REAL-TIME STEREO MATCHING: A CROSS-BASED LOCAL APPROACH

Jiangbo Lu [*,†], *Ke Zhang* [*,†], *Gauthier Lafruit* [†], *and Francky Catthoor* [*,†]

[*] Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium
[†] Multimedia Group, IMEC, Kapeldreef 75, B-3001, Leuven, Belgium

ABSTRACT

We propose an area-based local stereo matching algorithm that yields accurate disparity estimates, while achieving the real-time speed completely on the graphics processing unit (GPU). For a local stereo method, the key challenge is to decide an appropriate support window for the pixel under consideration. Our stereo method starts with computing an upright local cross adaptively for each anchor pixel, which defines a per-pixel support skeleton. Next, based on this compact local cross representation, we aggregate the matching costs in a shape adaptive full support region using two orthogonal integration steps. Approximating scene structures accurately, the proposed method is among the best-performing real-time stereo methods according to the benchmark Middlebury stereo evaluation. Additionally, our method is very easy to implement, memory efficient, and hence it is promising for many practical applications.

Index Terms— Stereo matching, local adaptation, GPGPU

1. INTRODUCTION

Stereo matching has been an active research topic for decades. Recent years have particularly witnessed a significant advance in this field, concerning the disparity estimation accuracy. However, these state-of-the-art stereo methods are still far from online processing, making them inappropriate for practical applications with a stringent real-time requirement, e.g., robot navigation and interactive view interpolation. As a result, real-time stereo matching has attracted increasing attentions, where the challenge is to generate accurate disparity maps while still maintaining the real-time execution speed.

In general, stereo methods can be categorized into two classes: local methods and global methods. Local methods, without involving complex optimization schemes, are arguably easy to implement in both software and hardware. Gong *et al.* [1] has evaluated several real-time local stereo methods, which lead to impressive disparity estimation speeds. Instead, enforcing the explicit disparity smoothness optimization, global stereo methods often yield superior disparity accuracy, but traditionally they are too time-consuming for real-time usage. Recently, Yang *et al.* [2] and Wang *et al.* [3] have shown that global methods are feasible for online execution, exploiting the horsepower of the modern graphics processing unit (GPU).

Aiming at achieving disparity accuracy comparable to global methods, while still preserving the implementation advantages of local methods, we are motivated to push forwards the accuracy of the local stereo solution under the real-time constraint. In this paper, we propose a novel local stereo algorithm based on locally adaptive cross representations. The key algorithmic ideas are two-fold. First, a locally adaptive upright cross is decided upon the color similarity, defining an initial support skeleton for the anchor pixel (see Fig. 1(a)). Second, we dynamically construct a shape adaptive full support region in the cost aggregation step, reusing the pre-computed

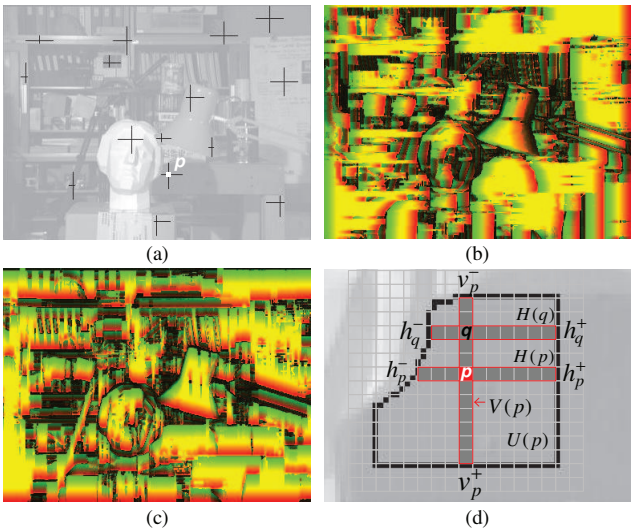


Fig. 1. Cross-based local support region representation and construction on the *Tsukuba* image [6]. (a) Sample local crosses for the anchor pixels, e.g., p. (b, c) For better viewing effects, the dense cross map M is split to show the horizontal and vertical cross arm lengths in (b) and (c), separately. In (b), {R,G,B,A} color channels are set to $15\times\{h_p^-, h_p^+, 0, 0\}$, while $15\times\{v_p^-, v_p^+, 0, 0\}$ in (c). (d) Configuration of a cross-based support region for the anchor pixel p.

neighboring cross configurations. The end result is that appropriate local support regions are efficiently derived from the fairly compact cross-based representation, leading to accurate disparity estimates for different pixel locations. In addition, we have implemented the proposed method on the GPU, optimizing its real-time performance.

Compared with the existing real-time methods, the proposed method has some unique advantages. First, yielding the similar disparity accuracy, our method is much easier to implement, more memory efficient (crucial for embedded platforms) than the leading real-time global methods [2, 3]. Furthermore, our method runs 1.8 times faster than Yang *et al.*'s method [2] on the identical GPU. Also, it can free up the CPU for other high-level tasks, unlike Wang *et al.*'s method [3] that has to use both the GPU and the CPU intensively. Second, under the real-time constraint, the proposed method outperforms all the existing real-time local methods [1, 4] in terms of disparity accuracy. It is particularly better than others [1, 3, 4] for the challenging depth discontinuities. Third, our cross-based local geometry approximation provides a real-time alternative to the popular yet offline mean-shift segmentation approach [5]. The latter is widely used in state-of-the-art stereo algorithms as a critical module.

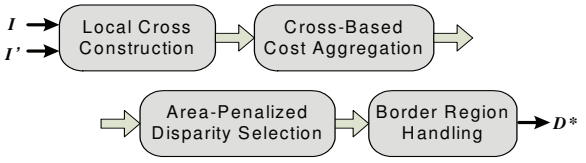


Fig. 2. Outline of the proposed stereo algorithm. $\{I, I'\}$ is the input stereo image pair, and D^* is the estimated dense disparity map.

2. OUR ALGORITHM

The proposed algorithm consists of the following major steps (Fig. 2). First, each image of a given stereo pair is $\{I, I'\}$ is independently analyzed to derive a dense local cross map. Second, we aggregate raw matching cost in a pixel-wise adaptive local region, combining the local support configurations of a pair of hypothetical correspondences symmetrically. Next, the aggregated cost is penalized differently according to the aggregation area size, before applying the Winner-Takes-All (WTA) strategy [7] for disparity selection. Finally, we perform a simple method to detect and handle unreliable disparity estimates in the image border regions. This results in a dense disparity map $D^* = \{d_p^* \,|\, p \in I\}$ for the left view.

2.1. Local cross construction

For accurate local stereo matching, it is important to decide an appropriate local support region for each pixel adaptively. In principle, this local support region should only contain the neighboring pixels from the same depth with the pixel under consideration. To this end, we propose a compact cross-based approach to represent and construct local support regions. We use the common assumption that pixels with similar intensity within a constrained area are likely from the same image structure, therefore having similar disparity.

The key idea of the proposed approach is to decide an upright cross for every pixel $p = (x_p, y_p)$ in the input left image I (similar for the pixel $p' \in I'$). As shown in Fig. 1(d), this pixel-wise adaptive cross consists of two orthogonal line segments, i.e., the horizontal segment $H(p)$ and the vertical one $V(p)$. These two segments intersect at the anchor pixel p, and they jointly define the local support skeleton for the pixel p. Instead of fixing the size of a local cross, we adaptively change its four arm lengths to reliably capture the local image structure. More specifically, for a given pixel $p \in I$, we first decide a parameter set $\{h_p^-, h_p^+, v_p^-, v_p^+\}$ that denotes the left, right, up, and bottom arm length, respectively. Then, this pixel-wise parameter set is packed into four channels (RGB + alpha) of a dense texture map M (see Fig. 1(b, c)), with one byte for each channel.

As a general concept, the local cross can be decided using various specific approaches. Here, we present an efficient approach based on color similarity under the connectivity constraint. Our approach starts with applying a separable 3×3 median filter to the input image I, suppressing the impact of image noise as well as subtle non-Lambertian effects. Next, the arm lengths $\{h_p^-, h_p^+, v_p^-, v_p^+\}$ are decided upon color similarity. Taking h_p^- as an example, we perform a color similarity testing for a consecutive set of pixels (with a preset maximum search range L), which reside on the left horizontal side of the pixel p. The purpose is to search for the largest left span r^*, where all the pixels covered, i.e., $\{p_r = (x_p - r, y_p) \,|\, 0 \le r \le r^*\}$, are similar to the anchor pixel p in color. More clearly, this means $\forall r \le r^*$, $\delta(p_r, p) = 1$, and $\delta(p_{r^*+1}, p) = 0$ (if $r^* < L$), where $\delta(p_1, p_2)$ is an indicator function. It signals whether two pixels p_1 and p_2 are similar based on all color bands:

$$\delta(p_1, p_2) = \begin{cases} 1\ , & \max\limits_{c \in \{\mathrm{R,G,B}\}} \Big(|I_c(p_1) - I_c(p_2)|\Big) \le \tau \\ 0\ , & \text{otherwise,} \end{cases} \tag{1}$$

where I_c is the intensity of the color band c. Set empirically, τ controls the confidence level of color similarity. Once the largest left span r^* is derived, we set the left arm length $h_p^- = \max(r^*, 1)$, enforcing a minimum support region of 3×3 for reliable matching. Based on the arm lengths $\{h_p^-, h_p^+, v_p^-, v_p^+\}$ decided for the pixel p, two orthogonal cross segments $H(p)$ and $V(p)$ are easily defined. For instance, $H(p) = \{(x, y) \mid x \in [x_p - h_p^-, x_p + h_p^+], y = y_p\}$.

Despite that only four parameters need to be stored for each pixel, a shape adaptive full support region $U(p)$ is readily determined for the pixel p. As shown in Fig. 1(d), the local support region $U(p)$ is delineated as an area union of multiple horizontal segments $H(q)$, sliding along the vertical segment $V(p)$ of the anchor pixel p:

$$U(p) = \bigcup_{q \in V(p)} H(q)\ , \tag{2}$$

where q is a support pixel located on the vertical segment $V(p)$. From the generated cross map M, $H(q)$ can also be easily retrieved.

2.2. Cross-based matching cost aggregation

As an area-based local stereo method, the proposed method places a key emphasis on the cost aggregation step. The goal of this step is to reduce the image ambiguity by collecting the support from the neighboring pixels of the same depth, for the pixel under estimation.

Prior to cost aggregation, we first compute the pixel-based raw matching cost between a pair of corresponding pixels, i.e., s in the left image I and s' in the right image I' (with a disparity value d),

$$e_d(s) = \min\Big(\sum_{c \in \{\mathrm{R,G,B}\}} \big|I_c(s) - I'_c(s')\big|\ ,\ T\Big) \times \frac{255}{T}\ , \tag{3}$$

where T controls the truncation limit of the matching cost. The truncated cost is scaled to 255 to make full use of the range of one byte.

For reliable cost aggregation, unlike most of local stereo methods [1, 8], we symmetrically consider both local support regions $U(p)$ and $U'(p')$ decided for the pixel p and p', respectively. If we only consider the local support region $U(p)$ for the left image, the matching cost aggregation will be polluted by outliers in the right image, i.e., pixels from different depths with the pixel p' in the support window. Therefore, combining two local support regions to define the aggregation region, the normalized matching cost $\overline{E}_d(p)$ between the pixel p and p' is computed as follows,

$$\overline{E}_d(p) = \frac{1}{\|U_d(p)\|} E_d(p) = \frac{1}{\|U_d(p)\|} \sum_{s \in U_d(p)} e_d(s)\ , \tag{4}$$

$$\text{with}\ \ U_d(p) = \{(x, y) \mid (x, y) \in U(p), (x - d, y) \in U'(p')\}\ .$$

In (4), $U_d(p)$ denotes the combined local support region that contains only the valid pixels, likely having similar disparities with the anchor pixels p and p' in both images. $\|U_d(p)\|$ is the number of support pixels in $U_d(p)$, used to normalize the aggregated cost $E_d(p)$.

To speed up the computation of the aggregated cost $E_d(p)$ in (4) significantly, we transform the double integral of raw matching costs $e_d(s)$ into two orthogonal iterated integrals, as follows,

$$E_d(p) = \sum_{s \in U_d(p)} e_d(s) = \sum_{q \in V_d(p)} \Big(\sum_{s \in H_d(q)} e_d(s)\Big) = \sum_{q \in V_d(p)} E_d^H(q)\ , \tag{5}$$

where $E_d^H(q)$ represents the result after the horizontal integration step. As with $U_d(p)$, $H_d(p)$ and $V_d(p)$ define the combined horizontal and vertical segments, treating two corresponding cross configurations symmetrically. Using only two rendering passes, the proposed cross-based cost aggregation can be efficiently performed.

To utilize the GPU's SIMD processing capability, we compute four adjacent disparity hypotheses at a time. The resulting average costs $\overline{E}_d(p)$ are packed into the four channels of a color image.

2.3. Area-penalized disparity selection

After cost aggregation, we use a Winner-Takes-All (WTA) strategy [7] to select the optimal disparity. Instead of only relying on the average matching cost $\overline{E}_d(p)$, we include an additional size penalty term $\rho(\|U_d(p)\|)$, when deciding the disparity d_p^* for the pixel p:

$$d_p^* = \arg\min_d \left(\overline{E}_d(p) + \rho(\|U_d(p)\|)\right) \ , \quad d \in [0, d_{max}], \qquad (6)$$

where d_{max} is the maximum value of possible disparities. The purpose of including such a penalty term is to encourage the selection of larger support areas, when the average matching costs associated with different disparities are approximately equal for untextured image regions [8]. Empirically, we set the value of $\rho(\|U_d(p)\|)$ by evaluating the area $\|U_d(p)\|$ against two preset thresholds as follows,

$$\rho(\|U_d(p)\|) = \begin{cases} 0.06 \times 255 \ , & \|U_d(p)\| \le A/4 \\ 0.03 \times 255 \ , & A/4 < \|U_d(p)\| \le A \\ 0 \ , & A < \|U_d(p)\| \ , \end{cases} \qquad (7)$$

where $A = (L+1) \times (L+1)$, and L is the maximum arm length. Lastly, a 3×3 median filter is applied to refine the disparity map [7].

2.4. Border region handling

As an area-based local stereo method, our algorithm does not reason about the disparities for the image border regions. But due to occlusions, some pixels in this region are only visible in the left view I, which lead to unreliable disparity estimates. To tackle this problem, we propose a simple border handling scheme without ruining the real-time speed. The basic idea is to detect *unreliable* pixels as those with an estimated correspondence residing outside of the right view I', i.e., whenever $x_p - d_p^* < 1$. We then apply a simple two-step approach to fill reliable disparities in the border regions:

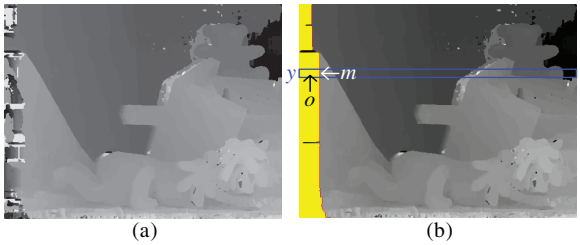
1. For each horizontal scanline y, we search for the rightmost unreliable border pixel $m=(x_m, y)$ with the maximum value x_m. We record $O(y) = x_m$ (see the red curve in Fig. 3(b)).
2. For each horizontal scanline y, and for any pixel o to the left of the pixel $v=(O(y)+1, y)$, its final disparity d_o^* is set to d_v^*.

Comparing Fig. 3(a) to Fig. 5(c), this border handling scheme reduces the *all* disparity error rate by 1.4% [6] with little overhead.

3. GPU OPTIMIZATION

Besides the general GPU-based optimization techniques [1, 3, 4], we have applied some particular schemes, further enhancing the performance of the proposed stereo method on the GPU.

Scaling arm length values. As we store the arm length (e.g., h_p^-) into a 8-bit color channel, we can scale up the value of each arm length by a factor of $255/L$ (i.e., $15 \times h_p^-$ when $L = 17$) without exceeding the range of a single byte. This overcomes the numerical precision problem on the GPU, when the arm length is very small.



(a) (b)

Fig. 3. Border handling on the *Teddy* image. (a) Without border handling. (b) Border pixels handled are marked in yellow (and red).

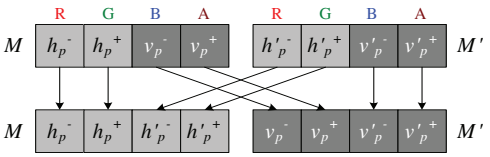


Fig. 4. Re-packing left-right cross maps $\{M, M'\}$ ($\forall p \in M$).

Re-packing left-right cross maps. In the preprocessing step, we have constructed a pair of local cross maps $\{M, M'\}$, each storing pixel-wise varying arm lengths for the given images $\{I, I'\}$ into RGBA channels. Since our cross-based cost aggregation is implemented with two orthogonal integration steps, either horizontal or vertical arm lengths are necessary and sufficient for one integration rendering pass. To maximize the GPU texture cache locality behavior, we hence re-pack the left-right cross maps as shown in Fig. 4.

Invoking narrow quads for border handling. In Sect. 2.4, a two-step approach is proposed for border region handling. As the maximum disparity value is d_{max}, the rightmost unreliable pixel's x-coordinate $O(y) \le d_{max}$. To avoid largely redundant pixel processing for the entire image, we apply the proposed bordering handling only to a rather small set of pixels. For the first step that decides the one-dimensional array $O(y)$, rendering a one-column quad is sufficient; while for the second step, a narrow quad with a width of d_{max} is rendered, covering all the possible unreliable border pixels.

4. EXPERIMENTAL RESULTS

The experiments are based on the benchmark Middlebury stereo database [6]. The parameters used in our stereo algorithm are set constant across all experiments, i.e., $L = 17$, $\tau = 25$, and $T = 70$.

Quantitative evaluation. Based on the Middlebury online disparity evaluation service [6], Table 1 reports the disparity error rates measured against the ground-truth disparities. Reusing the identical names posted online, our method is ranked between the RealTimeGPU [3] and CostRelax [9] approaches. It compares favorably over other existing real-time methods [10, 1, 11], particularly better for depth discontinuities. Though our method is slightly outperformed by the two leading real-time global methods [2, 3], it does have a few important advantages. The RealtimeBP method [2] consumes far more memory than ours, which demands eight textures to store the smoothness term. The RealTimeGPU method [3] has to transfer the intermediate results back to the CPU for the further processing, and also it does not perform well for depth discontinuities.

Visual results. Fig. 5 shows the estimated disparity maps. They are both piecewise smooth and accurate near depth discontinuities.

Table 1. Quantitative evaluation results for the new Middlebury stereo database [6]. Blank fields are not reported in the original paper.

Algorithm	*Tsukuba*			*Venus*			*Teddy*			*Cones*		
	nonocc.	all	disc.	nonocc.	all	disc.	nonocc.	all	disc.	nonocc.	all	disc.
RealtimeBP [2]	1.49	3.40	7.87	0.77	1.90	9.00	8.72	13.2	17.2	4.61	11.6	12.4
RealTimeGPU [3]	2.05	4.22	10.6	1.92	2.98	20.3	7.23	14.4	17.6	6.41	13.7	16.5
Our method	**2.80**	**4.84**	**7.29**	**2.14**	**3.40**	**11.5**	**9.67**	**16.3**	**18.8**	**5.85**	**13.7**	**12.1**
CostRelax [9]	4.76	6.08	20.3	1.41	2.48	18.5	8.18	15.9	23.8	3.91	10.2	11.8
Orthog. DP [10]	1.34	3.36	7.10	2.73	3.81	10.1	9.03	16.8	18.4	13.1	20.1	20.1
Adapt.Weight [1]	2.27	3.61	11.2	3.57	4.61	19.8	10.9	18.8	23.2	5.92	14.3	13.8
Realtime DP [11]	2.85		15.6	6.42		25.3						

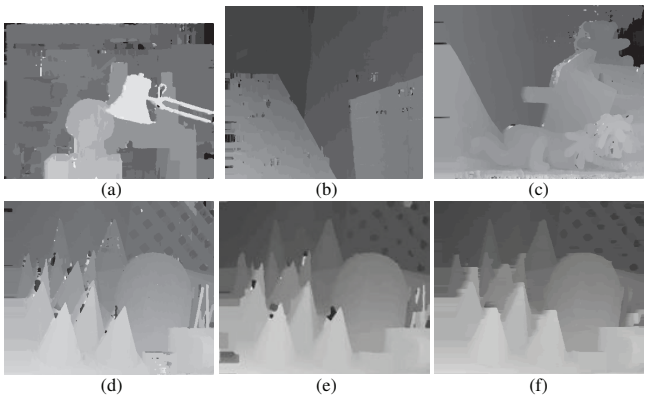


Fig. 5. Our resulting disparity maps estimated for *Tsukuba*, *Venus*, *Teddy*, and *Cones* stereo images (a-d). Disparity map of *Cones* estimated with (e) RealtimeBP [2] and (f) RealTimeGPU [3].

Fig. 6. Synthesized center views using the left-view disparity maps.

Compared to the leading real-time global methods [2, 3], our method preserves the subtle scene structures in the *Cones* more accurately.

Real-time speed. We tested the proposed algorithm on an NVIDIA GeForce 7900 GTX graphics card in a 3.2 GHz PC. For a stereo scene with resolution 384 $\times$ 288 and 16 disparity levels, the speed is about 17 fps using only the computational resources of the GPU. This speed measurement includes all the memory overhead.

Novel view synthesis. In Fig. 6, we show that the estimated disparity maps lead to visually plausible synthesized center views, when integrated into a depth-image-based rendering technique [12].

5. CONCLUSION

This paper proposes a cross-based local stereo matching algorithm, running in real-time on the GPU. Based on the generated pixel-wise compact cross, we perform cost aggregation in a shape adaptive local support region that approximates varying image structures accurately. Evaluation with the Middlebury stereo benchmark shows that the proposed method is ranked among the best-performing real-time stereo methods. In particular, it has the pronounced advantages in the execution time, memory consumption, as well as yielding accurate disparity estimates for the challenging depth discontinuities. In future work, we plan to further optimize our algorithm on an NVIDIA CUDA (Compute Unified Device Architecture) capable GPU.

6. REFERENCES

[1] M. Gong, R. Yang, L.Wang, and M. Gong, "A performance study on different cost aggregation approaches used in real-time stereo matching," *IJCV*, vol. 75, no. 2, pp. 283–296, 2007.

[2] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. BMVC*, 2006, pp. 989–998.

[3] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proc. 3DPVT*, 2006, pp. 798–805.

[4] R. Yang, M. Pollefeys, and S. Li, "Improved real-time stereo on commodity graphics hardware," in *Proc. of CVPRW*, 2004.

[5] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[6] Middlebury stereo page, http://vision.middlebury.edu/stereo/.

[7] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int'l Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, 2002.

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

[9] R. Brockers, M. Hund, and B. Mertsching, "Stereo vision using cost-relaxation with 3D support regions," in *Image and Vision Computing New Zealand*, 2005.

[10] M. Gong and Y.-H. Yang, "Real-time stereo matching using orthogonal reliability-based dynamic programming," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 879–884, 2007.

[11] S. Forstmann, J. Ohya, Y. Kanou, A. Schmitt, and S. Thuering, "Real-time stereo by using dynamic programming," in *Proc. CVPR Workshop*, 2004, pp. 29–36.

[12] J. Lu, G. Lafruit, and F. Catthoor, "Anisotropic local high-confidence voting for accurate stereo correspondence," in *Proc. SPIE-IS&T Electronic Imaging*, Jan. 2008, vol. 6812.