On Building an Accurate Stereo Matching System on Graphics Hardware

Xing Mei[1,2], Xun Sun[1], Mingcai Zhou[1], Shaohui Jiao[1], Haitao Wang[1], Xiaopeng Zhang[2]
[1]Samsung Advanced Institute of Technology, China Lab
[2]LIAMA-NLPR, Institute of Automation, Chinese Academy of Sciences
{xing.mei,xunshine.sun,mingcai.zhou,sh.jiao,ht.wang}@samsung.com,xpzhang@nlpr.ia.ac.cn

Abstract

This paper presents a GPU-based stereo matching system with good performance in both accuracy and speed. The matching cost volume is initialized with an AD-Census measure, aggregated in dynamic cross-based regions, and updated in a scanline optimization framework to produce the disparity results. Various errors in the disparity results are effectively handled in a multi-step refinement process. Each stage of the system is designed with parallelism considerations such that the computations can be accelerated with CUDA implementations. Experimental results demonstrate the accuracy and the efficiency of the system: currently it is the top performer in the Middlebury benchmark, and the results are achieved on GPU within 0.1 *seconds. We also provide extra examples on stereo video sequences and discuss the limitations of the system.*

1. Introduction

Stereo matching is one of the most extensively studied problems in computer vision [11]. Two major concerns in stereo matching algorithm design are the matching accuracy and the processing efficiency. Although many algorithms are introduced every year, the two concerns tend to be contradictory in the reported results: accurate stereo methods are usually time consuming [6, 17, 20], while GPU-based methods achieve high processing speed with relatively low disparity precision [10, 18, 24]. To the best of our knowledge, most top 10 Middlebury algorithms require at least 10 seconds to process a 384×288 image pair, and the only two GPU-based methods in top 20 are CostFilter [9] and PlaneFitBP [19], both working with near real-time performance.

The reason behind this contradiction is straightforward: some key techniques employed by accurate stereo algorithms are not suitable for parallel GPU implementation. A careful analysis of the leading Middlebury algorithms [6, 17, 20] shows that these algorithms have several common techniques in the matching process: they use large support windows for robust cost aggregation [5, 16, 21]; they formulate the disparity computation step as an energy minimization problem and solve it with slow-converging optimizers [14]; they extensively use segmented image regions as matching units [17], surface constraints [6, 20] or post-processing patches [2]. These techniques significantly improve the matching quality at the cost of considerable computation costs. Directly porting these techniques to GPU or other multi-core platforms, however, is tricky and cumbersome [4, 7, 18, 19]: large aggregation windows require extensive iterations over each pixel; some optimization, segmentation and post-processing methods require complex data structures and sequential processing. As a result, simple correlation-based techniques are more popular for GPU and embedded stereo systems. Designing a stereo matching system with good balance between accuracy and efficiency remains a challenging problem.

In this paper, we aim to meet this challenge by providing an accurate stereo matching system with near real-time performance. Currently (August 2011), our system is the top performer in the Middlebury benchmark. Briefly, we integrate several techniques into an effective stereo framework. These techniques guarantee high matching quality without involving expensive aggregation or segmented regions. Furthermore, they show moderate parallelism such that the entire system can be mapped on GPU for computation acceleration. The key techniques of our system include:

- An AD-census cost measure which effectively combines the absolute differences (AD) measure and the census transform. This measure provides more accurate matching results than common individual measures with robust aggregation methods. A similar measure has been adopted in a recent stereo algorithm [13].
- Improved cross-based regions for efficient cost aggregation. Support regions based on cross skeletons are first proposed by Zhang *et al.* [23], which allow fast aggregation with middle-ranking disparity results. We enhance this technique with more accurate cross construction and cost aggregation strategy.
- A scanline optimizer based on Hirschmüller's semi-

global matcher [2] with reduced path directions.

- A systematic refinement process which handles various disparity errors with iterative region voting, interpolation, depth discontinuity adjustment and sub-pixel enhancement. This multi-step process proves to be very effective for improving the disparity results.
- Efficient system implementation on GPU with CUDA.

2. Algorithm

Following Scharstein and Szeliski's taxonomy [11], our system consists of four steps: cost initialization, cost aggregation, disparity computation and refinement. We present a detailed description of these individual steps.

2.1. AD-Census Cost Initialization

This step computes the initial matching cost volume. Since the computation can be performed concurrently at each pixel and each disparity level, this step is inherently parallel. Our major concern is to develop a cost measure with high matching quality. Common cost measures include absolute differences (AD), Birchfield and Tomasi's sampling-insensitive measure (BT), gradient-based measures and non-parametric transforms such as rank and census [22]. In a recent evaluation by Hirschmüller and Scharstein [3], census shows the best overall results in local and global stereo matching methods. Although the idea of combining cost measures for improved accuracy seems straightforward, relatively less work has explored this topic. Klaus *et al.* [6] proposed to linearly combines SAD and a gradient based measure for cost computation. Their disparity results are impressive, but the benefits of the combination are not clearly elaborated.

Census encodes local image structures with relative orderings of the pixel intensities other than the intensity values themselves, and therefore tolerates outliers due to radiometric changes and image noise. However, this asset could also introduce matching ambiguities in image regions with repetitive or similar local structures. To handle this problem, more detailed information should be incorporated in the measure. For image regions with similar local structures, the color (or intensity) information might help alleviate the matching ambiguities; while for regions with similar color distributions, the census transform over a window is more robust than pixel-based intensity difference. This observation inspires a combined measure.

Given a pixel $\mathbf{p} = (x, y)$ in the left image and a disparity level d, two individual cost values $C_{census}(\mathbf{p}, d)$ and $C_{AD}(\mathbf{p}, d)$ are first computed. For C_{census}, we use a 9×7 window to encode each pixel's local structure in a 64-bit string. $C_{Census}(\mathbf{p}, d)$ is defined as the Hamming distance of the two bit strings that stand for pixel $\mathbf{p}$ and its correspondence $\mathbf{pd} = (x - d, y)$ in the right image [22]. C_{AD} is defined as the average intensity difference of $\mathbf{p}$ and $\mathbf{pd}$ in RGB channels:

$$C_{AD}(\mathbf{p}, d) = \frac{1}{3} \sum_{i=R,G,B} |I_i^{Left}(\mathbf{p}) - I_i^{Right}(\mathbf{pd})| \quad (1)$$

The AD-Census cost value $C(\mathbf{p}, d)$ is then computed as follows:

$$\begin{aligned} C(\mathbf{p}, d) = \rho(C_{census}(\mathbf{p}, d), \lambda_{census}) + \\ \rho(C_{AD}(\mathbf{p}, d), \lambda_{AD}) \end{aligned} \quad (2)$$

where $\rho(c, \lambda)$ is a robust function on variable c:

$$\rho(c, \lambda) = 1 - \exp(-\frac{c}{\lambda}) \quad (3)$$

The purpose of the function is twofold: first, it maps different cost measures to the range $[0, 1]$, such that equation (2) won't be severely biased by one of the measures; second, it allows easy control on the influence of the outliers with parameter λ.

To verify the effect of the combination, some close-up disparity results on the Middlebury data sets with AD, Census and AD-Census are presented in Figure 1. Cross-based aggregation is employed. Census produces wrong matches in regions with repetitive local structures, while pixel-based AD can not handle well large textureless regions. The combined AD-Census measure successfully reduces the errors caused by individual measures. For quantitative comparison, AD-Census reduces Census's non-occlusion error percentage by 1.96% (Tsukuba), 0.4% (Venus), 1.36% (Teddy) and 1.52% (Cones) respectively. And this improvement comes at low additional computation cost.

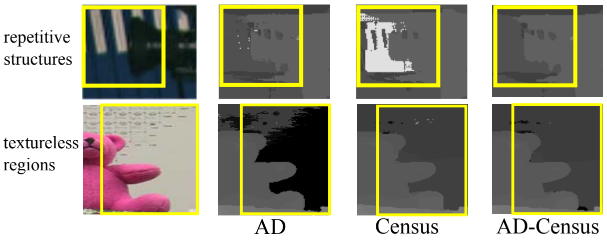
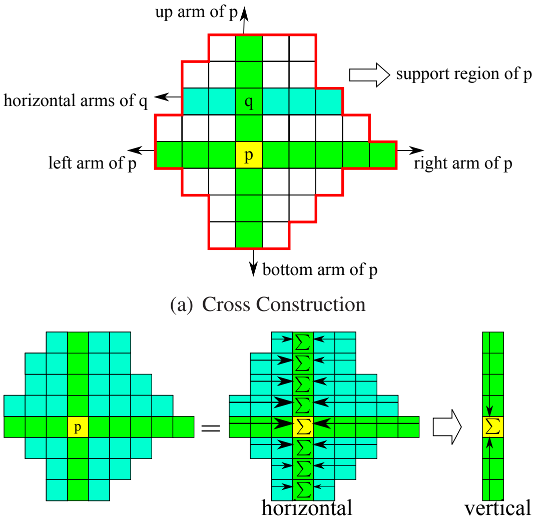


Figure 1. Some close-up disparity results on Tsukuba and Teddy image pair, which are computed with AD, Census, AD-Census cost measures and cross-based aggregation. AD-Census measure produces proper disparity results for both repetitive structures and textureless regions.

2.2. Cross-based Cost Aggregation

This step aggregates each pixel's matching cost over a support region to reduce the matching ambiguities and noise in the initial cost volume. A simple but effective assumption for aggregation is that neighboring pixels with similar colors should have similar disparities. This assumption has been adopted by recent aggregation methods, such as segment support [16], adaptive weight [21] and geodesic weight [5]. As stated in the introduction, these aggregation methods require segmentation operations or expensive iterations over each pixel, which are prohibitive for efficient GPU implementation. Although simplified adaptive weight techniques with 1D aggregation [13, 18] and color averaging [4, 19] have been proposed for GPU systems, the aggregation accuracy usually degenerates. Recently, Rhemann *et al.* [9] formulated the aggregation step as a cost filtering problem. By smoothing each cost slice with a guided filter [1], good disparity results can be achieved.

We instead focus on the cross-based aggregation method proposed recently by Zhang *et al.* [23]. We show that by improving support region construction and aggregation strategy, this method can produce aggregated results comparable to the adaptive weight method with much less computation time. Another advantage over the adaptive weight method is that an explicit support region is constructed for each pixel, which can be used in later post-processing steps.



(a) Cross Construction

(b) Cost Aggregation

Figure 2. Cross-based aggregation: In the first step, an upright cross is constructed for each pixel. The support region of pixel $\mathbf{p}$ is modelled by merging the horizontal arms of the pixels ($\mathbf{q}$ for example) lying on the vertical arms of pixel $\mathbf{p}$. In the second step, the cost in the support region is aggregated within two passes along the horizontal and vertical directions.

Cross-based aggregation proceeds by a two-step process, as shown in Figure 2. In the first step (Figure 2(a)), an upright cross with four arms is constructed for each pixel. Given a pixel $\mathbf{p}$, its left arm stops when it finds an endpoint pixel $\mathbf{p_l}$ that violates one of the two following rules:

1. $D_c(\mathbf{p_l}, \mathbf{p}) < \tau$, where $D_c(\mathbf{p_l}, \mathbf{p})$ is the color difference between $\mathbf{p_l}$ and $\mathbf{p}$, and τ is a preset threshold value. The color difference is defined as $D_c(\mathbf{p_l}, \mathbf{p}) = \max_{i=R,G,B} |I_i(\mathbf{p_l}) - I_i(\mathbf{p})|$.
2. $D_s(\mathbf{p_l}, \mathbf{p}) < L$, where $D_s(\mathbf{p_l}, \mathbf{p})$ is the spatial distance between $\mathbf{p_l}$ and $\mathbf{p}$, and L is a preset maximum length L (measured in pixels). The spatial distance is defined as $D_s(\mathbf{p_l}, \mathbf{p}) = |\mathbf{p_l} - \mathbf{p}|$.

The two rules pose limitations on color similarity and arm length with parameter τ and L. The right, up and bottom arms of $\mathbf{p}$ are built in a similar way. After the cross construction step, the support region for pixel $\mathbf{p}$ is modelled by merging the horizontal arms of all the pixels lying on $\mathbf{p}$'s vertical arms ($\mathbf{q}$ for example). In the second step (Figure 2(b)), the aggregated costs over all pixels are computed within two passes: the first pass sums up the matching costs *horizontally* and stores the intermediate results; the second pass then aggregates the intermediate results *vertically* to get the final costs. Both passes can be efficiently computed with 1D integral images. To get a stable cost volume, the aggregation step usually runs $2 - 4$ iterations, which can be seen as an anisotropic diffusion process. More details about the method can be found in [23].

The accuracy of cross-based aggregation is closely related to parameter L and τ, since they control the shape of the support regions with the construction rules. Large textureless regions may require large L and τ values to include enough intensity variation, but simply increasing these parameters for all the pixels would introduce more errors in dark regions or at depth discontinuities. We therefore propose to construct each pixel's cross with the following enhanced rules (we still use pixel $\mathbf{p}$'s left arm and the endpoint pixel $\mathbf{p_l}$ as an example):

1. $D_c(\mathbf{p_l}, \mathbf{p}) < \tau_1$ and $D_c(\mathbf{p_l}, \mathbf{p_l} + (1, 0)) < \tau_1$
2. $D_s(\mathbf{p_l}, \mathbf{p}) < L_1$
3. $D_c(\mathbf{p_l}, \mathbf{p}) < \tau_2$, if $L_2 < D_s(\mathbf{p_l}, \mathbf{p}) < L_1$.

Rule 1 restricts not only the color difference between $\mathbf{p_l}$ and $\mathbf{p}$, but also the color difference between $\mathbf{p_l}$ and its predecessor $\mathbf{p_l} + (1, 0)$ on the same arm, such that the arm won't run across the edges in the image. Rule 2 and 3 allow more flexible control on the arm length. We use a large L_1 value to include enough pixels for textureless regions. But when the arm length exceeds a preset value L_2 ($L_2 < L_1$), a much stricter threshold value τ_2 ($\tau_2 < \tau_1$) is used for $D_c(\mathbf{p_l}, \mathbf{p})$ to make sure that the arm only extends in regions with very similar color patterns.

For the cost aggregation step, we also propose a different strategy. We still run this step for 4 iterations to get stable cost values. For iteration 1 and 3, we follow the original method: the costs are first aggregated *horizontally* and then *vertically*. But for iteration 2 and 4, we switch the aggregation directions: the costs are first aggregated *vertically* and then *horizontally*. For each pixel, this new aggregation order leads to a cross-based support region different from the one in the original method. By altering the aggregation directions, both support regions are used in the iterative process. We find that such an aggregation strategy can significantly reduce the errors at depth discontinuities.

The Tsukuba disparity results computed by the original cross-based aggregation method and our improved method are presented in Figure 3, which shows that the enhanced cross construction rules and aggregation strategy can produce more accurate results in large textureless regions and near depth discontinuities.

The WTA disparity results with three aggregation methods (adaptive weight, the original cross-based aggregation method and our enhanced method) are evaluated. For adaptive weight, the parameters follow the settings in [21]. For the original cross-based method, $L = 17, \tau = 20$ and 4 iterations are used. The average error percentages on the four data sets (in *non-occlusion*, *discontinuity* and *all* regions) are given in Figure 4. Our enhanced method produces the most accurate results in all kinds of regions, especially around depth discontinuities. Our implementation of the adaptive weight method usually takes more than 1 minute on CPU to produce the aggregated volume, while our method requires only a few seconds.

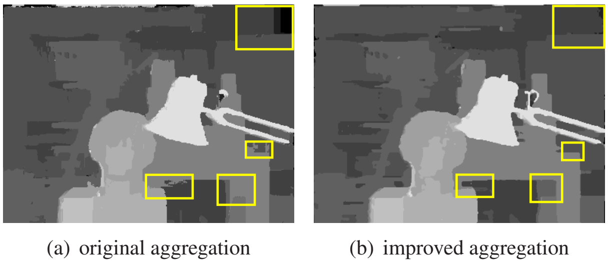

(a) original aggregation
(b) improved aggregation

Figure 3. Comparison of the original cross-based aggregation method and our improved method on the Tsukuba image pair. Our aggregation method can better handle large textureless regions and depth discontinuities.

2.3. Scanline Optimization

This step takes in the aggregated matching cost volume (denoted as C_1) and computes the intermediate disparity results. To further alleviate the matching ambiguities, an optimizer with smoothness constraints and moderate parallelism should be adopted. We employ a multi-direction scanline optimizer based on Hirschmüller's semi-global matching method [2].

Four scanline optimization processes are performed independently: 2 along horizontal directions and 2 along vertical directions. Given a scanline direction $\mathbf{r}$, the path cost

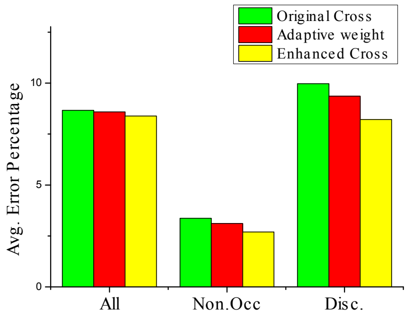


Figure 4. The average disparity error percentages in various regions for adaptive weight, the original cross-based aggregation method and our enhanced method.

$C_{\mathbf{r}}(\mathbf{p}, d)$ at pixel $\mathbf{p}$ and disparity d is updated as follows:

$$\begin{aligned} C_{\mathbf{r}}(\mathbf{p}, d) = C_1(\mathbf{p}, d) + \min(&C_{\mathbf{r}}(\mathbf{p}-\mathbf{r}, d), \\ &C_{\mathbf{r}}(\mathbf{p}-\mathbf{r}, d \pm 1) + P_1, \\ &\min_k C_{\mathbf{r}}(\mathbf{p}-\mathbf{r}, k) + P_2) - \min_k C_{\mathbf{r}}(\mathbf{p}-\mathbf{r}, k) \end{aligned} \tag{4}$$

where $\mathbf{p} - \mathbf{r}$ is the previous pixel along the same direction, and P_1, P_2 $(P_1 \leq P_2)$ are two parameters for penalizing the disparity changes between neighboring pixels. In practice, P_1, P_2 are symmetrically set according to the color difference $D_1 = D_c(\mathbf{p}, \mathbf{p} - \mathbf{r})$ in the left image and $D_2 = D_c(\mathbf{pd}, \mathbf{pd} - \mathbf{r})$ in the right image [8]:

1. $P_1 = \Pi_1, P_2 = \Pi_2$, if $D_1 < \tau_{SO}, D_2 < \tau_{SO}$.
2. $P_1 = \Pi_1/4, P_2 = \Pi_2/4$, if $D_1 < \tau_{SO}, D_2 > \tau_{SO}$.
3. $P_1 = \Pi_1/4, P_2 = \Pi_2/4$, if $D_1 > \tau_{SO}, D_2 < \tau_{SO}$.
4. $P_1 = \Pi_1/10, P_2 = \Pi_1/10$, if $D_1 > \tau_{SO}, D_2 > \tau_{SO}$.

where Π_1, Π_2 are constants, and τ_{SO} is a threshold value for color difference. The final cost $C_2(\mathbf{p}, d)$ for pixel $\mathbf{p}$ and disparity d is obtained by averaging the path costs from all four directions:

$$C_2(\mathbf{p}, d) = \frac{1}{4} \sum_{\mathbf{r}} C_{\mathbf{r}}(\mathbf{p}, d) \tag{5}$$

The disparity with the minimum C_2 value is selected as pixel $\mathbf{p}$'s intermediate result.

2.4. Multi-step Disparity Refinement

The disparity results of both images (denoted as D_L and D_R) computed by the previous three steps contain outliers in occlusion regions and at depth discontinuities. After detecting these outliers, the simplest refinement method is to fill them with nearest reliable disparities [11], which is only useful for small occlusion regions. We instead handle the disparity errors systematically in a multi-step process. Each step tries to remove the errors caused by various factors.

Outlier Detection: The outliers in D_L are first detected with left-right consistency check: pixel $\mathbf{p}$ is an outlier if $D_L(\mathbf{p}) = D_R(\mathbf{p} - (D_L(\mathbf{p}), 0))$ doesn't hold. Outliers are further classified into occlusion and mismatch points, since they require different interpolation strategy. We follow the method proposed by Hirschmüller [2]: for outlier $\mathbf{p}$ at disparity $D_L(\mathbf{p})$, the intersection of its epipolar line and D_R is checked. If no intersection is detected, $\mathbf{p}$ is labelled as 'occlusion', otherwise 'mismatch'.

Iterative Region Voting: The detected outliers should be filled with reliable neighboring disparities. Most accurate stereo algorithms employ segmented regions for outlier handling [2, 20], which are not suitable for GPU implementation. We process these outliers with the constructed cross-based regions and a robust voting scheme.

For an outlier pixel $\mathbf{p}$, all the reliable disparities in its its cross-based support region are collected to build a histogram $H_{\mathbf{p}}$ with $d_{\max} + 1$ bins. The disparity with the highest bin value (most votes) is denoted as $d^*_{\mathbf{p}}$. And the total number of the reliable pixels is denoted as $S_{\mathbf{p}} = \sum_{d=0}^{d=d_{\max}} H_{\mathbf{p}}(d)$. $\mathbf{p}$'s disparity is then updated with $d^*_{\mathbf{p}}$ if enough reliable pixels and votes are found in the support region:

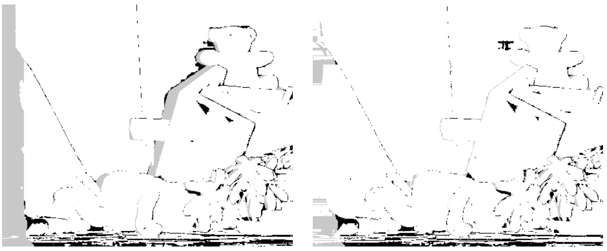
$$S_{\mathbf{p}} > \tau_S, \frac{H_{\mathbf{p}}(d^*_{\mathbf{p}})}{S_{\mathbf{p}}} > \tau_H \tag{6}$$

where τ_S, τ_H are two threshold values.

To process as many outliers as possible, the voting process runs for 5 iterations. The filled outliers are marked as 'reliable' pixels and used in the next iteration, such that valid disparity information can gradually propagate into occlusion regions.

Proper Interpolation: The remaining outliers are filled with a interpolation strategy that treats occlusion and mismatch points differently. For outlier $\mathbf{p}$, we find the nearest reliable pixels in 16 different directions. If $\mathbf{p}$ is an occlusion point, the pixel with the lowest disparity value is selected for interpolation, since $\mathbf{p}$ most likely comes from the background; otherwise the pixel with the most similar color is selected for interpolation. With region voting and interpolation, most outliers are effectively removed from the disparity results, as shown in Figure 5.

Depth Discontinuity Adjustment: In this step, the disparities around the depth discontinuities are further refined with neighboring pixel information. We first detect all the edges in the disparity image. For each pixel $\mathbf{p}$ on the disparity edge, two pixels $\mathbf{p}_1$, $\mathbf{p}_2$ from both sides of the edge are collected. $D_L(\mathbf{p})$ is replaced by $D_L(\mathbf{p}_1)$ or $D_L(\mathbf{p}_2)$ if one of the two pixels has smaller matching cost than $C_2(\mathbf{p}, D_L(\mathbf{p}))$. This simple method helps to reduce the small errors around discontinuities, as shown by the error maps in Figure 6.



(a) before outlier handling (b) after outlier handling

Figure 5. The disparity error maps for the Teddy image pair. The errors are marked in gray (occlusion) and black (non occlusion). The disparity errors are significantly reduced in the outlier handling process.

(a) before discontinuity adjustment (b) after discontinuity adjustment

Figure 6. The errors around depth discontinuities are reduced after the adjustment step.

Sub-pixel Enhancement: Finally, a sub-pixel enhancement process based on quadratic polynomial interpolation is performed to reduce the errors caused by discrete disparity levels [20]. For pixel $\mathbf{p}$, its interpolated disparity d^* is computed as follows:

$$d^* = d - \frac{C_2(\mathbf{p}, d_+) - C_2(\mathbf{p}, d_-)}{2(C_2(\mathbf{p}, d_+) + C_2(\mathbf{p}, d_-) - 2C_2(\mathbf{p}, d))} \tag{7}$$

where $d = D_L(\mathbf{p}), d_+ = d + 1, d_- = d - 1$. The final disparity results are obtained by smoothing the interpolated disparity results with a 3×3 median filter.

To verify the effectiveness of the refinement process, the average error percentages in various regions after performing each refinement step are presented in Figure 7. The four refinement steps successfully reduce the error percentage in *all* regions by 3.8%, but their contributions are distinct for different regions: for *non-occluded* regions, voting and sub-pixel enhancement are most effective for handling the mismatch outliers; for *discontinuity* regions, the errors are significantly reduced by voting, discontinuity adjustment and sub-pixel enhancement; most outliers in *all* regions are removed with voting and interpolation, and small errors due to discontinuities and quantization are reduced by adjustment and sub-pixel enhancement. A systematic integration of these steps guarantees a strong post-processing method.

3. CUDA Implementation

Compute Unified Device Architecture (CUDA) is a programming interface for parallel computation tasks on NVIDIA graphics hardware. The computation task is coded into a *kernel* function, which is performed concurrently on data elements by multiple threads. The allocation of the threads is controlled with two hierarchical concepts: *grid* and *block*. A $kernel$ creates a *grid* with multiple *block*s, and each *block* consists of multiple threads. The performance of the CUDA implementation is closely related to thread allocation and memory accesses, which needs careful tuning in various computation tasks and hardware platforms. Given image resolution $W \times H$ and disparity range D, we briefly describe the implementation issues of our algorithm.

Cost Initialization: This step is parallelized with $W \times H$ threads. The threads are organized into a 2D grid and the block size is set to 32×32. Each thread takes care of computing a cost value for a pixel at a given disparity. For census transform, a square window is require for each pixel, which requires loading more data into the shared memory for fast access.

Cost Aggregation: A grid with $W \times H$ threads is created for both steps of the aggregation process. For cross construction, we set the block size to W or H, such that each block can efficiently handle a scanline. For cost aggregation, we follow the method proposed by Zhang *et al*. [24], which works similar to the first step. Each thread sums up a pixel's cost values horizontally and vertically in two passes. Data reuse with shared memory is considered in both steps.

Scanline Optimization: This step is different from the previous steps, because the process is sequential in the scanline direction and parallel in the orthogonal direction. A grid with $W \times D$ or $H \times D$ threads is created according to the scanline direction. D threads are allocated for each scanline, such that path costs on all disparity levels can be computed concurrently. Synchronization between the D threads is needed for finding the minimum cost of the previous pixel on the same path.

Disparity Refinement: Each step of the refinement process works on the intermediate disparity images, which can be efficiently processed with $W \times H$ threads.

4. Experimental Results

We test our system with the Middlebury benchmark [12]. The test platform is a PC with Core2Duo 2.20GHz CPU and NVIDIA GeForce GTX 480 graphics card. The parameters are given in Table 1, which are kept constant for all the data sets.

The disparity results are presented in Figure 8. Our system ranks first in the Middlebury evaluation, as shown in Table 2. Our algorithm performs well on all the data sets, and gives the best results on the Venus image pair with minimum errors both in non-occluded regions and near depth discontinuities. Compared to algorithms such as CoopRegion [17], the results on the Tsukuba image pair are not competitive. The Tsukuba image pair contains some very dark and noisy regions near the lamp and the desk, which lead to incorrect cross-based support regions for aggregation and refinement.

λ_{AD}	λ_{Census}	L_1	L_2	τ_1	τ_2
10	30	34	17	20	6
Π_1	Π_2	τ_{SO}	τ_S	τ_H	
1.0	3.0	15	20	0.4	

Table 1. Parameter settings for the Middlebury experiments

We run the algorithm both on CPU and on graphics hardware. For the four data sets (Tsukuba, Venus, Teddy and Cones), the CPU implementation requires 2.5 seconds, 4.5 seconds, 15 seconds and 15 seconds respectively, while the GPU implementation requires only 0.016 seconds, 0.032 seconds, 0.095 seconds and 0.094 seconds respectively. The GPU-friendly system design brings an impressive 140× speedup in the processing speed. The average proportions of the GPU running time for the four computation steps are 1%, 70%, 28% and 1% respectively. The iterative cost aggregation step and the scanline optimization process dominate the running time.

Finally, we test our system on two stereo video sequences: a 'book arrival' scene from the HHI database (512×384, 60 disparity levels), and an 'Ilkay' scene from Microsoft i2i database (320×240, 50 disparity levels). To test the generalization ability of the system, we use the same set of parameters as the Middlebury datasets, and no temporal coherence information is employed in the computation process. The snapshots for the two examples are presented in Figure 9, and a video demo that runs at about 10FPS is available at `http://xing-mei.net/resource/video/adcensus.avi`. Our system performs reasonably well on these examples, but the results are not as convincing as the Middlebury datasets: artifacts are visible around depth boarders and occlusion regions.

We briefly discuss the limitations of the current system with the video examples. The disparity errors come from several aspects: first, the support regions defined by the cross skeleton rely heavily on color and connectivity constraints. For practical scenes the cross construction process can be easily corrupted by dark regions and image noise. Small regions without enough support area can be produced, which brings significant errors for later computation steps such as cost computation and region voting. Bilateral filtering might be used as a pre-process to reduce the noise while preserving the image edges [1, 15]. Second, the well-designed multi-stage mechanism is a double-edged sword. It help us to get accurate results and remove the errors step by step in a systematic way, but it also brings a large set of parameters. By carefully tuning individual parameters, the disparity quality can be improved, but such a scheme is usually laborious and impractical for various real-world applications. A possible solution is to analyze the robustness of the parameters with ground truth data and adaptively set the 'unstable' parameters with different visual contents. Automatic parameter estimation within an iterative framework [25] might also be used to avoid the tricky parameter tuning process.

5. Conclusions

This paper has presented a near real-time stereo system with accurate disparity results. Our system is based on several key techniques: AD-Census cost measure, cross-based support regions, scanline optimization and a systematic refinement process. These techniques significantly improve the disparity quality without sacrificing performance and parallelism, which are suitable for GPU implementation. Although our system presents nice results for the Middlebury data sets, applying it in real world applications is still a challenging task, as shown by the video examples. Real world data usually contains significant image noise, rectification errors and illumination variation, which might cause serious problems for cost computation and support region construction. And robust parameter setting methods are also important to produce satisfactory results. We would like to explore these topics in the future.

Acknowledgment

The authors would like to thank Daniel Scharstein for the Middlebury test bed and personal communications.

References

[1] K. He, J. Sun, and X. Tang. Guided image filtering. In *Proc. ECCV*, 2010.

[2] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE TPAMI*, 30(2):328–341, 2008.

[3] H. Hirschmüller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE TPAMI*, 31(9):1582–1599, 2009.

[4] A. Hosni, M. Bleyer, and M. Gelautz. Near real-time stereo with adaptive support weight approaches. In *Proc. 3DPVT*, 2010.

[5] A. Hosni, M. Bleyer, M. Gelautz, and C. Rheman. Local stereo matching using geodesic support weights. In *Proc. ICIP*, pages 2093–2096, 2009.

[6] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *ICPR*, pages 15–18, 2006.

[7] J. Liu and J. Sun. Parallel graph-cuts by adaptive bottom-up merging. In *Proc. CVPR*, pages 2181 – 2188, 2010.

[8] S. Mattoccia, F. Tombari, and L. D. Stefano. Stereo vision enabling precise border localization within a scanline optimization framework. In *Proc. ACCV*, pages 517–527, 2007.

[9] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *Proc. CVPR*, 2011.

[10] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. A. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In *Proc. ECCV*, pages 6311–6316, 2010.

[11] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[12] D. Scharstein and R. Szeliski. Middlebury stereo evaluation - version 2, 2010. `http://vision.middlebury.edu/stereo/eval/`.

[13] X. Sun, X. Mei, S. Jiao, M. Zhou, and H. Wang. Stereo matching with reliable disparity propagation. In *Proc. 3DIMPVT*, pages 132–139, 2011.

[14] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE TPAMI*, 30(6):1068–1080, 2008.

[15] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. ICCV*, pages 839–846, 1998.

[16] F. Tombari, S. Mattoccia, and L. D. Stefano. Segmentation-based adaptive support for accurate stereo correspondence. In *Proc. PSIVT*, pages 427–438, 2007.

[17] Z. Wang and Z. Zheng. A region based stereo matching algorithm using cooperative optimization. In *Proc. CVPR*, pages 1–8, 2008.

[18] Y. Wei, C. Tsuhan, F. Franz, and C. H. James. High performance stereo vision designed for massively data parallel platforms. *IEEE TCSVT*, 99:1–11, 2010.

[19] Q. Yang, C. Engels, and A. Akbarzadeh. Near real-time stereo for weakly-textured scenes. In *Proc. BMVC*, pages 80–87, 2008.

[20] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. *IEEE TPAMI*, 31(3):492–504, 2009.

[21] K.-J. Yoon and I.-S. Kweon. Adaptive support-weight approach for correspondence search. *IEEE TPAMI*, 28(4):650–656, 2006.

[22] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proc. ECCV*, pages 151–158, 1994.

[23] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE TCSVT*, 19(7):1073–1079, 2009.

[24] K. Zhang, J. Lu, G. Lafruit, R. Lauwereins, and L. V. Gool. Real-time accurate stereo with bitwise fast voting on cuda. In *Proc. ICCV Workshop*, 2009.

[25] L. Zhang and S. M. Seitz. Estimating optimal parameters for mrf stereo from a single image pair. *IEEE TPAMI*, 29(2):331–342, 2007.

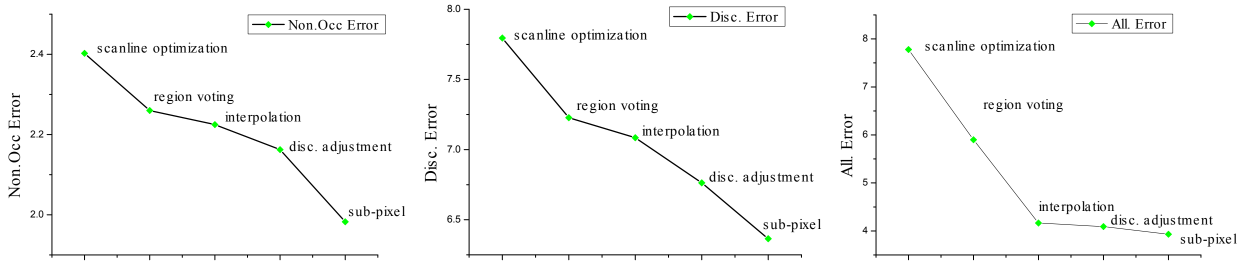


Figure 7. The average error percentages in *non-occlusion*, *discontinuity* and *all* regions after performing each refinement step.

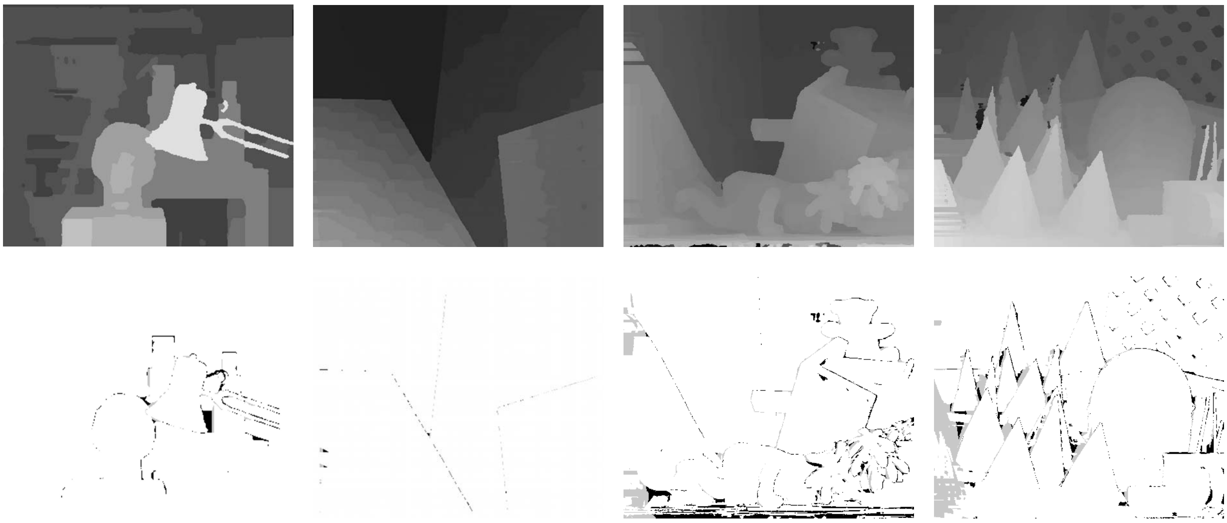


Figure 8. Results on the Middlebury data sets. First row: disparity maps generated with our system. Second row: disparity error maps with threshold 1. Errors in unoccluded and occluded regions are marked in black and gray respectively.

Algorithm	Avg.	Tsukuba			Venus			Teddy			Cones		
	Rank	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
Our method	5.8	1.07_{12}	1.48_{10}	5.73_{14}	$\mathbf{0.09}_2$	0.25_7	$\mathbf{1.15}_2$	4.10_4	6.22_3	10.9_4	2.42_3	7.25_5	6.95_4
AdaptingBP [6]	7.2	1.11_{15}	1.37_6	5.79_{15}	0.10_3	0.21_4	1.44_4	4.22_6	7.06_6	11.8_7	2.48_4	7.92_9	7.32_7
CoopRegion [17]	7.2	0.87_3	1.16_1	4.61_2	0.11_4	0.21_3	1.54_6	5.16_{14}	8.31_{10}	13.0_{11}	2.79_{12}	7.18_4	8.01_{16}
DoubleBP [20]	9.7	0.88_5	1.29_3	4.76_5	0.13_7	0.45_{17}	1.87_{11}	3.53_3	8.30_9	9.63_2	2.90_{17}	8.78_{24}	7.79_{13}

Table 2. The rankings in the Middlebury benchmark. The error percentages in different regions for the four data sets are presented.

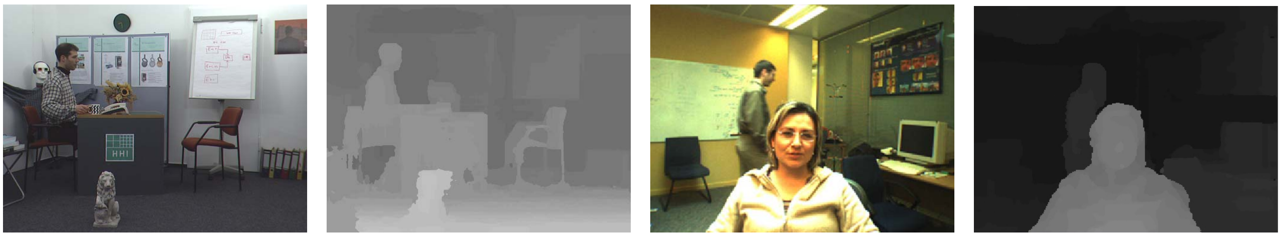


Figure 9. Snapshots on 'book arrival' and 'Ilkay' stereo video sequences.