International Journal of Computer Vision 75(2), 283–296, 2007

DOI: 10.1007/s11263-006-0032-x

A Performance Study on Different Cost Aggregation Approaches Used in Real-Time Stereo Matching

MINGLUN GONG
Department of Math & Computer Science, Laurentian University, Sudbury, ON, Canada

RUIGANG YANG AND LIANG WANG
Department of Computing Science, University of Kentucky, Lexington, KY, USA

MINGWEI GONG
Department of Computer Science, University of Calgary, Calgary, AB, Canada

Received August 5, 2006; Accepted December 27, 2006

First online version published in February, 2007

Abstract. Many vision applications require high-accuracy dense disparity maps in real-time and online. Due to time constraint, most real-time stereo applications rely on local winner-takes-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. Six recent cost aggregation approaches are 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.

Keywords: real-time stereo matching, cost aggregation algorithms, programmable graphics hardware

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 (2002). 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.

As an extension to our preliminary work (Wang et al., 2006), 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, most existing real-time algorithms—those can produce disparity maps of 320 $\times$ 240 resolution at 10 fps or more—still rely on local winner-takes-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 new cost aggregation approaches are proposed recently (Gong and Yang, R., 2005; Veksler, 2003; Yoon and Kweon, 2005). 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 (Veksler, 2003; Yoon and Kweon, 2005), it is worthy to investigate whether they can be adopted into real-time stereo applications and how their relative performances are.

Compared to our preliminary work (Wang et al., 2006), this paper provides more detailed discussions on the implementation of different cost aggregation approaches. One more adaptive-window based approach is included in the evaluation. The optimal range of the truncation parameter used in cost initialization step and the effect of disparity refinement step are also investigated.

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 selects the one that gives the smallest average cost (Bobick and Intille, 1999; Fusiello et al., 1997). An extension has also been proposed, which picks several windows from surrounding square windows and computes the sum of their average costs (Hirschmuller et al., 2002).

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 (Boykov et al., 1998;Kanade and Okutomi, 1994; Veksler, 2003).

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 (Tao and Sawhney, 2000; Wang et al., 2004). These approaches can produce nice results when disparities of pixels in the same segmentation are similar. 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 (Gong and Yang, R., 2005).

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 (Yoon and Kweon, 2005). 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 developed (Kim et al., 2005). 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 (Forstmann et al., 2004; Gong and Yang, R., 2005; Gong and Yang, Y.-H., 2005; Hirschmuller et al., 2002; Yang et al., 2006; Yang and Pollefeys, 2003; Yang, et al., 2004). 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 (Forstmann et al., 2004; Hirschmuller et al., 2002) and the pixel/vertex shading instructions for the Graphics Processing Units (GPUs) that are available on modern programmable graphics hardware (Gong and Yang, R., 2005; Gong and Yang, Y.-H., 2005; Yang et al., 2006; Yang and Pollefeys, 2003; Yang et al., 2004).

Due to the time constraint, most 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 (Forstmann et al., 2004). This problem can be addressed using the 2-pass DP approach (Gong and Yang, Y.-H., 2005; Kim et al., 2005). 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 (Scharstein and Szeliski, 2002, 2003). To make our evaluation results comparable with those reported in other literatures, we use the datasets downloaded from their Middlebury stereo evaluation 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.

It should be noted that dedicated hardware for stereo processing is available (Woodfill and Von Herzen, 1997). Due to the high cost in developing hardware, these systems typically implemented fairly standard stereo algorithm based on WTA methods. We do not investigate their quality and speed in this paper.

2. Implementations

A testing platform is developed and several cost aggregation approaches are implemented using Visual C++ with DirectX 9.0 and pixel shader 2.0. Our implementation is modularized and can be easily extended to include other aggregation approaches in the future. The source code is available at http://www.cs.laurentian.ca/gong/research/CostCompare/. Details about the implementation are discussed below.

2.1. The Testing Platform

The testing platform contains three modules: cost initialization, disparity optimization, and disparity refinement. They can be used together to generate disparity maps without the cost aggregation step. The implementations for these modules are optimized on GPU for speed. Without cost aggregation, the application can achieve over 530 fps for the Tsukuba dataset on an ATI Radeon X800 graphics card. Therefore, when evaluating the processing speed of different cost aggregation approaches, the computation time spent on these three modules (less than 2 ms for the Tsukuba dataset) can be omitted.

2.1.1. Cost Initialization Module. The cost initialization module computes the initial matching cost $C(u, v, d)$ for assigning disparity hypothesis d to pixel (u, v) Similar to existing real-time stereo implementations (Forstmann et al., 2004; Gong and Yang, R., 2005; Gong and Yang, Y.-H., 2005; Yang et al., 2006; Yang and Pollefeys, 2003) the matching costs obtained are stored in 8-bit integers for lower computational cost. In order to make the best use of the range that a single byte offers, the costs are calculated using the truncated and scaled absolute intensity differences between corresponding pixels. That is:

$$C(u, v, d) = \min(|I(u, v) - I^{\text{ref}}(u - d, v)|, C_{\max}) \times \frac{255}{C_{\max}} \quad (1)$$

where $I(u, v)$ and $I^{\text{ref}}(u, v)$ are the intensities of pixel (u, v) in the center and the reference images, respectively. The parameter $C_{\max}$ $(0 < C_{\max} \leq 255)$ is the truncation value. When $C_{\max}$ is set to 255 the above equation simply computes the absolute intensity difference between the two pixels.

We also evaluated the effect of applying the sampling insensitive matching criterion (Birchfield and Tomasi, 1998) in matching cost calculation. The results show that this criterion helps when the truncation value used is too small, but offers little or no improvement when a reasonable truncation value is selected. This finding is consistent with Scharstein and Szeliski's survey (Scharstein and Szeliski, 2002). Since this paper focuses on the performances of different aggregation approaches under optimal truncation values, the sampling insensitive criterion is not applied.

Similar to previous GPU-based approaches (Gong and Yang, R., 2005), 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. An example is shown in Fig. 1.

This process to initialize the cost volume 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.

2.1.2. Disparity Optimization Module. The disparity optimization step computes an optimal disparity map

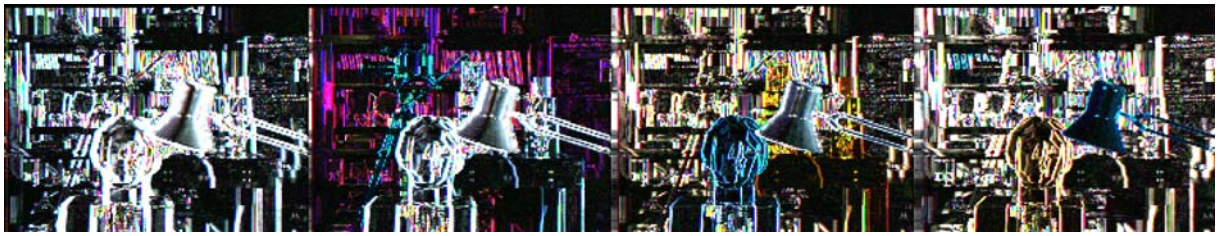

Figure 1. The 2D matching cost textures. The four color channels of a single pixel in the texture store the matching costs of a pixel under four different disparity hypotheses.

$D(u, v)$ 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. That is:

$$D(u, v) = \arg\min_{d}(A(u, v, d)) \tag{2}$$

where $A(u, v, d)$ represents the matching cost obtained after cost aggregation for assigning disparity hypothesis d to pixel (u, v).

2.1.3. 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 six different cost aggregation approaches are implemented. Some of them are initially designed for running on the GPU (Gong and Yang, R., 2005), while others are adopted from the original CPU-based algorithm (Kim et al., 2005; Yoon and Kweon, 2005). 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.

2.2.1. Square-window Approach. The square-window can be implemented using a box filter (Scharstein and Szeliski, 2002). Since box filter is separable, two rendering passes are used in our GPU-based implementation. The first rendering pass updates the cost of every pixel with the average costs of its neighboring pixels along horizontal scanlines, while the second one does the same along vertical scanlines. That is, when the window size is set to $(2r + 1) \times (2r + 1)$, the aggregated matching cost A^r_{square} is calculated using:

$$\begin{aligned} T^r(u, v, d) &= \frac{1}{2r+1} \sum_{m=-r}^{r} C(u + m, v, d) \\ A^r_{\text{square}}(u, v, d) &= \frac{1}{2r+1} \sum_{n=-r}^{r} T^r(u, v + n, d) \end{aligned} \tag{3}$$

where T^r is the cost matrix holding the intermediate results obtained after the first rendering pass.

Please note that the incremental property of the box filter is not used here as the GPU is designed for parallel processing and cannot execute iterative algorithms efficiently. The computational cost is therefore linearly dependent on parameter r.

2.2.2. Shiftable-window Approach. The effect of shiftable-window approach can be achieved using a box filter followed by a min filter (Scharstein and Szeliski, 2002). Since a min filter is also separable, two additional rendering passes are used to compute shiftable-window results based on the square-window ones. Each rendering pass replaces costs of different pixels with the minimum costs of their neighboring pixels along either horizontal or vertical scanlines. That is, the aggregated matching cost for shiftable-window approach, $A^r_{\text{shiftable}}$, can be calculated based on A^r_{square} using:

$$\begin{aligned} T^r(u, v, d) &= \min_{-r \le m \le r} \left(A^r_{\text{square}}(u + m, v, d)\right) \\ A^r_{\text{shiftable}}(u, v, d) &= \min_{-r \le n \le r} (T^r(u, v + n, d)) \end{aligned} \tag{4}$$

2.2.3. Oriented-rod Approach. A rather complicated aggregation approach is used in Kim et al. (2005). 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 does not classify pixels. Instead, a small

3×3 square-window filter is applied to all pixels before the oriented-rod filter is applied. This gives similar effect as computing the average cost within 3-pixel wide rods, which is more robust to noises than applying the oriented-rod filter directly. When the length of the rod is $2r + 1$, the aggregated matching cost A^r_{rod} is calculated using:

$$A^r_{\text{rod}}(u, v, d) = \min_{\theta \in \Theta} \left(\frac{1}{2r+1} \sum_{k=-r}^{r} A^1_{\text{square}} \times (u + k \cdot \cos\theta, v + k \cdot \sin\theta, d) \right) \quad (5)$$

where r controls the length of the rod. Θ is a set of angles that specify all the sampling orientations.

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 rod configurations are used.

2.2.4. Adaptive-window Approach. Most adaptive-window approaches pick an optimal window from a set of windows of different shapes and/or sizes at each pixel location, then search for the best disparity hypothesis using the average costs within the optimal window. When only square windows are used, the corresponding algorithm can be cast as an aggregation approach that computes the final cost as a function of the aggregated costs obtained using square/shiftable windows of different sizes. For example, in the fast variable-window algorithm, the computation of the final aggregated cost involves three terms: the average cost within the window, the variance of the costs in the window, and a bias term that favors large windows (Veksler, 2003).

Due to the use of integral image and dynamic programming, the original variable-window algorithm can hardly be implemented on GPUs for real-time stereo matching. Instead, a simplified adaptive-window based aggregation algorithm is derived and evaluated in this paper. The approach starts with aggregating the initial matching costs using shiftable-windows of different sizes. This produces a set of matching costs for each disparity assignment, based on which the final aggregated cost is calculated. Three simple ways of computing the final aggregated costs are tested. The first method sets the final cost for a given disparity assignment to the average of different matching costs obtained under different window sizes, while the next two methods set it to the minimum or maximum value among them. That is, when the largest window size is set to $(2r + 1) \times (2r + 1)$, the aggregated costs are computed using:

$$\begin{aligned} A^r_{\text{adaptive-avg}}(u, v, d) &= \frac{1}{r} \sum_{k=1}^{r} A^k_{\text{shiftable}}(u, v, d) \\ A^r_{\text{adaptive-min}}(u, v, d) &= \min_{1 \le k \le r} A^k_{\text{shiftable}}(u, v, d) \\ A^r_{\text{adaptive-max}}(u, v, d) &= \max_{1 \le k \le r} A^k_{\text{shiftable}}(u, v, d) \end{aligned} \quad (6)$$

Our experiments show that the last method yields the most accurate disparity maps among the three. It is therefore used in our evaluation.

2.2.5. Boundary-guided Approach. The boundary-guided filter is initially proposed for the GPU (Gong and Yang, R., 2005). 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.

This approach starts with applying Symmetric Neighborhood Filter (Harwood et al., 1987) multiple times to enhance edges in the image. It then calculates the intensity gradients along both the horizontal and vertical direction. The local maximums of the gradients are labeled as boundaries. Cost aggregation is achieved by convoluting the initial matching costs with a 5×5 boundary-guided filter one or more times according to the desired maximum window size. For efficiency, the convolution process is separated into a horizontal and a vertical filtering passes. Both passes replace the cost of each pixel with the average cost of selected neighboring pixels based on nearby color boundary information (Gong and Yang, R., 2005).

2.2.6. Adaptive-weight Approach. When aggregating matching costs, the original adaptive-weight approach computes the weighted average of adjacent matching costs, with the weights generated using both stereo images (Yoon and Kweon, 2005). Under the left-and-right stereo setting, where assigning disparity hypothesis d to pixel (u, v) in the center (left) view means matching the pixel with pixel $(u - d, v)$ in the reference (right) view, the weighted average is calculated using:

$$\frac{\sum_{m,n \in [-r,r]} (w(u, v, m, n) \cdot w'(u - d, v, m, n) \cdot C(u + m, v + n, d))}{\sum_{m,n \in [-r,r]} w(u, v, m, n) \cdot w'(u - d, v, m, n)} \quad (7)$$

where w and w' are the weights calculated based on the center and the reference images, respectively (Yoon and Kweon, 2005).

Due to its high computational cost, the original algorithm takes about a minute to generate a disparity map. Two simplifications to the algorithm are made in this paper for achieving real-time performance. The first one

is to omit the weight term obtained using the reference image. Consequently, the same weight is used when handling different disparity hypotheses of the same pixel, making it possible to compute the aggregated matching costs for different disparity hypotheses in parallel. The second simplification is to approximate the weighted average of matching costs within the 2D square window using a two-pass approach—the first pass computes the weighted average along the horizontal scanline, while the second pass computes along the vertical scanline (Wang et al., 2006). This further reduces the complexity of the aggregation approach from $O(r^2)$ to $O(r)$. As a result, in our simplified approach, the aggregated costs are calculated using:

$$T^r(u, v, d) = \frac{\sum_{m=-r}^{r} (w(u, v, m, 0) \cdot C(u + m, v, d))}{\sum_{m=-r}^{r} w(u, v, m, 0)}$$
$$A^r_{\text{weight}}(u, v, d) = \frac{\sum_{n=-r}^{r} (w(u, v, 0, n) \cdot T^r(u, v + n, d))}{\sum_{n=-r}^{r} w(u, v, 0, n)} \quad (8)$$

For a given dataset, our GPU-based implementation starts with computing all the weights for each pixel based on the color information in the center image. The weights obtained are normalized and stored into textures. Two additional steps are used to calculate the weighted averages for different pixels along horizontal and vertical scanlines, respectively. The number of rendering passes involved in these steps depends on the window size used.

The experiments show that the disparity maps we generated using the simplified approach are not as good as those reported in the original paper. However, we believe the simplification 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 performed on a 3 GHz P4 equipped with an ATI Radeon X800 graphics card. The four datasets used (Tsukuba, Venus, Teddy, and Cones) are downloaded from the Middlebury stereo evaluation site (Scharstein and Szeliski, 2002). The disparity maps generated are also evaluated using the Middlebury site with error threshold set to 1. That is, an estimated disparity is considered correct if it is within ground truth disparity ± 1. Since none of the cost aggregation approaches tries to handle half-occluded regions, the mismatches in these regions are not included.

3.1. *Effect of the Truncation Value used in Cost Initialization*

The first experiment tries to determine the effects of the truncation value $C_{\max}$ used in the cost initialization step. The accuracy of disparity maps generated using the six aggregation approaches under different truncation values are plotted in Fig. 2. The results indicate that:

- All aggregation approaches yield poor results when the absolute intensity differences are used directly as matching costs, i.e. $C_{\max}$ is set to 255. This is because matching costs are kept in 8-bit integers and small differences in cost aggregation results cannot be distinguished. Our investigation shows that representing costs using 16-bit floating point textures alleviates this problem, but leads to a huge speed penalty.
- Truncating-then-scaling the matching costs helps to obtain more accurate results since it can make better use of the range of a single byte. This is especially true for the approaches with complex computations involved, such as the boundary-guided and the adaptive-weight approaches.
- Once the truncation values drop below a threshold, the error rates increase dramatically. The actual threshold value depends on the intensity differences between corresponding pixel pairs in the input images. The Cones dataset has the highest threshold value among the four, likely due to the existence of non-Lambertian surface in the scene.
- Previous evaluation suggests that avoiding outliers by shifting window is preferable to limiting their influence using truncated cost function (Scharstein and Szeliski, 2002). However, the comparison between Fig. 2(a) and 2(b) indicates that this is no longer the case when costs are represented using 8-bit integers—shifting window actually increases the error rates.
- Generally speaking, the selection of truncation value has an important effect on result accuracy. It appears that good truncation values range from 16 to 32 for all approaches evaluated.

3.2. *Performances of Different Aggregation Approaches Under Different Settings*

The cost aggregation approaches we implemented all have parameters controlling the size of the support region to be used. To evaluate the effects of these control parameters, Fig. 3 plots the overall error rate of the disparity maps generated by different approaches with different settings. The truncation value $C_{\max}$ is set to 25 in all the experiments. 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

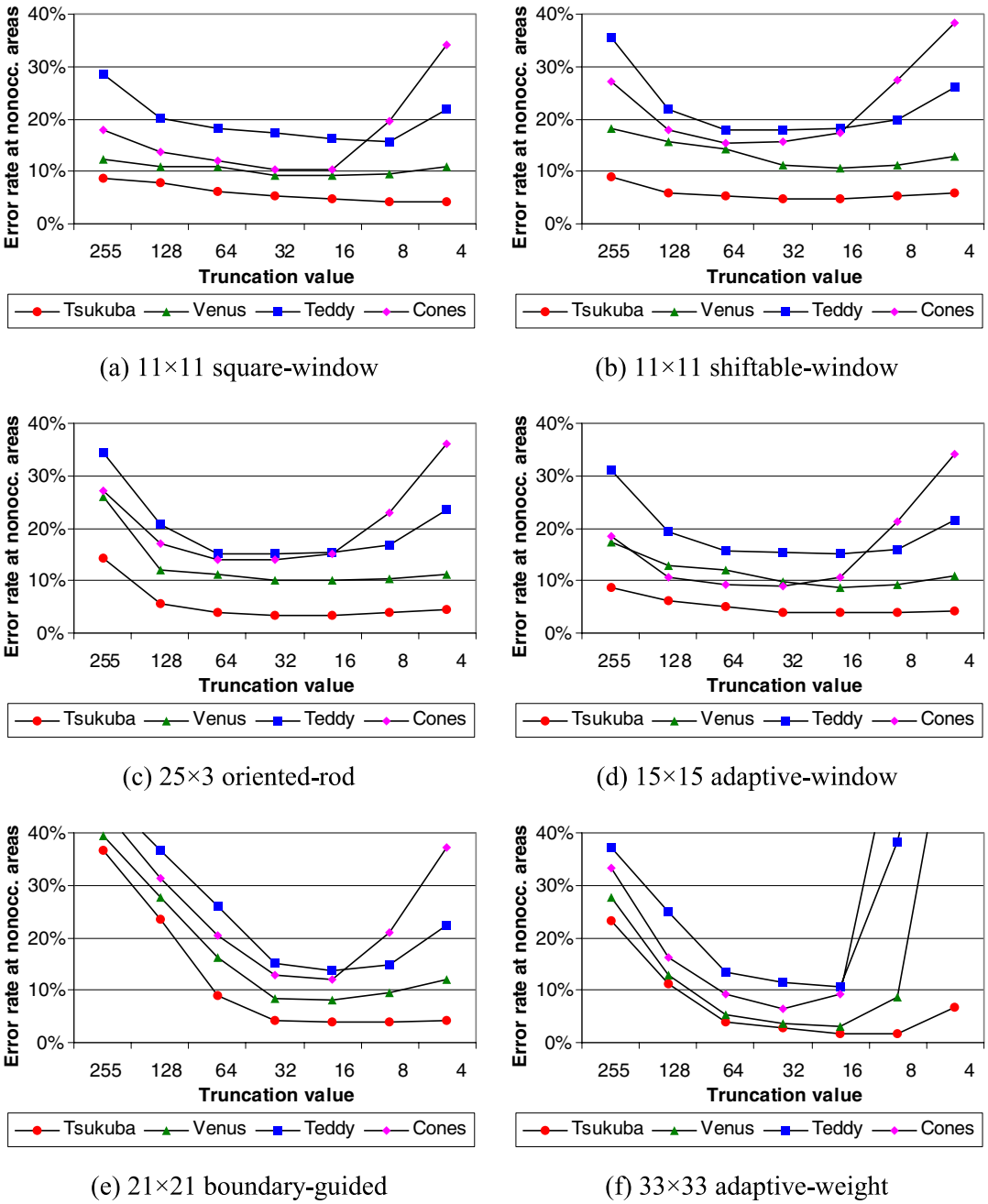

Figure 2. The error rates of disparity maps generated under different truncation values. The support regions used in different approaches are set to proper sizes, which are shown in the figure.

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.
- 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 9×9 for the Tsukuba dataset and 17×17 for the Venus one.
- The above problems are not as important for adaptive aggregation approaches as the parameters in these approaches only control the maximum window sizes. The outliers in the window may be excluded in the

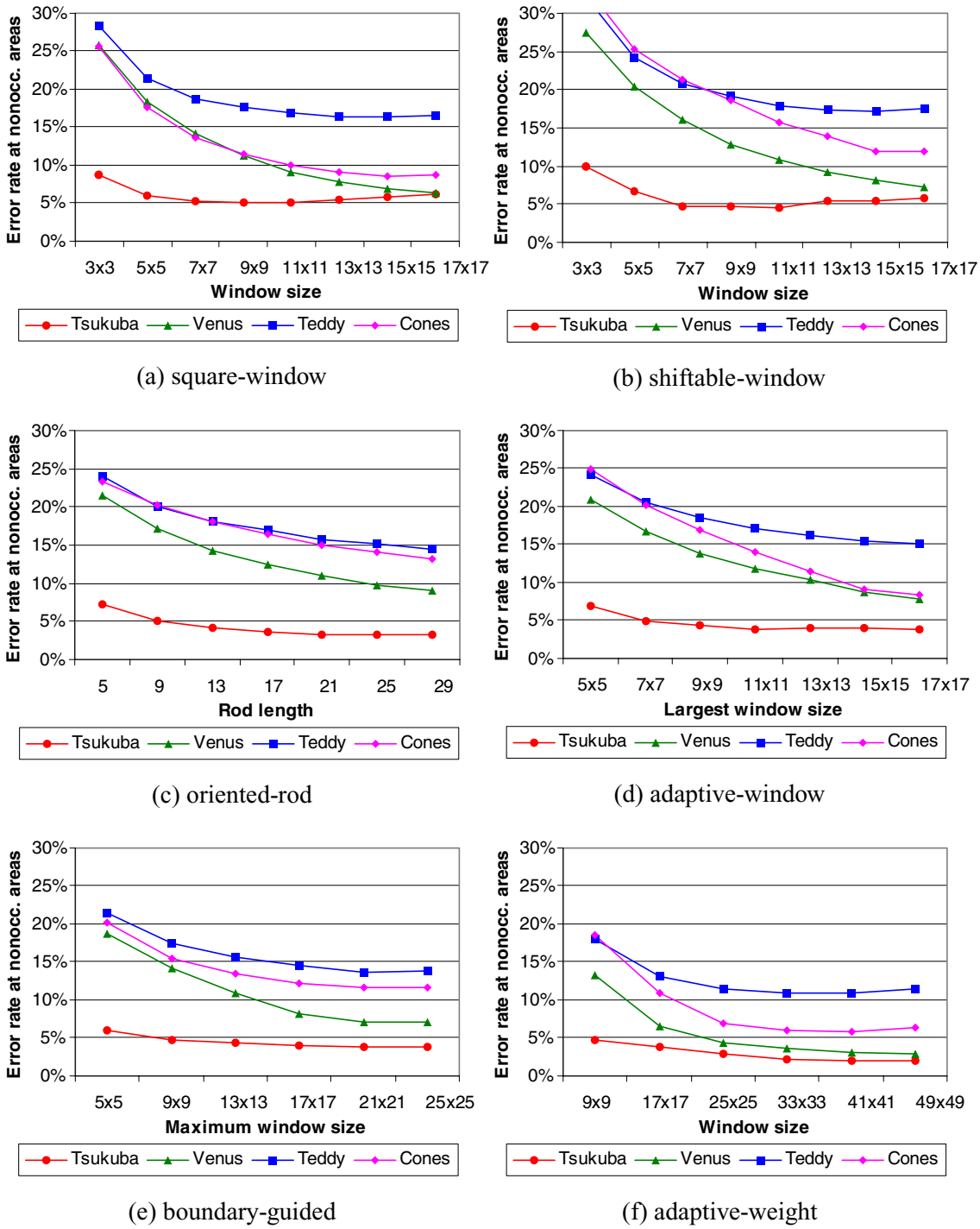

Figure 3. The overall error rates (non-occluded areas only) of disparity maps generated by different approaches.

cost calculation or may have very small weights. Their influence is therefore minimized.

To evaluate the performance of different approaches in areas near depth discontinuities, Fig. 4 plots the error rates in these regions. It suggests that:

- For the square-window approach, in three of the four datasets, 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

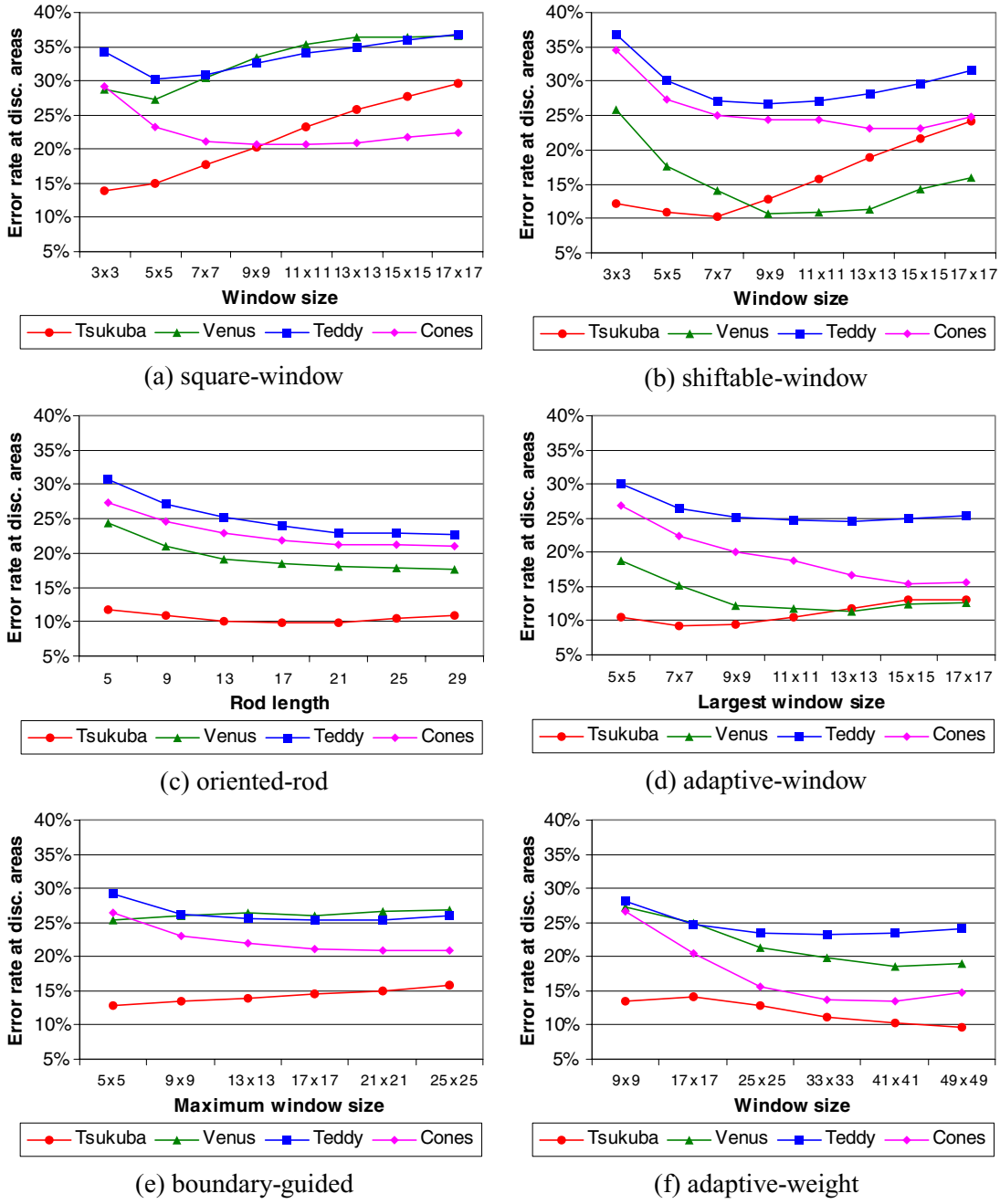

Figure 4. The error rate in areas near depth discontinuity boundaries for different approaches.

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, in three of the four datasets, increasing the length of the rod does not increase mismatches at all because it is likely that one of the oriented rods does not cross the depth discontinuity boundary.
- Adaptive aggregation approaches also perform well in near discontinuity areas. Increasing the size of support region either decreases or slightly increases mismatches in these areas.

Finally, the computation times required by different aggregation approaches as functions of the parameters used are plotted in Fig. 5. The figure supports the following observations:

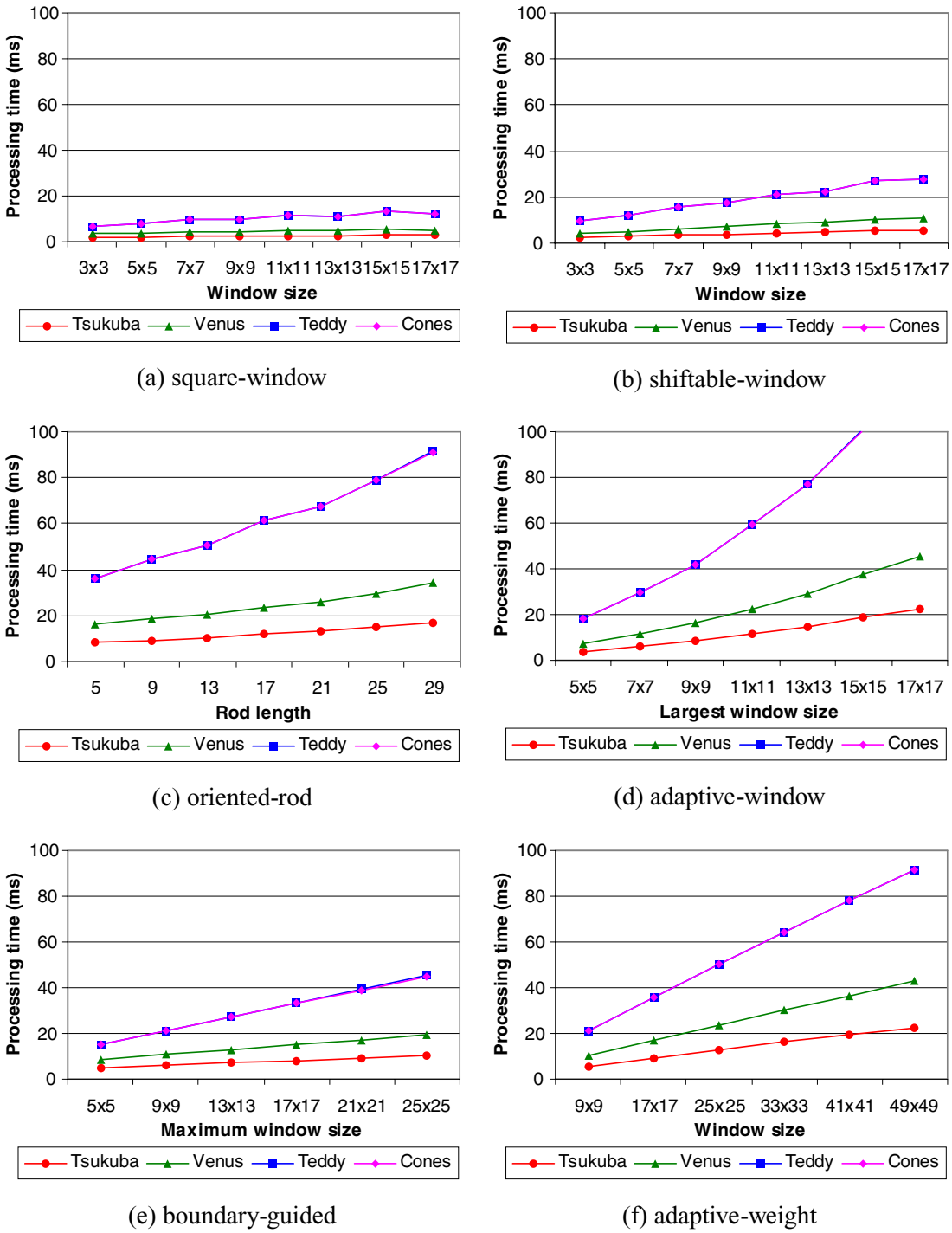

Figure 5. The computation time needed for different aggregation approaches.

- The curves for Teddy and Cone datasets, which has the same image resolution, overlap in all graphs as the computation times required for these two datasets are the same. This confirms that the processing speeds of all aggregation approaches are independent of the scene contents.
- As the size of the support region increases, the processing time increases quadratically for the adaptive-window approach and increases linearly for the remaining approaches. This agrees with the algorithm complexity analysis for different approaches.

- The total processing time is less than 100 ms under almost all settings. That is, by using the graphics hardware, more than 10 fps can be achieved for even the most demanding datasets (Teddy and Cones).

3.3. *Disparity Map Comparison*

In this section, the best settings for different aggregation approaches are picked based on the previous evaluation results and are used to compare among different approaches. In particular, we set window size to 11×11 for both square-window and shift-window approaches, to 33×33 for adaptive-weight approach, rod length to 25 for oriented-rod approach, and maximum window sizes to 15×15 and 21×21 for adaptive-window and boundary-guided approaches, respectively.

The disparity maps generated under these settings are shown in Fig. 6 for visual comparison. The numerical accuracies of these disparity maps and those generated by other related algorithms are listed in Table 1. The overall performances (speed vs. quality) of different aggregation approaches are plotted in Fig. 7. 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. As we discussed earlier, 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. The lamp arm in the Tsukuba scene shows up nicely in the disparity map and the error rates at near depth discontinuity areas are the lowest in two of the four datasets. 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 (Kim et al., 2005), this problem can be addressed by applying an additional shiftable-window filter to homogeneous pixels and/or by using DP-based optimization instead of local WTA. Nevertheless, as shown in Fig. 7, the computational cost of oriented-rod filter is already high. Making the algorithm more complex may make it unsuitable for real-time stereo matching.

In terms of accuracy, the simple adaptive-window approach we implemented performance overall better than the above three approaches. The generated disparity maps are less noisy and the depth discontinuity boundaries are

Table 1. The accuracy of disparity maps generated by different approaches as evaluated by Middlebury stereo site. (The bottom seven 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 implementation	Square-window	5.13	7.11	23.2	9.18	10.3	35.4	16.9	24.5	34.0	9.94	18.9	20.8
	Shiftable-window	4.46	6.03	15.7	10.9	11.9	**11.0**	17.9	24.6	27.1	15.8	22.9	24.3
	Oriented-rod	3.29	5.09	**10.5**	9.82	11.0	17.9	15.2	22.3	**22.9**	14.1	21.7	21.3
	Adaptive-window	4.01	5.90	13.0	8.80	10.1	12.4	15.4	23.3	24.9	9.07	18.0	15.3
	Boundary-guided	3.88	5.88	15.0	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 approach	Adaptive-weight (Yoon and Kweon, 2005)	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 (Kim et al., 2005)	1.53		8.25	0.94		5.72						
	Variable-window (Veksler, 2003)	2.35		12.2	1.23		13.4						
	Multiple-window (Hirschmuller et al., 2002)	4.25		15.0	1.53		12.3						
	Boundary-guided (Gong and Yang, R., 2005)	4.91		12.6	9.43		20.7						
Existing realtime approach	Realtime BP (Yang et al., 2006)	1.49	3.40	7.87	0.77	1.90	9.00	8.72	13.2	17.2	4.61	11.6	12.4
	Reliability-DP (Gong and Yang, Y.-H., 2005)	1.36	3.39	7.25	2.35	3.48	12.2	9.82	16.9	19.5	12.9	19.9	19.7
	Realtime DP (Forstmann et al., 2004)	2.85		15.6	6.42		25.3						
	GPU (AW4) (Yang et al., 2004)	9.68			15.7								

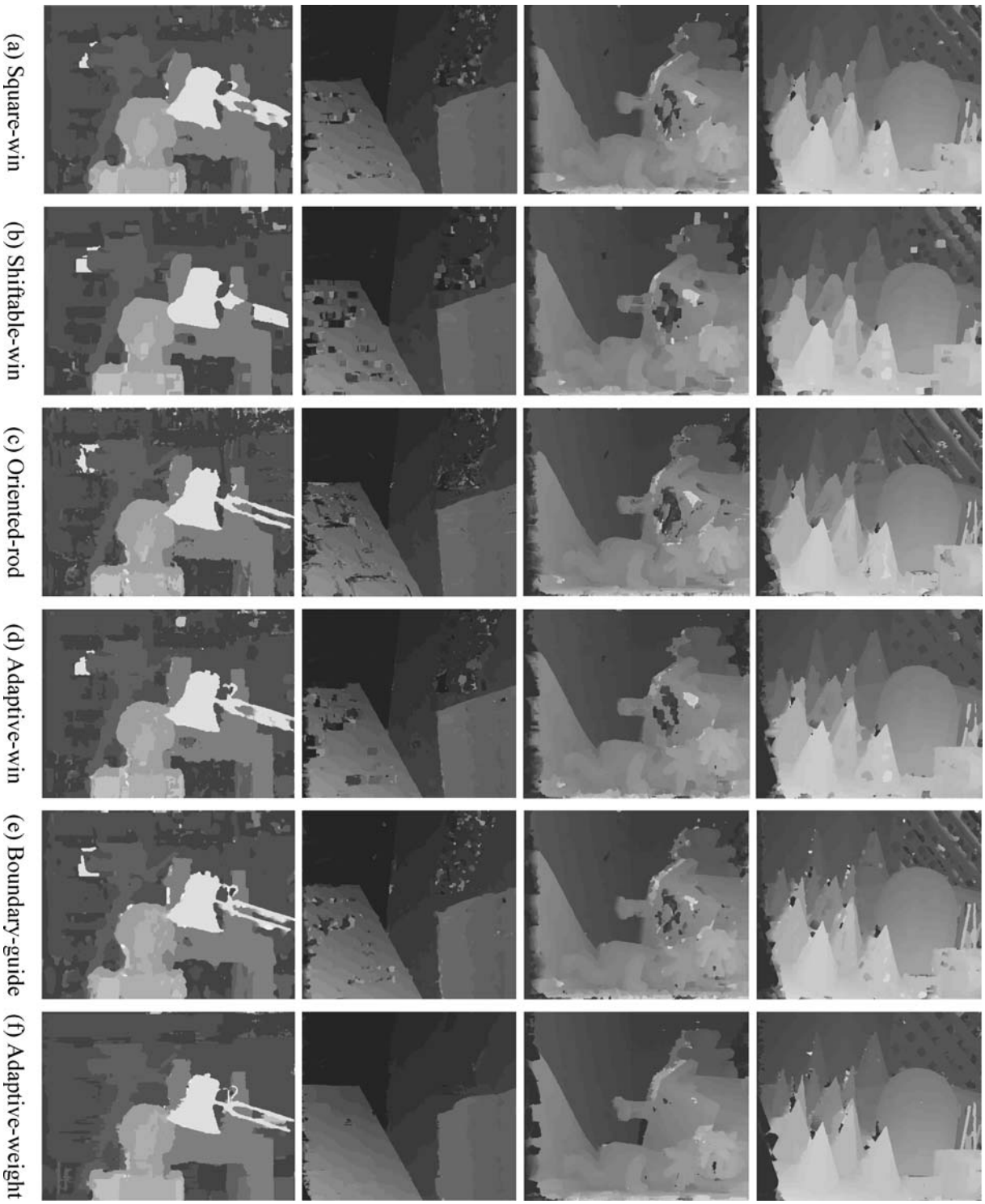

Figure 6. Disparity maps generated by different approaches using selected parameter settings.

preserved quite well. However, since our straightforward implementation requires applying the shiftable-window filter multiple times with different window sizes, it has the slowest processing speed among the six evaluated approaches. More efficient implementation may be available and will be investigated in the future.

Due to the larger support window size and the additional disparity refinement process, our implementation

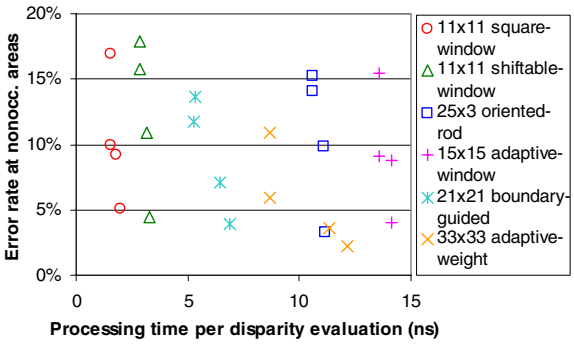

Figure 7. The speed vs. quality plot for different approaches. For boundary-guided and adaptive-weight approaches, the per disparity estimation processing time differs a lot among different datasets due to the overhead cost for calculating the support weights or for detecting edges.

for the boundary-guided approach produces slightly better disparity maps than the original approach does (Gong and Yang, R., 2005). This approach performs reasonable good in areas with enough details, but not so well in weakly textured areas. It is also much faster than the other adaptive aggregation approaches.

Among the six aggregation approaches evaluated, the adaptive-weight approach yields the most accurate results for all four datasets. The disparity maps generated are clean and detailed. Weakly textured regions, such as the roof area of the Teddy scene, are handled properly. As shown in Table 1, even though our simplified implementation does not perform as good as the original offline algorithm (Yoon and Kweon, 2005), it is already better than some real-time algorithms (Forstmann et al., 2004; Gong and Yang, R., 2005; Yang et al., 2004) and is also comparable with the DP-based near-real-time algorithm (Gong and Yang, Y.-H., 2005). Further investigation on how to better utilize adaptive-weight aggregation in real-time stereo applications seems very promising. In terms of processing speed, this approach is a bit slow though.

3.4. *Effect of Disparity Refinement Step*

In the above experiments, a median-filter-based disparity refinement step is used to remove isolated noises from disparity maps obtained with local WTA optimization. The experiment in this section evaluates how much the refinement step contributes to the final results and how well different aggregation approaches perform without the refinement.

In the experiment, the parameters for different aggregation approaches are set to the same optimal values as above. The average error rates over the four datasets are calculated and are plotted in Fig. 8. The result suggests that:

- All six aggregation approaches are benefited from the disparity refinement step. Depending on the approaches, the refinement step lowers the overall error rates in nonoccluded areas by 0.27∼0.96% and the error rates at depth discontinuity areas by 0.05∼0.58%.
- The oriented-rod and the adaptive-weight approaches benefit more from the refinement step than the remaining approaches. This indicates that, without refinement, there will be more isolated noises in the disparity maps generated using these two approaches.
- Whether or not the refinement step is applied does not affect the relative performance of different aggregation approaches.

4. Conclusions

This paper evaluates the performances of six 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

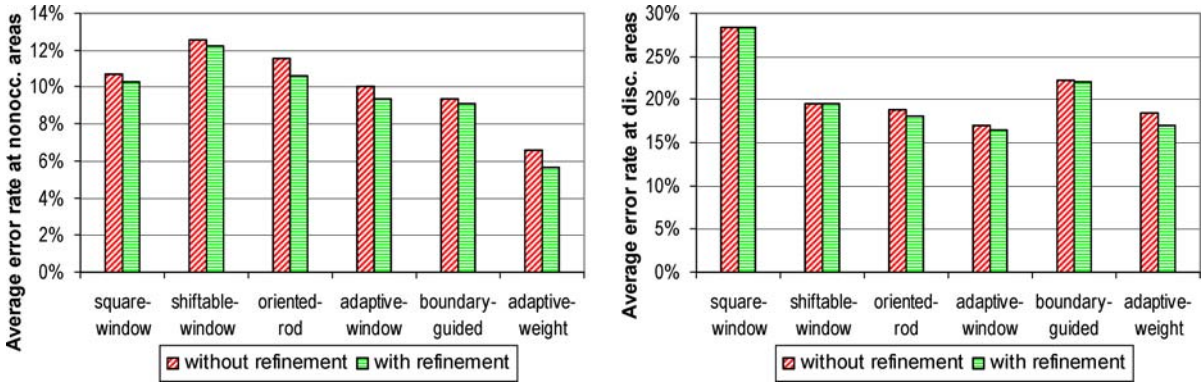

Figure 8. The quality (average error rate over four datasets) of disparity maps obtained with and without the refinement step; left: error rate at all visible areas; right: error rate at areas near depth discontinuity boundaries.

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, we find that representing matching costs using 8-bit integers—an approach widely used in real-time stereo for lower computational cost—makes accurate disparity estimation more challenging since small cost differences cannot be presented due to the very limited precision of a single byte. This precision problem can be effectively addressed by truncating-then-scaling the original matching costs obtained. Nevertheless, the resulting algorithms are more sensitive to the selection of the truncation value than their offline counterparts.

Our evaluation also suggests that properly designed cost aggregation approaches can significantly improve the quality of the disparity maps without introducing too much computational cost. For example, the simplified adaptive-weight 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 test the performances of these algorithms with fast global optimization techniques such as dynamic programming (Gong and Yang, Y.-H., 2005) and belief propagation (Yang et al., 2006) We plan to extend our study to these areas in the future.

Acknowledgments

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

References

Birchfield, S. and Tomasi, C. 1998. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406.

Bobick, A.F. and Intille, S.S. 1999. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200.

Boykov, Y., Veksler, O., and Zabih, R. 1998. A variable window approach to early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1283–1294.

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

Fusiello, A., Roberto, V., and Trucco, E. 1997. Efficient stereo with multiple windowing. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, pp. 858–863.

Gong, M. and Yang, R. 2005. Image-gradient-guided real-time stereo on graphics hardware. In *Proc. International Conference on 3-D Digital Imaging and Modeling*, Ottawa, ON, Canada, pp. 548–555.

Gong, M. and Yang, Y.-H. 2005. Near real-time reliable stereo matching using programmable graphics hardware. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, pp. 924–931.

Harwood, D., Subbarao, M., Hakalahti, H., and Davis, L. 1987. A new class of edge-preserving smoothing filters. *Pattern Recognition Letters*, 6:155–162.

Hirschmuller, H., Innocent P.R., and Garibaldi, J. 2002. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47:1–3.

Kanade, T. and Okutomi, M. 1994. Stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932.

Kim, J.-C., Lee, K.M., Choi, B.-T., and Lee, S. U. 2005. A dense stereo matching using two-pass dynamic programming with generalized ground control points. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1075–1082.

Scharstein, D. and Szeliski, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–3):7–42.

Scharstein, D. and Szeliski, R. 2003. High-accuracy stereo depth maps using structured light. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, pp. 195–202.

Tao, H. and Sawhney, H.S. 2000. Global matching criterion and color segmentation based stereo. In *Proc. Workshop on the Application of Computer Vision*, pp. 246–253.

Veksler, O. 2003. Fast variable window for stereo correspondence using integral images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 556–561.

Wang, L. Gong, M., Gong, M., and Yang, R. 2006. How far can we go with local optimization in real-time stereo matching: A performance study on different cost aggregation approaches. In *Proc. International Symposium on 3D Data Processing, Visualization and Transmission*, Chapel Hill, NC, USA.

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

Wang, L., Liao, M., Gong, M., Yang, R., and Nister, D. 2006. High quality real-time stereo using adaptive cost aggregation and dynamic programming. In *Proc. International Symposium on 3D Data Processing, Visualization and Transmission*, Chapel Hill, NC, USA.

Woodfill, J. and Von Herzen, B. 1997. Real-time stereo vision on the PARTS reconfigurable computer. In *Proc. IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 201–210.

Yang, Q., Wang, L., Yang, R., Wang, S., Liao, M., and Nister, D. 2006. Real-time global stereo matching using hierarchical belief propagation. In *Proc. British Machine Vision Conference*, Edinburgh, UK.

Yang R. and Pollefeys, M. 2003. Multi-resolution real-time stereo on commodity graphics hardware. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, USA, pp. 211–220.

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

Yoon, K.-J. and Kweon, I.-S. 2005. Locally adaptive support-weight approach for visual correspondence search. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 924–931.