CUDA Implementation of Disparity Map Computation using Block Matching and Sum of Hamming Distances

Akshay Hodigere Arunkumar
Rutvij Girish Karkhanis

Final Project Report
ECEN 5593 Advanced Computer Architecture
May 12, 2014

CONTENTS

1 INTRODUCTION

This project proposes a modified block matching algorithm for generating disparity map of stereoscopic images based on a feature matching technique, using adaptive threshold for achieving it. In this algorithm, the disparity values are calculated by comparing set of pixels in left image with different sets of pixels in the same row of pixels in the right image.

The disparity value of an image feature is taken as the magnitude of difference between the positions of the matched feature in each image. The computation being non-intrusive between each other, can easily be implemented on a parallel computing environment like the NVIDIA CUDA based GPUs.

The project also involves implementation of Sum of Hamming Distances technique in Image feature comparison and matching for the formulation of disparity map. CUDA Implementation of this algorithm is used for the computation.

The project in total involves, implementation of the two algorithms on MATLAB, C platform (CPU Computation), and CUDA (GPU Computation), and comparison of the performance in each of them.

1.1 Motivation

We, Humans have the capacity to perceive depth in our field of vision because of the fact that we have two sources of vision (i.e two eyes), which is used by the brain to frame a 3 Dimensional perception of the environment in the field of vision. The field of study which attempts to replicate this perception of depth is known as stereoscopy. This involves having two dimensional images using camera arrangement analogous to the human visual system.

The intricate differences in the two images contribute to the depth information of the objects in the given pair of images. If the cameras are arranged for side by side stereoscopy the features that encompass the objects have a shift in their positions in the left to right direction or vice versa depending on the choice of reference image.

To extend the ability of humans in which one can perceive depth, to a computing environment requires development of algorithms to map the intricate differences in the stereoscopic image set. This project proposes an algorithm to compute this map of differences, known as the disparity map using NVIDIA CUDA technology, inclusive of a technique to incorporate subtle differences in intensities of various features in the images.

2.1 Block diagram

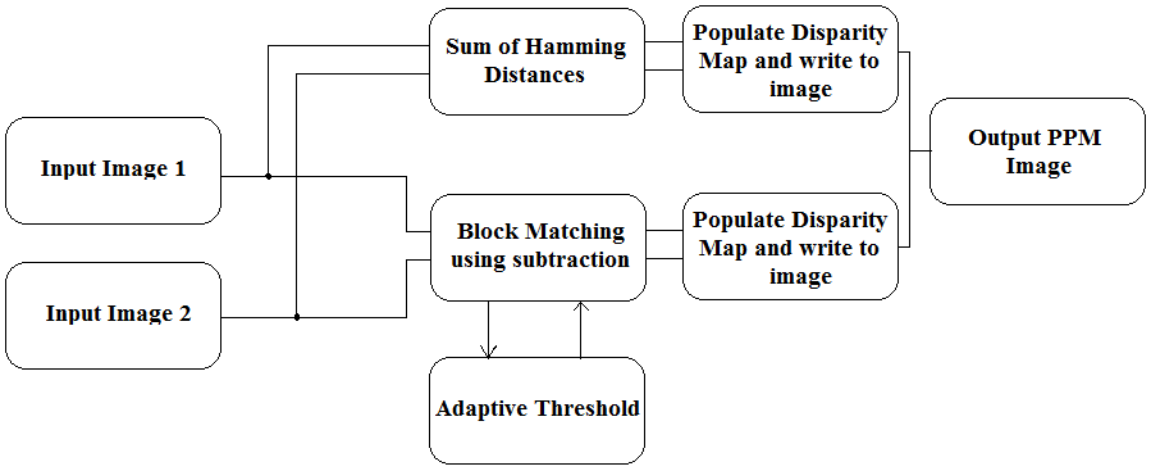


Figure: 1 High level overview

2 BLOCK MATCHING ALGORITHM

2.1 Overview and Steps Involved

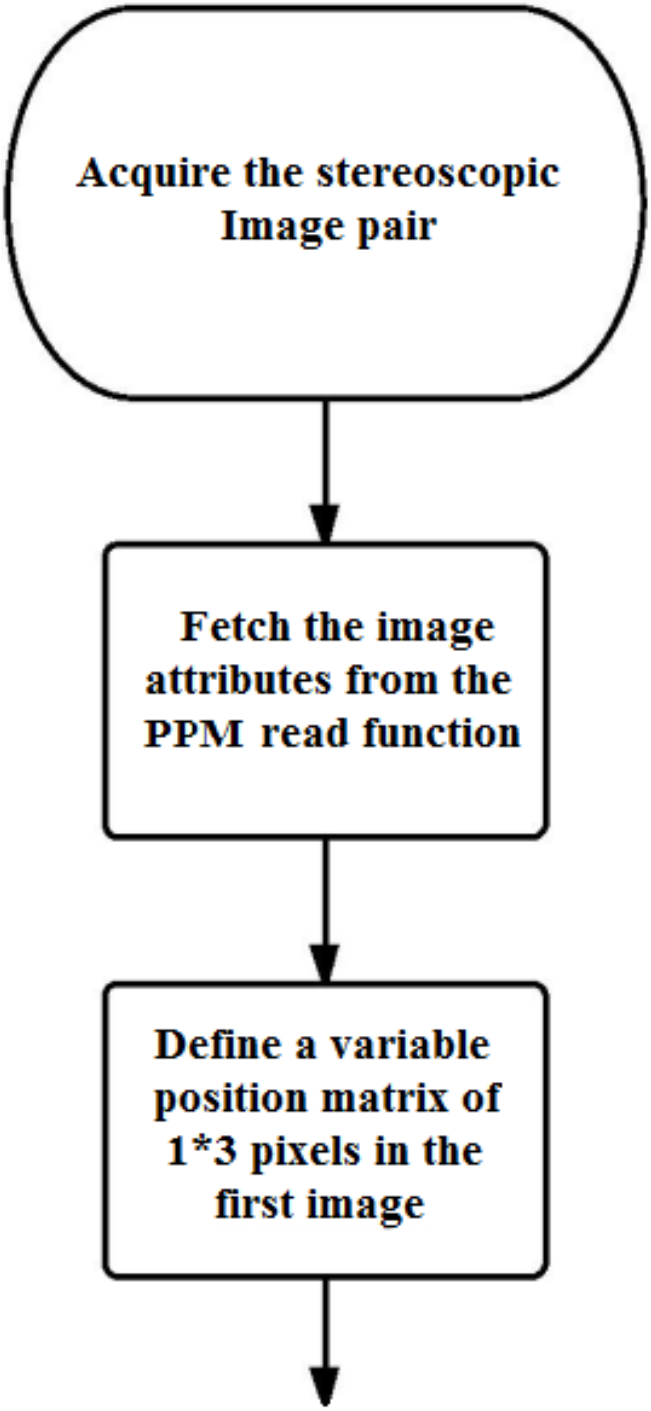
The Algorithm iterates through the following steps:

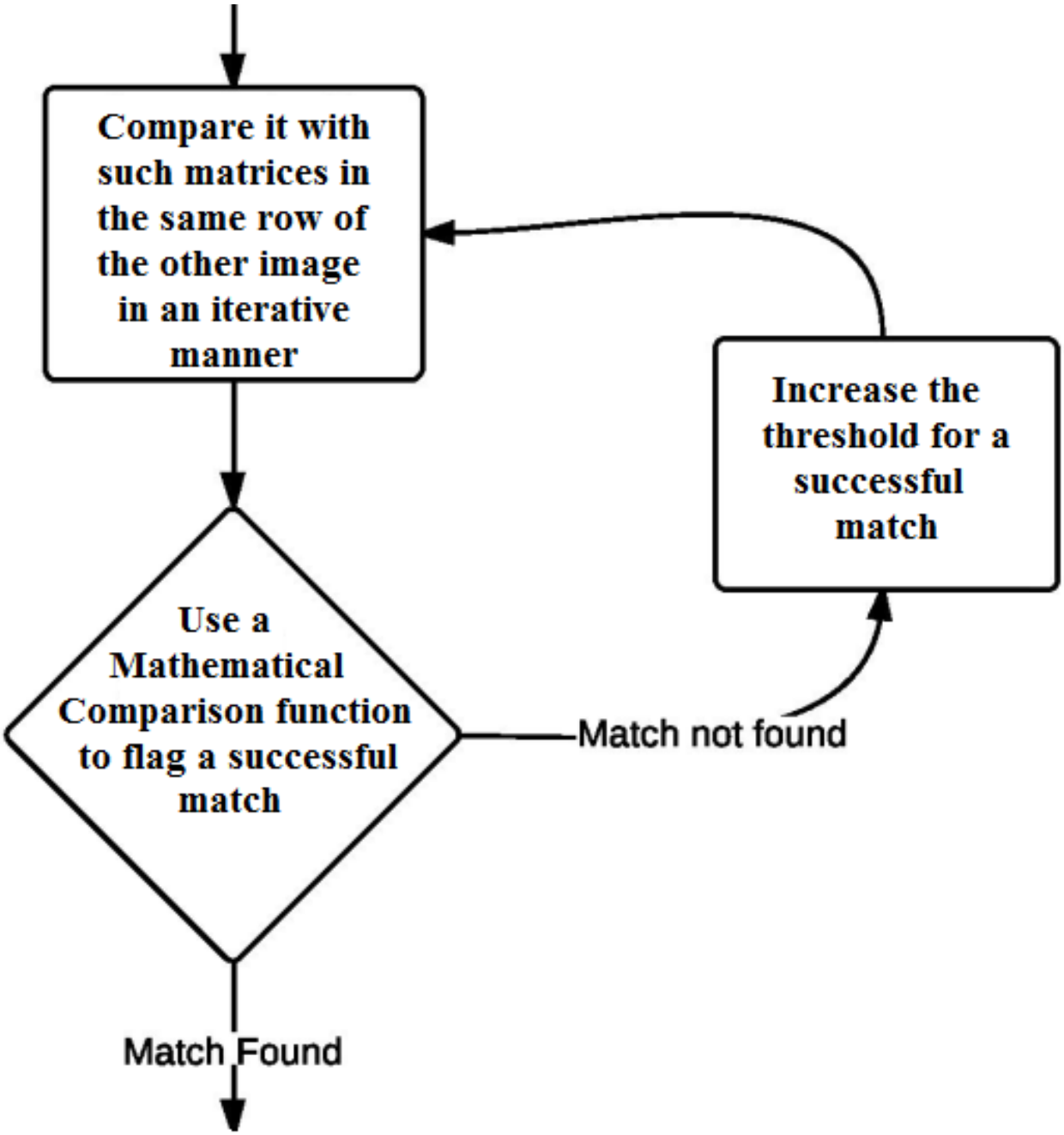
1. Acquire the stereoscopic image pair i.e. left and right image.
2. Define a matrix of 1x3 pixels in both the images.
3. Consider one row and consider a feature in the image defined by the matrix, A Mathematical comparison function such as subtraction is performed and its corresponding match is found in the other image.
4. If a match is not found then the strictness of the threshold is reduced and the effort is made to find the match in the other image of the stereoscopic pair.
5. Based on the apparent shift in the feature corresponding to the pixels where the match has been found the disparity is said to be computed.
6. Features which are would closer have moved more and features which are farther would have move less.
7. Disparity map image is created based on the information obtained in the previous step.

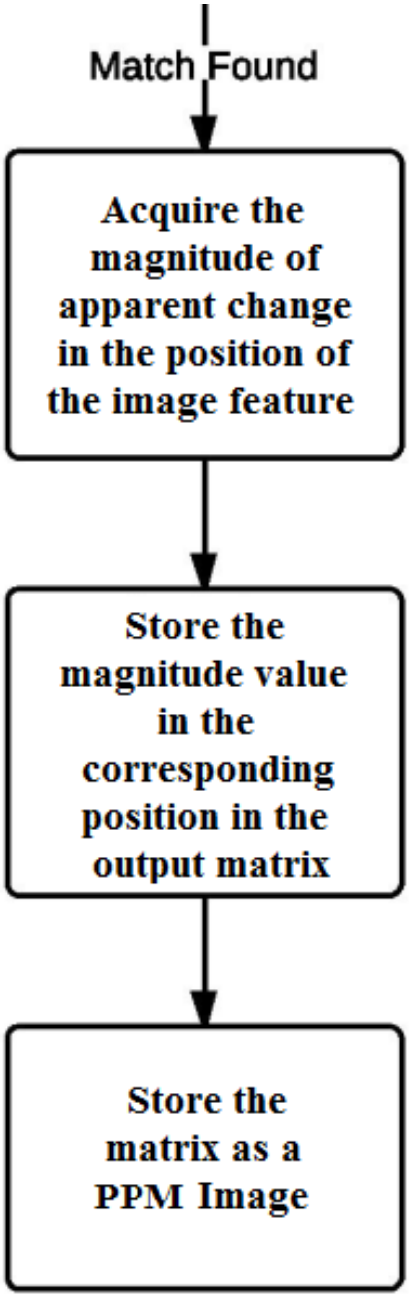
The algorithm is implemented on MATLAB, C code for CPU and CUDA Code for GPU. We have developed this algorithm based on the concept of block matching, with the addition of a feature for varying threshold for detection flag.

The MATLAB Code takes approximately 10 seconds to run this algorithm on an Intel i3 based computer with 6 GB RAM running Windows 8.1, Which is not comparable to CPU or GPU Execution time.

2.2 Flow Chart




Compare it with such matrices in the same row of the other image in an iterative manner
Use a Mathematical Comparison function to flag a successful match
Match not found
Increase the threshold for a successful match
Match Found



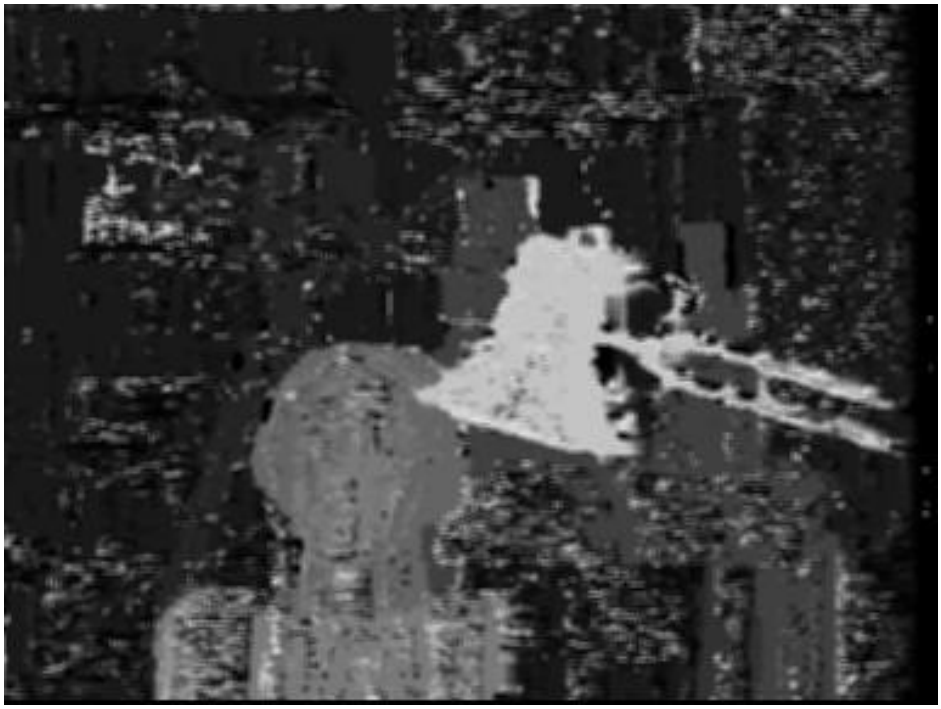
The Above steps are similar in the Hamming Distances algorithm, with the exception of using adaptive threshold, which is not required for it, and that the comparison mechanism used is to compute the sum of hamming distances between the comparison matrices

2.3 Disparity Map

The following are some of the Disparity Maps obtained using the algorithm on GPU

Image Set 1:

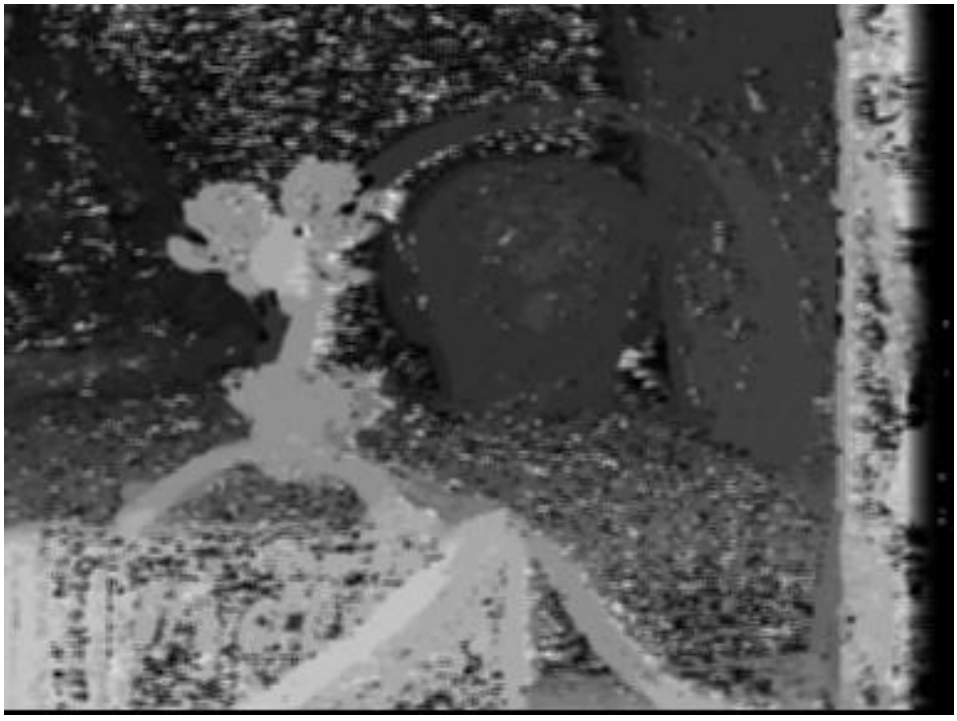
Input Images [2][3]



Output Image

Image Set 2:

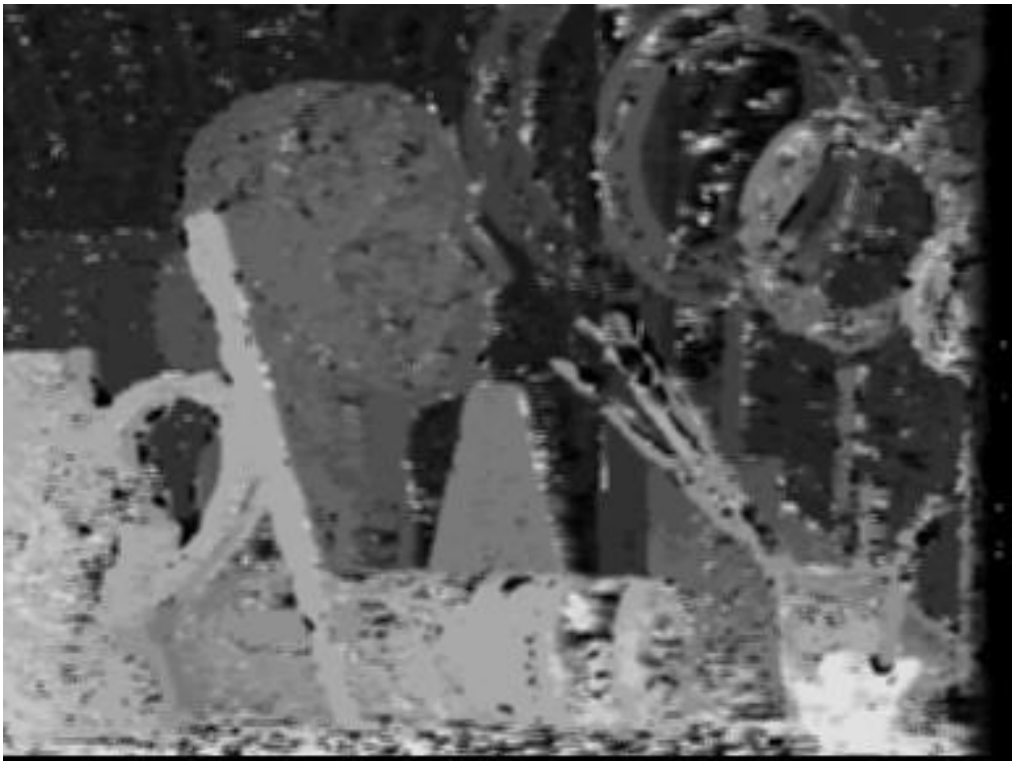
Input Images [2][3]



Output Image

Image Set 3:

Input Images [2][3]



Output Image

Note: Image results of MATLAB, CPU and GPU Implementation for the entire image set is included in the file submission.

3 SUM OF HAMMING DISTANCES ALGORITHM

3.1 Overview and Steps involved

The core equation for the feature matching is the sum of hamming distances:

$$\sum_{(i,j)\in W} I_1(i,j)\ bitwiseXOR\ I_2(x+i,y+j) \quad [1]$$

The Algorithm iterates through the following steps, which is similar to the first algorithm:

1. Acquire the stereoscopic image pair i.e. left and right image.
2. Define a matrix of 1x5 pixels in both the images and acquire them in each iteration.
3. Consider one row and consider a feature in the image defined by the matrix, A Mathematical comparison function such as bitwise XOR operation is performed and its corresponding match is found in the other image by matching the hamming distance between the pixel sets.
4. Based on the apparent shift in the feature corresponding to the pixels where the match has been found the disparity is said to be computed.
5. Features which are would closer have moved more and features which are farther would have move less.
6. Disparity map image is created based on the information obtained in the previous step.

The MATLAB Code takes approximately 90 seconds to run this algorithm on an Intel i3 based computer with 6 GB RAM running Windows 8.1, Which is not comparable to CPU or GPU Execution time..

The MATLAB Code used is a modified version of code by Siddhant Ahuja [1]. We have implemented CPU Code and CUDA Code based on it.

3.2 Disparity Map

The following are some of the Disparity Maps obtained using the algorithm on GPU

Image Set 1:

Input Images [2][3]

Output Image

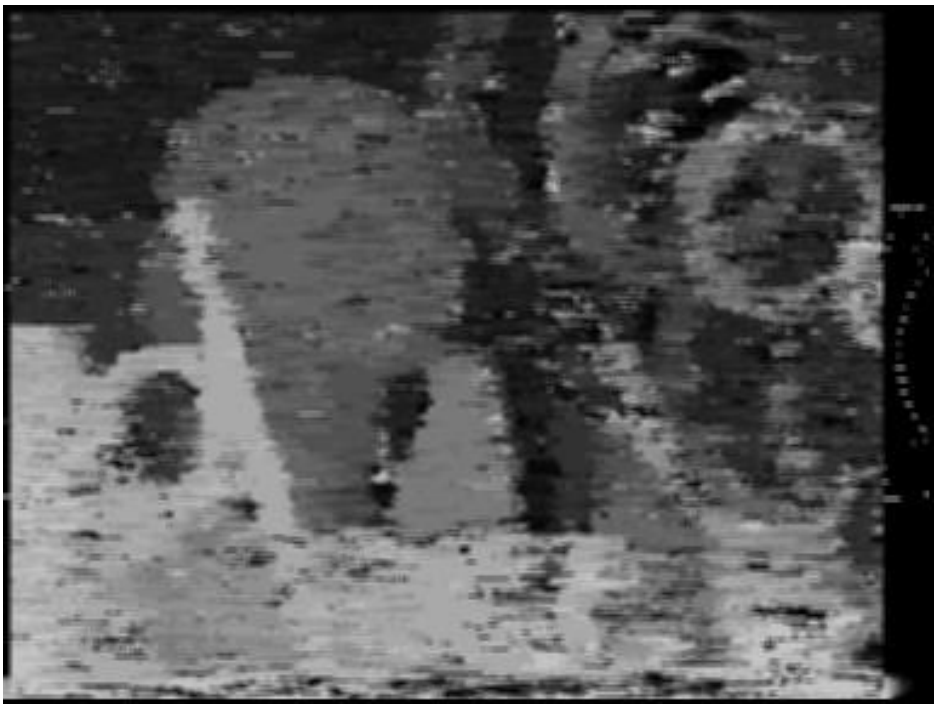
Image Set 2:

Input Images [2][3]

Output Image

Image Set 3:

Input Images [2][3]



Output Image

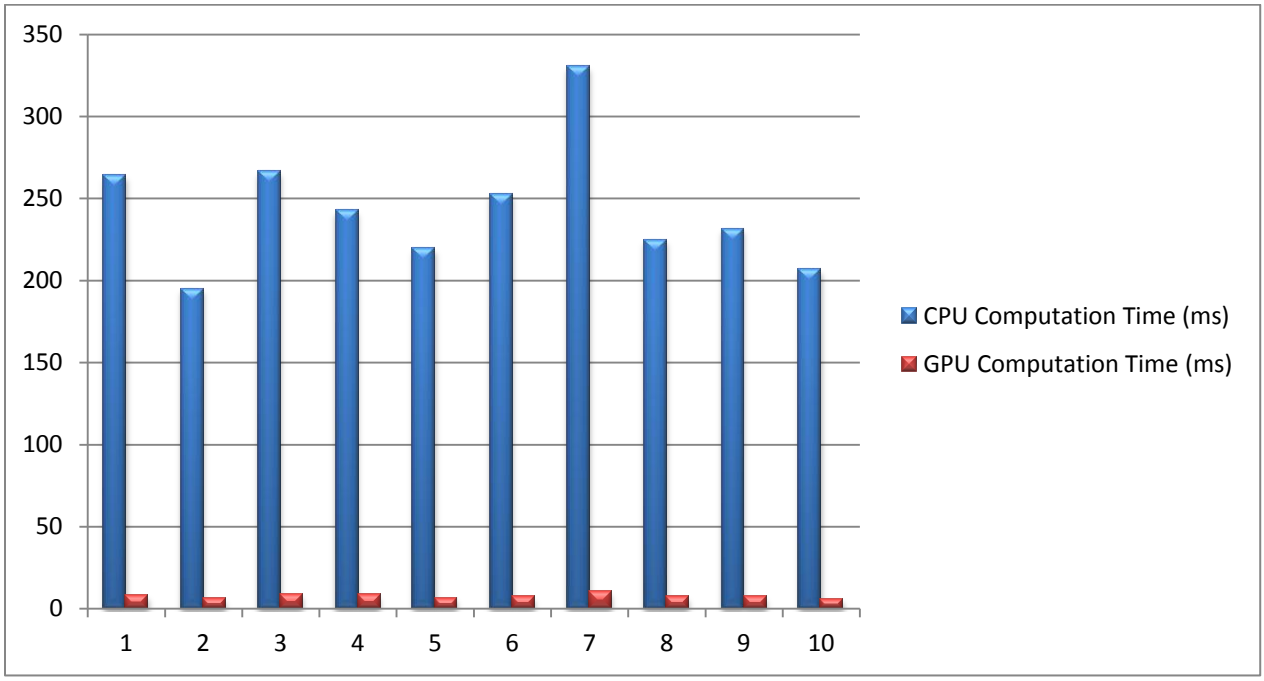
Note: Image results of MATLAB[1], CPU and GPU Implementation for the entire image set is included in the file submission.

4 RESULTS

4.1 Result Metrics

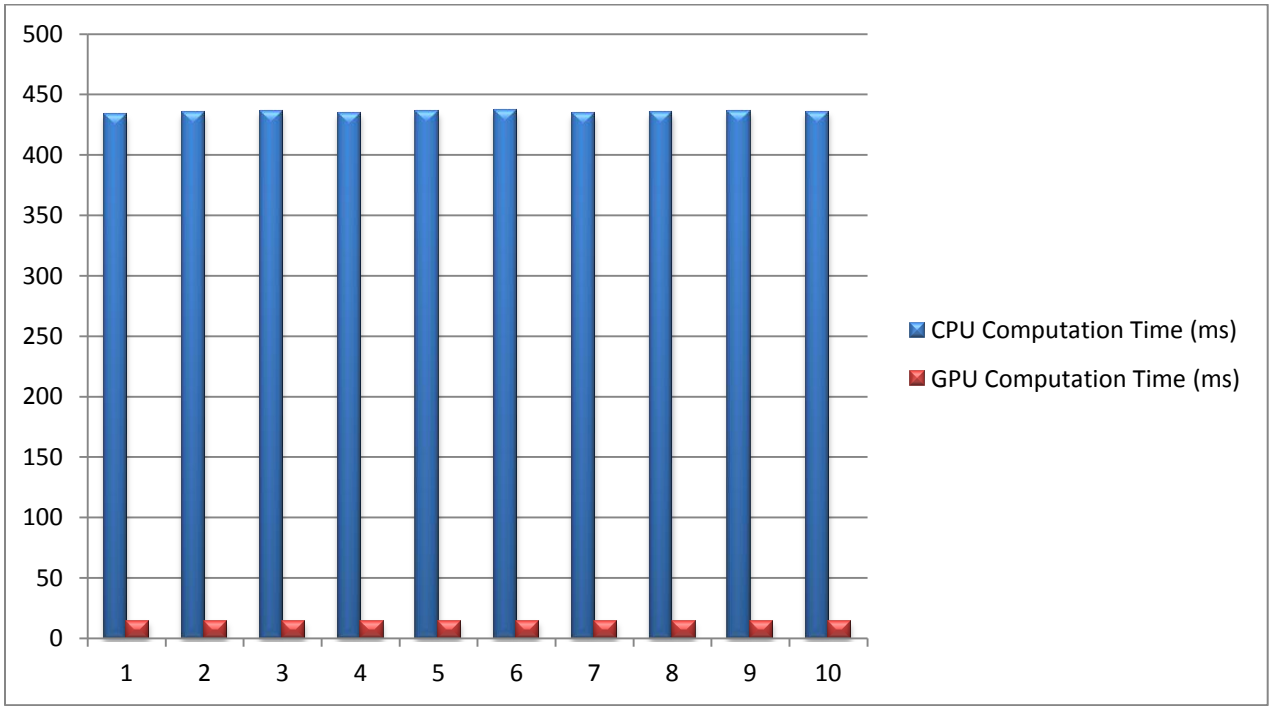
Block Matching with Adaptive Threshold:

Image Set	CPU Computation Time (ms)	GPU Computation Time (ms)	GPU Memory Operation Time (MS)	Speedup	Reduction in sequential overhead(%)
1	263.606995	8.201000	0.343000	32.143272	96.888931
2	194.972000	6.268000	0.298000	31.105934	96.785179
3	266.542999	8.999000	0.298000	29.619181	96.623817
4	242.531998	8.498000	0.297000	28.539890	96.496132
5	219.608994	6.453000	0.300000	34.032078	97.061592
6	252.757004	7.553000	0.298000	33.464451	97.011757
7	330.302002	10.510000	0.300000	31.427402	96.818062
8	224.626999	7.355000	0.298000	30.540720	96.725685
9	231.192001	7.652000	0.311000	30.213278	96.690201
a	206.330002	5.863000	0.295000	35.191883	97.158432



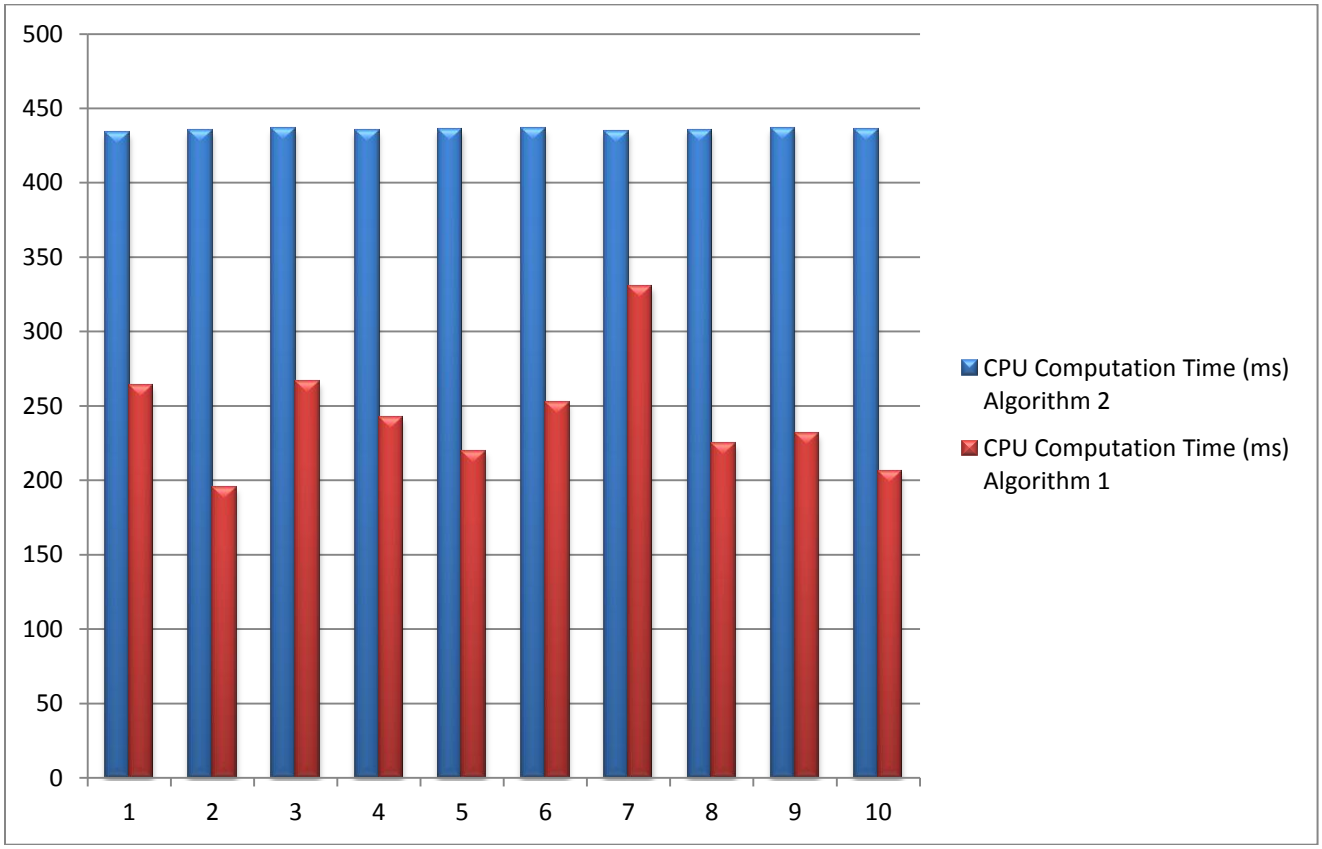
Sum of Hamming Distances:

Image Set	CPU Computation Time (ms)	GPU Computation Time (ms)	GPU Memory Operation Time (MS)	Speedup	Reduction in sequential overhead(%)
1	433.764008	14.240000	0.295000	30.460957	96.717110
2	434.984009	14.260000	0.317000	30.503786	96.721718
3	436.436005	14.251000	0.297000	30.624939	96.734688
4	434.843994	14.277000	0.294000	30.457659	96.716751
5	435.984985	14.282000	0.381000	30.526886	96.724197
6	436.700989	14.253000	0.298000	30.639233	96.736214
7	434.584991	14.303000	0.294000	30.384184	96.708809
8	435.341003	14.249000	0.294000	30.552391	96.726936
9	436.406006	14.272000	0.295000	30.577774	96.729645
a	435.648010	14.275000	0.295000	30.518250	96.723274



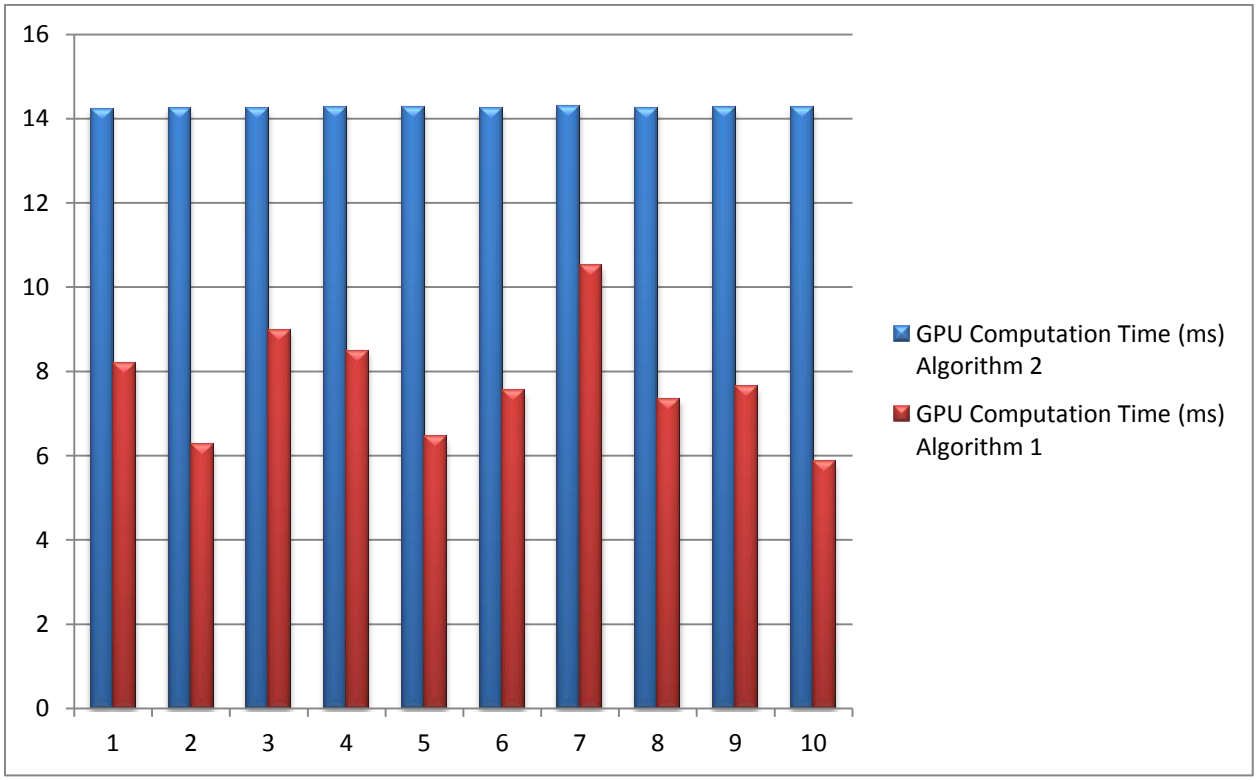
Comparison Between CPU Computation time of the two algorithms:

Image Set	CPU Computation Time (ms) Algorithm 2	CPU Computation Time (ms) Algorithm 1
1	433.764008	263.606995
2	434.984009	194.972000
3	436.436005	266.542999
4	434.843994	242.531998
5	435.984985	219.608994
6	436.700989	252.757004
7	434.584991	330.302002
8	435.341003	224.626999
9	436.406006	231.192001
a	435.648010	206.330002



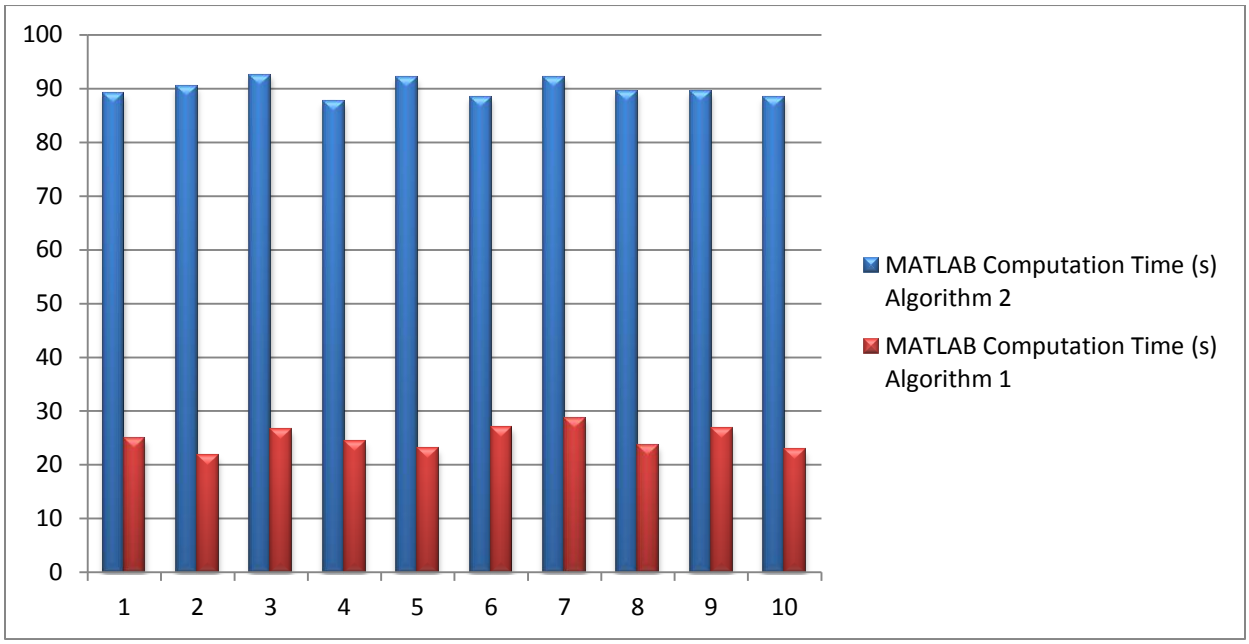
Comparison Between GPU Computation time of the two algorithms:

Image Set	GPU Computation Time (ms) Algorithm 2	GPU Computation Time (ms) Algorithm 1
1	14.240000	8.201000
2	14.260000	6.268000
3	14.251000	8.999000
4	14.277000	8.498000
5	14.282000	6.453000
6	14.253000	7.553000
7	14.303000	10.510000
8	14.249000	7.355000
9	14.272000	7.652000
a	14.275000	5.863000



Comparison Between MATLAB Computation time of the two algorithms:

Image Set	MATLAB Computation Time (s) Algorithm 2	MATLAB Computation Time (s) Algorithm 1
1	89.1340	24.9352
2	90.2889	21.6839
3	92.5234	26.5495
4	87.5646	24.3204
5	92.0614	22.9816
6	88.3860	26.9277
7	92.1565	28.6899
8	89.4270	23.6172
9	89.3614	26.6705
a	88.3694	22.8051



4.2 Scope for Improvement

There is scope for improvement in both the algorithms in the following ways

1. Both the algorithms can be improved further to obtain a balance between output quality and computation time.
2. The SHD Algorithm can be improved by using rectangular windows of various aspect ratios to obtain a more accurate result.
3. The Block Matching Algorithm can be improved by using a comparison mechanism similar to subtraction, but which provides a more accurate result.
4. MATLAB Code can be optimized by using C function calls which perform the core computation.
5. CPU/GPU Implementation can be improved by providing option of using various image formats like BMP/JPG etc.
6. CPU/GPU and MATLAB Implementation can be improved by providing option of using images of different resolutions for the disparity map computation.

5 CHALLENGES AND ISSUES ENCOUNTERED

We encountered the following challenges and issues during the project:

1. We encountered issues with passing two dimensional arrays (Image data) to the GPU Kernel using pointers. We resolved it by converting the image data into a single dimensional matrix and passing the pointer of the same to the kernel.

2. During the GPU Computation of the Sum of Hamming Distances algorithm, The output of the algorithm was found to be distorted because of concurrent access of the image pixels in the same region and different rows by multiple threads. We resolved it by modifying the algorithm such that each thread has access to one particular row.

3. We found it challenging to port the algorithm from MATLAB to C and also to optimize the speedup for both the algorithms on GPU by implementing various combinations of blocks and thread assignments .

6 ACKNOWLEDGEMENTS

We would like to sincerely thank Prof. Connors for offering the excellent course on Advanced Computer Architecture. We would like to thank authors of various sources that we have utilized for the success of this project. We thank the ECEE department for providing servers with GPU access essential for the coursework and project.

7 REFERENCES

[1] https://siddhantahuja.wordpress.com/tag/matlab-code/
[2] http://vision.middlebury.edu/stereo/data/scenes2005/
[3] http://vision.middlebury.edu/stereo/data/scenes2001/
[4] http://stackoverflow.com/questions/2693631/read-ppm-file-and-store-it-in-an-array-coded-with-c
[5] http://www.ravenousbirds.com/eolson/papers/stereovision/stereovisionpaper.pdf
[6] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002.
[7] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007.
[8] http://docs.nvidia.com/cuda/cuda-c-programming-guide/#abstract
[9] http://www.cs.unc.edu/~prins/Classes/633/Readings/CUDA_C_Programming_Guide_4.2.pdf

9 APPENDIX

9.1 Project files inclusive of code

All the project files inclusive of code, test images, output images of MATLAB, CPU and GPU Implementation for the entire image set are included in the following Dropbox folder.

Link: https://www.dropbox.com/sh/pktayvikg79ro6x/AADWCCHc33KB_e8j3SrB5sRQa