Real-Time Semi-Global Matching Using CUDA Implementation

Robert Mahieu
Stanford University
Department of Electrical Engineering
rmahieu@stanford.edu

Michael Lowney
Stanford University
Department of Electrical Engineering
mlowney@stanford.edu

***Abstract*—With recent rise of technology such as augmented reality and autonomous vehicles, there comes a necessity for speedy and accurate depth estimation to allow these products to effectively interact with their environments. Previous work using local methods to produce depth maps has generally been fast but inaccurate and work using global methods has been accurate but too slow. A new technique referred to as semi-global matching combines local and global methodologies to balance speed and accuracy, producing particularly useful results. This project focuses on the implementation of slight variations on the original algorithm set forth by Hirschmuller [2] to increase accuracy and using CUDA to accelerate the runtime. Results show sufficiently low error, though runtime was found to be imperfectly optimized.**

1. Introduction

Acquiring depth information from sets of images is incredibly important in many emerging fields such as augmented reality, robotics, autonomous vehicles, etc. However, these applications rely on the produced depth information to be both accurate and generated in a short amount of time—ideally close to real-time—to ensure safety of the system and users, as well as for reliable pose tracking.

Many algorithms for computing this information have been previously explored, either looking at localized information or global information throughout the entire image. Local techniques such as Winner-Takes-All (WTA) or scanline optimization (SO) [7] compute results for pixels independently and require minimal computation, but due to lack of consideration for global trends typically result in inaccurate conclusions. The dynamic programming [5] approach is also computationally efficient, but because the algorithm only looks at a single row per iteration, it also lacks consideration for global trends and commonly causes streaking patterns to show up in the output. On the other hand, while global techniques such as a Graph Cuts [1] and Belief Propagation [9] produce more accurate results and better avoid the errors encountered in the local methods, these techniques are significantly more memory intensive and end up being much slower.

To obtain both reasonable accuracy as well as real-time performance, we instead move to what can be referred to as a semi-global matching technique [2], which makes some use of both local and global methods. Additionally, offloading data calculation onto the GPU, which is ideal for handling SIMD (single instruction multiple data) computation, allows us to exploit the parallelizable nature of our image-based calculations and significantly reduce computation time for estimating the optimal disparity map.

2. Problem Statement

In order to tackle this problem, we make the assumption that the input images are a rectified stereo pair. This is inherently the case when two cameras are orthogonal to the baseline, and point in the same direction. Mpopular stereo vision datasets, such as the Middlebury dataset, which is used in this paper [3][6][8], provide stereo pairs that have been rectified. The benefit to having rectified images is that the epipolar lines are horizontal and corresponding lines are at the same height in each image. This simplifies the problem because corresponding points will lie on the same epipolar line, and so we only have to search in horizontal directions.

Local methods are more prone to noise in their disparity maps due to the fact that there may be several local minima in their cost function. Because of this the semi global approach uses a model which penalizes changes in disparity values in local neighborhoods. This causes the resulting disparity map to be smoother by attenuating high frequency noise, which provides a clearer estimate of the true relative depth of the objects in the scene.

3. Technical Content

The implementation of the semi-global matching method comes down to minimizing an energy function describing the quality of a potential disparity image. This is represented by the expression below:

$$E(D) = \sum_p \left(C(p, D_p) + \sum_{q \in N_p} P_1 \mathbf{1}\{|D_p - D_q| = 1\} + \sum_{q \in N_p} P_2 \mathbf{1}\{|D_p - D_q| > 1\} \right)$$

Where D is the disparity map, p is a pixel location on the map, D_p is the disparity value at pixel p, N_p is the neighborhood of pixels around p, $\mathbf{1}\{?\}$ is an indicator function that is equal to one if the argument within the braces is true and zero if false, and P_1 and P_2 represent penalty value given to various changes in disparity within the local neighborhood. $C(p, d)$ represents an initial cost function which is based on the absolute difference between gray levels at a pixel p in the base reference image and pixel p shifted by d along the epipolar line in the matching image (assumed to be the right image in the stereo pair):

$$C(p, d) = |I_b(p_x, p_y) - I_m(p_x - d, p_y)|$$

Note also as stated in section 2 that the images are assumed to be rectified. The energy function thus penalizes heavily (with P_2) large jumps in the disparity map and less heavily (with P_1) small changes that may represent sloped surfaces. Note that $P_1 < P_2$. This allows us to reduce high frequency noise in the resulting disparity image.

Optimizing this energy function using global minimization in 2-dimensions is NP-complete, and requires too much computation to solve for many practical applications. On the other side of the spectrum, minimizing in 1-dimension over image rows, such as in the dynamic programming approach, is light on computation, but suffers from accuracy issues as discussed above. To handle this problem, semi-global matching leverages several 1-dimensional minimization functions to more efficiently construct an adequate estimate of the solution.

The first step in the actual implementation of the algorithm is to calculate initial costs for all pixels in the image pair at disparities ranging from 0 to some selected d_{max}. We found that using a d_{max} value of 64 was more than enough for all images we tested. This computation can be efficiently carried out on the GPU with a kernel that gives each thread a computation for one pixel and one disparity value. The system takes an RGB stereo pair as an input and then converts the image to grayscale. The cost function uses the difference in grayscale intensity values as a metric to determine how good or bad a potential match is. Once the images are converted to grayscale they are store in texture memory on the GPU to increase speed. Texture memory is cached on the chip and allows for much faster reads than global memory.

To carry out the next step denoted "cost aggregation", we iterate and compute the energy function locally over 8 directions (two horizontal, two vertical, two for each diagonal). An example for the recursive expression is shown below for the horizontal direction, going from left to right across the image:

$$\begin{aligned} E(p_x, p_y, d) = C(p, d) \\ + \min(&E(p_x - 1, p_y, d), \\ &E(p_x - 1, p_y, d - 1) + P_1, \\ &E(p_x - 1, p_y, d + 1) + P_1, \\ &\min_i(E(p_x - 1, p_y, i) + P_2)) \end{aligned}$$

This step can be parallelized by having a different block for each direction, and within each block having each thread handle one iteration for one disparity. Note that after each step along the direction, the threads must be synchronized. This can be done by utilizing the CUDA command `__syncthreads()`.

Once all paths have been traversed, results are compiled into a single value:

$$S(p, d) = \sum_r w_r * E_r(p, d)$$

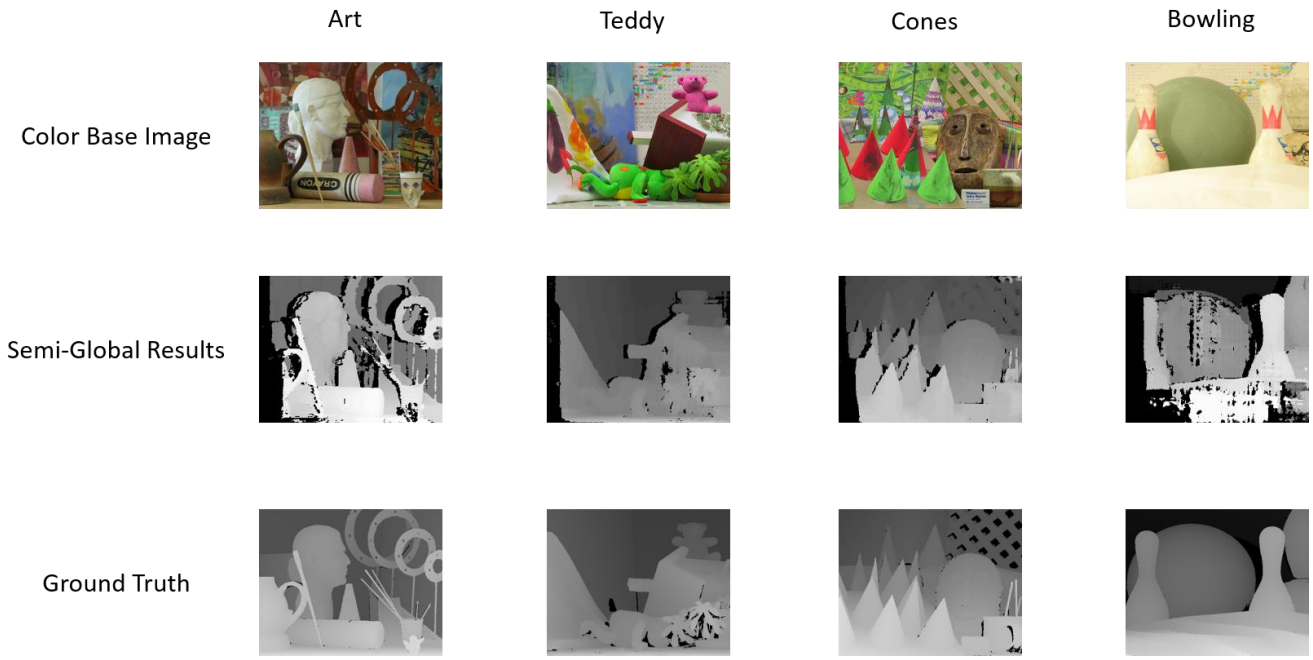
Where r represents a given direction and w_r represents a weight value for that particular direction. By introducing this weighting term, we account for the fact that results obtained from a certain path orientation may still lead to better estimates than the results from the other orientations. Therefore, we are able to weight each direction accordingly based on the scene type.

The final step is to, for each pixel, iterate over the calculated energy values and determine which disparity value corresponds to the minimum energy. This is represented by the following function:

$$D(p) = \underset{d}{\operatorname{argmin}}\, S(p, d)$$

This can be parallelized on the GPU by having each thread handle all disparities for a given pixel. The disparities returned from this step represent the optimal disparity map that minimizes the energy function.

Once we have a reasonable disparity map, the next step



is to determine areas of occlusion in the image, meaning areas that are visible in the base image, but blocked in the matching image. This can be done by running a slightly modified version of the algorithm as defined above again to generate a disparity map for the match image. The only modification is in the initial cost function which becomes:

$$C(p,d) = |I_m(p_x, p_y) - I_b(p_x + d, p_y)|$$

Once we have disparity maps for both the base and match image we can compare the results to identify occluded regions. For each pixel in the base disparity map we sample the disparity value. We then compare this to the value in the match disparity map at the same pixel location shifted by the base disparity value we just sampled. If these two values are the same (within some small tolerance), we judge them to be true correspondences, otherwise we make them as occluded pixel and set them to zero in the base disparity map. This technique outputs a refined base disparity map.

Finally, to eliminate residual noise in the output we filter the disparity map using a median filter. Good results were observed while using a small kernel of 3x3. This allows us to keep the major edges and details in the map while removing the unwanted high frequency components.

A consideration worth noting is the memory requirements of this algorithm. The amount of memory used scales like $O(mnd_{max})$, where m it the number of rows in the image, n is the number of columns in the image, and d_{max} is the number is the maximum number of disparity values. Storing the initial cost matrix and the matrices for the 8 search directions may exceed the total amount of memory on the GPU. For this paper the algorithm was run on a laptop with a NVIDIA GeForce 940M GPU with 2GB of memory. In order to stay below this 2GB threshold the input images must be down sampled. Unless otherwise specified the images will be down sampled so that the number of columns will be 450, and the number of rows will be scaled accordingly.

4. Results

The quality of our algorithm was tested using the Middleburry dataset. Figure 1 shows the results of our algorithm compared to the ground truth of the depth map for various image pairs. For these trials we used the values suggested in [4] for P_1 , P_2. All w_r were set to the same value to ensure equal weighting. Table 1 shows the values of P_1 and P_2 used for our results.

Table 1: Penalty Values

	↔	↕	↖↘	↙↗
P_1	22.02	17.75	14.93	10.67
P_2	82.79	80.87	23.30	28.80

Qualitatively the results appear to be quite a close match to the ground truth. The regions towards the left border of the image are consistently unlabeled. This is due to the fact that they represent pixels that are only seen in the base image. Only pixels that are in the field of view of both cameras will result in accurate disparity values.

Table 2: MSE and runtime for images in Midlleburry dataset

IMAGE PAIR	MSE	Runtime
aloe	0.0296	3715ms
books	0.0687	3394ms
dolls	0.0712	3385ms
laundry	0.0935	3517ms
pots	0.1053	3607ms
baby	0.0385	3759ms
bowling	0.1083	3692ms
art	0.0949	3385ms
cones	0.0409	3152ms
wood	0.0671	3403ms

Table 2 shows the mean-squared error (MSE) between our experimental depth maps and their respective ground truths. Note that the error values are generally quite low, never reaching any higher than around 10% for any of the images we tested. Although differences in scaling of depth to grayscale may be present between the experimental results and the ground truth, this appears to be minimal and therefore the MSE should still provide a good metric for analyzing the success of the algorithm.

While the performance of the algorithm seems to decrease in highly cluttered scenes such as the Art image pair or scenes with many strong gradients such as the Pots image pair, in general, variation in the scene structure appears to affect the quality of the produced depth map very little, both qualitatively and quantitatively.

These results also appear to indicate that our decision to weight the results from all path directions equally, as well as our decision to keep constant the penalty values throughout all tests, had relatively small impact on the quality of our depth maps. The results from Michael et al. [4] report that training these values for specific scenes does increase quality noticeably, however our results seem to show that it is still possible to get reasonably good results without worrying about changing these parameters for every scene.

Unfortunately, due to time constraints on the project, we were unable to spend much time optimizing the CUDA implementation, so tests on runtimes have returned sub-optimal results. In the current somewhat naïve implementation, we are still able to get runtime down to around one frame/sec for images with sizes below about 250x217 (54250 pixels). The relationship between runtime and input image size is illustrated in Figure 2. As shown in the figure, the Semi-Global Matching algorithm has a runtime that scales linearly with the number of pixels in the input images. It is also worth noting that the graphics cards used in modern stereo research are much more powerful than the ones used in this paper (consumer grade laptop GPUs). This difference in hardware is a main contributor to the longer runtimes found in this project.

It is important to note that one of the most significant factors in the runtime, however, is actually the resizing of images that occurs at the start of the program after being read in by the CPU. This is necessary to ensure that we do not overload the GPU memory, however the time cost is very high. When images do not need to be resized on-the-fly, total speeds are greatly increased (about 2x speedup). In future implementations, intelligent use of shared memory and memory access within warps should be able to dramatically increase performance. Some of these techniques are outlined by Michael et al. [4].

For posterity, to demonstrate the robustness of our algorithm we also selected an arbitrary stereo image pair

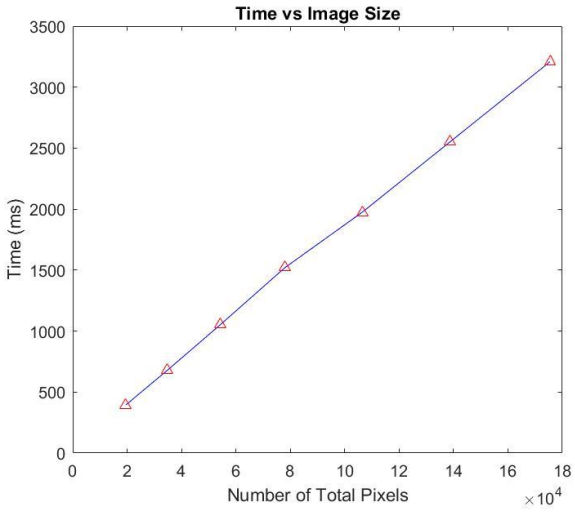


Figure 2- Runtime vs Image Size

from reddit (https://www.reddit.com/r/crossview) of a Batman figurine. The results are displayed in Figure 3. While some poorly defined occlusion areas create a good bit of distracting noise in the depth map image, the large non-occluded areas actually show quite a high amount of detail. Looking closely, we can even make out definition at

the level of the muscles on the figurine.

Figure 3- Batman robustness test

5. Conclusions

In this paper we demonstrate the effectiveness of the Semi-Global Matching technique for depth estimation from a set of stereo images. Leveraging a combination of global and local techniques our results exhibited low error when compared to ground truths. With more powerful hardware and optimized CUDA implementation, our initial results imply that this would be possible to run in real time. This has been shown to be the case in [2][4].

The error could be further reduced in the future by changing how we handle occlusions. Currently we are able to detect occlusions by comparing the depth map for both the base and match image. Once occlusions are found they are set to zero, which can cause an increase in the measured MSE. By adapting an interpolation technique as suggest by [2] we can increase clarity of the depth map and increase the accuracy.

Implementing a mutual information based initial cost function can also further increase the accuracy of the depth maps. We experimented with a basic mutual information approach, but were unable to produce reasonable outputs. We believe this is due to the need of multiple iterations for the depth map to converge. Future work would also explorer further the concept of mutual information, and finding a way to incorporate it into stereo matching with as little iterations as possible.

The results from this project further indicate that Semi-Global Matching is a robust approach to estimating depth information from a scene. Semi-Global Matching well balances the tradeoffs of both speed and accuracy, making it a strong contender for use in new technologies.

Link to code:
https://github.com/rmahieu/SemiGlobalMatching

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11):1222–1239, 2001.

[2] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 807-814 vol. 2.

[3] H. Hirschmüller and D. Scharstein. Evaluation of cost functions for stereo matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN, June 2007.

[4] Michael, M., Salmen, J., Stallkamp, J., & Schlipsing, M. (2013, June). Real-time stereo vision: Optimizing semi-global matching. In *Intelligent Vehicles Symposium (IV), 2013 IEEE* (pp. 1197-1202). IEEE.

[5] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, and L. Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming. International Journal of Computer Vision, 47(1/2/3):275–285, April-June 2002.

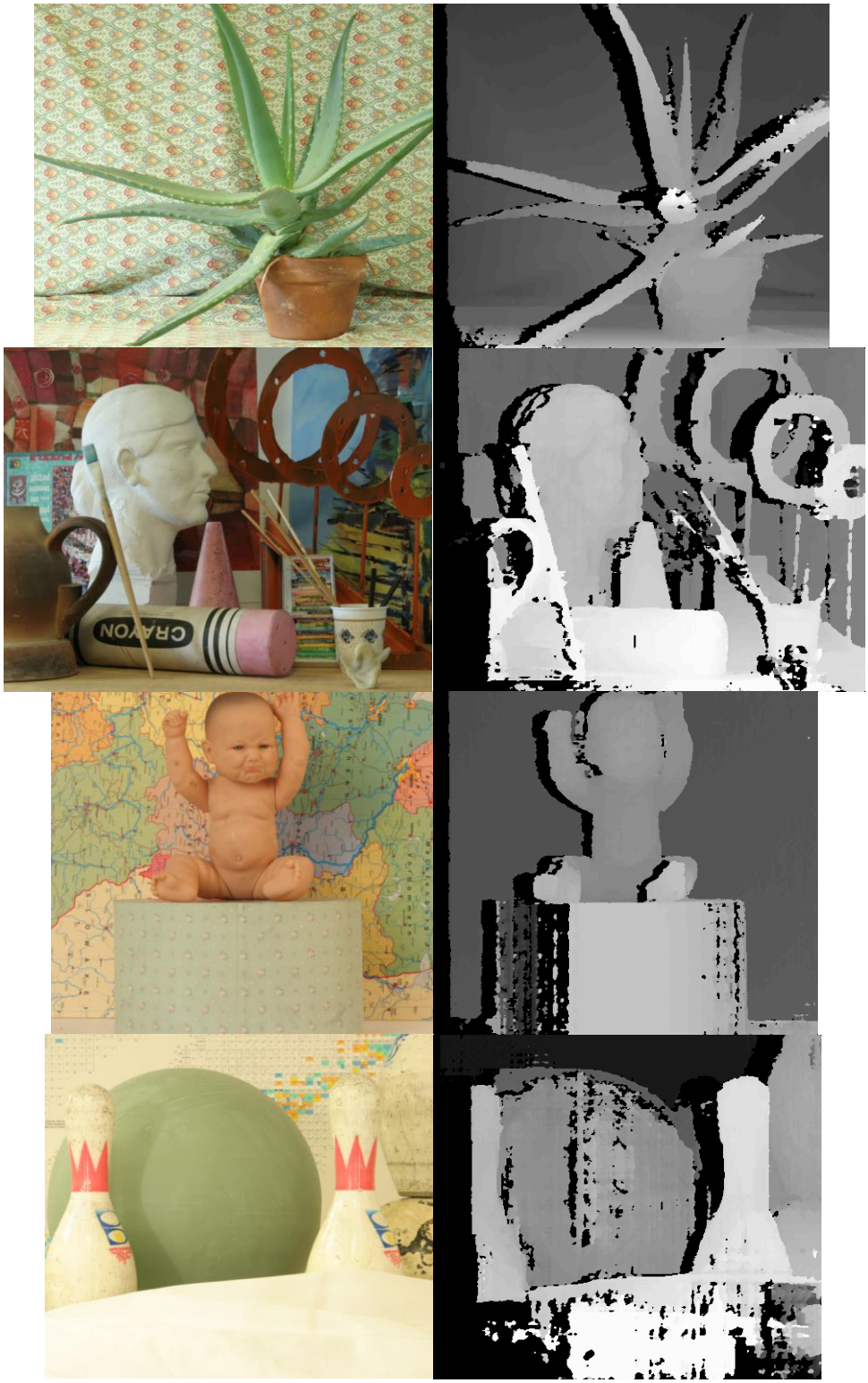
[6] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003),* volume 1, pages 195-202, Madison, WI, June 2003.

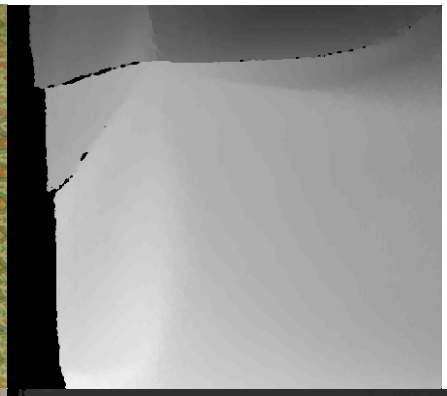
[7] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," International Journal of Computer Vision, vol. 47, pp. 7–42, 2002.

[8] 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.

[9] J. Sun, H. Y. Shum, and N. N. Zheng. Stereo matching using belief propagation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(7):787–800, July 2003

Appendix:





Fisherbrand
Safety Matches