How Far Can We Go with Local Optimization in Real-Time Stereo Matching

—— A performance study on different cost aggregation approaches

Liang Wang [*], Mingwei Gong [†], Minglun Gong [‡], and Ruigang Yang [*]

[*] *Department of Computing Science, University of Kentucky, Lexington, KY, USA*
[†] *Department of Computer Science, University of Calgary, Calgary, AB, Canada*
[‡] *Department of Math & Computer Science, Laurentian University, Sudbury, ON, Canada*

Abstract

Applications such as robot navigation and augmented reality require high-accuracy dense disparity maps in real-time and online. Due to time constraint, most real-time stereo applications rely on local winner-take-all optimization in the disparity computation process. These local approaches are generally outperformed by offline global optimization based algorithms. However, recent research shows that, through carefully selecting and aggregating the matching costs of neighboring pixels, the disparity maps produced by a local approach can be more accurate than those generated by many global optimization techniques. We are therefore motivated to investigate whether these cost aggregation approaches can be adopted in real-time stereo applications and, if so, how well they perform under the real-time constraint.

The evaluation is conducted on a real-time stereo platform, which utilizes the processing power of programmable graphics hardware. Several recent cost aggregation approaches are also implemented and optimized for graphics hardware so that real-time speed can be achieved. The performances of these aggregation approaches in terms of both processing speed and result quality are reported.

1 Introduction

Intensity-based stereo algorithms try to estimate dense disparity maps based on two or more images captured from different views. Most of the previous works are nicely surveyed by Scharstein and Szeliski [11]. As they suggested, a stereo vision algorithm generally performs the following four steps: 1) cost initialization, in which the matching costs for assigning different disparity hypotheses to different pixels are calculated; 2) cost aggregate, where the initial matching costs are aggregated spatially over support regions; 3) disparity optimization, in which the best disparity hypothesis for each pixel is computed so that a local or global cost function is minimized; and 4) disparity refinement, where the generated disparity maps are post-processed to remove mismatches or to provide sub-pixel disparity estimates.

This paper integrates several cost aggregation approaches into a real-time stereo platform and evaluates their performances in terms of both the computation time required and the quality of the disparity maps generated. Our work is motivated by the following two observations.

First, even though global optimization techniques are widely used in many state-of-the-art stereo algorithms, real-time algorithms today still rely on local winner-take-all (WTA) optimization. The performances of these real-time algorithms highly depend on the cost aggregation approaches used. Hence, a thorough evaluation on different cost aggregation approaches can serve as a useful guide for future development of real-time algorithms.

Secondly, several cost aggregation approaches are proposed recently [6, 14, 18]. They demonstrate that, with proper cost aggregation approaches, algorithms based on local WTA optimization can outperform many global optimization based techniques. Even some of these aggregation approaches cannot achieve real-time speed yet [14, 18], it is worthy to investigate whether they can be adopted into real-time stereo vision applications and how their relative performances are.

1.1 Previous cost aggregation approaches

The output of the cost initialization step is a 3D cost volume. All cost aggregation approaches try to update each cost value in this 3D volume based on cost values in its local support regions. Different aggregation approaches differ in the way of selecting the support region, as well as the function used for calculating the new costs.

A straightforward aggregation approach is to replace the cost of assigning disparity d to a given pixel p with the average cost of assigning d to all pixels in a square window centered at pixel p. This square-window approach implicitly assumes that all pixels in the square window have similar disparity value as the center pixel does. It performs badly in areas near depth discontinuity boundaries, where the above assumption does not hold. To better handle these areas, the shiftable window approach can be used, which considers multiple square windows

centered at different locations and uses the one that gives the smallest average cost [2, 5].

For both square-window and shiftable-window approaches, the size of the support window is fixed and is difficult to pick − It should be large enough to include enough intensity variation for reliable matching, as well as be small enough to avoid disparity variation inside the window. The ideal window size may vary within a single image: larger ones for areas with weak textures and smaller ones for areas with geometry details. To this end, several adaptive-window approaches have been proposed, which select the optimal window sizes and/or shapes automatically based on local information [3, 9, 13, 14].

In many cases, the depth discontinuity boundaries in the scene are also the color discontinuity boundaries. Based on this property, segmentation-based approaches select the sizes and shapes for support windows according to the segmentation results of the input image [12, 15]. These approaches can produce nice results when the assumption − disparities of all pixels in the same segmentation are similar − holds. However, they require color segmentation as *a prior*, which is difficult to obtain in real-time stereo matching. To remove this constraint, approach that based on edge detection results has been proposed [6].

Instead of searching for an optimal support window with arbitrary shape and size, the recently proposed adaptive-weight algorithm adjusts the support-weight of each pixel in a fix-sized square window [18]. The support-weight of each pixel in the window is calculated based on both the color similarity and the Euclidean distance to the center pixel. The experiment results show that this approach can produce disparity maps comparable with those generated using global optimization techniques. However, the computation cost is high and the typical running time is about one minute.

The support regions used in all aggregation approaches above are 2D at a fixed disparity value and therefore favoring fronto-parallel surfaces. To better handle slanted surfaces, the oriented-rod approach is also used [10]. By assuming that the intersections between slanted surfaces in the scene and fronto-parallel planes are close to straight lines, this approach aggregates costs along a group of line segments with different orientations and uses the one that gives the smallest average cost.

1.2 Other related work

As the computation power of standard PCs increases dramatically in the past few years, how to perform real-time stereo matching without special hardware becomes an active research topic [4, 6-8, 16, 17]. To achieve real-time speed many of the proposed algorithms utilize hardware-dependent instructions, such as the Multi-Media Extension (MMX) instructions for recent CPUs [4, 8] and the pixel/vertex shading instructions for the Graphics Processing Units (GPUs) that are available on modern programmable graphics hardware [6, 7, 16, 17].

Due to the time constraint, almost all real-time stereo algorithms to date are either using local WTA optimization or optimizing different scanlines independently using dynamic programming (DP). The traditional DP technique does not enforce the consistency constraint across different scanlines and hence suffers from the "streaking" artifact [4]. This problem can be addressed using the 2-pass DP approach [7, 10]. However, due to the complexity of the algorithm, these approaches are limited to non-real-time or near-real-time speed. As a result, real-time stereo applications will likely rely on local optimization in the near future, which makes it worthy to investigate how accurate and how fast we can get with local optimization when different cost aggregation approaches are used.

Our paper is closely related to Scharstein and Szeliski stereo survey [11]. To make our evaluation results comparable with those reported in other literatures, we use the datasets downloaded from their Middlebury stereo vision site, as well as their online service for evaluating disparity maps. However, our evaluation is mainly focused on the performances of different cost aggregation approaches when they are used for real-time stereo matching. The performance metrics we used include both result quality and processing speed. To satisfy the real-time constraint, the testing platform we developed and the implementation for different aggregation approaches make full use of the processing power of the GPU. The evaluation results support some interesting observations that are somewhat contradict with those reported by [11].

2 Implementations

A testing platform is developed and several cost aggregation approaches are implemented using C++ with DirectX and pixel shader 2.0. The implementation is modularized and can be easily extended to include other aggregation approaches in the future. Details about the implementation are discussed below.

2.1 The testing platform

The testing platform contains three modules, which can be used together to generate disparity maps without the cost aggregation step. The implementations for these modules are optimized for speed. Without cost aggregation, the application can achieve over 500 fps for the Tsukuba dataset on an ATI 800 graphics card.

Cost initialization module

Three different matching criteria are used: absolute intensity differences (AD), truncated absolute differences (TAD), and sampling insensitive matching criterion (SIC)

[1]. They are implemented based on the following equations:

$$C_{AD}(u,v,d) = \left| I(u,v) - I^{ref}(u+d,v) \right|$$

$$C_{TAD}(u,v,d) = \min\left(\left| I(u,v) - I^{ref}(u+d,v) \right|, C_{\max} \right) \times \frac{255}{C_{\max}}$$

$$C_{SIC}(u,v,d) = \min_{k=\{-0.5,0,+0.5\}} \begin{pmatrix} \left| I(u,v) - I^{ref}(u+d+k,v) \right|, \\ \left| I(u+k,v) - I^{ref}(u+d,v) \right| \end{pmatrix}$$

where $I(u,v)$ and $I^{ref}(u,v)$ are the intensities of pixel (u,v) in the center and the reference images, respectively. In the TAD approach, C_{max} is the truncation value used and is set to 25 in our experiments. The costs are scaled to the range of [0,255] after truncation.

Similar to previous GPU-based approaches [6], the 3D cost volume is treated as a stack of 2D grayscale images. The grayscale images for four adjacent disparity hypotheses are packed into one color image to utilize the vector processing capacity of GPU. The color images are tiled together to form a large matching cost texture.

This process for generating matching cost texture contains multiple rendering passes. In each rendering pass, a pixel shader takes the two stereo images as input textures and initializes one of the tiles in the matching cost texture. For a stereo dataset whose disparity range is R, a total of $R/4$ rendering passes is required.

Disparity optimization module

The disparity optimization step computes an optimal disparity map using the local WTA approach. The pixel shader developed takes the matching cost texture as input and generates a disparity texture. For each output pixel, the pixel shader goes through the corresponding pixels in different tiles and searches for the disparity value that gives the smallest matching cost. Due to the limitation on the number of instructions one can use in pixel shader 2.0, the pixel shader can handle up to 5 tiles in a single rendering pass. The total number of rendering passes needed is therefore $R/20$.

Disparity refinement module

Since local optimization does not enforce the smoothness constraint, the disparity maps generated may contain spurious mismatches. A 3×3 median filter is employed in the disparity refinement step to remove these mismatches. This median filtering process is implemented on the GPU using a single rendering pass.

2.2 Implementation of aggregation approaches

A total of five different cost aggregation approaches are implemented. Some of them are initially designed for running on the GPU [6], while others are adopted from the original CPU-based algorithm [10, 18]. For the latter group, the correctness of our GPU based implementations is verified using our implementation on the CPU. All GPU implementations are carefully optimized for speed as well.

Square-window and shiftable-window approaches

The square-window can be implemented using a box filter and the shiftable-window can be implemented using a box filter following by a min filter [11]. Since both box and min filters are separable, they are implemented using two rendering passes. The first rendering pass updates costs of different pixels with the average (or minimum) costs of their neighboring pixels along horizontal scanlines, while the second one does the same along vertical scanlines.

Please note that the incremental property of the box filter and the min filter is not used here as the GPU is designed for parallel processing and cannot execute iterative algorithms efficiently.

Oriented-rod approach

A rather complicated aggregation approach is used in [10]. The process starts with classifying pixels in the input stereo image into homogeneous pixels and heterogeneous ones. After the oriented-rod filter is applied to all pixels, an additional shiftable-window filter is applied to homogeneous pixels.

In order to achieve real-time speed and to focus on the effect of the oriented-rod filter, our GPU-based implementation only contains the oriented-rod filtering step. Multiple rendering passes are used to process rods of different orientations. In each rendering pass, the pixel shader we developed calculates the average cost along the current rod and, if necessary, updates the minimum cost. In our experiments, a total of 24 orientations are used.

Boundary-guided approach

The boundary-guided filter is initially proposed for the GPU [6]. Their original paper proposes two approaches: the first one assumes color segmentations of input images are available and the second does not. Since generating accurate segmentation in real-time is a difficult problem itself, only the second approach is evaluated here.

The process starts with detecting color discontinuity boundaries using local intensity gradient information. The 5×5 boundary-guided filter is then applied one or more times according to the desired maximum window size.

Adaptive-weight approach

Due to its high computational cost, the original adaptive-weight algorithm takes about a minute to generate a disparity map [18]. We have to simplify the algorithm to achieve real-time speed.

In our GPU implementation, instead of calculating the weights based on both stereo images, they are calculated based on the center image only. This makes it possible to pre-calculate all the weights for each pixel in the center image and use them in the weighted-average pixel shader

we developed. Even though the disparity maps we generated using one-image-based adaptive weights are not as good as those reported in the original paper, we believe our approach is a good tradeoff as it achieves two orders of magnitude speedup when implemented on the GPU.

3 Experiment results

Several experiments are conducted to evaluate the performances of different cost aggregation approaches we implemented. All experiments are conducted on a 3GHz P4 equipped with an ATI X800 graphics card.

3.1 Effects of cost initialization approaches

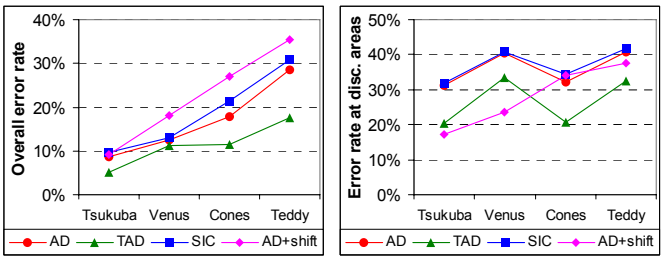


Figure 1: The quality of disparity maps generated using different cost initialization approaches; left: error rate at all visible areas; right: error rate at areas near depth discontinuity boundaries.

The first experiment compares the three different cost initialization approaches (AD, TAD, and SIC). The initial matching costs are aggregated using a 9×9 square window before they are used for computing disparity maps. Since previous evaluation suggests that avoiding outliers by shifting window is preferable to limiting their influence using truncated cost function [11], the results obtained using AD with 9×9 shiftable window is also included in the evaluation.

As shown in Figure 1, somewhat contradict to previous belief, neither the SIC nor the shiftable window helps to reduces the overall mismatches (the shiftable window approach does reduce mismatches in near depth discontinuity areas for the first two datasets). This finding is confirmed by our CPU-based implementation and the investigation indicates that the precision limitation in real-time stereo matching is the cause. To achieve high processing speed, most real-time stereo algorithms store matching costs in 8-bit integers [4, 6, 7, 16] and therefore small differences in cost aggregation results cannot be distinguished. Representing costs using 16-bit floating numbers alleviates the problem, but leads to a huge performance penalty. Under this situation, truncating the matching costs and then scaling them to [0,255] can make the best use of the range of 8-bit integers.

Our experiments show that TAD gives the best performance for other aggregation approaches as well (not shown here for space limit). Therefore it is used in the rest experiments.

3.2 Performances of different approaches under different settings

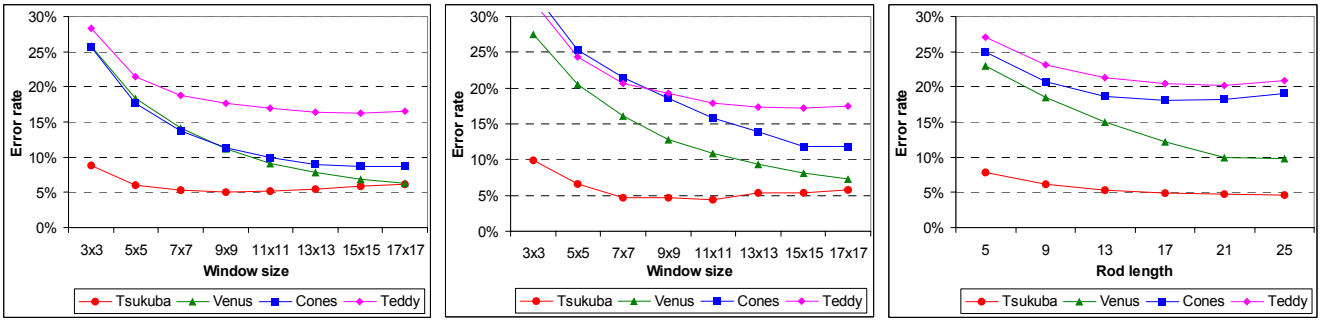
The cost aggregation approaches we implemented all have parameters controlling the size of the support region to be used. Figure 2 plots the overall error rate of the disparity maps generated by different approaches with different settings. Since none of the cost aggregation approaches tries to handle half-occluded regions, the mismatches in these regions are not included. The following can be observed from the result:

- In general the error rate is high when small support regions are used and starts to decrease as the support regions get larger. This is reasonable as small support regions cannot effectively suppress noises.
- For approaches using fix-sized support window (square-window and shiftable-window), increasing the support window size over a threshold will also increase the error rate. As shown in the next experiment, this is mainly because larger support windows introduce more mismatches at areas near depth discontinuity boundaries.
- The above problem is not as important for approaches using adaptive window/weight as the parameters in these approaches only control the maximum window sizes. The outliers in the window may be excluded in the cost calculation or may have very small weights. Their influence is therefore minimized.
- It is generally difficult to pick a good parameter for fix-sized support window approaches as it is highly depends on the input dataset. For example, the best window size in square-window approach is 7×7 for the Tsukuba dataset and 17×17 for the Venus one.

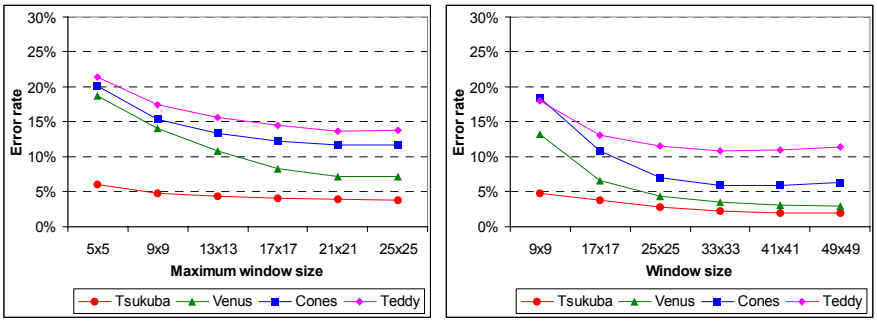
To evaluate the performance of different approaches in areas near depth discontinuities, Figure 3 plots the error rates in these regions. It suggests that:

- For the square-window approach, the number of mismatches in near discontinuity areas starts to increase after the support window size passes 5×5.
- Using shiftable windows helps to reduce mismatches in these areas. However, once the support window size passes 9×9, the number of mismatches still increases. This suggests that, in areas that contain geometry details, changing the location of the window only is not good enough.
- For the oriented-rod approach, increasing the length of the rod does not increase mismatches in near discontinuity areas dramatically because it is likely that one of the oriented rods does not cross the boundary.
- Approaches with adaptive window/weight perform reasonable well in near discontinuity areas.

Finally, the computation times required by different aggregation approaches as functions of the parameters used are plotted in Figure 4. It is noteworthy that the curves for Teddy and Cone datasets overlap as the computation times required are the same. As expected, the processing time increases linearly as the size of the support region increases. In addition, the total processing time is less than 65ms under most settings. That is, more than 15 fps can be achieved for even the most demanding datasets (Teddy and Cones).

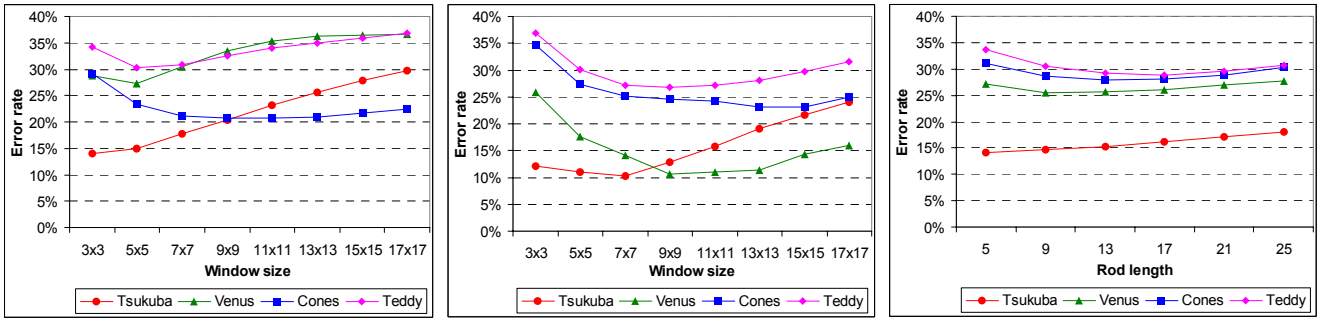


(a) square-window (b) shiftable-window (c) oriented-rod

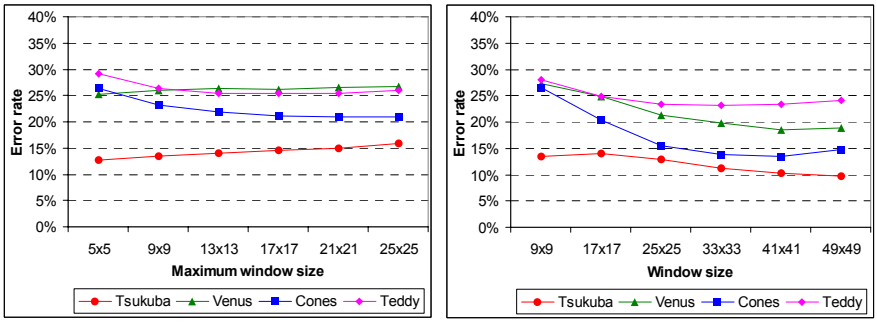


(d) boundary-guided (e) adaptive-weight

Figure 2: The overall error rates (non-occluded areas only) of disparity maps generated by different approaches



(a) square-window (b) shiftable-window (c) oriented-rod



(d) boundary-guided (e) adaptive-weight

Figure 3: The error rate in areas near depth discontinuity boundaries for different approaches

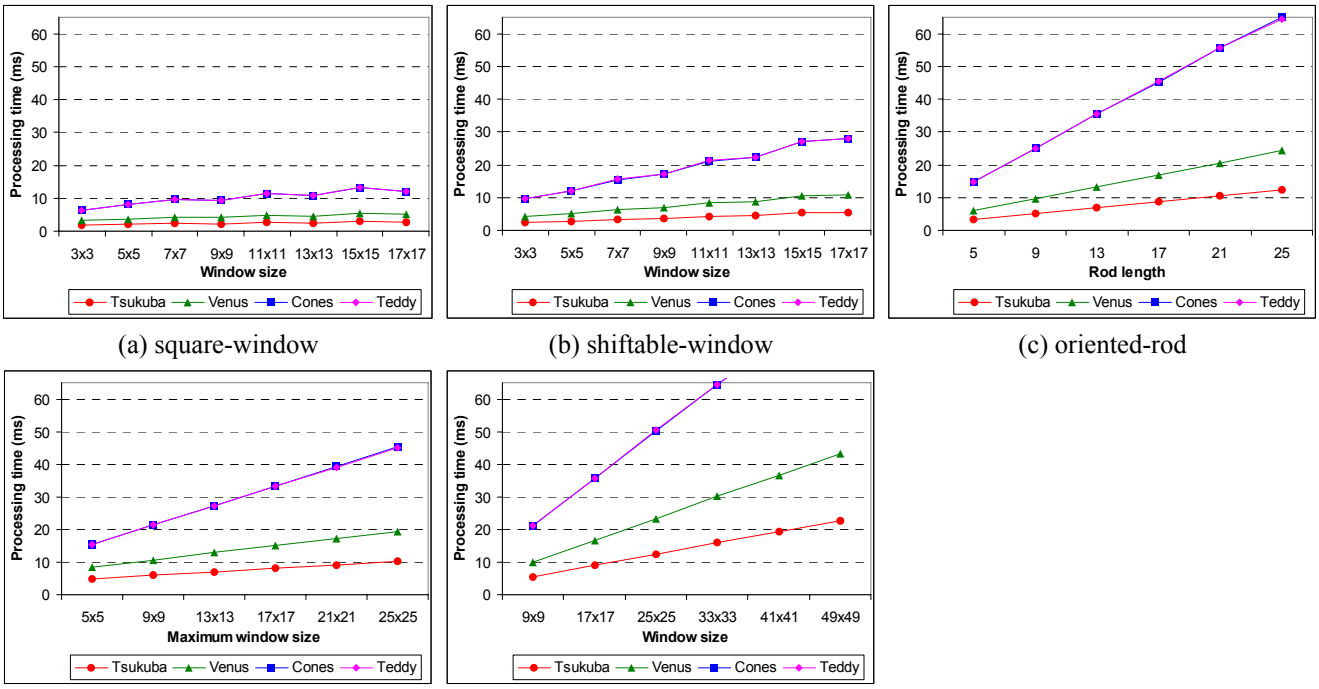
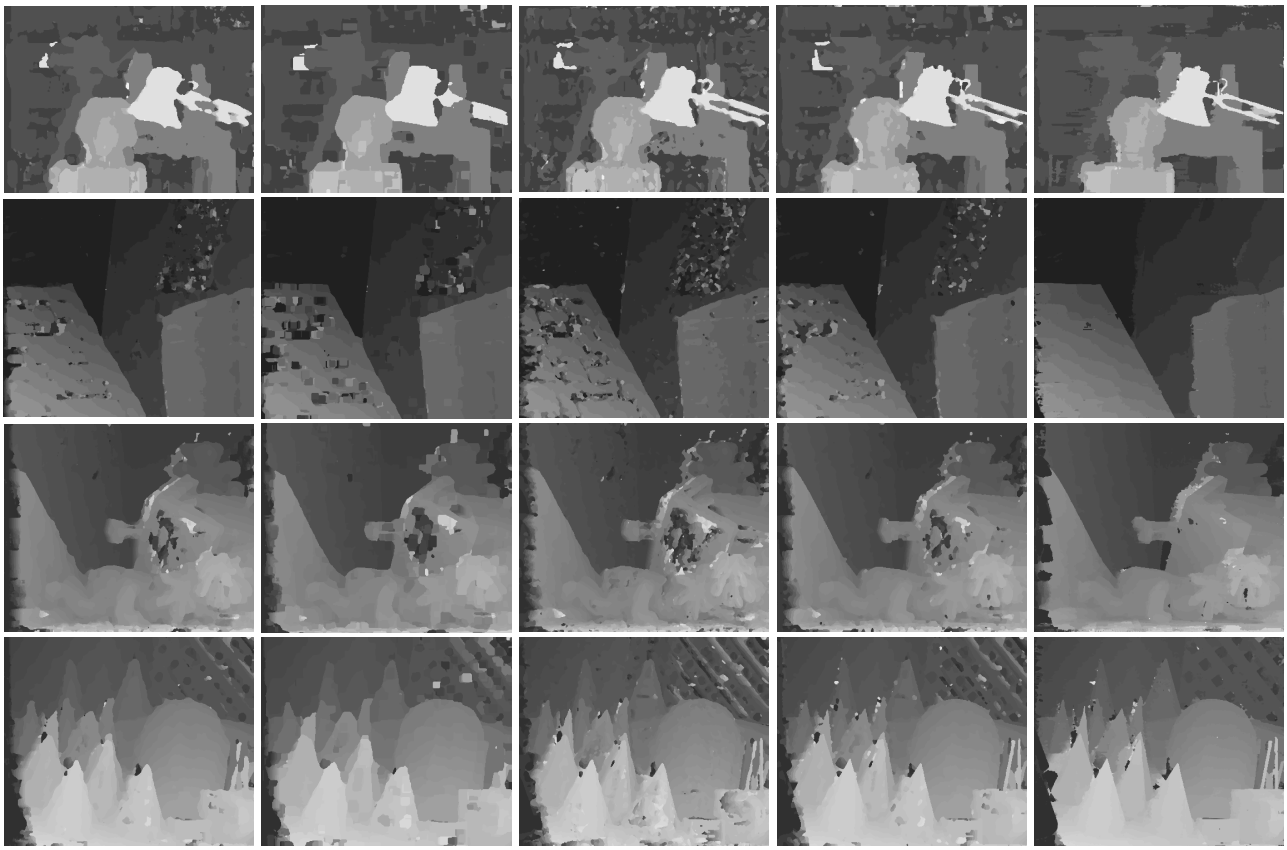


Figure 4: The computation time needed for different aggregation approaches



(a) square-window (b) shiftable-window (c) oriented-rod (d) boundary-guided (e) adaptive-weight

Figure 5: Disparity maps generated by different approaches using selected parameter settings.

TABLE 1: THE ACCURACY OF DISPARITY MAPS GENERATED BY DIFFERENT APPROACHES AS EVALUATED BY MIDDLEBURY STEREO SITE. (THE BOTTOM 3 ROWS ARE EVALUATION RESULTS REPORTED BY THE ORIGINAL PAPERS. SOME FIELDS ARE LEFT BLANK AS NOT ALL MEASURES ARE REPORTED IN THESE PAPERS.)

		Tsukuba			Venus			Teddy			Cones		
		vis	all	disc	vis	all	disc	vis	all	disc	vis	all	disc
Our GPU-based implementations	Square-window	5.13	7.11	23.2	9.18	10.3	35.4	16.9	24.5	34	9.94	18.9	20.8
	Shiftable-window	4.46	6.03	15.7	10.9	11.9	11	17.9	24.6	27.1	15.8	22.9	24.3
	Oriented-rod	5.68	7.77	14.8	16.6	17.8	25.5	22.1	29.4	29.8	24	31.6	28.1
	Boundary-guided	3.88	5.88	15	7.12	8.34	26.6	13.7	21.4	25.4	11.7	20.4	20.9
	Adaptive-weight	2.27	3.61	11.2	3.57	4.61	19.8	10.9	18.8	23.2	5.92	14.3	13.8
Related aggregation approaches	Adaptive-weight [18]	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26
	DP w/ oriented-rod [10]	1.53			0.94								
	Boundary-guided [6]	4.91			9.43								

3.3 Result comparison on Middlebury datasets

The best settings for different approaches are picked based on the previous evaluation results and are used for the comparison among different approaches. In particular, we set window size to 11×11 for both square-window and shift-window approaches, rod length to 21 for oriented-rod approach, and maximum window sizes to 21×21 and 33×33 for boundary-guided and adaptive-weight approaches, respectively.

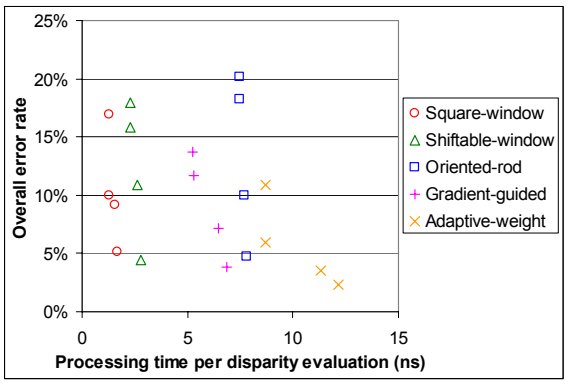


Figure 6: The speed vs. quality plot for different approaches. Due to the overhead cost for calculating the support weights or for detecting edges, the per disparity estimation processing time for both gradient-guided and adaptive-weight approaches is different for different datasets.

The disparity maps generated under these settings are evaluated using the Middlebury site. The numerical accuracies of these disparity maps and those generated by other related algorithms are listed in Table 1. The disparity maps are also shown in Figure 5 for visual comparison. The overall performances (speed vs. quality) of different algorithms are plotted in Figure 6. The comparison results support following conclusions:

As expected, the square window approach has the fastest speed. However, it has difficulties in handling both near depth discontinuity areas and weakly textured areas. Details such as the chopsticks in the Cones scene are lost and mismatches show up in the roof area of the Teddy scene.

As shown in Table 1, using shiftable windows helps to reduce mismatches in near discontinuity areas, but it actually increases the overall error rates on three of four datasets. We believe this is mainly due to the limited precision issue caused by representing costs using single bytes.

Using oriented rods instead of square windows helps to preserve geometry details. However, since 1D rod does not offer enough samples for reliable matching in weakly textured areas, using it alone yields noisy results. As suggested in previous research [10], this problem can be addressed by classifying pixels into homogeneous and heterogeneous groups and applying an additional shiftable-window aggregation process to homogeneous pixels. However, as shown in Figure 6, the computational cost of oriented-rod filter is already high. Making the algorithm more complex may make it unsuitable for real-time stereo matching.

Our implementation for the boundary-guided approach produces slightly better disparity maps than the original approach does [6]. We believe this is due to the larger support window we used, as well as the additional disparity refinement process. Generally speaking, this approach performs reasonable good in areas with enough details, but not so well in weakly textured areas.

Even though our simplified adaptive-weight implementation does not perform as good as the original offline algorithm [18], it is already better than many other real-time algorithms [4, 6, 17]. The disparity maps generated are clean and detailed. Further investigation on how to better utilize adaptive-weight aggregation in real-

time stereo applications seems very promising. However, in terms of processing speed, this approach is a bit slow.

4 Conclusions

This paper evaluates the performances of five different cost aggregation approaches in terms of both result quality and processing speed. The objective is to find out how accurate the generated disparity maps can be when these aggregation approaches are used in a local optimization based real-time stereo application. To achieve high processing speed, these algorithms are implemented on the GPU and are integrated into the same testing platform. Many observations are obtained from the evaluation, which can be useful guides to design better cost aggregation approaches and novel real-time stereo applications in the future.

In particular, our evaluation shows that techniques such as sampling insensitive matching criterion and shiftable windows, which are widely used in offline stereo algorithms, are not helpful for real-time stereo applications. This is because most real-time stereo algorithms represents matching costs with single bytes to optimize processing speed. This poses a new challenge as small cost differences cannot be presented due to the very limited precision of a single byte.

Our evaluation also suggests that, properly designed cost aggregation approaches can significantly improve the quality of the disparity maps generated without introducing too much computational cost. For example, the adaptive-weight aggregation approach we implemented can produce reasonably accurate disparity maps in real-time. Therefore, further study on new cost aggregation approaches appears to be a promising research direction.

The research presented in this paper is focused on the performances of different cost aggregation algorithms when implemented on the GPU and used with local optimization technique. It will be useful to include real-time CPU-based cost aggregation algorithms in the evaluation, as well as to test the performances of these algorithms when used with fast global optimization techniques such as dynamic programming. We plan to extend our study to these areas in the future.

Acknowledgements

This work is supported in part by Laurentian University, National Science and Engineering Council of Canada, University of Kentucky Research Foundation, US Department of Homeland Security, and US National Science Foundation grant IIS-0448185.

References

[1] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *IJCV*, vol. 35, no. 3, 1999.

[2] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *IJCV*, vol. 33, no. 3, pp. 181-200, 1999.

[3] Y. Boykov, O. Veksler, and R. Zabih, "A variable window approach to early vision," *IEEE TPAMI*, vol. 20, no. 12, pp. 1283-1294, 1998.

[4] S. Forstmann, J. Ohya, Y. Kanou, A. Schmitt, and S. Thuering, "Real-time stereo by using dynamic programming," *Proc. CVPR Workshop on Real-time 3D Sensors and Their Use*, Washington, DC, USA, pp. 29-36, 2004.

[5] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," *Proc. CVPR*, Puerto Rico, pp. 858-863, 1997.

[6] M. Gong and R. Yang, "Image-gradient-guided real-time stereo on graphics hardware," *Proc. 3DIM*, Ottawa, ON, Canada, pp. 548-555, 2005.

[7] M. Gong and Y.-H. Yang, "Near real-time reliable stereo matching using programmable graphics hardware," *Proc. CVPR*, San Diego, CA, USA, pp. 924-931, 2005.

[8] H. Hirschmuller, P. R. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *IJCV*, vol. 47, no. 1-3, 2002.

[9] T. Kanade and M. Okutomi, "Stereo matching algorithm with an adaptive window: theory and experiment," *IEEE TPAMI*, vol. 16, no. 9, pp. 920-932, 1994.

[10] J.-C. Kim, K. M. Lee, B.-T. Choi, and S. U. Lee, "A dense stereo matching using two-pass dynamic programming with generalized ground control points," *Proc. CVPR*, pp. 1075-1082, 2005.

[11] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, vol. 47, no. 1-3, pp. 7-42, 2002.

[12] H. Tao and H. S. Sawhney, "Global matching criterion and color segmentation based stereo," *Proc. Workshop on the Application of Computer Vision*, pp. 246-253, 2000.

[13] O. Veksler, "Stereo correspondence with compact windows via minimum ratio cycle," *IEEE TPAMI*, vol. 24, no. 12, pp. 1654-1660, 2002.

[14] O. Veksler, "Fast variable window for stereo correspondence using integral images," *Proc. CVPR*, pp. 556-561, 2003.

[15] L. Wang, S. B. Kang, H.-Y. Shum, and G. Xu, "Cooperative segmentation and stereo using perspective space search," *Proc. Asian Conference on Computer Vision*, Jeju Island, Korea, pp. 366-371, 2004.

[16] R. Yang and M. Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware," *Proc. CVPR*, Madison, WI, USA, pp. 211-220, 2003.

[17] R. Yang, M. Pollefeys, and S. Li, "Improved real-time stereo on commodity graphics hardware," *Proc. CVPR Workshop on Real-time 3D Sensors and Their Use*, Washington, DC, USA, 2004.

[18] K.-J. Yoon and I.-S. Kweon, "Locally adaptive support-weight approach for visual correspondence search," *Proc. CVPR*, pp. 924-931, 2005.