maps that have a larger set of invalidated pixels.

After that, all the invalidated pixels can be filled according to the described completion stage in Chapter 6.

Figure 4.14 shows the obtained disparity map for the Teddy images through the proposed steps.

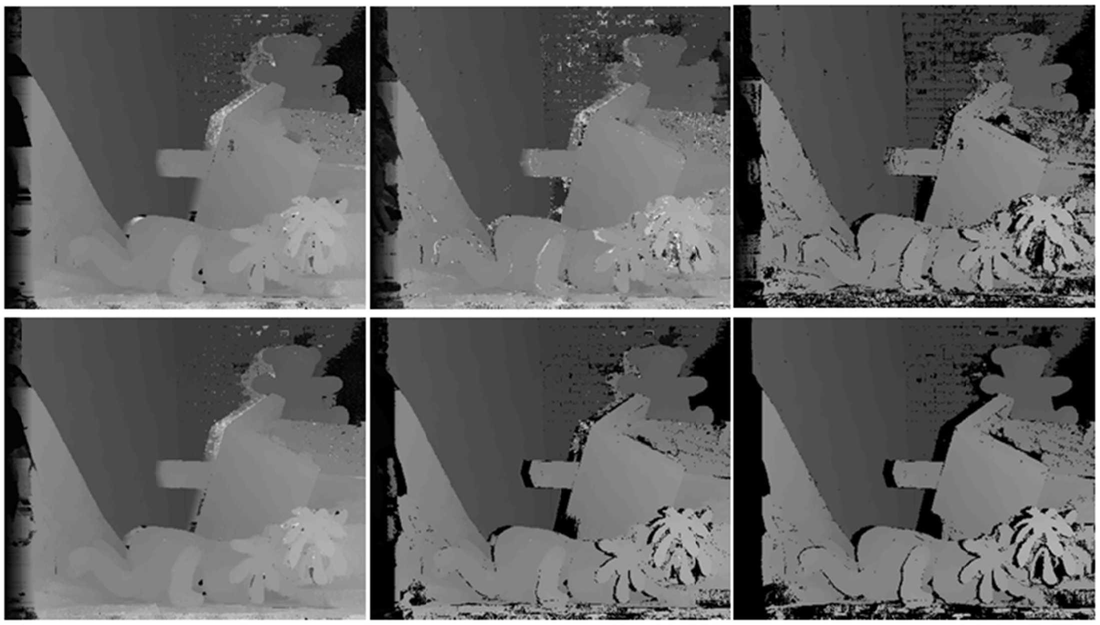

(a) (b) (c)
(d) (e) (f)

Fig. 4.14 (a) Initial disparity map as by [19], (b) Initial map through gradient image, (c) Combined validated map, (d) Filled map through nearest to median value, (e) Validated map, (f) LRC validated map

It is necessary to note that the obtained map can be further improved as later discussed in this work.

CHAPTER 5

CLASSIFICATION

As already noted, it is beyond our scope to cover everything related to classification since we concentrate on solving the stereo matching problem as a classification problem. Mainly, the classification related ideas are developed in a rather similar manner to the works in [85-89]. This chapter presents the proposed ideas regarding classification and these are afterwards related to stereo matching in the next chapter. So, this chapter does not deal with any idea regarding stereo matching.

5.1 The Proposed Method

This section deals with the different enhancements used for improving the boosting techniques applied in this work. The handled boosting techniques include AdaboostM1, GentleBoost and LogitBoost. These methods were briefly presented in Chapter 2.

The premise behind the method implemented here is that there exists a possibility of finding a better learner solution out of combining some of the classifiers instead of using all of them. This, as a very basic example, can be observed in some cases when the boosting method achieves better accuracy if limited to a lower number of iterations rather than a greater one due to many reasons that include over-fitting, quality of the learners, etc...

In other words, the basic concept is that instead of selecting all of the classifiers found in the different iterations as in the Adaboost algorithm, or the best classifier amongst the set of classifiers as in the approach introduced in [85,86,87,88,89], one selects a group of classifiers that aim at improving the accuracy of classification.

Thus, and with these ideas in mind, a search amongst the solutions needs to be conducted in order to find the best possible solution. But since for *(T)* classifiers, there are almost *(2^T)* possible combinations, it is of great importance to find a way, whether optimal or near optimal, to choose between the learners. Definitely, if *(T)* was small, it is directly possible to try the different combinations.

Two main methods are considered for selecting the classifiers, with both relying on the training data available and searching through the space of possible combinations. After performing both of these methods, the best set amongst the two obtained ones is selected to have the final chosen set.

The techniques aim at minimizing the error in the training data, which does not necessarily mean minimizing the error on the testing data. And as already emphasized, the main problem to avoid is over-fitting and for this purpose, the handling of the training data is modified to evade this trouble.

The used schemes along with the modification or the two steps used for training are explained next.

5.1.1 Modified Training

The aim here is to avoid over-fitting. The approach used is rather straightforward and has generally shown to lead to acceptable behavior that escapes over-fitting and ensures performance enhancement.

The training data is divided into two parts. The first part comprises most of the data, as in 90% of the data. The second part comprises the remaining part, as in the 10%

remaining of the data. Then, the major part is used as the data for training. This large data part is the part that the proposed selection approaches explained afterwards depends on which means that the methods aim to minimize the error on this part of the data as a first step. The remaining data is rather small and is used for verifying that the obtained set of classifiers is able to achieve acceptable results when subjected to the remaining data as a second step. This is rather similar to the RE method which also divided the training set into different parts.

Therefore, and this is performed after the selection approaches, the small set of data is tested on the obtained set of classifiers and the found error is compared with the error of the original set. And if the found error is better than the original, the found set of classifiers is accepted while otherwise the selected set becomes the original one.

5.1.2 Combining Classifiers Search

Before directly explaining this method, it is worth noting that the two approaches used here are similar except that the first one builds from small blocks while the other removes from large blocks. The two methods are then used together to ensure better results.

In this search, one considers having a (T) number of classifiers. The basic idea is to start from all the classifiers which are of cardinality one or consisting of one classifier and then trying the different combinations with the remaining classifiers. Then, out of the found combinations, an (X) set of combinations is selected based on having the best results so far, and so forth… This process is continued for a finite number of steps or as a maximum until one has all the classifiers in one block, which is the original set. This can be called a “small to large”, or building and combining approach, since it starts with smaller components and gradually joins the blocks together until no more is possible. Figure 5.1 shows a flowchart of this method.

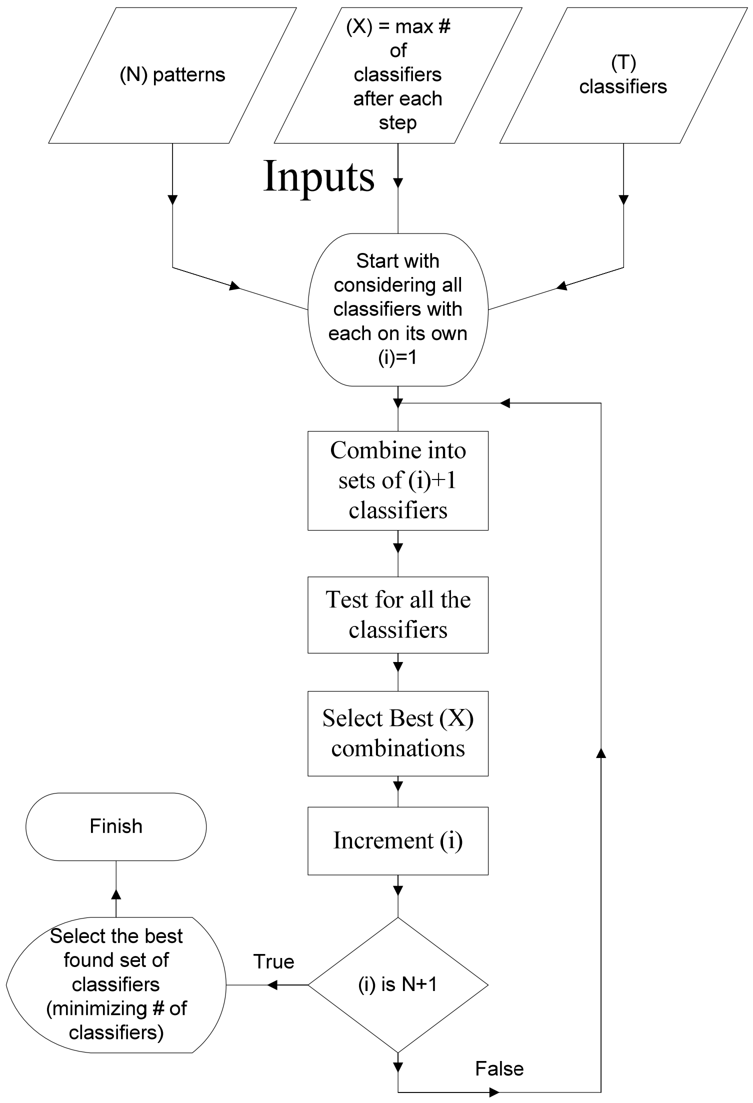

(N) patterns
(X) = max # of classifiers after each step
(T) classifiers
Inputs
Start with considering all classifiers with each on its own (i)=1
Combine into sets of (i)+1 classifiers
Test for all the classifiers
Select Best (X) combinations
Increment (i)
(i) is N+1
True
False
Select the best found set of classifiers (minimizing # of classifiers)
Finish

Fig.5.1. Flowchart of Combining Classifiers Search

Figure 5.2 shows an example of a few stages of this binding search, where the number of classifiers is (T=7) and the selected set (X=3).

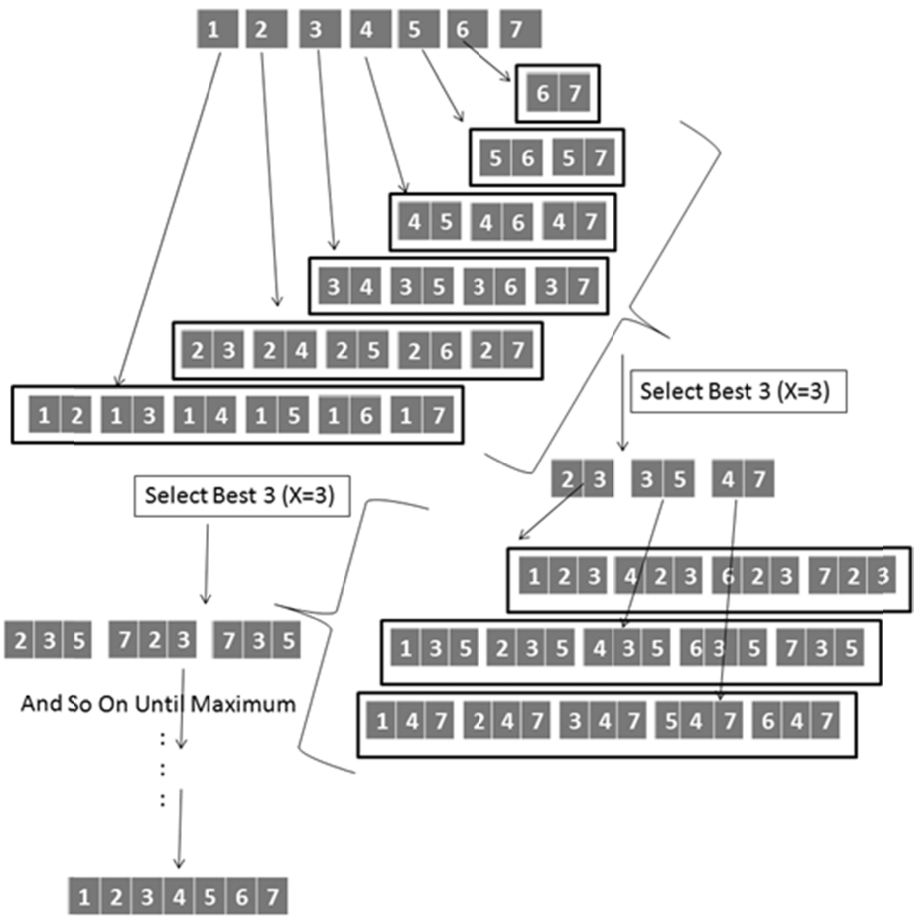


Fig.5.2. Example of the Combining Approach for T=7 and X=3

Analyzing the number of searches conducted through this technique is rather straightforward. The first stage consumes $\sum_{i=1}^{T-1} i$ searches. Then, *(X)* of these are chosen and each one leads to a maximum of *(T-2)* combinations meaning that there is a maximum of *(X*(T-2))* combinations. This stage is repeated with *(T-s)* combinations, *(s)* being the stage number, being found for each block of *(X)*, so each stage tests *(X*(T-s))* combinations.

Since *(s)* varies from 2 to *(T-1)*, this leads to having the following maximum number of possible combinations:

$$Space\,Size \leq \sum_{i=2}^{T} \left(X * T - 1 - i(X-1) \right) \tag{5.1}$$

At the end of this search procedure one can select the set or sets with the lowest error or in the vicinity of this error, as in greater or equal to the (error-1).

It is important to highlight here that this combination method is similar to the RE procedure except that it maintains additional sets which are further tested for accuracy. So RE is a special case of this where *(X=1)*. This proposed approach should produce better results than RE which is also validated when presenting the results.

5.1.3 Separating Classifiers Search

This method is rather the opposite of the other one. It starts by considering all the classifiers as one which could be the last step in the previous approach or which is what is generally adopted in boosting and bagging algorithms. Then different incremental eliminations are tried while selecting the best *(X)* sets at each step and so on for a certain number or until we end up with a set of separated classifiers. This can be called a "large to small" or a reduction approach since it starts with all classifiers and ends up with a smaller number.

Figure 5.3 shows the basic structure of the noted search procedure.

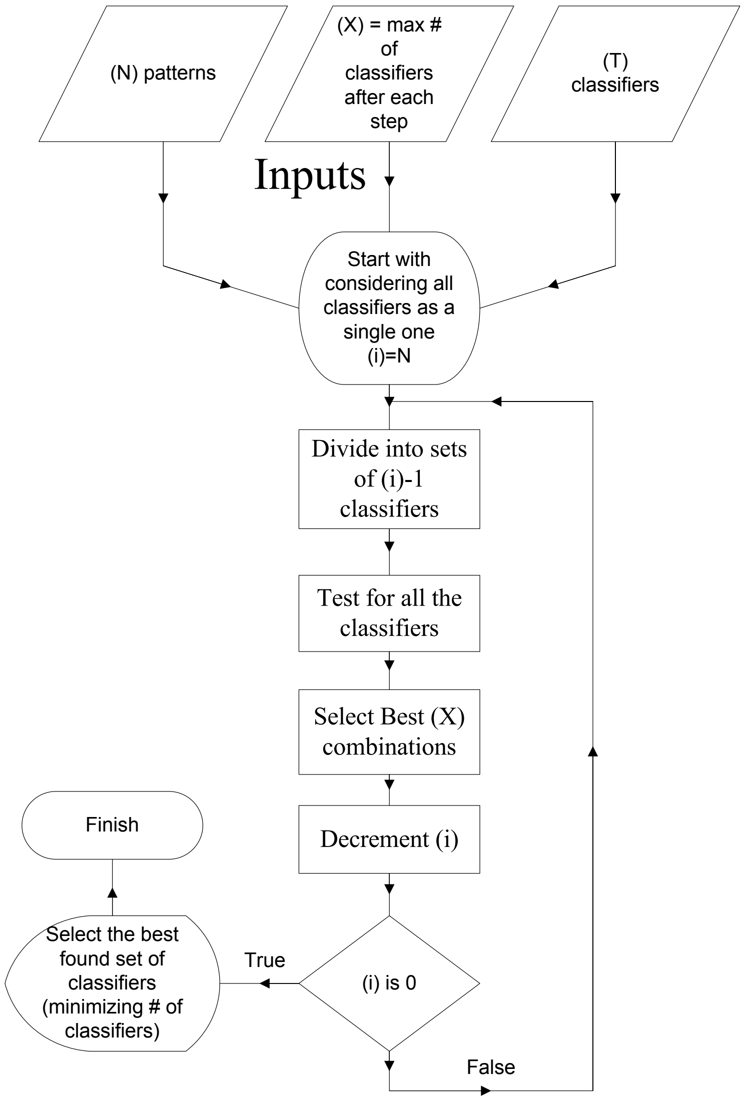

(N) patterns
(X) = max # of classifiers after each step
(T) classifiers
Inputs
Start with considering all classifiers as a single one (i)=N
Divide into sets of (i)-1 classifiers
Test for all the classifiers
Select Best (X) combinations
Decrement (i)
(i) is 0
True
False
Select the best found set of classifiers (minimizing # of classifiers)
Finish

Fig.5.3. Flowchart of Separating Classifiers Search

Also, Figure 5.4 shows an example of a stage of this separating or eliminating method, where, as before, the number of classifiers is (T=7) and the number of selected combinations is (X=3).

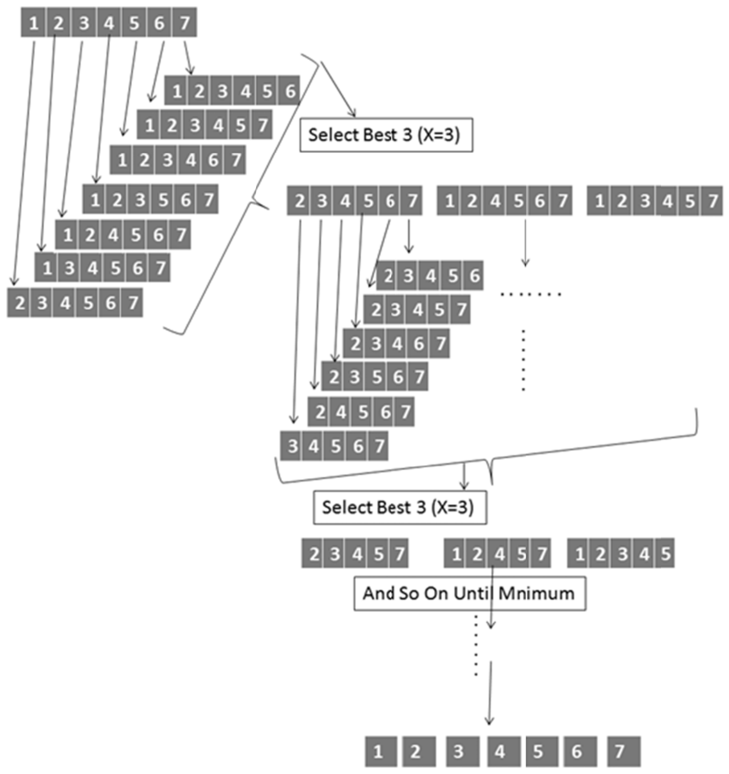


Fig.5.4. Example of the Separating Approach for (T=7) and (X=3)

In a similar manner as before, a basic analysis of the number of searches is noted. The first stage consumes *(T)* searches. Then, *(X)* of these are chosen and each one leads to a maximum of *(T-1)* combinations leading to a maximum of *(X*(T-1))* combinations. This stage is repeated with *(T+1-s)* combinations, s being the stage number, being found for each block of (X), so each stage tests *(X*(T+1-s))* combinations. Since (s) then varies from 2 to *(T-1)*, this leads to having the following maximum number of possible combinations:

$$Space\,Size \leq T + \sum_{i=2}^{T-1} X * i \tag{5.2}$$

Also, and like the previous search, one can select the set or sets with the lowest error which could lead to selecting the original set of classifiers if it was found to be of lower error.

5.1.4 Discussion and Combining the Approaches

Each of the two proposed methods leads to a tree of possible selections. Each selection in this tree is made of a set of classifiers of varying cardinality and having a certain training error value. Afterwards, it is possible to select the minimum error set(s), or even up to certain cardinality or pruning level according to the problem at hand and the different possible limitations which may include data storage and speed.

In this work, each approach is generally found to lead to sets of lowest error. Then, these different sets are tested using the small data part. Out of the different sets, the set that provides the lowest error on the total training data is selected.

Still, this selection needs to be accepted or rejected. So, the small data error of the original or total set of classifiers is compared with that of the selected set. Only if this error and the error of the large data part are lower than the original set errors, the set is accepted. If the found set was rejected, the original set is chosen as the selected solution which means that there is a good probability of having the original set as the selected one.

This training approach and acceptance criteria is used against over-fitting, since in different cases and although the training error of the selected set was the lowest, the original set performed better on new data. So, using these ideas provided higher confidence when testing the selected classifier set on new data and these can be noted later when presenting the results although in many cases the testing error was very close to the original one due to the accuracy of the original set and the acceptance and rejection criteria.

Besides this, it was found that in most cases and as the simulations and results will later show, an improvement in at least the cardinality or the training error was achieved.

5.1.5 Simulated Annealing Based Approach

In addition to the presented strategies, we use the well-known heuristic optimization technique, simulated annealing to search for a better set of classifiers. It is worth mentioning here that optimization was previously used in literature for online boosting algorithms as in [14].

In general terms, SA is a statistical heuristic search method that aims at achieving the global optimal solution of a certain problem. It is based on heat annealing which uses varying heating and cooling especially through slow cooling metallic materials to harden and change the properties of the matter into a better atomic arrangement. In the optimization version, and similar to the metallic process, the SA method generates a random solution at each iteration according to certain initial limits. The solution is either accepted or rejected based on the obtained function value and the temperature variable (T) which is analogously decreased as in the cooling process. The randomness of the solution is high when T is high in order to escape local optima and it decreases to near the optimal as T decreases. It should be noted that performing SA multiple times from a different start or initial guess enhances the possibility of nearing the global optimum.

SA is performed for a number of iterations and in this work we further modify it to lead to several solutions by saving the history of tested sets of classifiers. So, we select the top five or (X) performing selections in a similar manner to the previous strategies, meaning that we perform SA on the modified training data and save the top solutions. Out of the (X) sets, we choose the best performing set on the remaining training data. Besides, we utilize the acceptance criteria noted in the previous section to accept or reject the found solution and validate the performance of the selected set of classifiers.

It is important to note that we initialized the total set of classifiers as the starting solution and restricted SA to perform a maximum of 500 iterations to reduce time consumption. This directly means that no more than 500 possible combinations are checked.

5.2 Classifiers or Learners

As already noted, ensemble learning methods like boosting use a certain classifier or learner of which the most common one is decision trees. Although it is possible to consider various learners such as the Mean Squared Generalized Inverse Method, or the naïve Bayes Classifier, we concentrate on decision trees noting that preliminary results using other learners have led to similar conclusions.

Basic decision trees are the most famous weak learners used in boosting and bagging algorithms [93, 94, 95] with so many references on them, and this is due to many reasons, which include having the ability to be very simple and basic in terms of stumps, or much more complex with many conditions and settings. Furthermore, they can be either strict binary classifiers having a direct decision like (0,1) or (0,1,2) according to the number of classes, or regression classifiers leading to a numerical value, which could be useful in a bigger range of problems, where a value needs to be found.

In this work, they are used in their weakest or simplest form, in which they are also called stumps, where one node is found in every tree. In general, each node in a tree basically accounts for one of the features at its condition to predict the outcome of the input.

It is worth mentioning that for a 2 feature vector case, or 2D Data, with x and y values, basic decision trees are either horizontal or vertical lines placed at a certain point in the graph.

5.3 Real Life Scenario

The example here is not intended to be a validation of the ideas considered for selecting classifiers, but rather an example of using the concept of selecting classifiers in real life. Also, it is not by any means based on this work or invented here since it is a simple situation that many times occurs in real life. Moreover, some could argue against such an example and that is very much acceptable since the work does not aim at the proof of this and the problem is not necessarily a classification one.

The main problem assumes that there a quarrel or conflict happens between two groups of people on a certain issue, no matter what that is. One way to solve the problem is to send another group of people to try to amend the different issues and solve the quarrel or at least reduce the tension between the conflicting parties. So, if we assume we have 50 people that could be available and are usually ready to contribute and visit both parties, the question then lies in the procedure used to involve the willing people.

Thus, it would seem that one could either send all the people to solve the issue and have each one of them say their word. A second solution would be to choose a group of the people of whom all should say their word, and in such a case the problem becomes in choosing this group. Also, as a third solution, it is further possible to combine the two methods by choosing a group of people to have their say and sending the remaining people with them for moral support, or without having their say. Of course, there are other possible solutions, as in having all the people contribute their saying before going and sending a part as representatives to deliver the main ideas, but that would be rather similar to the second approach if the representatives were well prepared. Another possible idea is to divide the whole set into groups and send each group on its own, which seems like a combination of the first two approaches and there lies a difficulty in choosing the groups.

Concerning the first approach, one can argue that too much people would tend to waste more time, and could cause a bit of problem if not prepared to work together. Concerning the second, the problem becomes choosing the representatives since they must be adequate, prepared, and well coordinate between each other. The third approach has the advantages of both with one rather outside disadvantage concerning the willing people, because involving everyone at such varying fashions could cause an issue between them.

So, and in analogy to the classification problem and selecting classifiers, the two first ideas are applicable and one could argue for the second against the first since the first is the one used in boosting, bagging and others, while the second is the proposed according to the work here. For the third proposition, it is definitely worth investigation although it is a bit harder since moral support is rather hard to consider in the context of classification problems.

Thus and as already stressed, this is not meant to be a proof or exact corresponding scenario, but rather an analogous idea to argue for the proposed methods or show their possible correspondence to real scenarios.

CHAPTER 6

STEREO MATCHING WITH CLASSIFICATION

Chapter 5 concentrated on classification which could seem unrelated if one considers the general stereo matching literature, but in this work we combine stereo matching with classification in a novel manner. So, the method used to perform the proposed ideas is explained in what follows.

6.1 Stereo Matching Using Classification

After presenting the classification techniques, it is of great importance to relate them to the stereo matching problem and explain the adopted approach here, which is the main contribution of this work.

As already emphasized, the stereo matching aims at computing the disparity which is first performed using a certain selected approach which can be a local basic method as in SAD. After that, it is possible to check whether the found disparity is correct or not, which is a binary problem and therefore a candidate classification problem.

Besides this, if one checks the neighboring disparity values of a certain pixel, it is highly likely that the correct disparity of the current pixel is one of its neighbors and this can be seen in the next figure. For example, assume that SAD was performed according to the explained approach in this work, then for every pixel, let's consider a neighborhood of pixels according to the SAD radius used and if one is to choose for every pixel the nearest to the true disparity one can obtain a disparity map that is of rather high accuracy as shown in Figure 6.1 for the teddy image which shows the possibilities available through these ideas.

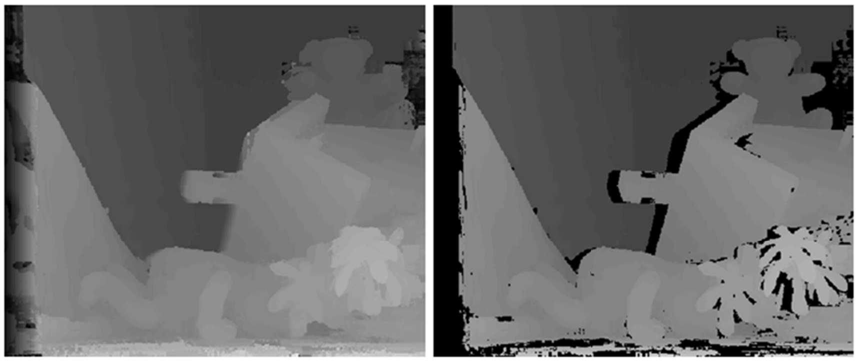


Fig.6.1.Best Possible Map through checking neighboring disparities and the Correct Obtained Pixels

The shown disparity map is of high accuracy except at the right boundaries, where it is not possible to check the whole disparity range as shown in the previous figure.

Thus, by finding the erroneous disparity and substituting a disparity from the neighboring disparities in its place according to a certain criteria could lead to an accurate map.

So, this work aims at finding erroneous disparities by classifying whether the disparity value is acceptable or not and afterwards correctly replacing them through a certain filling stage.

And although the features are not yet detailed, Figure 6.2 shows the obtained map through SAD and after performing classification through AdaboostM1, where the black pixels are set to unknown. The map shows that the errors, especially due to occlusion, are greatly reduced.

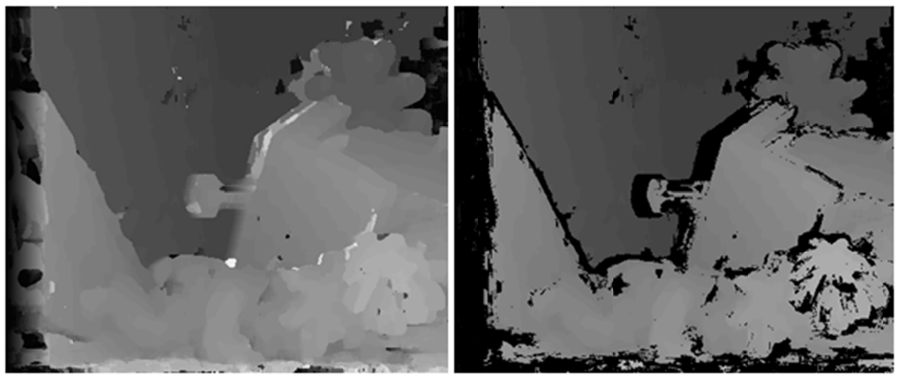


Fig.6.2. Initial Disparity Map and Obtained Disparity Map

It is important to mention that the proposed classification relies on the confident disparity map previously discussed in Chapter 4. Also, this is not the only thinkable classification approach since there are other possibilities, and some of these are discussed later.

6.2 Proposed Features

Perhaps the most important block of building a classifier is selecting the features that well suit the subject of study, since this determines the accuracy of the system. So, this section presents the suggested features.

The main initially available data are the stereo images and these are used to compute a preliminary disparity map using any of the matching methods handled here. Thus, the data becomes the image (I) or the left one (L) in the Middlebury images, with the initial corresponding disparity map (D).

In any case, there are many possible features, since it is possible to use the data in many ways. The proposed features were chosen to be simple and fast to compute. Also, the selected features try to account for local and global effects. The main features used in this work are eleven and they are listed next with additional discussions later.

Moreover, the selected set of features was not always the same for each stereo matching algorithm to ensure better results. The selected set for each matching method is shown in the results section.

Besides the eleven features, an additional twelfth feature is presented based on segmentation information. But this additional feature is not always accounted for as part of the total set since it requires much additional computations.

6.2.1 Feature 1: Mean Intensity Difference

The first feature is selected out of the image itself. The idea is to account for the discrepancies between the pixel *(x,y)* and its surroundings. This feature finds the difference of the intensity value of the pixel with all the pixels in a square neighborhood with radius *Rad* around it. This difference constitutes a small matrix that one can select different measures including the mean, median, standard deviation and others. The mean was selected for ease of computation.

The mean being

$$f_1 = \frac{1}{n} \sum_{i=x-Rad}^{x+Rad} \sum_{j=y-Rad}^{y+Rad} \left| I(i,j) - I(x,y) \right| \tag{6.1}$$

6.2.2 Feature 2: Mean Edge Disparity Value

The second feature is depicted from the disparity map. This is not direct however, since it is based on finding the edge disparity map E_D according to the Al-Alaoui edge detection filter [59]. An example of the edge map for the NCC obtained map is shown in Figure 6.3.

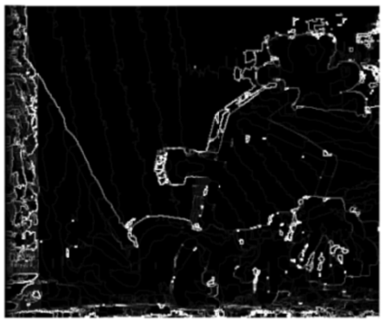


Fig.6.3. Example of the Edge Image of the Disparity Map

Afterwards, for every pixel, the average edge disparity value is found. This is important since the edge map indicates the regions where the problems arise and the areas with the highest concentration of errors especially due to occlusion.

$$f_2 = \frac{1}{n} \sum_{i=x-Rad}^{x+Rad} \sum_{j=y-Rad}^{y+Rad} E_D(i,j) \tag{6.2}$$

6.2.3 Features 3 and 4: Similar Disparities and Mean Disparity Difference

It is highly likely that if most of the surrounding pixels have a different disparity from the current one, the disparity is incorrect. So, the disparity of the current pixel is checked with its neighbors. At first, the number of pixels that have a dissimilar disparity in the current neighborhood is computed. In the kernel, only pixels that are close in intensity (difference in intensity $\leq i_D$) to the current pixel are considered. The radius here is set to *Rad*2* instead of *Rad* since that led to better results.

The idea is that if there are many pixels that are similar in intensity but different in disparity, the associated disparity value is probably wrong, which is somewhat analogous to segmentation, since neighboring similar intensity value pixels usually belong to same segments.

$$f_3 = length\left(\sum_{i=x-Rad*2}^{x+Rad*2} \sum_{j=y-Rad*2}^{y+Rad*2} \left|D(i,j) - D(x,y)\right| > T_D\right) \quad \text{Condition: } \left|I(i,j) - I(x,y)\right| \le i_D \tag{6.3}$$

This work assumes a disparity similar to another if the absolute difference is $\le T_D$. T_D was already discussed and i_D is set to 15.

Moreover, the mean difference between the disparity value and the surrounding disparities is computed which is the fourth feature.

$$f_4 = \frac{1}{n} \sum_{i=x-Rad*2}^{x+Rad*2} \sum_{j=y-Rad*2}^{y+Rad*2} \left|D(i,j) - D(x,y)\right| \tag{6.4}$$

6.2.4 Feature 5: Disparity Value

This feature is basically the disparity value $f_5 = D(x,y)$. This was used because it was noticeable that the lower disparities were more often true than the higher ones, especially in the SAD case. Also, this included no computational cost.

6.2.5 Feature 6: Candidate Accurate Disparity

The main idea here is to check if the confident disparity value D_A(x,y) is available or not. *As already noted, D_A* is very important and is used in several of the features since it is easy to compute and has proven to enhance the accuracy. The first one of these features is feature six which is a logical value that indicates whether the value exists or not.

$$f_6 = 1 \; if \; D_A(x,y) \; exists; 0 \; otherwise \tag{6.5}$$

6.2.6 Feature 7: Histogram Disparity Value Difference

Many of the features here combine the intensity and the disparity value and this feature is one of these. This is based on D_A in addition to the intensity of the pixels. The idea is to determine a histogram that indicates the average of each gray scale intensity value *(gi)*. This is straightforward because *gi* varies between 0 and 255 for 8 bit quantization and between 0 and 63 for 6 bits per pixel. So if one checks the intensity value of each known disparity pixel, the average disparity is found.

$$H_{DA}(gi) = \frac{1}{N = \#\text{ of pixels with } gi} \sum_{i=1}^{N} D_A(I = gi) \qquad (6.6)$$

And after obtaining the histogram *(H_{DA})*, the intensity of the pixel to be classified is checked to find the equivalent disparity in the histogram. This disparity is then subtracted from the candidate disparity which is feature seven.

$$f_7 = \left| H_{DA}(I(x,y)) - D(x,y) \right| \qquad (6.7)$$

6.2.7 Features 8 and 9: Regional Histogram Disparity Value Differences

These two features are rather similar and that is why they are grouped together. Like the previous feature, these rely on (*D*), *(D_A)* and *(I)*. The ideas of the two features are the same, but each one is depicted from a different map, the first is from (D_A) and the second is from (*D*). So, feature eight is found by dividing (D_A) into four equal parts like a two by two array and assigning for each part a histogram similar to that in feature seven. Thus, there would be four histograms indicating the average disparity for each intensity value (*$H1_{DA}$*), (*$H2_{DA}$*), (*$H3_{DA}$*), and (*$H4_{DA}$*).

Now, and since any pixel would lie in either part, its feature is extracted from the histogram of the part it belongs to, in a similar manner to feature seven. This means that the difference between the average and the candidate disparity is computed.

$$f_8 = \left| Hi_{DA}(I(x,y)) - D(x,y) \right| \tag{6.8}$$

The same applies for feature nine which is similar to feature eight but depicted from the regular disparity map (D) which leads to have ($H1_D$), ($H2_D$), ($H3_D$), and ($H4_D$).

$$f_9 = \left| Hi_D(I(x,y)) - D(x,y) \right| \tag{6.9}$$

6.2.8 Features 10 and 11: Local Histogram Disparity Value Differences

In a similar manner like features eight and nine, features ten and eleven are from (D_A) and (D) respectively. Also, these are obtained by dividing the maps into equal parts but instead of four, there are nine parts like a three by three array to lead to ($h1_{DA}$),..., ($h9_{DA}$) and ($h1_D$),..., ($h9_D$) . This means that there are nine histograms and each pixel is assigned to the histogram in the part it belongs to. Then, the absolute difference between the disparity and the corresponding histogram value is computed as in the previous features.

$$f_{10} = \left| hi_{DA}(I(x,y)) - D(x,y) \right| \tag{6.10}$$

$$f_{11} = \left| hi_D(I(x,y)) - D(x,y) \right| \tag{6.11}$$

6.2.9 Feature 12: Segmentation Feature

This feature is only used with segmentation. So, the features are used without it for all the methods and afterwards, the features are tested with and without it in the case involving segmentation. The feature basically computes the absolute difference between the pixel's disparity and the median or mean disparity value of each segment.

$$f_{12} = \left| D(x,y) - \frac{1}{\#\, pixels\ in\ Segment(I(x,y))} \sum_{pixel(i,j) \in Segment(I(x,y))} D(i,j) \right| \quad (6.12)$$

6.3 Completion Stage

As already mentioned when discussing the method used to relate classification and stereo matching, some disparities are left unknown, and so they must be determined. This means that the completion stage decides the overall accuracy and speed of the system. It is worth emphasizing that any stereo matching algorithm that performs LRC requires a similar stage.

Before explaining the proposed method, it is important to note that there are many references as in [50] that initially rely on finding the maps for a set of points. In this work, the initially found points constitute a large percentage and were always more than 70% of all the pixels. Also, if the initial method was accurate, more than 90% of the pixels would be known. This means that the unknown pixels are rapidly found.

Definitely, there are several works that have a completion or filling stage and many of these rely on choosing nearby disparities and weighted median filtering as in [152, 153, 154]. In this work, a new heuristic approach is proposed to achieve acceptable accuracy with fast performance. This proposed stage can be considered as a minor contribution of this work. This is done by combining two techniques.

Before describing the techniques, it is important to note that the lower accuracy methods had very erroneous data before 60% of the disparity range. So, these pixels were set to unknown.

The first technique is comprised of few steps and can be thought as a simple segmentation approach. At first, the gray image (*I*) is quantized into 10 values

(255/25.5), or the RGB image into 7 values in each scale (255/36) to yield new image (*Iq*). Then, each connected empty disparity region is treated as a single region. An empty region is a region where the disparities are currently unknown since they were found erroneous.

The region dimensions are limited by min((# of rows in Image)/2, (# of columns in Image)/2). So, if a dimension bypasses the limit, the region is subdivided.

Then, we check each empty region. In the region, we consider the *Iq* values with unknown disparities and assign them with the median disparity of similar known *Iq* values in the vicinity of the region. The vicinity of the region is every known disparity pixel that neighbors the region as in Figure 6.4. The obtained map is termed (D1).

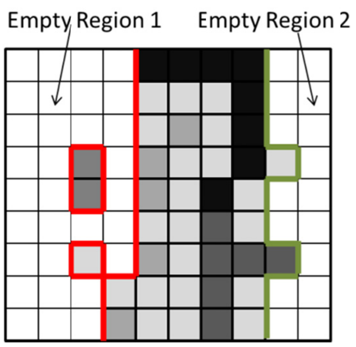


Fig.6.4. Example of empty regions where each (gray level) pixel with a colored border being in the vicinity of the empty (white) region

The second technique is to use the basic approach in [50, 51, 52], termed (*Cb*), where each unknown disparity is filled with the nearest spatially known disparity along the same row to yield the map (D_2).

These two techniques are then combined meaning that if the two disparity values have an absolute difference $\leq T_D$ as in feature 3, they are averaged, while otherwise the value is set to unknown again.

This validation approach leaves a set of unknown pixels which are now filled by the first process unless no similar *Iq* known value exists. These unknown *Iq* values are filled with (*Cb*) to yield the map (D_3).

An idea that enhanced the results is to special treat the pixels that lie before 60% of the disparity range. These disparities are set to the maximum value between that of (D_3) and (D_2) leading to the final map.

Figure 6.5 shows an example of the obtained maps after the filling stage for the Teddy SAD obtained map through AdaboostM1.

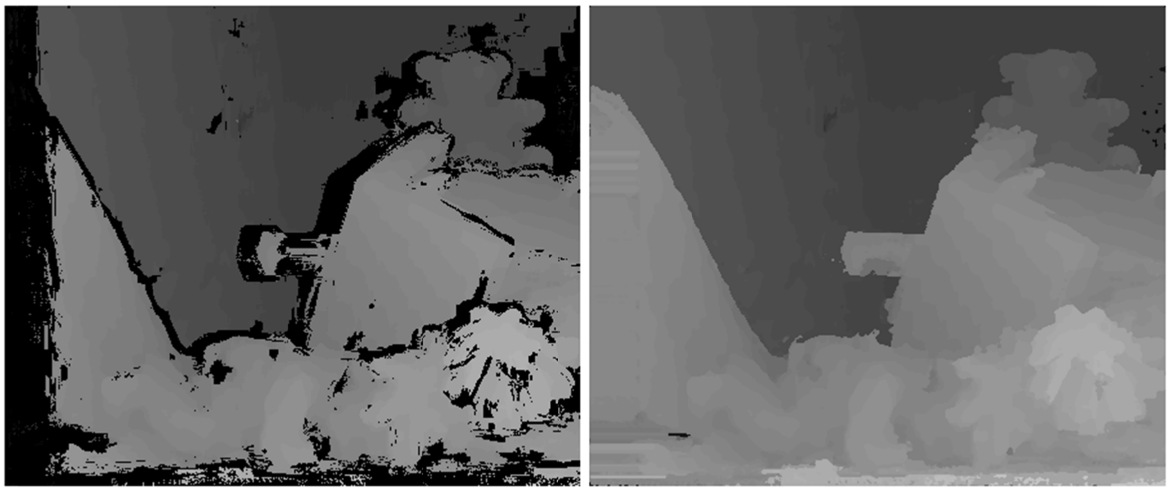


Fig.6.5. Obtained Disparity Map through SAD+AdaboostM1+Completion

It is important to mention that after the completion stage, median filtering is performed since it reduces the noise and improves the results.

A very vital notion here is that the completion stage could not have been any useful if the classification stage did not well identify the erroneous pixels.

CHAPTER 7

3D RECONSTRUCTION

3D Reconstruction is considered as a domain on its own. The work here does not emphasize this part, since the research is not concentrated on this idea. We do not aim to show this part as a contribution but rather as a typical implementation of stereo matching. Still, we present some basic ideas of the used 3D Reconstruction. It is worth noting that the main results shown here are used for the figure shown next and depicted from [155].

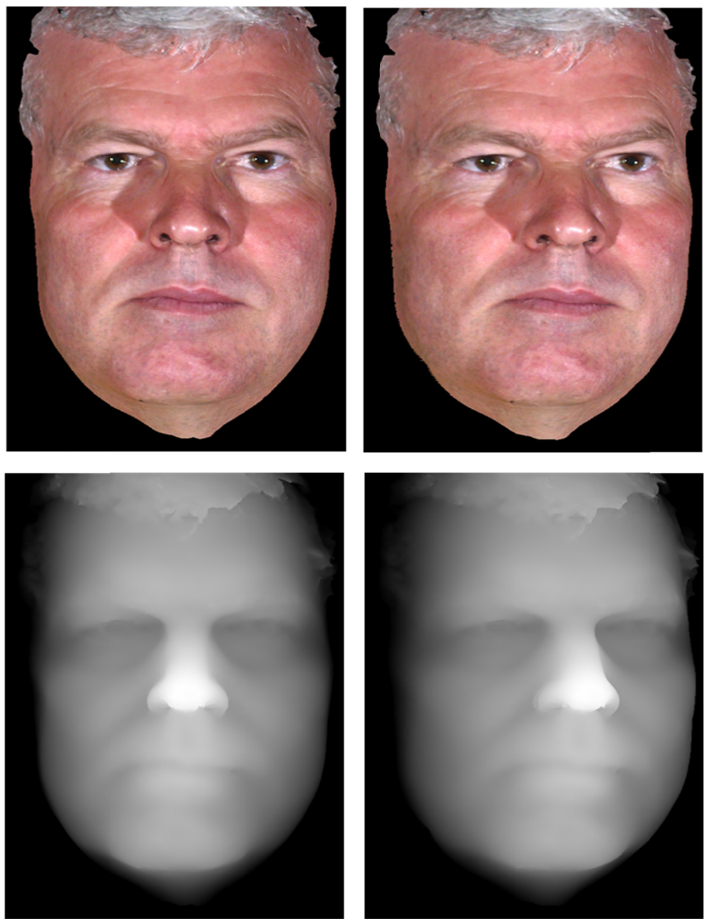


Fig.7.1. Stereo Pair from [155] with the corresponding disparity maps

7.1 Proposed Method

The approach relies on direct usage of the disparity map. The basic idea is that after obtaining the corresponding left and right disparity maps, a 3D reconstruction of the scene should be possible. This is done through using the 3D coordinates of each pixel which are simply x or column position, y or row position, and z or the disparity value according to the preset range. Of course, it is possible to use the different camera parameters in order to obtain the real world coordinates, but this is not necessary here. So, the notion basically maps the pixels to their 3D locations in the virtual environment.

The software used here is not DirectX which leaves another option. We use OpenGL, where the mapping is directly done after computing the disparity into the Graphics Buffer to show the 3D obtained view. The approach here is also straightforward since no lighting, or any other complex graphics procedures are used. It is worth noting that it is possible to use some libraries for the reconstruction like Points Cloud Library (PCL), but they rely on using OpenGL so using OpenGL is the faster way especially when considering the interoperation between CUDA and OpenGL.

An example of the obtained 3D scene is shown in the next figure and it is directly possible to check the different parts of the scene in its 3D environment.

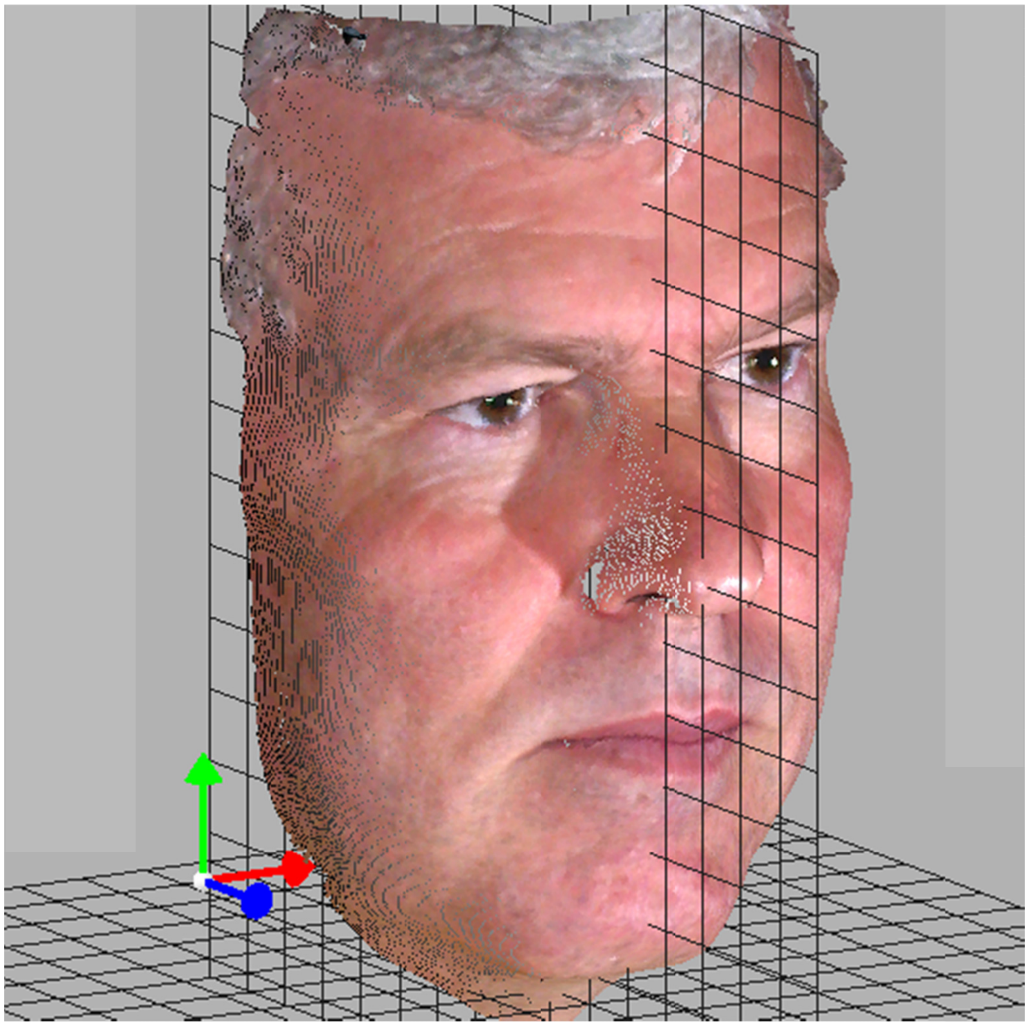


Fig.7.2. Point 3D Reconstructed Scene

These figures directly map each pixel into the (x,y,z) position as a point or quad or sphere with a color as the pixel's color. And since the map shows 3D points which can be thought of as small spheres, it is clear that there are some missing points and the figure seems discontinued which is a problem here.

So, the solution to the missing points' problem is to use interpolation between the pixels by considering each pixel as a rectangle or four sided quad that is connected with the neighboring pixels at its different sides. This can be easily explained according to Figure 7.3.

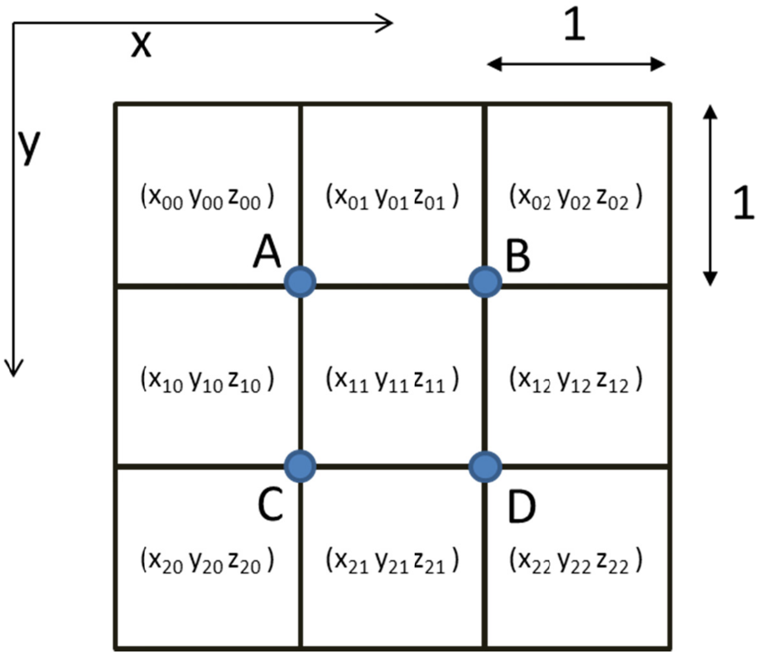

x
y
1
1
$(x_{00}\ y_{00}\ z_{00})$
$(x_{01}\ y_{01}\ z_{01})$
$(x_{02}\ y_{02}\ z_{02})$
A
B
$(x_{10}\ y_{10}\ z_{10})$
$(x_{11}\ y_{11}\ z_{11})$
$(x_{12}\ y_{12}\ z_{12})$
C
D
$(x_{20}\ y_{20}\ z_{20})$
$(x_{21}\ y_{21}\ z_{21})$
$(x_{22}\ y_{22}\ z_{22})$

Fig.7.3. Basic Proposed Points Interpolation

In Figure, the idea is that the quad is determined by four points and each one can be found through interpolating the value of its neighbors. So, as an example, for point A:

$$(x_A, y_A, z_A) = \left(x_{11} - 0.5, y_{11} - 0.5, \frac{z_{00} + z_{01} + z_{10} + z_{11}}{4} \right) \tag{7.1}$$

And the same goes for the remaining points

$$(x_B, y_B, z_B) = \left(x_{11} + 0.5, y_{11} - 0.5, \frac{z_{02} + z_{01} + z_{12} + z_{11}}{4} \right) \tag{7.2}$$

$$(x_C, y_C, z_C) = \left(x_{11} - 0.5, y_{11} + 0.5, \frac{z_{10} + z_{11} + z_{20} + z_{21}}{4} \right) \tag{7.3}$$

$$(x_D, y_D, z_D) = \left(x_{11} + 0.5, y_{11} + 0.5, \frac{z_{11} + z_{12} + z_{21} + z_{22}}{4} \right) \tag{7.4}$$

An example of the obtained 3D scene through this method is shown next.

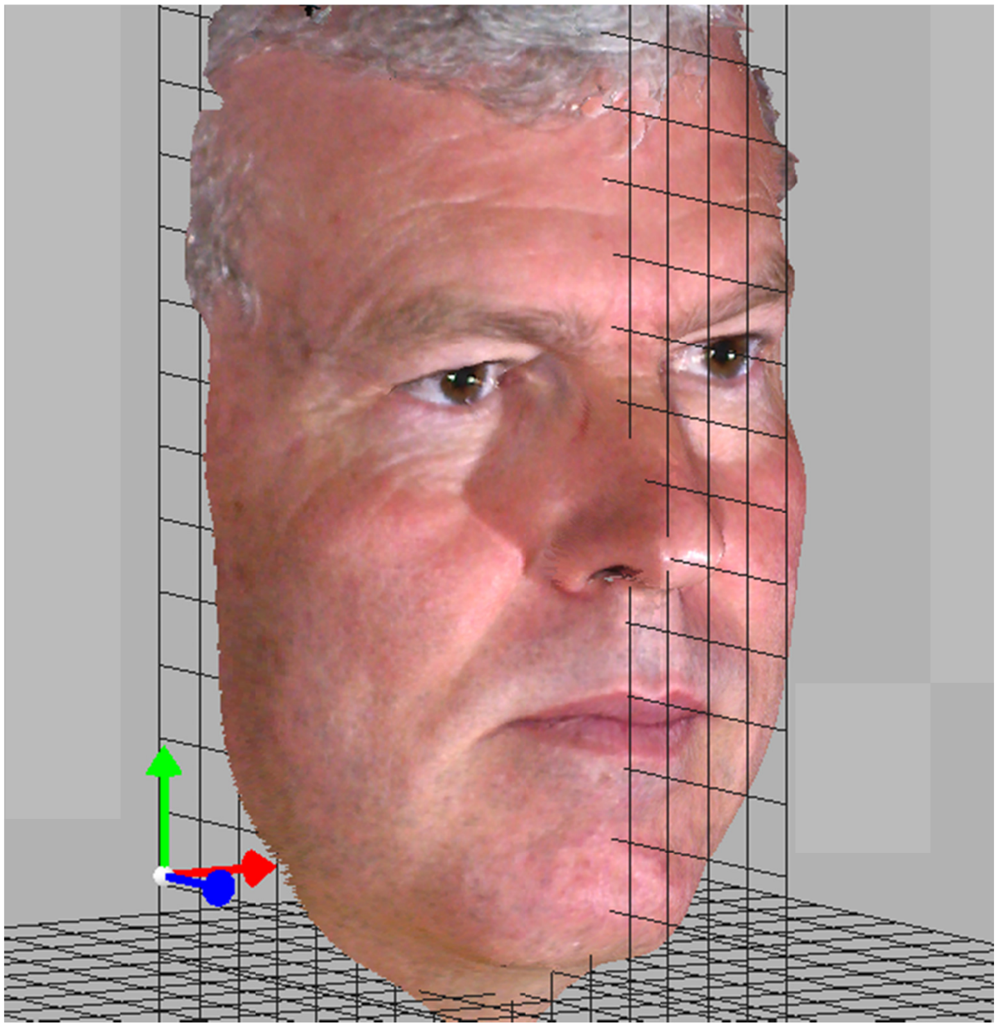


Fig.7.4. Interpolated 3D Reconstructed Scene

A worthy idea here is that this work may be argued for being a 3D mapping rather than 3D reconstruction, since the scenes still lack some missing sections and that is rather true, but in any case there are limits to what the stereo images or the eyes see and the whole scene is impossible to reconstruct unless it is rather flat or simple or according to a known pattern.

So, and in order to present a better idea of 3D reconstruction, we generate a set of images that tend to complete the scene in a set of 2D images rather than generate the whole 3D model. This can be achieved by using the left and right disparity maps and generating a set of maps that start from the left map and end up at the right map or in the opposite direction. This is better explained through an example.

At first, consider that we have the stereo images with the stereo disparity maps, which is a total of four images as already shown in the first figure.

Then, and in order to start from one figure and end at the other, one must generate an intermediate set of images that slowly move from one image to the other pair whether that is for the stereo or disparity pair. So and using the map, each pixel is gradually shifted from one image into its pair. This can be easily seen by checking the difference distance between the pair, which corresponds to the disparity and then dividing this distance into a set of gradual shifts. For example, if the distance of a certain pixel is 30, and the number of shifts for the images is set to 10, on each step, the pixel is shifted by 3, and when each step is performed to the whole image, a new intermediary image is obtained which corresponds to the scene at that position. There are some problems however, and these mainly include unknown pixels, since each shift leaves a set of pixel values that are not known or blank, and these need to be found to have a complete image. So, interpolation can be used to find these pixels, or choosing the nearest lower disparity pixel and duplicating it, or even duplicating the higher disparity pixel. This is mainly done by using the nearest neighbors and it should work in a very good manner when the image is not crowded or smooth. Figure 7.5 shows an example of the set of disparities obtained by assuming ten shifts between the pair on the disparity map.

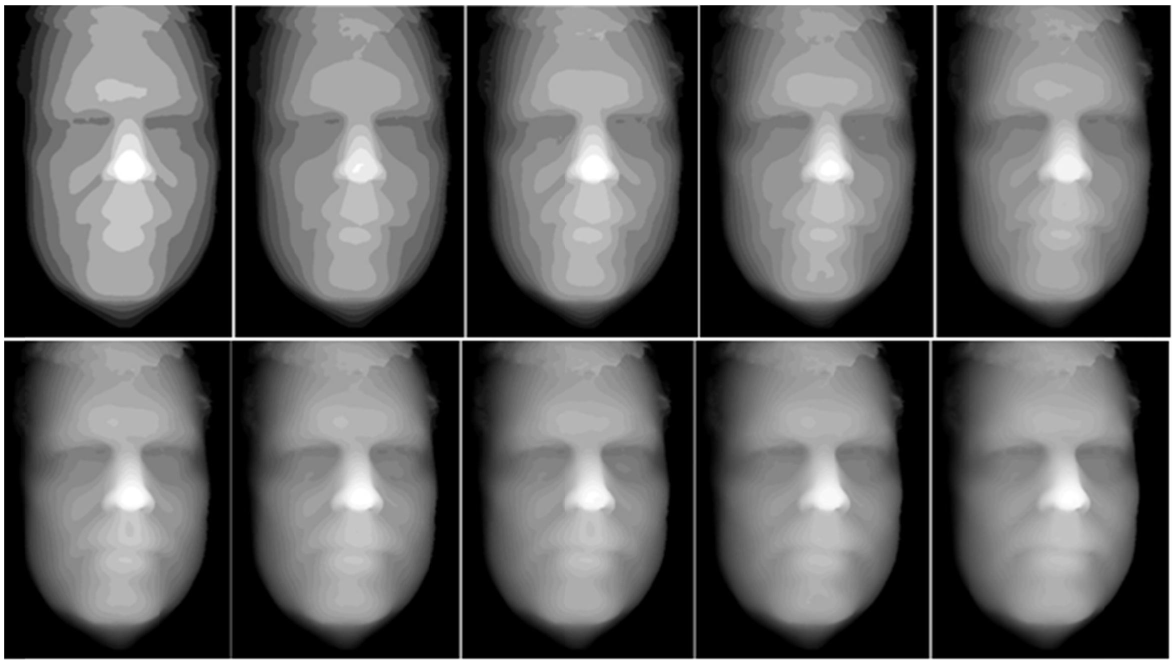


Fig.7.5. Interpolated Disparity Maps

Figure 7.6 shows the application on the regular images which are obtained after performing interpolation and using the obtained disparity maps.

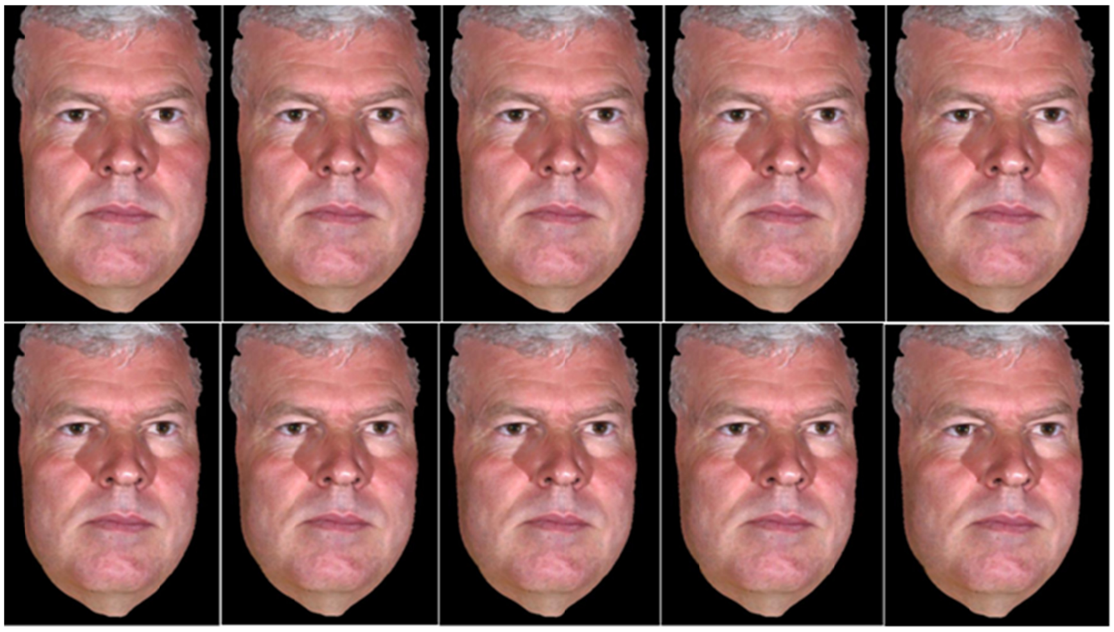


Fig.7.6. Reconstructed Images through Interpolated Disparity Maps

It is worth noting here that combining these different images or sequence in a video or even a gif image leads to obtaining 3D like effects. Also, these figures can be obtained through any coding software with image manipulation like Matlab, C, C++ with its different libraries like OpenCV and OpenGL.

CHAPTER 8

CUDA IMPLEMENTATIONS

This chapter considers CUDA with many of the ideas addressed in this work. These include edge detection and stereo matching with or without classification. It is important to emphasize that the CUDA related work does not cover all the work and can be further updated, so this chapter might seem to miss some ideas.

8.1 Notion about CUDA

The CUDA architecture was introduced by NVIDIA in order to bring general purpose programming to the Graphics processor (GPU) thus enabling the use of a highly parallel easy to program architecture suitable for image processing algorithms such as image edge detection. The CUDA platform is chosen in this work due to its popularity, availability, ease of programming and highly parallel design. CUDA is becoming more and more the primary non CPU parallel programming platform due to these reasons.

Moreover, and since this work aims at achieving real time performance, CUDA was chosen especially that basic CPU implementations were rather slow whether in Matlab, C or C++.

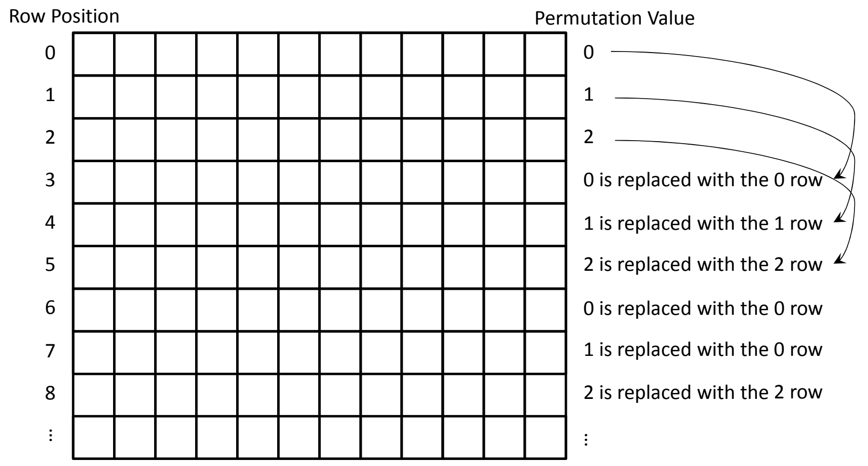
It is worth mentioning that CUDA devices have several types of memory that include global, texture, shared and local memory. The method used to deal with these memories in regards to organizing the accesses and using the faster ones often leads to an enhanced algorithm for any implementation.

8.2 Edge Detection using CUDA

Edge detection related to this work relied on the notion of masks as already explained and the operation used to apply the masks is convolution of a 3x3 kernel. So, for the purpose of obtaining a parallel and faster implementation for the edge detection method, and due to the increasing popularity of parallel hardware whether GPUs or FPGAs, we targeted a CUDA approach since it is rather easy and efficient for GPUs. Thus, and to suite the method to CUDA, a faster algorithm was proposed. The approach here is general and aims at performing the mask operation which is a main step in most applications related to image processing in this work or in others. It is important to note here that the main advantages of using CUDA lie in the data handling in memory which is noted here.

In order to explain this well, it is important to emphasize that the filtering stage is a convolution or correlation one, where an image is convolved with a mask like Sobel, Prewitt, or difference of Gaussian, etc... The adopted algorithm relies on exploiting the shared memory of the CUDA GPU device which is much faster than global memory. And before directly explaining the proposed implementation, it is necessary to explain the shared memory implementation of Nvidia [129]. The basic idea of this is to divide the input image into a set of blocks that are possible to store in shared memory and access these blocks as necessary for every pixel. The full explanation of the algorithm is provided in [156] in which separable filters and memory coalescence are implemented.

Concerning the proposed method, the idea is to store the required values until no longer needed and reduce the accesses to global memory. Thus, and since the convolution of each pixel considers a neighborhood or kernel of pixels around it, which are in turn to be computed, the main reduction would be to send each pixel value once. This is explained to work for an *(n x m)* kernel where *n* is for rows and *m* is for columns with both of them being odd, and in particular for the most common *(3 x 3)* mask. Thus, we start by sending the initial *n* rows to shared memory and since after finding the values for the *(n/2+1/2)th* row the first row is no longer necessary, it is replaced with the *(n+1)th* row. And in each step, we replace the required row with the one no longer needed. An important problem here is that shifting all the rows would require many operations, so we consider permutation variables to know the order of the rows. In Figure 6, the *(3 x 3)* mask is considered for an input image. Initially, rows 0, 1 and 2 are sent to the shared memory, so we have shared rows 0, 1 and 2. After computing the resulting pixel values for row 1, the values saved in shared row 0 are replaced with the values of row 3. Then, the pixels of row 2 are computed noting the order or permutation of the rows since the shared row order is 1, 2, 0. Then the values saved in shared row 1 are no longer needed, so they are replaced with those in row 4 and so on...



Row Position
Permutation Value
0
1
2
3
4
5
6
7
8
⋮
0
1
2
0 is replaced with the 0 row
1 is replaced with the 1 row
2 is replaced with the 2 row
0 is replaced with the 0 row
1 is replaced with the 0 row
2 is replaced with the 2 row
⋮

Fig.8.1. Explanation of the Algorithm Used for Filtering Images in CUDA for a (3 x 3) mask

Using the proposed approach, the access to global memory is reduced according to the size of the filter with higher sizes leading to better performance.

The code for the (3 x 3) kernel implementation is available for checking and download at [157]. This code can be used in a similar manner as the Sobel Filter source code implementation provided by NVIDIA in the GPU SDK [129].

A necessary note here is that not all the row is used since that could exceed the allowed shared memory and slow the procedure and not all the rows are considered in a single pass, but are rather divided into different sections. Different tests were done to achieve the typical values mentioned next. It is worth mentioning that it is further possible to use the columns instead of the rows, and the initial approach of moving along the columns is rather similar to the stereo matching algorithm in [158].

Besides the mentioned ideas, it is important to mention that memory coalescence, in a similar manner to [156] is used to enhance the performance.

8.3 Stereo Matching using CUDA

This work concentrates on Stereo Matching and in particular using local stereo matching methods such as SAD, SSD and NCC. In particular, SAD and SSD are faster to compute than NCC, although the approaches are rather similar since they both rely on accounting for differences between local neighborhoods in the stereo pair.

Concerning CUDA stereo matching based implementations and as already discussed there are plenty, and in particular for the SAD and SSD, a parallel algorithm stands up and seems to be the fastest. This was done and explained in [158]. The main approach relies on exploiting the repeated calculations in a fast manner to reduce memory transfers and calculations. For a radius (*Rad=5*), the proposed algorithm in [158] consumes about 4 ms on a Geforce GTX 460 GPU for the Teddy image.

By testing the different possible approaches which included checking the rows or different blocks of rows and columns, the proposed implementation which is based on the columns was indeed found to be the fastest one due to the nature of the problem and the disparity that indicates the difference between the pixels in terms of the number of columns. Still there seemed to be a room for improvement by further reducing global memory accesses and further exploiting the local memory available. So, this was done here and it led to an improvement of more than 20% for the different tested images.

The modifications mainly included local storage of the intermediary distances since the SAD or SSD approaches aimed at minimizing these distances. Another idea is storing repeated local values instead of recalculating them more than once on every thread. Moreover, storing the current disparity value of the pixel locally enhanced the results since the repeated access of this in global memory slowed down the implementation.

It is important to mention here that, perhaps, the main reason that these modifications were not considered in the original implementation was that local memory was more limited than that of current GPUs, so this work mainly exploited some of the properties of updated GPUs.

In our implementation we used the texture memory of the CUDA device to store the image and its map. Also, we used the shared memory for all the features related to classification, since this is essential for reducing the accesses to global or texture memory to obtain real time speeds.

In regards to the initial processing, this is relatively fast since the histograms are easy to compute using atomic operations in CUDA [159].

Regarding classification, training is time consuming since it tries to determine the parameters of the classification method, with each approach using a different training operation. But, this is only performed once and prior to the implementation of the stereo matching system. Also, the training does not need a GPU implementation.

Concerning the testing or implementation stage of classification, this is required for every pixel which is computationally acceptable as explained next.

The vital idea here is that the proposed classification is independent of the disparity range since it is a post processing stage. This means that classification consumes the same time on same sized images.

Concerning the features, and although there are eleven, excluding the segmentation feature, and they require little time. Features 1 and 2 are direct and can be computed using separable convolution [156]. The work in [156] reported speeds of 1.4 (Giga pixels per second) GPPS for large arrays. Also, Feature 5 is already available.

For feature 6, the map (DA) requires checking four values per pixel which is straightforward. Concerning features 7, 8, 9, 10 and 11, the histograms can be computed with a speed of 5.5 GPPS [159] for histograms of 256 bins and 10 GPPS for histograms of 64 bins for large arrays.

The main time consuming features are 3 and 4 since they are performed on bigger neighborhoods as in a (21x21) kernel as opposed to an (11x11) neighborhood.

The required time can be checked by finding the number of operations done per pixel. So, if we study the exhaustive algorithm case of the SAD computation, each pixel would require $dr \times (Rad \times 2 + 1)^2$ set of operations. dr is the disparity range and Rad is the Radius of the kernel. The operations include additions and conditions. This is compared with the hardest to compute features, features 3 and 4. These would require about $2\times((Rad\times 2)\times 2+1)^2$ different operations where the $2\times$ is for the map and the image, and the first $\times 2$ is for the larger radius. So, if $(Rad=5\ \&\ dr=60)$, the SAD algorithm would require almost eight times the computation of features 3 and 4 which are found together. In fact, the computation of these two features is comparable to a SAD disparity map computation performed at a larger radius $Rad\times 2$ but at a small disparity range $dr\leq 3$.

After finding the features, classification is performed. This is fast especially according to boosting which requires checking a set of conditions according to the tree classifier and determining the class of each pixel. It is worth mentioning here that the total time required for the features and AdaboostM1 is about 40% of the SAD map computational time in the Teddy and Cones images for an (11x11) kernel.

After the classification, the completion of the unknown pixels mainly requires checking the neighboring pixels and the possible known disparities. The time consumed here is about 60% of the SAD map required time since this is performed only for a set of pixels that are grouped together. The completion stage mainly relies on labeling connected components which is fast on both the CPU and the GPU [57]. This can be further developed in future work.

After the filling stage, median filtering is used and this is also fast since it is performed on a small neighborhood (Rad=1).

It is important to note here that the usage of the basic completion stage (Cb) directly means less total time, albeit at the expense of reduced accuracy. (Cb) consumes about 0.2 ms. Additional results are presented in the next chapter.

It is important to remark that the completion stage requires the most time in the SAD case since it was the most erroneous, so using other matching methods should be faster.

More details on the CUDA implementation are provided in the next chapter.

CHAPTER 9

RESULTS

This chapter presents the results regarding the main proposed work. It is divided into several parts to accommodate the different parts of this work. At first, the results regarding the proposed classification approach, excluding stereo matching, are presented. Afterwards, we present the results regarding stereo matching. The results include results with and without both segmentation and classification. Moreover, and since parts of the stereo matching proposed work were developed on CUDA, we present the related time results. Definitely, it is important to discuss the various obtained results, so various ideas are discussed throughout this chapter.

9.1 Results on Classification

The proposed work has indeed led to better results in many cases, and in this section these claims are validated. Various datasets are considered with different number of features. All the datasets are binary ones although similar results were obtained using AdaboostM2 for multiclass datasets. The different datasets are selected from various references.

And since the different methods of boosting are tested, it is important to well divide them for showing the results, so each table indicates the different boosting method that was used amongst AdaboostM1, GentleBoost and LogitBoost.

The tables initially show the results of the boosting methods *(% Initial)*. Also, the tables present the results obtained by the two approaches *(% Comb)* and *(% Sep)*, each on its own without the training acceptance check in addition to the final results obtained through the combination of the two approaches along with the training acceptance check *(% New)*. We further show the errors obtained by the RE method which is a particular case of the combining classifiers search. The RE method selects the best performing classifiers' combination on the training data as opposed to selecting a pruning level or percentage.

In addition to the error percentages, the tables indicate the average number of classifiers of the selected set after the acceptance test *(# trees)*, which means that it is possible to have an average equal to the original number of classifiers especially that it is always one of the candidate sets, at least in the separating procedure and due to the acceptance conditions.

The results present the testing amongst 50 learners with further generalizations possible whether less or more. It is important to note that similar tests were done to 100 and 20 learners and they generally led to similar results. An additional idea that is worth highlighting is that a lower number of classifiers, like 20, usually allowed for the proposed selection method to reach better solutions in more databases as compared to the original set and this is mainly due to the smaller search space and the initial relatively low accuracy. On the other hand, if a larger number of classifiers was used, like 100, very similar improvements as in the used case, 50 classifiers, were obtained although the procedures were more time consuming due to the larger search space.

Furthermore, the results are considered for tenfold validation which divided the data into 90% for training and 10% for testing in each of the tenfold cases, and computing the average error with rounding the results. These tests were performed for ten times per each dataset, meaning that each dataset underwent at least 100 runs per the boosting method since the tenfold test includes 10 runs. Of course, other scenarios are possible like twofold, fivefold, etc… It is important to note here that this 90%-10% test is independent of the modified training used for selection since that worked on the 90% used for training and divided it into two parts according to the suggested approach.

It is worth mentioning that it is possible to present the training only errors, but these are not included for brevity.

9.1.1 Datasets

In this part, a brief idea of the different used datasets is provided. These are a total of 20 binary or two class classification datasets. Table 9.1 provides basic information of these databases.

Table 9.1 Datasets Basic Information

Dataset	**# Features**	**# Patterns**
Australian Credit [160]	15	690
Blood Transfusion [161]	5	748
Cancer [161]	9	699
Circular	2	400
Crab [161]	6	200
Flare [161]	14	1389
Fourclass [162]	3	862
German Credit [160]	25	1000
Glass [161]	9	214
Iono [161]	35	351
KC2 [163]	22	522
Magic04 [161]	11	19020
Mammography [161]	6	961
Mushroom [161]	22	8124
Pima [161]	9	768
Skin [161]	4	245057
Spiral [164]	2	900
TicTac [165]	10	958
Titanic [166]	4	2201
Twonorm [167]	21	7400

As it can be seen the different databases are selected from different sources that are all publicly available. These definitely include the UCI database [161] in several cases. Statlog [160] is used in two of the databases in addition to several other resources as in [162, 164, 165, 166, 167].

Concerning the Spiral dataset, it is a 2 class 2 features database where each class looks like a part of the spiral. It can be generated randomly and is based on the work in [164].

Also, the circular dataset can be randomly generated since it is a 2 features 2 class dataset where class 1 consists of samples inside a circle and class 2 consists of samples surrounding Class 1 by a ring with added noise to the different elements. This can be easily extended to multiclass problems.

9.1.2 AdaboostM1 with Selection Results

This provides the results for AdaboostM1. Table 9.2 shows the tenfold results according to the previous discussion. It is important to re-note that the different tables for this method and the others show the error percentages *(%)*, so the lower the better.

Table 9.2 AdaboostM1 with Selection Error Results

Dataset	**% Error Initial**	**% Error Comb**	**% Error Sep**	**% Error RE**	**% Error New**	**% Error SA**	**% Error SA-C**	**# Trees New**	**# Trees SA-C**
Australian	14.2(3.8)	14.8(3.9)	14.4(3.5)	14.9(4.0)	**14.1(3.8)**	14.2(3.9)	14.2(3.9)	49.1	46.2
Blood Transfusion	20.8(5.4)	20.4(5.1)	**20.2(5.0)**	20.4(5.1)	20.5(5.1)	21.0(5.3)	20.9(5.4)	49.7	49.1
Cancer	4.3(2.5)	**4.1(2.2)**	4.5(2.5)	4.2(2.2)	4.2(2.1)	4.3(2.7)	**4.1(2.2)**	49.2	40.7
Circular	14.7(6.0)	14.3(5.1)	**14.0(4.9)**	14.5(5.4)	14.6(6.0)	14.4(6.1)	14.7(6.0)	49.8	47.8
Crab	13.8(9.1)	**12.8(9.0)**	13.4(9.1)	13.0(9.0)	13.3(9.1)	13.7(9.0)	13.8(9.1)	46.5	50
Flare	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	50	48.9
Fourclass	15.6(4.1)	10.3(3.7)	**9.3(3.3)**	10.6(3.6)	10.1(3.6)	10.8(3.0)	11.3(3.1)	44.3	32.3
German	24.9(4.7)	24.4(4.6)	**24.3(4.5)**	24.5(4.4)	24.7(4.5)	24.8(4.9)	24.8(4.7)	48.7	47.6
Glass	4.2(4.2)	**3.7(3.5)**	4.2(3.4)	3.9(3.5)	4.2(3.5)	4.2(4.2)	4.2(4.2)	49.6	50
Iono	7.9(4.5)	8.3(4.7)	8.5(4.8)	8.4(4.7)	**7.8(4.5)**	8.2(4.6)	7.9(4.5)	49.5	50
KC2	14.8(5.3)	15.2(5.2)	15.0(5.2)	15.3(5.2)	**14.7(5.3)**	**14.7(5.0)**	14.9(5.8)	48.3	45.8
Magic	17.0(1.0)	16.3(0.9)	**16.1(0.9)**	16.3(0.9)	16.2(0.9)	16.5(1.2)	16.5(1.1)	41.3	44.6
Mammography	17.3(3.3)	17.5(3.1)	17.6(3.2)	17.6(3.2)	**17.1(3.4)**	17.6(3.0)	17.4(2.9)	47.2	41.3

Table 9.2 Continued

Mushroom	0.3(0.2)	**0.1(0.1)**	**0.1(0.1)**	**0.1(0.1)**	**0.1(0.1)**	**0.1(0.1)**	**0.1(0.1)**	45.4	37.4
Pima	**23.6(4.3)**	24.6(5.2)	24.4(5.0)	24.7(5.2)	**23.6(4.3)**	24.6(4.8)	23.7(4.4)	50	46.6
Skin	8.2(0.1)	5.9(0.1)	**5.3(0.1)**	6.1(0.1)	**5.3(0.1)**	6.1(0.1)	6.1(0.1)	41.5	27.3
Spiral	9.8(3.1)	8.6(2.7)	8.4(2.4)	8.6(2.7)	8.8(2.8)	**8.3(2.8)**	8.7(2.9)	42.1	34.1
TicTac	22.3(4.7)	22.1(4.6)	**19.7(4.3)**	22.4(4.7)	20.3(4.5)	22.5(4.8)	22.3(4.8)	39.7	44.7
Titanic	22.2(2.7)	22.2(2.6)	**21.9(2.7)**	22.2(2.6)	**21.9(2.7)**	22.2(2.7)	22.2(2.7)	47.8	48.4
Twonorm	**4.1(0.6)**	4.3(0.7)	4.4(0.8)	4.4(0.8)	**4.1(0.6)**	**4.1(0.6)**	**4.1(0.6)**	50	49.8
Average Value	***13.3%***	***12.8%***	***12.6%***	***12.9%***	***12.6%***	***12.9%***	***12.9%***	***47.0***	***44.1***
Average Rank	***4.6***	***3.4***	***3.3***	***4.6***	***2.8***	***4.6***	***4.8***		

9.1.3 GentleBoost with Selecting Results

This provides the results for GentleBoost. Table 9.3 shows the related testing results.

Table 9.3 GentleBoost with Selection Error Results

Dataset	**% Error Initial**	**% Error Comb**	**% Error Sep**	**% Error RE**	**% Error New**	**% Error SA**	**% Error SA-C**	**# Trees New**	**# Trees SA-C**
Australian	**14.9(3.7)**	15.6(4.1)	15.5(4.2)	15.6(4.1)	**14.9(3.7)**	15.2(4.0)	**14.9(3.8)**	49.6	48.2
Blood Transfusion	**20.6(4.7)**	21.0(4.6)	20.9(4.6)	21.2(4.8)	**20.6(4.7)**	20.9(4.7)	**20.6(4.7)**	49.7	48.1
Cancer	**3.8(2.1)**	3.9(2.2)	3.9(2.2)	3.9(2.2)	**3.8(2.1)**	3.9(2.2)	**3.8(2.1)**	50	48.5
Circular	13.7(5.7)	13.6(5.7)	13.7(5.6)	13.6(5.7)	13.6(5.7)	**13.5(5.7)**	13.7(5.7)	49.8	47.6
Crab	10.5(7.3)	10.8(7.7)	11.0(7.7)	10.9(7.7)	10.5(7.3)	**10.4(7.3)**	10.5(7.3)	50	50
Flare	5.8(1.9)	5.8(1.9)	**5.7(1.9)**	**5.7(1.9)**	**5.7(1.9)**	**5.7(1.9)**	5.8(1.9)	47.5	45.8
Fourclass	8.1(3.5)	**6.3(3.2)**	6.4(3.4)	6.4(3.3)	6.7(3.3)	6.5(3.1)	6.9(3.0)	42.2	44.7
German	**23.8(4.8)**	24.3(4.9)	24.5(5.0)	24.4(5.0)	**23.8(4.8)**	24.1(4.8)	**23.8(4.8)**	49.9	49.2
Glass	**4.2(4.4)**	4.8(4.9)	4.9(4.7)	5.0(4.9)	**4.2(4.4)**	**4.2(4.4)**	**4.2(4.4)**	50	50
Iono	**7.7(4.0)**	8.0(4.1)	8.3(4.2)	8.1(4.1)	**7.7(4.0)**	7.8(4.0)	**7.7(4.0)**	50	50
KC2	15.8(5.7)	16.4(6.1)	16.2(5.9)	16.8(6.2)	15.8(5.7)	**15.7(5.7)**	15.8(5.7)	50	48.7
Magic	15.9(0.8)	15.5(1.1)	**15.1(0.9)**	15.6(1.0)	15.3(0.9)	15.6(1.0)	15.6(1.0)	45.1	44.4
Mammography	17.9(3.3)	18.1(2.8)	18.2(3.2)	18.1(3.2)	**17.8(3.1)**	18.2(3.5)	18.0(3.3)	48.4	43.2
Mushroom	0.2(0.1)	**0.1(0.1)**	**0.1(0.1)**	**0.1(0.1)**	**0.1(0.1)**	***0.1*(0.1)**	***0.1*(0.1)**	47.5	44.3
Pima	24.5(4.1)	**24.4(4.1)**	**24.4(4.1)**	**24.4(4.1)**	**24.4(4.1)**	24.9(4.3)	24.5(4.2)	49.4	48.9
Skin	4.5(0.2)	3.4(0.1)	**3.0(0.1)**	3.4(0.1)	**3.0(0.1)**	3.4(0.1)	3.4(0.1)	37.7	36.3
Spiral	9.0(3.0)	**8.1(2.8)**	**8.1(2.7)**	8.2(2.7)	8.3(2.8)	**8.1(2.9)**	8.3(2.7)	44.7	38.3
TicTac	21.8(4.7)	20.4(4.8)	**18.8(4.4)**	20.6(4.9)	19.2(4.5)	22.0(4.9)	21.4(4.6)	44.3	45.4
Titanic	22.2(2.7)	22.2(2.7)	22.1(2.7)	22.2(2.7)	22.1(2.7)	**22.0(2.7)**	22.1(2.7)	49.8	41.1
Twonorm	**3.7(0.6)**	3.8(0.7)	**3.7(0.6)**	3.8(0.7)	**3.7(0.6)**	**3.7(0.6)**	**3.7(0.6)**	50	49.9
Average Value	***12.4%***	***12.3%***	***12.2%***	***12.4%***	***12.1%***	***12.3%***	***12.2%***	***48.0***	***46.1***
Average Rank	***4.1***	***4.2***	***4.0***	***5.1***	***2.7***	***3.8***	***4.2***		

9.1.4 LogitBoost with Selecting Results

Table 9.4 shows the different error testing results according to the previous discussion for the LogitBoost method.

Table 9.4 LogitBoost with Selection Error Results

Dataset	% Error Initial	% Error Comb	% Error Sep	% Error RE	% Error New	% Error SA	% Error SA-C	# Trees New	# Trees SA-C
Australian	**13.4(3.4)**	13.8(3.1)	13.5(2.9)	13.7(3.0)	**13.4(3.4)**	13.6(3.2)	**13.4(3.4)**	50	49.4
Blood Transfusion	20.7(5.0)	21.1(5.3)	20.5(5.0)	21.3(5.4)	**20.6(5.0)**	21.2(5.1)	20.7(5.1)	49.6	48.2
Cancer	**3.9(2.4)**	4.2(2.6)	**3.9(2.4)**	4.3(2.6)	**3.9(2.4)**	4.2(2.5)	4.0(2.4)	49.5	46.9
Circular	13.9(5.7)	14.0(5.9)	**13.6(5.4)**	14.1(5.6)	**13.6(5.3)**	14.1(6.0)	14.0(5.8)	49.3	46.5
Crab	15.1(7.2)	**13.9(6.8)**	14.0(6.5)	**13.9(6.7)**	14.9(6.9)	14.0(6.4)	14.5(6.8)	49.8	47.8
Flare	**5.7(2.0)**	**5.7(2.0)**	5.8(2.0)	5.8(2.0)	**5.7(2.0)**	5.8(2.0)	**5.7(2.0)**	50	48.1
Fourclass	7.0(2.9)	4.8(2.3)	**4.5(2.2)**	5.1(2.4)	4.6(2.2)	4.9(2.1)	4.9(2.1)	46.6	35.5
German	24.4(4.4)	**24.0(3.2)**	25.0(5.0)	**24.0(3.1)**	24.4(3.5)	24.8(4.6)	24.5(4.5)	50	45.7
Glass	**4.5(4.4)**	**4.5(4.4)**	**4.5(4.4)**	**4.5(4.4)**	**4.5(4.4)**	**4.5(4.4)**	**4.5(4.4)**	50	50
Iono	**6.8(4.3)**	7.0(4.3)	7.3(4.4)	7.0(4.3)	**6.8(4.3)**	7.1(4.5)	**6.8(4.3)**	50	50
KC2	15.3(5.3)	16.2(5.4)	16.1(5.6)	16.4(5.6)	**15.2(5.4)**	15.4(5.4)	15.4(5.3)	49.7	48.9
Magic	15.4(0.9)	15.2(0.8)	15.1(0.8)	15.2(0.8)	**15.0(0.9)**	15.3(0.9)	15.3(0.8)	44.8	47.3
Mammography	**17.0(3.0)**	17.6(2.9)	17.4(2.9)	17.7(3.1)	**17.0(3.0)**	18.2(3.4)	17.4(3.2)	50	44.1
Mushroom	0.9(0.3)	0.3(0.1)	**0.1(0.1)**	0.3(0.1)	**0.1(0.1)**	0.2(0.2)	0.2(0.2)	44.1	26.9
Pima	24.2(3.8)	24.2(3.9)	24.3(4.1)	24.3(4.0)	**24.1(3.8)**	24.3(3.8)	24.2(3.7)	47.7	45.7
Skin	3.6(0.1)	2.9(0.1)	2.4(0.1)	2.9(0.1)	**2.3(0.1)**	2.4(0.1)	2.4(0.1)	34.1	35.6
Spiral	10.1(3.1)	**8.4(2.1)**	9.1(2.2)	8.6(2.1)	8.9(2.5)	8.6(2.0)	9.3(2.7)	46.3	37.3
TicTac	23.4(5.0)	22.1(4.4)	**21.5(4.6)**	22.2(4.5)	22.0(4.5)	23.6(5.1)	23.2(5.3)	41.2	40.9
Titanic	22.2(2.7)	21.2(2.4)	**21.1(2.4)**	21.4(2.3)	21.6(2.4)	21.2(2.4)	21.3(2.5)	47.6	32.3
Twonorm	**3.7(0.6)**	**3.7(0.6)**	**3.7(0.6)**	**3.7(0.6)**	**3.6(0.6)**	**3.7(0.6)**	**3.7(0.6)**	49.8	49.8
Average Value	***12.6%***	***12.2%***	***12.2%***	***12.3%***	***12.1%***	***12.4%***	***12.3%***	***47.9***	***43.8***
Average Rank	***4.2***	***3.7***	***3.3***	***5.1***	***2.4***	***5.1***	***4.3***		

9.1.5 Bagging with Selecting Results

In this part, we provide the results for the bagging method and applying the different proposed methods for selecting classifiers. Table 9.5 shows the varying results in this case.

Table 9.5 Bagging with Selection Error Results

Dataset	% Error Initial	% Error Comb	% Error Sep	% Error RE	% Error New	% Error SA	% Error SA-C	# Trees New	# Trees SA-C
Australian	**14.5(3.8)**	**14.5(3.8)**	**14.5(3.8)**	**14.5(3.8)**	**14.5(3.8)**	**14.5(3.8)**	**14.5(3.8)**	50	50
Blood Transfusion	**23.8(4.9)**	23.9(5.0)	23.8(4.9)	23.9(5.0)	**23.8(4.9)**	**23.8(4.9)**	**23.8(4.9)**	50	50
Cancer	7.3(3.4)	**5.6(3.1)**	5.8(3.2)	5.7(3.2)	**5.6(3.2)**	5.7(3.0)	6.8(3.2)	7.3	46.1
Circular	35.7(7.6)	**34.8(7.4)**	35.0(7.3)	35.0(7.4)	35.2(7.6)	35.5(7.3)	35.7(7.6)	3.2	45.3
Crab	39.3(10.6)	**32.1(11.7)**	32.4(13.2)	32.3(11.8)	33.2(12.3)	33.1(12.1)	32.6(11.1)	3.7	34.2
Flare	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	**5.7(2.0)**	50	50
Fourclass	32.8(5.4)	**26.4(5.0)**	28.7(5.7)	26.5(5.1)	26.8(5.1)	30.7(4.8)	31.3(5.1)	9.4	35.4

Table 9.5 Continued

German	**29.9(5.1)**	30.0(4.8)	30.2(5.0)	30.1(4.9)	30.0(4.9)	**29.9(5.1)**	**29.9(5.1)**	2.3	50
Glass	**7.8(7.2)**	8.3(7.3)	8.9(7.4)	8.5(7.4)	**7.8(7.2)**	**7.8(7.2)**	**7.8(7.2)**	46.2	50
Iono	17.4(6.4)	16.8(6.3)	**16.7(6.5)**	16.8(6.4)	17.1(6.2)	17.3(6.7)	17.4(6.4)	6.9	46.7
KC2	16.1(5.2)	16.5(4.1)	16.2(4.9)	16.6(4.3)	**16.0(4.4)**	16.5(5.7)	16.0(5.3)	8.1	47.7
Magic	27.1(0.9)	**26.4(0.8)**	**26.4(0.8)**	**26.4(0.8)**	26.5(0.8)	26.6(0.9)	26.7(0.9)	23.7	28.1
Mammography	**18.1(3.2)**	**18.1(3.2)**	**18.1(3.2)**	**18.1(3.2)**	**18.1(3.2)**	**18.1(3.2)**	**18.1(3.2)**	50	50
Mushroom	**20.9(1.5)**	**20.9(1.5)**	**20.9(1.5)**	**20.9(1.5)**	**20.9(1.5)**	**20.9(1.5)**	**20.9(1.5)**	50	50
Pima	27.3(3.9)	**25.7(4.6)**	25.8(4.2)	25.8(4.4)	26.2(4.3)	26.7(3.3)	27.2(3.8)	3.9	37.6
Skin	**11.7(0.2)**	**11.7(0.2)**	**11.7(0.2)**	**11.7(0.2)**	**11.7(0.2)**	**11.7(0.2)**	**11.7(0.2)**	50	50
Spiral	28.5(4.3)	29.3(4.8)	29.2(4.7)	29.3(4.8)	28.7(4.4)	**28.4(4.3)**	28.5(4.3)	41.2	47.3
TicTac	**29.8(4.9)**	**29.8(4.9)**	**29.8(4.9)**	**29.8(4.9)**	**29.8(4.9)**	**29.8(4.9)**	**29.8(4.9)**	50	50
Titanic	**22.4(2.7)**	**22.4(2.7)**	**22.4(2.7)**	**22.4(2.7)**	**22.4(2.7)**	**22.4(2.7)**	**22.4(2.7)**	50	50
Twonorm	24.4(4.9)	17.6(2.4)	17.7(2.4)	17.6(2.4)	**17.5(2.4)**	18.7(2.6)	19.0(2.5)	12.4	23.4
Average Value	***22.0%***	***20.8%***	***21.0%***	***20.9%***	***20.9%***	***21.2%***	***21.3%***	***28.4***	***44.6***
Average Rank	***4.7***	***3.3***	***3.9***	***4.1***	***3.6***	***4.1***	***4.4***		

9.1.5 Discussion and Insight

The obtained results show the validity of the denoted approaches. The search methods provide improvements in reducing the number of learners and thus storage in many cases. Also, the proposed scheme generally improves the accuracy of the set of learners and thus enhances boosting. In comparison with the RE method, the results here are better especially that RE is a special case of the combining classifier search.

These ideas can be further tested on other methods to check their performance whether in terms of training or testing errors which are often related albeit not always due to over-fitting. It is worth mentioning here that initial tests on other boosting algorithms like AdaboostM2 [93] led to similar conclusions in multi class problems.

Definitely, the improvement involved in the implemented methods can be argued as low since there was no or slight advance in several cases in the datasets. This is certainly an issue that indicates the existence of a limit to possible enhancements when trying to select classifiers of the same type. And this can be considered as a drawback of similar approaches that try to select amongst diverse learners of similar type, as the literature argues for especially in the boosting case which is known to be generally diverse. Still, the enhancement exists in several cases with almost no losses and this probably relies on the dataset itself according to the features, etc... This could be the subject of further studies.

Also, it is possible here to use Wilcoxon test and other non-parametric tests to check the validity of the approaches and whether they lead to enhancements or not, but this seemed unnecessary since the results are almost always equal to or better than the original results. Additionally, it should be possible to carry noise related tests on the proposed approaches in the future.

Of course, these search methods are not the only possible ones. For example, it is possible to use 80% or 85% of the training data instead of 90%, as the large data part, or other combinations and indeed this was tested. Similar results were generally obtained in the different boosting techniques although they were generally less accurate in the AdaboostM1 case. This rather indicates the dependence of the different boosting methods on the data which could require further studies.

Moreover, the selection of the classifiers depends on the training data and this is highly related to the availability of this data, the number of available patterns, features, and if the training data well represents the whole data or not.

Also, it is possible to perform the search using different parts of the training data since the 90%-10% training data was randomly generated. This idea can be repeated for a number of times and different sets can be obtained, but this would consume more time.

In addition, the same approaches should be carried out to different weak learners besides the classification and regression trees, like the Bayes classifier or other stronger learners. Also, there is a possibility for combining the different kinds of classifiers of the boosting techniques together which is an issue that is sometimes considered in literature.

Additionally, there are several observations that were found when applying the noted methods. A notable idea is that the best classifier of the whole set, usually the first, was almost always among the best found group of the classifiers in a similar situation to being a team captain. Another important note is that arranging the learners in terms of accuracy and then directly selecting a top set of them is not a solution since that was generally not the best found set. Moreover, and since the search procedures lead to a tree of solutions, one can select a set of certain limited size if this is required as already mentioned regarding storage and speed although the search itself requires additional time.

Thus, these varying ideas lead to several questions regarding quality versus quantity and teamwork. These ideas provide notions about such concepts which are generally known, like having the best people may not necessarily lead to the best results and similar concepts in various fields.

Also, there are possible scenarios that could lead to enhancing the different ideas proposed here. One possible development is to search for different sets of classifiers, with each set handling different data based on the features, or other criteria, which could enhance accuracy. This means performing search on classifiers and features which is more complicated than the main ideas discussed here.

9.2 Results on Stereo Matching

In this section, the main results regarding stereo matching are presented. These are shown for the Middlebury evaluation set [1]. Also and since we are using a classification based approach, it is important to create a training dataset which is also based on images selected from [1].

We considered several methods to validate our proposed classification scheme. These include the works of [29], [30], [44] and [36].

The images used for training included Bull, Aloe, Art and Books except for [44] and [36] since their data is not readily available. In particular, 25% of the pixels of each image were randomly chosen for training which makes a total of 167350 pixels. 25% means that there are 4 parts. This was performed for 10 times in each case which means a total of 40 runs were made. The results tabulated here are the average of the different runs with the figures being of a random case. Definitely, each method had a separate training stage since each one had different types of error.

The main errors in the training stages varied between 3 to 13 % for all the classification techniques in all the methods.

In regards to the error calculation and the comparison of the disparity with the true one, we used the same measure as [1]. This basically finds the percentage of pixels whose disparity differs from the true disparity by a certain difference threshold according to equation (9.1):

$$\%\,error = \frac{\#\,pixels\ \ with\,(|disp - true\,disp| > thresh)}{\#\,all\ pixels} * 100 \qquad (9.1)$$

The threshold usually varies between 0.5 and 2 according to [1] and in this work we set it equal to T_D or 1 for Tsukuba and Venus, and 2 for the Teddy and Cones, as already mentioned.

Figure 9.1 shows the SAD obtained disparity maps through the proposed approach. Moreover, all the figures that are shown afterwards were obtained through AdaboostM1 since the proposed classification method did not greatly affect the results visually. It is important to mention that all the proposed features excluding 9 and 11 were considered for classification using this approach.

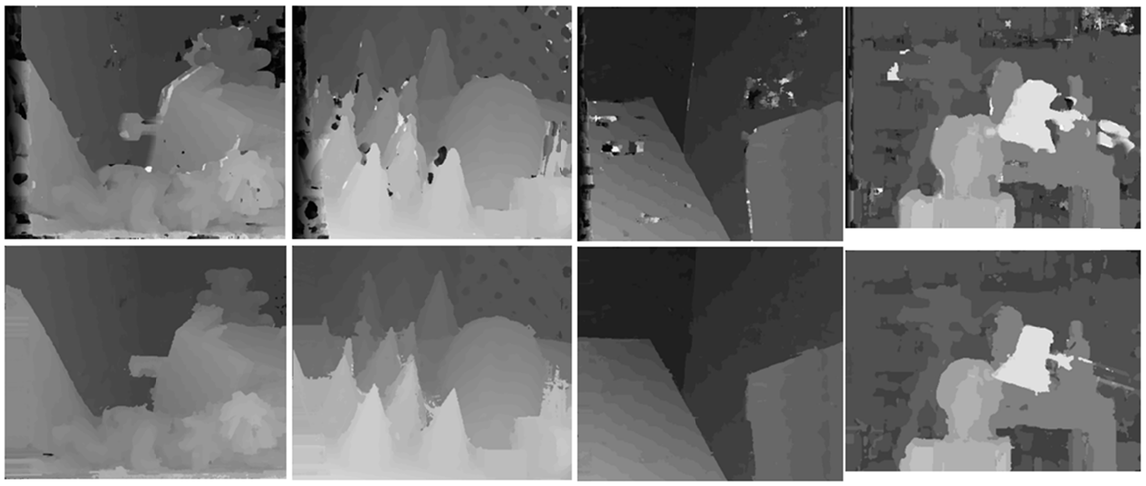


Fig.9.1. Disparity Maps using SAD (Top Row) and through SAD+Proposed Approach (Bottom Row)

Figure 9.2 shows the NCC obtained disparity maps before and after using the proposed enhancement. Also, and like SAD, all the features were chosen excluding 9 and 11.

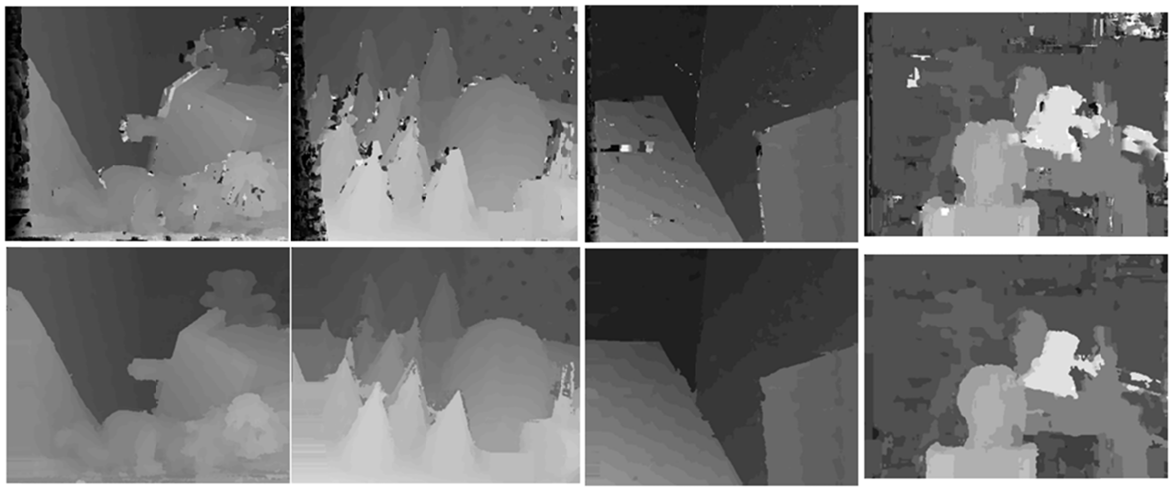


Fig.9.2. Disparity Maps using NCC (Top Row) and through NCC+Proposed Approach (Bottom Row)

Figure 9.3 shows the obtained disparity maps through [29] and after enhancing the results with the proposed approach. For this method, the selected features were the same as NCC and SAD.

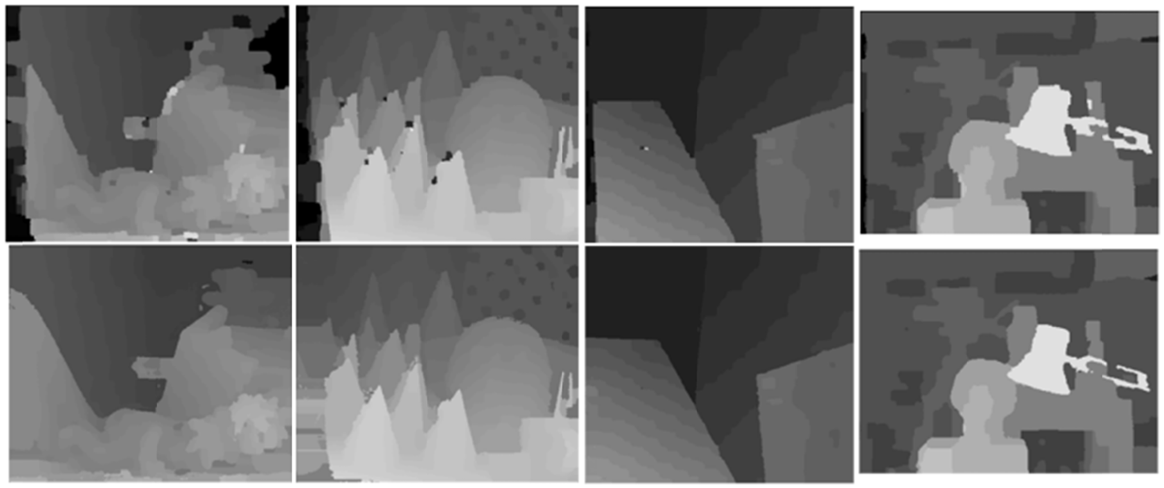


Fig.9.3. Disparity Maps using [29] (Top Row) and through [29]+Proposed Approach (Bottom Row)

Figure 9.4 shows the results obtained by performing the method of [30] before and after the proposed approach. The same set of features as before was chosen with this approach.

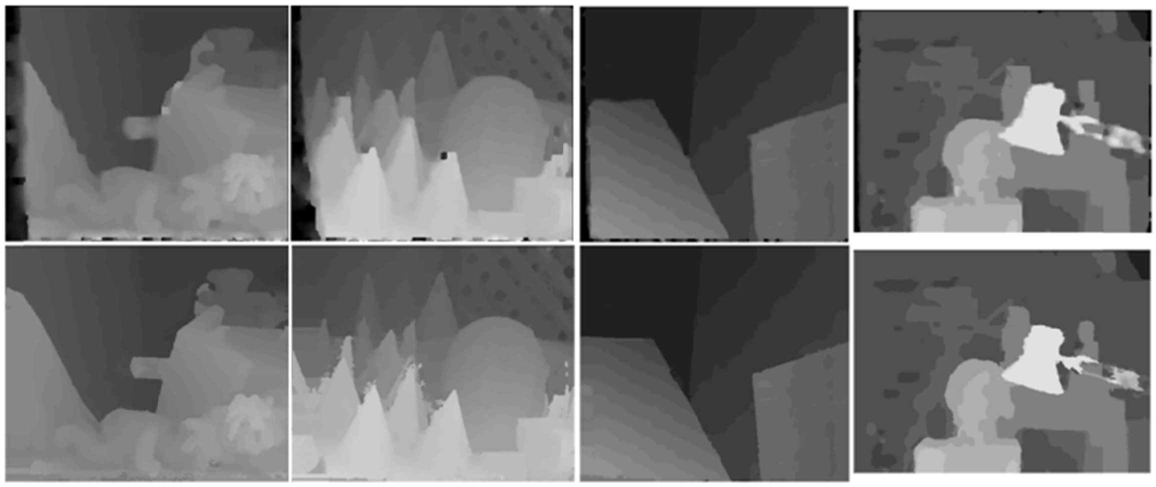


Fig.9.4. Disparity Maps using [30] (Top Row) and through [30]+Proposed Method(Bottom Row)

Also, Figure 9.5 shows the disparity maps obtained through [44] before and after using the proposed enhancement. The selected features were features 2 to 9.

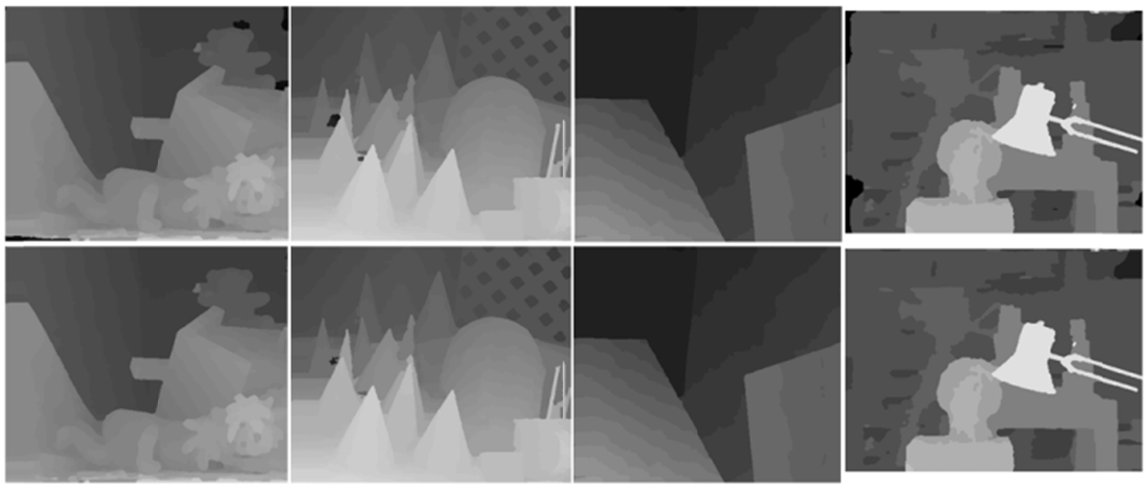


Fig.9.5. Disparity Maps using [44] (Top Row) and using [44]+Proposed Approach (Bottom Row)

Figure 9.6 shows the disparity maps obtained through [36] before and after performing the proposed implementation. For this method, the set of selected features was composed of features 1, 3, 5, 6, 7, 9 and 11.

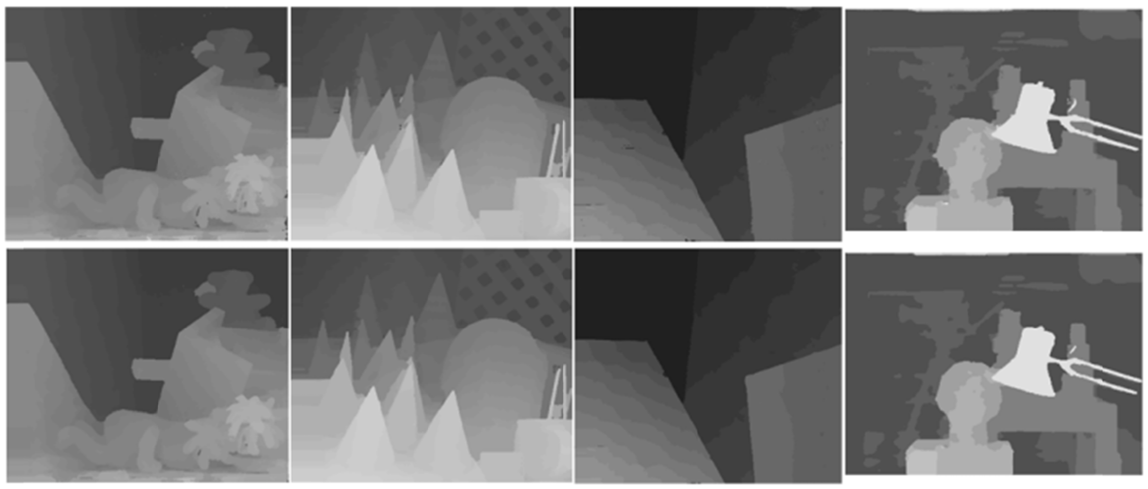


Fig.9.6. Disparity Maps using [36] (Top Row) and using [36]+Proposed (Bottom Row)

Table 9.6 shows the results evaluated according to equation (21) with a threshold value equal to 2 or 1 according to the disparity range of the image. The table shows the original error and the found error after applying the proposed approach for the different used methods. The initial processing results are presented when possible. It is worth remarking that preprocessing was not performed to the methods of [44] and [36] and is not needed for NCC that is illumination invariant.

Table 9.6 Results on the Middlebury Test Images Measured by Percentage Error Threshold Difference for Proposed Approach

Error / Image / Method	*Teddy*			*Cones*			*Venus*			*Tsukuba*		
	Original	Initial	Proposed	Original	Initial	Proposed	Original	Initial	Proposed	Original	Initial	Proposed
SAD+AdaBoostM1	26.8	23.2	*11.5*	20.2	19.5	*12.5*	8.4	8.4	*2.6*	10.1	10.1	*5.2*
SAD+Proposed	26.8	23.2	*11.3*	20.2	19.5	*12.4*	8.4	8.4	*2.4*	10.1	10.1	*5.1*

Table 9.6 Continued

NCC+AdaBoostM1	21.3	21.3	*11.6*	19.7	19.7	*12.5*	6.9	6.9	*2.7*	11.8	11.8	*5.9*
NCC+ Proposed	21.3	21.3	*11.1*	19.7	19.7	*12.4*	6.9	6.9	*2.6*	11.8	11.8	*5.7*
[29]+AdaBoostM1	22.6	20.9	*11.9*	19.4	14.3	*11.7*	2.8	2.8	*1.2*	5	5	*2.9*
[29]+ Proposed	22.6	20.9	*11.6*	19.4	14.3	*11.5*	2.8	2.8	*1.2*	5	5	*2.8*
[30]+AdaBoostM1	19.3	18.8	*11.5*	16.8	16.0	*11.9*	4.7	4.7	*1.9*	5.2	5.2	*4.5*
[30]+ Proposed	19.3	18.8	*11.3*	16.8	16.0	*11.7*	4.7	4.7	*1.8*	5.2	5.2	*4.3*
[44]+AdaBoostM1	7.0	NA	*6.6*	6.1	NA	*6.1*	0.5	NA	*0.4*	2.3	NA	*1.9*
[44]+ Proposed	7.0	NA	*6.5*	6.1	NA	*6.1*	0.5	NA	*0.4*	2.3	NA	*1.9*
[36]+AdaBoostM1	7.9	NA	*7.7*	7.2	NA	*7.0*	0.7	NA	*0.5*	1.8	NA	*1.6*
[36]+ Proposed	7.9	NA	*7.5*	7.2	NA	*7.0*	0.7	NA	*0.5*	1.8	NA	*1.6*

Besides, this work argues that it is possible to perform occlusion handling through classification which should be validated. This is achieved by testing the different methods with LRC in comparison with classification. The methods tested here are the NCC, SAD, method of [29] and method of [30]. The works of [44] and [36] are not tested since their code and right disparity maps are not available. For each method, we present the results achieved after performing filling through both our proposed completion stage (*Cp*) and the basic regular method (*Cb*).

In order to ensure a fair comparison, all the images underwent initial processing. In addition, the disparities prior to 60% of the disparity range were set to unknown in all the cases since it generally improved the results. Table 9.6 shows the comparison between the methods. In Table 9.7, {1} means the proposed approach using AdaboostM1 with (*Cp*) as in Table 9.5 and {2} means the proposed approach using AdaboostM1 with (*Cb*) and median filtering. Also, {3} means LRC followed by (*Cp)* and {4} means LRC followed by (*Cb*) and median filtering.

The results validate that classification performs better than LRC in most cases, albeit not all, for the tested methods. LRC performs better on the Teddy and Cones images using the methods of [29] and [30]. Also, the results prove that *(Cp)* performs better or similar to *(Cb)*.

Table 9.7 Occlusion Handling Results on the Middlebury Test Images Measured by Percentage Error Threshold Difference

Error / Method / Image	*NCC {1}*	*NCC {2}*	*NCC {3}*	*NCC {4}*	*SAD {1}*	*SAD {2}*	*SAD {3}*	*SAD {4}*	*[29] {1}*	*[29] {2}*	*[29] {3}*	*[29] {4}*	*[30] {1}*	*[30] {2}*	*[30] {3}*	*[30] {4}*
Teddy	11.6	11.9	12.3	12.7	11.5	13.9	14	16	11.9	13.6	11.7	13.5	11.5	13.4	11.3	11.7
Cones	12.5	12.6	14.5	14.5	12.5	12.6	12.6	13	11.7	12.4	10.1	11.9	11.9	13.2	11.5	12.4
Venus	2.7	2.9	4.3	4.7	2.6	3.1	5.3	5.6	1.2	1.3	1.4	1.5	1.9	2.6	3.4	3.4
Tsukuba	5.9	6.4	9.3	9.7	5.2	5.8	8.0	8.2	2.9	3.5	3.1	3.9	4.5	5.2	4.1	5.5

Besides these results, it is important to show the results regarding the proposed segmentation with ASW which was discussed in section 4.3.3. So, Figure 9.7 shows the obtained disparity maps using the initial approach with the completion stage.

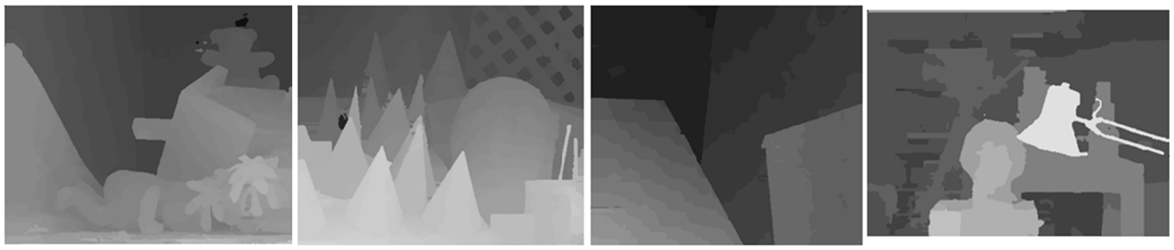


Fig.9.7. Disparity Maps using segmentation with ASW

In addition, Figure 9.8 shows the obtained maps using the segmentation ASW approach in addition to the classification stage. Although the results show improvements, these are still somewhat minor.

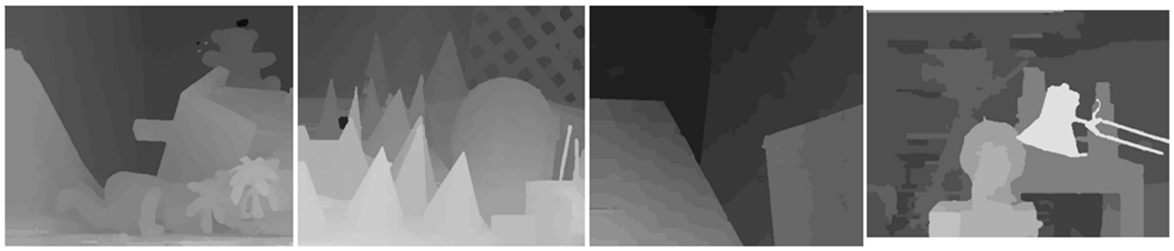


Fig.9.8. Disparity Maps using segmentation with ASW+ AdaboostM1

Furthermore, Figure 9.9 shows the obtained maps for the same method, but while accounting for feature 12. The improvement here is clearer and surpasses that without feature 12.

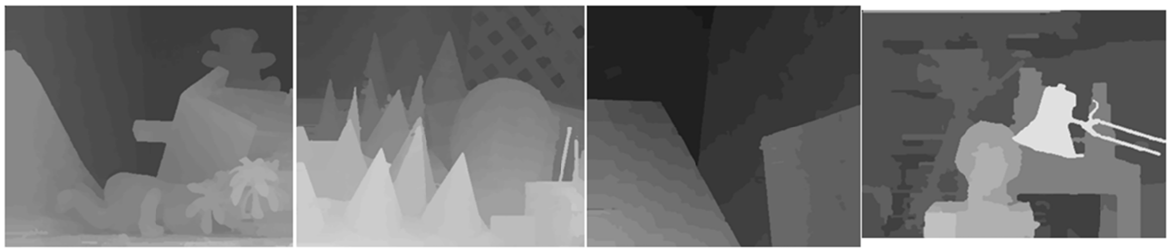


Fig.9.9. Disparity Maps using segmentation with ASW+ AdaboostM1 with feature 12

Moreover, Table 9.8 shows the results of this approach with and without considering the twelfth feature. All the results presented in Table 9.7 were achieved in this work. It is important to note that the proposed classification approach did not improve on boosting in this case, so the results are not repeated in the table.

Table 9.8 Results on the Middlebury Test Images Measured by Percentage Error Threshold Difference for the Proposed Segmentation with ASW

Error / Image / Method	*Teddy*			*Cones*			*Venus*			*Tsukuba*		
	Original	Initial	Proposed	Original	Initial	Proposed	Original	Initial	Proposed	Original	Initial	Proposed
SEG+AdaBoostM1	5.8	5.8	*5.6*	6.5	6.5	*6.4*	0.5	0.5	*0.4*	1.7	1.7	*1.7*
SEG+Feature 12+ AdaBoostM1	5.8	5.8	*5.4*	6.5	6.5	*6.2*	0.5	0.5	*0.4*	1.7	1.7	*1.6*

9.2.1 Discussion about Features

Features were selected based on speed and accuracy. Some features seemed redundant but were used since they were easy to compute. Some candidate features improved the accuracy but were dismissed due to extra computations. Various tests were performed with and without the used features. The tests included random combinations of different features and testing the possible set excluding each contending feature. These tests led to the described set of features.

Definitely, there are similarities between the features and this is mainly because they are depicted from the intensity and disparity data. Also, some features are indeed alike as in features 8 and 9 and features 10 and 11. In such cases, it is possible to use one of each since they were slightly different.

It is important to note here that in order to ensure faster results; each of the histograms used 64 bins instead of 256 bins. This did not diminish the accuracy and helped obtain higher speeds. Furthermore, few of the histograms had some empty bins, so, these were filled with the nearest known values.

Also, there could be some ambiguities as in accounting for a true disparity which is usually in sub-pixel accuracy. For this, if the disparity error was ≤ maximum (3% of the disparity range, and 1 value in difference), it was accepted as true. The same idea was used for the confident disparity map (D_A), so if the difference was ≤ 3% or 1 pixel difference, the disparities were considered similar. This led to T_D.

Moreover, there are many possibilities for the features. The first one is including the SAD or the NCC distances. It is also possible to choose different sized neighborhoods. In addition, it is possible to use the median instead of the mean for several of the features. Another simple choice is to divide the disparity map into different sized parts and so on. Also, one can generate other confident disparity maps by accounting for various combinations from the neighboring pixels. Furthermore, different features can be combined through normalization and multiplication which could improve accuracy.

Additionally, it is possible to use the edge filtered map of the stereo image or use the Canny [50] Edge map for the stereo images and maps, but this is generally time demanding.

Besides, Local Binary Patterns (LBP) and Histogram Oriented Gradients (HOG) were tested for both the image and the map, but they provided little improvement. This is probably due to redundancy that is available in features related to edges and in the nature of the classification problem here.

Moreover, the inclusion of feature 12 in the case of segmentation proved to further enhance the accuracy of the system. This validates that additional features are required to improve the more accurate methods.

It is important to stress that including further features would require additional processing and slow the system.

Obviously, there are numerous possibilities and countless features to try, which can be carried in future work.

9.2.2 Discussion about Results

The best obtained results varied according to the tested image and the chosen algorithm. Also, high enhancement was not always achieved as in the Tsukuba and Venus images which were less enhanced using [44] and [36].

Moreover, it is easily notable that the results were more improved in the lower accuracy methods as opposed to the higher ones. This varying enhancement is due to the error being more systematic in the lower methods with the existence of bigger room for improvement. Another reason for this is that the features concentrated on the lower methods and additional features are probably required to enhance the better methods. Besides, a vital idea is that the accurate methods already handled occlusion using LRC which reduced the main source of error.

Also, and as an argument in favor of this work, many of the best results in literature are obtained through time demanding approaches performed on the high resolution colored images. This is different from using the low resolution gray scale ones as in this work. A typical example of this is [168] which used segmentation and matching to achieve high accuracy at the expense of time.

Moreover, the found results achieve better accuracy and speeds than those in [120] and [121]. Besides, the enhanced SAD results show a clear improvement over those obtained using [29] and [30].

9.2.3 Advantages and Limitations

The main advantage of using classification for stereo matching in the unique suggested manner is the speed that the system improves accuracy. This is achieved without the need for LRC and regardless of the number of disparities. Still, there are arguable drawbacks.

First, the enhancement for lower accuracy images is more as already discussed with the possibility of having no improvement if erroneous pixels were not found.

Besides, the data and storage requirements of the system might seem high due to having eleven features per pixel. These features need to be computed for every pixel which could require a relatively high amount of data to store if not well treated. In any case, the memory storage is acceptable since the worst case scenario does not exceed (13) times the image size, but this is discouraged since it would require additional global memory accesses.

Besides, it is clear that the proposed approach is totally dependent on the initial method used in addition to the classification method and the features used.

Furthermore, the system is totally reliant on the quality of the training data and its relation to the testing data. Thus, if the training data is too dependent on a certain scene, it is expected to perform well on similar scenes and less on different ones.

Also, the completion stage needs to be accurate and the overall accuracy of the system directly depends on this stage, so it is not just a matter of classification.

Still, some of the limitations may be claimed as advantages. The approach can be used with any initial matching method which makes it generic. Also, the system delivers according to what you give it, so the dependence on training is convenient when stereo matching is to be performed in a known environment. Another very significant advantage is that the ideas presented here constitute a new technique for handling occlusion without computing the opposite map.

Moreover, there are several gains from the presented approach. The first one is that the system's dependence on features means that it can be improved with other ones. Also, one can use segmentation since similar disparities are usually obtained in same segments. Segmentation can be used for the stereo pair and the disparity map since errors usually occur together. In addition, using the color information should enhance the accuracy of the system, although more computations would be required. Also, one can use other classifiers such as other boosting methods, SVM, Bagging, etc…

Furthermore, there are other possibilities for the classification problem since one can check the correctness of a set of disparities as opposed to checking only one as done here. This can be useful since it is highly likely that the correct disparity would be in the selected set. Of course, the more there are candidate disparities the more computations are required. All these ideas prove that there are many prospects for future improvements.

9.3 CUDA Related Results

This section presents the different results related to the implemented CUDA parts. These include the proposed parallel edge detection approach in addition to the proposed stereo matching and classification based approach.

9.3.1 CUDA Edge Detection Results

We proposed a new parallel edge detection algorithm, so we need to analyze its performance. At first, it is important to note that a regular single core CPU implementation of a (3 x 3) kernel operation usually consumes about 2 milliseconds (ms) on 2 Ghz processors. For the work here, the performance varies according to the used GPU and the number of threads and blocks. And for better comparison, the Image Filter implementation based on shared memory provided in the GPU Computing SDK [2] is considered. So, concerning the proposed approach, the timing required for a (512 x 512) pixels grayscale image with a (3 x 3) kernel was about .16 (ms) on an old very inexpensive CUDA capable GPU GT 240 as opposed to .20 (ms) consumed by the SDK

[129] implementation. Same sized images led to having less than 0.05 ms on a GTX 460 GPU in comparison with the original implementation which consumed .06 (ms) and less than 0.02 (ms) on a Tesla C2070. These results mean that there is about 100 times improvement for the GPU over the CPU. Also, any OpenMP or a parallel CPU enhancement should not yield more than (number of CPUs) times improvement, which proves the benefit of the GPU due to the nature of the pleasantly parallelizable problem.

For a (5 x 5) filter, performance over SDK [129] code improved up to 22%. For an (11 x 11) filter, the increase was 23% to 26%. Concerning the CUDA variables used, the block width or number of threads per block was set to 128 and divided by 4 for coalescence; the number of rows passed through was set to 32 for smaller filters and increased for larger ones meaning that a single thread had to deal with 32 or more pixels along the column. Table 9.9 shows some examples of the main parameters used for images of different sizes. A necessary idea here is that these timings exclude the processes preceding or following the implementation.

Table 9.9 Example Parameters for CUDA Implementation

Kernel Size	Image Size	Blocks	Threads	Improvement
3 x 3	256 x 256	4 x 8	16 x 1	12 - 14 %
3 x 3	512 x 512	4 x 16	32 x 1	15 - 20 %
3 x 3	640 x 640	4 x 20	40 x 1	16 - 20 %
5 x 5	256 x 256	4 x 8	16 x 1	13 - 15 %
5 x 5	512 x 512	4 x 16	32 x 1	18 - 22 %
5 x 5	640 x 640	4 x 20	40 x 1	18 - 22 %

It is important to mention that using memory coalescence in the applying masks implementation leads to more than 200% improvement over an implementation without it.

Definitely, there are issues that must be taken into account and these are mainly memory related. The main bottleneck when dealing with images, or large data, in case of GPUs is in sending the data from and to the main memory in relation with the GPU global memory. These memory transfers often consume more time than very basic operations done on the GPU such as performing (3 x 3) mask operations on images, unless this process is repeated and the image is stored in the GPU global memory. This directly means that any algorithm for performing similar operations, even with 0.00 ms time consumption, will not be beneficial unless big kernels are used or processes are repeated as in the mentioned applications. Another main condition is that the memory transfers between the GPU and the main memory must be minimal. Thus, the effect of any enhanced algorithm could be of importance according to the required application and not in very basic operations, because in such scenarios, it is preferred to use serial implementations on the CPU since the memory transfers are reduced rather than any parallel architecture.

9.3.2 CUDA Stereo Matching Results

As already denoted in this work, the combination of a fast and relatively accurate system is a property that is achieved here. Thus, it is important to analyze the speed of the system. The analysis is reported according to a parallel implementation on a CUDA device. In particular, we used an Nvidia Geforce GTX 460 GPU and obtained the kernel execution time results using the Visual Profiler by Nvidia.

After the discussion of the main ideas regarding the CUDA implementation in Chapter 8, we need to show to show the portion of each computational part in the SAD and AdaboostM1 approach. So, Table 9.10 shows the portions for the Teddy Map

TABLE 9.10 Portion In Overall Time Of Each Component For The Teddy MAP Computation Using SAD+AdaboostM1

Part	***Time (Milliseconds)***	***Portion***
Initial Processing	1.3	13.5
SAD Stereo	3.9	40.6
Feature 1	0.1	1
Feature 2	0.2	2.1
Features 3 and 4	0.4	4.2
Feature 5	0	0
Feature 6	0.1	1
Feature 7, 8, 9, 10, 11	0.5	5.2
Classification (AdaboostM1)	0.3	3.1
Completion (Cp)	2.4	25
Median Filtering (MF)	0.4	4.2
Total	9.6	100%

Also, we need to report the different time results for all the images, so, Table 9.11 shows the execution times obtained by using SAD with the proposed stages in comparison with [29] and [30] without the proposed enhancement. In all the cases, initial processing and median filtering are included.

It is necessary to note here that the literature usually compares between algorithms on different GPUs as in [136], [137] and [138], which can be hard to follow. So, we opt to account for the time of the SAD approach on a (11x11) window as a benchmark for comparison. The reported time for this SAD does not include preprocessing and median filtering. This is done since the GPU algorithms in literature require more time than the SAD approach even with LRC. Through this, it would become easier to compare related stereo matching research. So, Table 9.11 shows the ratio of the execution time of each method in comparison with the SAD (11x11) approach.

It is worth mentioning that each of the approaches in [136], [137] and [138] consume more than 40 ms on the Teddy image noting that they achieve higher accuracy than this work. So, the proposed approach consumes relatively little time, but, we cannot be very sure about the actual speedup over these methods due to the different setups.

TABLE 9.11 Execution Time (ms) Comparison With Other Stereo Matching Methods

Method	*Teddy*	*Cones*	*Venus*	*Tsukuba*	*SAD Ratio*
SAD (11x11)	3.9	3.9	2.1	0.9	1
SAD(11x11) +LRC+Cb	9.7	9.6	5.8	3.4	2.6
SAD(11x11) +LRC+Cp	13.9	12.5	7.8	4.9	3.6
Proposed (SAD+Boost)	9.6	9.6	6.7	5.2	2.9
[29]	217	217	104	37	53
[30]	79	79	39	31	21

CHAPTER 10

CONCLUSION

The work presented interesting ideas regarding various domains that include stereo matching, classification, edge detection and others. This chapter presents the main conclusions.

This work dealt with the two subjects of boosting and stereo matching. Several ideas were developed and work considered many ideas including edge detection, segmentation and parallel programming.

In regards to classification or more particularly boosting, different methods are considered in this work including Adaboost, GentleBoost and LogitBoost. The concept of selecting classifiers out of the total set of classifiers is tested in a novel manner. This is a post processing stage that can be performed to any of the noted concepts and has proven to be of enhancing effects in many cases with the improvement rate varying according to the dataset and method used. Two complimentary search methods are presented and combined together to find a suitable candidate set of classifiers. This set performs well on the training data. The selection procedure is based on the training data that is treated in a selective manner in order to avoid over-fitting and reject or accept the chosen set of classifiers as opposed to the original boosting set.

Concerning stereo matching, we present an original approach that relies on classification to reduce errors and handle occlusion [169]. The images are first processed using a novel method that utilizes histogram information for reducing illumination variances. Afterwards, different standard and new methods of stereo matching are considered to obtain the initial disparity map. These include a new method that uses edge based segmentation for addressing stereo matching, where edge detection used a new mask for edge detection. The map and the image are then used as the main data for obtaining several features that are used in a classification scheme. This scheme succeeds in remedying different errors of the initially used algorithm. The main contribution here is that the proposed system is generic, since any algorithm can be used for the initial matching. This also renders the approach as a novel verification stage that can be used for solving the occlusion problem. A rather minor contribution is the suggested completion stage which can be used in any stereo matching method after occlusion handling. There are several merits of this work. First, handling occlusion in the proposed manner saves the computation of the opposite map if not desirable. Second, the features used are easy to compute and can be improved. Third, the classification is performed per pixel in a straightforward manner. Fourth, most of the parts are parallelizable and fast yielding real time performance through parallel architectures like CUDA, which was emphasized throughout the document. Fifth, the accuracy is acceptable since low resolution gray scale images were used.

Besides these ideas, the work considers several of the possible improvements that can be later carried in the future. Concerning classification, different ideas and insights are discussed to add to the prospects of the work. This work presents further possibilities to enhancing classification and it should hopefully lead to more related research whether on the analytical or experimental domains of boosting, bagging and others through selecting classifiers whether of similar or different types. Moreover and concerning stereo matching, this work further discusses the different prospects noting that the timing added to the implemented system is minimal and justifies the increase in performance. And so, all these ideas provide room for future work.

REFERENCES

[1] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, 2003, pp. I-195-I-202 vol.1.

[2] http://en.wikipedia.org

[3] Gonzalez and R. E. Woods, Digital Image Processing. Upper Saddle River, NJ ; Harlow: Pearson/Prentice Hall, 2008.

[4] B. Zitova and J. Flusser, "Image registration methods: a survey," Image and Vision Computing, Vol. 21, No. 11. pp. 977-1000, October 2003

[5] M. Wyawahare, P. Patil, and H. Abhyankar, "Image Registration Techniques: An overview," International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 2, No.3, pp.11-27 September 2009.

[6] M. Sdika, "A Fast Nonrigid Image Registration With Constraints on the Jacobian Using Large Scale Constrained Optimization," IEEE Transactions On Medical Imaging, pp.271-281, February 2008.

[7] W. Chu, L. Ma, J. Song, and T. Vorburger, "An Iterative Image Registration Algorithm by Optimizing Similarity Measurement," Journal of Research of the National Institute of Standards and Technology, pp.1-6, 2010.

[8] C. J. Taylor and A. Bhusnurmath, "Solving Image Registration Problems Using Interior Point Methods," European Conference on Computer Vision, pp.638–651, October 2008.

[9] S. Klein, M. Staring, and J. P. W. Pluim, "Evaluation of Optimization Methods for Nonrigid Medical Image Registration Using Mutual Information and B-Splines," IEEE Transactions On Image Processing, pp.2879-2890, December 2007.

[10] Y. Huang, T. Tong, W. Liu, Y. Fan ,H. Feng and C. Li, "Accelerated Diffeomorphic Non-rigid Image Registration with CUDA Based on Demons Algorithm," International Conference of Bioinformatics and Biomedical Engineering (iCBBE), pp.1-4, 2010.

[11] P. Muyan-Ozcelik, J. Owens, J. Xia, and S. Samant, "Fast Deformable Registration on the GPU: A CUDA Implementation of Demons," International Conference on Computational Sciences and Its Applications ICCSA pp. 223-233, 2008.

[12] X. Gu, H. Pan, Y. Liang, R. Castillo, D. Yang, D. Choi, E. Castillo, A. Majumdar, T.Guerrero and S. B Jiang, "Implementation and evaluation of various demons

deformable image registration algorithms on a GPU," physics in Medicine and Biology, pp.207–219, 2010.

[13] Zhang Lei, Gao Xianwei, Zhao Cheng and Dong Xiuze, "Research on the technology of extracting 3D face feature points on basis of binocular vision," in Image and Signal Processing, 2009. CISP '09. 2nd International Congress on, 2009, pp. 1-4.

[14] http://vision.middlebury.edu/stereo

[15] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 20, pp. 401-406, 1998.

[16] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: theory and experiment," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 16, pp. 920-932, 1994.

[17] M. Gong, R. Yang, L. Wang and M. Gong, "A Performance Study on Different Cost Aggregation Approaches Used in Real-Time Stereo Matching," International Journal of Computer Vision, vol. 75, pp. 283-296, 2007.

[18] Kuk-Jin Yoon and In-So Kweon, "Locally adaptive support-weight approach for visual correspondence search," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, pp. 924-931 vol. 2.

[19] F. Tombari, S. Mattoccia and L. Di Stefano, "Segmentation-based adaptive support for accurate stereo correspondence," in Advances in Image and Video TechnologyAnonymous Springer, 2007, pp. 427-438.

[20] C. L. Zitnick and S. B. Kang, "Stereo for image-based rendering using image over-segmentation," International Journal of Computer Vision, vol. 75, pp. 49-65, 2007.

[21] M. Bleyer, C. Rother and P. Kohli, "Surface stereo with soft segmentation," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 1570-1577.

[22] V. Vineet and P. J. Narayanan, "CUDA cuts: Fast graph cuts on the GPU," in Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, 2008, pp. 1-8.

[23] Myung-Ho Ju and Hang-Bong Kang, "A new method for stereo matching using pixel cooperative optimization," in Image Processing (ICIP), 2009 16th IEEE International Conference on, 2009, pp. 2105-2108.

[24] 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, 2008, pp. 1-8.

[25] J. Y. Goulermas, P. Liatsis and T. Fernando, "A constrained nonlinear energy minimization framework for the regularization of the stereo correspondence problem," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 15, pp. 550-565, 2005.

[26] E. Park and K. Wohn, "Stereo and motion correspondences using nonlinear optimization method," Comput. Vision Image Understanding, vol. 101, pp. 194-203, 3, 2006.

[27] W. Kim, J. Park and K. Lee, "Stereo Matching Using Population-Based MCMC," International Journal of Computer Vision, vol. 83, pp. 195-209, 2009.

[28] Grauer-Gray and C. Kambhamettu, "Hierarchical belief propagation to reduce search space using CUDA for stereo and motion estimation," in Applications of Computer Vision (WACV), 2009 Workshop on, 2009, pp. 1-8.

[29] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," Int. Journal of Computer Vision,vol.70, p. 41-54, 2006.

[30] Q. Yang, L. Wang and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 1458-1465.

[31] Jian Sun, Yin Li, S. B. Kang and Heung-Yeung Shum, "Symmetric stereo matching for occlusion handling," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, pp. 399-406 vol. 2.

[32] Jiejie Zhu, Liang Wang, Ruigang Yang and J. Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps," in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, 2008, pp. 1-8.

[33] Qingxiong Yang, Liang Wang, Ruigang Yang, H. Stewenius and D. Nister, "Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 31, pp. 492-504, 2009.

[34] I. Ernst and H. Hirschmuller, "Mutual information based semi-global stereo matching on the GPU," in Proceedings of the 4th International Symposium on Advances in Visual Computing, Las Vegas, NV, 2008, pp. 228-239.

[35] J. Gibson and O. Marques, "Stereo depth with a unified architecture GPU," in Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, 2008, pp. 1-6.

[36] Kuk-Jin Yoon and In-So Kweon, "Stereo matching with the distinctive similarity measure," in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, 2007, pp. 1-7.

[37] Le Thanh Sach, K. Atsuta, K. Hamamoto and S. Kondo, "A robust stereo matching method for low texture stereo images," in Computing and Communication Technologies, 2009. RIVF '09. International Conference on, 2009, pp. 1-8.

[38] Y. Ruichek, "Multilevel- and neural-network-based stereo-matching method for real-time obstacle detection using linear cameras," Intelligent Transportation Systems, IEEE Transactions on, vol. 6, pp. 54-62, 2005.

[39] T. Pock, C. Zach and H. Bischof, "Mumford-shah meets stereo: Integration of weak depth hypotheses," in Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, 2007, pp. 1-8.

[40] B. Cyganek and J. P. Siebert, An Introduction to 3D Computer Vision Techniques and Algorithms. Wiley. com, 2011.

[41] T. Liu, P. Zhang and L. Luo, "Dense stereo correspondence with contrast context histogram, segmentation-based two-pass aggregation and occlusion handling," Advances in Image and Video Technology, pp. 449-461, 2009.

[42] C. Rother, T. Minka, A. Blake and V. Kolmogorov, "Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs," in Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, 2006, pp. 993-1000.

[43] H. Bay, V. Ferraris and L. Van Gool, "Wide-baseline stereo matching with line segments," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, pp. 329-336.

[44] Dongbo Min, Jiangbo Lu and M. N. Do, "Joint Histogram-Based Cost Aggregation for Stereo Matching," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, pp. 2539-2545, 2013.

[45] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, 2007, pp. 1-8.

[46] I. Jung, J. Sim and C. Kim, "Histogram-based stereo matching under varying illumination conditions," in Visual Communications and Image Processing (VCIP), 2012 IEEE, 2012, pp. 1-5.

[47] Y. S. Heo, K. M. Lee and S. U. Lee, "Simultaneous color consistency and depth map estimation for radiometrically varying stereo images," Computer Vision, 2009 IEEE 12th Conference on, 2009, pp. 1771-1778.

[48] M. Antunes and J. Barreto. "Efficient stereo matching using histogram aggregation with multiple slant hypothesis". IbPRIA 2013.

[49] Y. Han, G. Xuan, C. Li, D. Li and H. Han, "Removing illumination from image pair for stereo matching," in Audio, Language and Image Processing (ICALIP), 2012 International Conference on, 2012, pp. 508-512.

[50] J. Canny, "A Computational Approach to Edge Detection," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. PAMI-8, pp. 679-698, 1986.

[51] R. Deriche, "Using Canny's criteria to derive a recursively implemented optimal edge detector," Int. J. Computer Vision, Vol. 1, pp. 167–187, April 1987

[52] F. G. Franklin, J. D. Powell, and A. Emami-Naeini: Feedback Control of Dynamic Systems. Reading, Mass.: Addison-Wesley, 1994.

[53] E. C. Ifeachor andB. W. Jervis: Digital Signal Processing A Practical Approach, Second Edition. Harlow, Prentice-Hall, Pearson Education Limited, 2002.

[54] S. K. Mitra: Digital Signal Processing, Third Edition. New York,:McGraw-Hill, 2006.

[55] C. L. Philips and H. T. Nagle: Digital Control System Analysis and Design, 3rd Edition, Chapter 11. NJ: Prentice-Hall, 1995.

[56] J. G. Proakis and D. G. Manolakis: Introduction to Digital Signal Processing, Third Ed. New Jersey: Prentice-Hall, 1996.

[57] W. J. Tompkins and J. G. Webster: Design of Microcomputer-based Medical Instrumentation. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[58] M. A. Al-Alaoui, "Novel FIR Approximations of IIR Differentiators with Applications to Image Edge Detection,", ICECS 2011, Beirut , Lebanon, December 11-14, 2011

[59] M. A. Al-Alaoui, " Direct Approach to Image Edge Detection Using Differentiators" proceedings of the 17th IEEE International Conference on Electronics, Circuits and Systems, Athens, Greece, pp. 154-157, December 12-15, 2010.

[60] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by simulated annealing", Science 1983, vol 220, pp. 671- 680.

[61] N. Otsu, "A threshold selection method from gray-level histograms," Automatica, vol. 11, pp. 23-27, 1975.

[62] L. G. Shapiro and G. C. Stockman (2001): "Computer Vision", pp 279-325, New Jersey, Prentice-Hall

[63] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, pp. 881-892, 2002.

[64] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 12, pp. 629-639, 1990.

[65] J. J. Clark, "Singularity theory and phantom edges in scale space," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 10, pp. 720-727, 1988.

[66] Y. Cheng, "Mean shift, mode seeking, and clustering," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 17, pp. 790-799, 1995.

[67] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, pp. 603-619, 2002.

[68] C. M. Christoudias, B. Georgescu and P. Meer, "Synergism in low level vision," in Pattern Recognition, 2002. Proceedings. 16th International Conference on, 2002, pp. 150-155.

[69] J. Shi and J. Malik, "Normalized cuts and image segmentation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, pp. 888-905, 2000.

[70] F. J. Estrada, A. D. Jepson and C. Chennubhotla, "Spectral embedding and min cut for image segmentation." in Bmvc, 2004, pp. 1-10.

[71] K. Haris, S. N. Efstratiadis, N. Maglaveras and A. K. Katsaggelos, "Hybrid image segmentation using watersheds and fast region merging," Image Processing, IEEE Transactions on, vol. 7, pp. 1684-1699, 1998.

[72] Mumford, David, Shah, Jayant (1989), "Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems", Communications on Pure and Applied Mathematics XLII (5): 577–685

[73] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," International Journal of Computer Vision, vol. 59, pp. 167-181, 2004.

[74] M. Brejl and M. Sonka, "Object localization and border detection criteria design in edge-based image segmentation: automated learning from examples," Medical Imaging, IEEE Transactions on, vol. 19, pp. 973-985, 2000.

[75] P. A. Hajare and P. A. Tijare, "Edge Detection Techniques for Image Segmentation," International Journal of Computer Science and Applications, vol. 4, 2011.

[76] S. Lakshmi and D. V. Sankaranarayanan, "A Study of edge detection techniques for segmentation computing approaches," IJCA Special Issue on "Computer Aided Soft Computing Techniques for Imaging and Biomedical Applications" CASCT, 2010.

[77] G. M. Hadi and N. H. Salman, "A STUDY AND ANALYSIS OF DIFFERENT EDGE DETECTION TECHNIQUES".

[78] S. S. Al-Amri, N. Kalyankar and S. Khamitkar, "IMAGE SEGMENTATION BY USING EDGE DETECTION." International Journal on Computer Science & Engineering, vol. 2, 2010.

[79] R. Kumar and A. Arthanariee, "A Comparative Study of Image Segmentation Using Edge-Based Approach," .

[80] R. Muthukrishnan and M. Radha, "EDGE DETECTION TECHNIQUES FOR IMAGE SEGMENTATION." International Journal of Computer Science & Information Technology, vol. 3, 2011.

[81] N. Senthilkumaran and R. Rajesh, "Edge detection techniques for image segmentation–a survey of soft computing approaches," International Journal of Recent Trends in Engineering, vol. 1, pp. 250-254, 2009.

[82] P. Arbelaez, C. Fowlkes and D. Martin, "The berkeley segmentation dataset and benchmark," See Http://www.Eecs.Berkeley.edu/Research/Projects/CS/vision/bsds, 2007.

[83] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, 2001, pp. 416-423.

[84] R. E. Schapire and Y. Freund, Boosting: Foundations and Algorithms. MA: MIT Press, 2012.

[85] M. A. Al-Alaoui, some Applications of Generalized Inverse to Pattern Recognition (Ph.D.Thesis Abstr.), Information Theory, IEEE Transactions on, vol. 22, pp. 633, September, 1976.

[86] M. A. Al-Alaoui, A New Weighted Generalized Inverse Algorithm for Pattern Recognition, Computers, IEEE Transactions on, vol. C-26, pp. 1009, Oct., 1977.

[87] M. A. Al-Alaoui, R. Mouci, M. M. Mansour and R. Ferzli, Systems, A Cloning Approach to Classifier Training, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 32, pp. 746, Nov, 2002.

[88] M. A. Al-Alaoui, "Application of constrained generalized inverse to pattern classification," Pattern Recognition, vol. 8, pp. 277, 1976.

[89] M. A. Al-Alaoui, "Al-Alaoui Pattern Recognition Algorithm: A MSE Asymptotic Bayesian Approach to Boosting", Proc. of the 2009 International Conference on Image Processing and Computer Vision, IPCV 2009, Las Vegas Nevada, USA, pp. 683-689, July 13-16 2009.

[90] D. Helmbold, and M. Warmuth, "On Weak Learning." University of California, Santa Cruz, Computer Research Laboratory, 1992.

[91] N. C. Oza, "Online bagging and boosting," in Systems, Man and Cybernetics, 2005 IEEE International, 2005, pp. 2340.

[92] B. Babenko, Ming-Hsuan Yang and S. Belongie, "A family of online boosting algorithms," in Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th, 2009, pp. 1346.

[93] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences, vol. 55, pp. 119, 1997.

[94] L. Breiman, "Bagging Predictors." Machine Learning 26, pp. 123–140, 1996.

[95] L. Breiman, Classification and Regression Trees. Chapman & Hall, Boca Raton, 1993.

[96] A.Webb, Statistical Pattern Recognition, New York, NY: Oxford University Press Inc., 1999.

[97] J. Friedman, T. Hastie, and R. Tibshirani. "Additive logistic regression: A statistical view of boosting." Annals of Statistics, Vol. 28, No. 2, pp. 337–407, 2000.

[98] A. Demiriz, K. P. Bennett and J. Shawe-Taylor, "Linear programming boosting via column generation," Mach. Learning, vol. 46, pp. 225-254, 2002.

[99] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning, second edition. Springer, New York, 2008

[100] M. Hristakeva, "Different Boosting Algorithms and Underlying Optimization Problems," 2008.

[101] Y. Freund, "Boosting a Weak Learning Algorithm by Majority," Information and Computation, vol. 121, pp. 256, 1995.

[102] Y. Freund, "An adaptive version of the boost by majority algorithm," in Proceedings of the Twelfth Annual Conference on Computational Learning Theory, 2000, pp. 102-113.

[103] C. Domingo and O. Watanabe, "MadaBoost: A modification of AdaBoost," in Proc. of the Thirteenth Annual Conference on Computational Learning Theory, 2000, pp. 180-189.

[104] MATLAB version 7.14.0. Natick, Massachusetts: The MathWorks Inc., 2012.

[105] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and, vol. 42, pp. 463, July, 2012.

[106] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in Icml, 1997, pp. 211-218.

[107] G. Martinez-Muoz, D. Hernández-Lobato and A. Suárez, "An analysis of ensemble pruning techniques based on ordered aggregation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 31, pp. 245-259, 2009.

[108] CC. De Stefano, G. Folino, F. Fontanella and A. Scotto di Freca, "Using bayesian networks for selecting classifiers in GP ensembles," in Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, 2011, pp. 173-174.

[109] G. Martínez-Muñoz and A. Suárez, "Using boosting to prune bagging ensembles," Pattern Recog. Lett., vol. 28, pp. 156-165, 2007.

[110] B. D. Ripley, Pattern Recognition and Neural Networks. Cambridge university press, 2007.

[111] J. Meynet and J. Thiran, "Information theoretic combination of pattern classifiers," Pattern Recognit, vol. 43, pp. 3412-3421, 2010.

[112] W. Hoff and N. Ahuja, "Surfaces from stereo: integrating feature matching, disparity estimation, and contour detection," Pattern Analysis and Machine Intelligence, IEEE Transactions, vol.11,pp. 121-136, 1989.

[113] G. Pajares, J. M. Cruz and J. A. LÃpez-Orozco, "Improving stereovision matching through supervised learning," Pattern Analysis and Applications, vol. 1, pp. 105-120, 1998.

[114] G. Pajares and J. M. de la Cruz, "Stereovision matching through support vector machines," Pattern Recog. Lett., vol. 24,pp. 2575-2583, 11, 2003.

[115] T. Sun, "Stereo Matching Using Synchronous Hopfield Neural Network," vol. 24, pp. 336-347, 2009.

[116] M. S. Lew, T. S. Huang and K. Wong, "Learning and feature selection in stereo matching," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 16, pp. 869-881, 1994.

[117] H. Saito and M. Mori, "Application of genetic algorithms to stereo matching of images," Pattern Recog. Lett., vol. 16, pp. 815-821, 8, 1995.

[118] M. Chessa, V. Bianchi, M. Zampetti, S. P. Sabatini and F. Solari, "Real-time simulation of large-scale neural architectures for visual features computation based on GPU," Network: Comput. Neural Syst., vol. 23, pp. 272-291, 2012.

[119] L. Nalpantidis and A. Gasteratos, "Biologically and psychophysically inspired adaptive support weights algorithm for stereo correspondence," Robotics and Autonomous Systems, vol. 58, pp. 457-464, 2010.

[120] W. Wang and S. Goto, "Stereo matching with pixel classification and reliable disparity propagation," in Circuits and Systems (ISCAS), 2012 IEEE International Symposium on, 2012, pp. 1891-1894.

[121] Ke Zhang, Jiangbo Lu and G. Lafruit, "Cross-Based Local Stereo Matching Using Orthogonal Integral Images," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 19, pp. 1073-1079, 2009.

[122] Jiang Ze-tao, Zheng Bi-na, Wu Min and Chen Zhong-xiang, "A 3D reconstruction method based on images dense stereo matching," in Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, 2009, pp. 319-323.

[123] H. Kim, Seung-jun Yang and Kwanghoon Sohn, "3D reconstruction of stereo images for interaction between real and virtual worlds," in Mixed and Augmented Reality, 2003. Proceedings. the Second IEEE and ACM International Symposium on, 2003, pp. 169-176.

[124] S. Se and P. Jasiobedzki, "Photo-realistic 3D model reconstruction," in Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, 2006, pp. 3076-3082.

[125] Unsang Park and Anil K. Jain, "3D face reconstruction from stereo video," in Computer and Robot Vision, 2006. the 3rd Canadian Conference on, 2006, pp. 41-41.

[126] M. S. Hossain, M. Akbar and J. D. Starkey, "Inexpensive construction of a 3D face model from stereo images," in Computer and Information Technology, 2007. Iccit 2007. 10th International Conference on, 2007, pp. 1-6.

[127] R. Fransens, C. Strecha and L. Van Gool, "Parametric Stereo for Multi-pose Face Recognition and 3D-Face Modeling," vol. 3723, pp. 109-124, 2005.

[128] H. Rara, S. Elhabian, A. Ali, M. Miller, T. Starr and A. Farag, "Face recognition at-a-distance based on sparse-stereo reconstruction," in Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on, 2009, pp. 27-32.

[129] Nvidia, "Nvidia Cuda Programming Guide 3.2," 2010.

[130] K. Pauwels, M. Tomasi, J. Diaz Alonso, E. Ros and M. Van Hulle, "A Comparison of FPGA and GPU for Real-Time Phase-based Optical Flow, Stereo, and Local Image Features," Computers, IEEE Transactions on, vol. PP, pp. 1-1, 2011.

[131] R. Kalarot and J. Morris, "Comparison of FPGA and GPU implementations of real-time stereo vision," in Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, 2010, pp. 9-15.

[132] X. Mei, X. Sun, M. Zhou, S. Jiao, H. 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, 2011, pp. 467-474.

[133] Zhu, Ke. Butenuth, Matthias. Angelo, Pablo., "Comparison of dense stereo using CUDA," in ECCV'2010 Workshop on Computer Vision on GPUs (CVGPU2010), Greece, 2010.

[134] In Kyu Park, N. Singhal, Man Hee Lee and Sungdae Cho, "Efficient design and implementation of visual computing algorithms on the GPU," in Image Processing (ICIP), 2009 16th IEEE International Conference on, 2009, pp. 2321-2324.

[135] Jiangbo Lu, Ke Zhang, G. Lafruit and F. Catthoor, "Real-time stereo matching: A cross-based local approach," in Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on, 2009, pp. 733-736.

[136] K. Zhang, J. Lu, Q. Yang, G. Lafruit, R. Lauwereins and L. Van Gool, "Real-time and accurate stereo: a scalable approach with bitwise fast voting on CUDA," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 21, pp. 867-878, 2011.

[137] W. Yu, T. Chen, F. Franchetti and J. C. Hoe, "High performance stereo vision designed for massively data parallel platforms," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 20, pp. 1509-1519, 2010.

[138] J. Kowalczuk, E. T. Psota and L. C. Pérez, "Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 23, pp. 94-104, 2013.

[139] K. Zhu, M. Butenuth and P. d'Angelo, "Comparison of dense stereo using CUDA," in Trends and Topics in Computer Vision, Springer, 2012, pp. 398-410.

[140] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal and P. Dubey, "Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU," in Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 2010, pp. 451-460.

[141] P. Bui and J. Brockman, "Performance Analysis of Accelerated Image Registration Using GPGPU," Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, pp. 38-45 2009.

[142] M. Baydoun, and M. A. Al-Alaoui, "Parallel Edge Detection Based on Digital Differentiator Approximation", ICCIT 2013, Beirut, Lebanon, June 19-21, 2013.

[143] M. A. Al-Alaoui, "Novel Digital Integrator and Differentiator"; IEE Electronics Letters, Vol. 29, No. 4, pp. 376-378, 18 February 1993.

[144] M. A. Al-Alaoui, "Filling the Gap Between the Bilinear and the Backward Difference Transforms: An Interactive Design Approach"; International Journal of Electrical Engineering Education. Vol. 34, No. 4, pp. 331-337, October 1997.

[145] M. A. Al-Alaoui, "Al-Alaoui Operator and the New Transformation Polynomials for Discretization of Analogue Systems", Electrical Engineering, Springer Berlin/Heidelberg, Vol. 90, Number 6, pp. 455-467, June 2008.

[146] V. Radisavljevic, "An Alternative Derivation of the Al-Alaoui Operator", Signal Processing Letters, IEEE Volume:15, pp. 881 – 882, 2008

[147] F. Auger, " Some New Developments on the Al-Alaoui and the Pei and Hsu s-to-z Transforms", Circuits and Systems II: Express Briefs, IEEE Transactions, Volume: 57 , Issue: 6 , pp.471 – 475, 2010 ,

[148] Jack E. Bresenham, "Algorithm for computer control of a digital plotter", IBM Systems Journal, Vol. 4, No.1, January 1965, pp. 25–30

[149] T. Arici, S. Dikbas and Y. Altunbasak, "A histogram modification framework and its application for image contrast enhancement," Image Processing, IEEE Transactions on, vol. 18, pp. 1921-1935, 2009.

[150] M. Baydoun, and M. A. Al-Alaoui, “”Enhancing Stereo Matching with VaryingIllumination Through Histogram Information and Normalized Cross Correlation”, IWSSIP, Bucharest,Romania, July 7-9, 20134.

[151] Qingxiong Yang, Ruigang Yang, J. Davis and D. Nister, "Spatial-depth super resolution for range images," in Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, 2007, pp. 1-8.

[152] A. Hosni, C. Rhemann, M. Bleyer, C. Rother and M. Gelautz, "Fast Cost-Volume Filtering for Visual Correspondence and Beyond," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, pp. 504-511, 2013.

[153] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz and C. Rother, "Real time local stereo matching using guided image filtering," in Multimedia and Expo (ICME), IEEE International Conference on, 2011, pp. 1-6.

[154] Won-Hee Lee, Yumi Kim and Jong Beom Ra, "Efficient stereo matching based on a new confidence metric," in Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, 2012, pp. 1139-1143.

[155] Gupta, S.; Castleman, K. R.; Markey, M. K.; Bovik, A. C., "Texas 3D Face Recognition Database," Image Analysis & Interpretation (SSIAI), 2010 IEEE Southwest Symposium on, pp.97-100, 23-25 May 2010, Austin, TX.

[156] V. Podlozhnyuk, “Image Convolution with CUDA”, NVIDIA, June 2007.

[157] http://webfea.fea.aub.edu.lb/dsaf/CUDACodeFilter.htm

[158] J. Stam, “Stereo Imaging with CUDA”, NVIDIA, January 2008.

[159] V. Podlozhnyuk, “Histogram calculation in CUDA”, NVIDIA, November, 2007.

[160] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, “Machine Learning, Neural and Statistical Classification”.Ellis Horwood Limited, London, 1994.

[161] C. Blake and C. J. Merz, "{UCI} Repository of machine learning databases," 1998.

[162] T. K. Ho and E. M. Kleinberg, "Building projectable classifiers of arbitrary complexity," in Pattern Recognition, 1996., Proceedings of the 13th International Conference on, 1996, pp. 880-885.

[163] NASA/WVU IV&V Facility, Metrics data program. [Online]. Available: http://mdp.ivv.nasa.gov

[164] S. Haykin, "A comprehensive foundation," Neural Networks, vol. 2, 2004.

[165] D. W. Aha, P. Clark, S. Salzberg and G. Blix, "Incremental constructive induction: An instance-based approach," 1991.

[166] Journal of Statistics Education. [Online]. Available: http://www.amstat.org/publications/jse/jse_data_archive.htm

[167] L. Breiman, "Bias, variance, and arcing classifiers," 1996.

[168] M. Bleyer, C. Rother, P. Kohli, D. Scharstein and S. Sinha, "Object stereo—joint stereo matching and object segmentation," in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, 2011, pp. 3081-3088.

[169] M. Baydoun, and M. A. Al-Alaoui, "Enhancing Stereo Matching with Classification", Access, IEEE, vol. 2, pp. 485-499, 2014.