15th IAPR International Conference on Machine Vision Applications (MVA)
Nagoya University, Nagoya, Japan, May 8-12, 2017.

Two-Stage Cross-Based Stereo Disparity Refinement

Zonglin Xu
Waseda University, Japan
xuzhonglin@fuji.waseda.jp

Sei-ichiro Kamata
Waseda University, Japan
kam@waseda.jp

Qieshi Zhang
Shaanxi Normal University, China
qieshi.zhang@ieee.org

Abstract

This paper proposed a disparity refinement method based on two-stage cross. First stage is anti-texture cross-based support region construction to build proper support regions for error pixels without being influenced by texture. Based on the support regions, second stage of the method is proposed, which is called weighted cross-based updating method. The experiments show that the proposed method could build the support region accurately and improve the accuracy of the disparity map in final results with fast speed, compared to other tree-based algorithms. It also outperforms the existing disparity refinement methods in preserving the boundaries of objects in the final disparity map.

1. Introduction and related work

Stereo matching has traditionally been, and continued to be one of the most heavily investigated topics in computer vision (*CV*). Generating disparity map from a pair of stereo images is a popular topic in *CV*, because it plays an important role in many applications.

Stereo matching methods are divided into global algorithms and local algorithms [1]. Global algorithms usually could get a very high accuracy with high computation complexity, while the local algorithms are more efficient. Due to the real-time algorithms are drawing more attention, local algorithms are focused.

In recent years, lots of local methods have been proposed. All the local methods need to build local support regions to find similar pixels for pixels. Such as, Zhang and Lu [2] developed an adaptive cross-based local stereo matching method that could build support region with high accuracy by using cross. Then Mei *et al.* [3]improved this method by changing the rules of the cross. However, because these two methods above are based on color similarity and distance, they suffer a lot from the texture in the color image. The texture with changeable intensity often has no effect with the disparity, but could influent extent of the cross arm. The support region generated often smaller than it should be, result in increasing the computation cost and lacking of true pixels in the support region. More importantly, the original cross could not identify the edges of objects (structure), causing boundary blurring in the final disparity map.

Recently, Yang [4] proposed a non-local aggregation method based on bilateral filter with fast speed and accuracy. In Yang's method, the image is treated as a four-connected, undirected planar graph, where each pixel corresponds to a node and each pair of neighboring nodes is connected by an edge. Edges are sorted and selected based on the weight to build the minimum spanning tree (MST). During the cost aggregation step, the information is propagated from pixel to its neighboring pixels. After that, segment-tree (ST) [5] was proposed to improve the tree structure. It divided one MST into several trees to be the support region. However, these tree-based algorithms also suffer from the texture in the image. Because the texture has no effect with the disparity, but can cause the weight of the edge to be very large. In this case, the information should propagate within the texture edges, but is impeded, which is a deficiency of the MST structure.

Textures refer to surface patterns that are similar in appearance and local statistics. They are often small in scale. Structures are often big in scale, the edges of which are actually edges of objects. Extracting the structure has been widely studied. Also, lots of edge-preserving filters have been proposed, such as bilateral filter [6] and relative total variant filter [7].

Disparity refinement, which is an important step in stereo matching, are paid more and more attention. The previous steps could generate the raw disparity maps from color image. However, because of the occluded area in raw color image, there are lots of outliers in the generated disparity maps. The pixels in outliers are also called error pixels. Disparity refinement is a step to detect and correct these error pixels. Ji *et al.* [8] proposed an efficient and accurate stability-based tree disparity refinement method based on MST. But this method is also influenced by texture.

2. Proposed method

In order to solve the problems mentioned above, we proposed a disparity refinement method based on two-stage cross. First, anti-texture cross is proposed to build support region. Then based on anti-texture cross, a new updating method called weighted cross-based updating method is proposed. We follow the existing framework of disparity refinement method including error pixel checking, image pre-processing, support region construction and updating method. Here, we use left-and-right check [8] to do the error pixel checking. The flow chart of the proposed method can be seen in

Fig. 1.

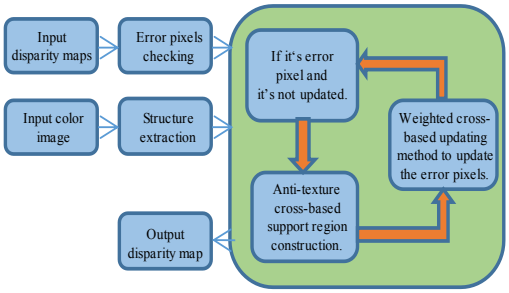

Figure 1. Flowchart of proposed method.

2.1 Color image pre-processing

In order to reduce the influence of textures in raw color image, we need to eliminate these textures while preserving the structural edges.

Reminding that the edge-preserving filters, could realize the function above. But experiment results show that just use these filters one time could also eliminate part of the structural information. We need to extract structural information from the detail layer. Our structure extraction method is proposed.

We firstly apply edge-preserving filter (RTV [7] is recommended) to the original image I to get the first structural layer S_1. Then the first detail layer $D_1=I-S_1$. We apply the same decomposition to D_1 to get the second structural layer S_2. The second detail layer $D_2=D_1-S_2$. So the original image I is equal to

$$I=S_1+S_2+D_2 \quad (1)$$

The final structure S equals to the combination of first structural layer S_1 and second structural layer S_2, which denoted as:

$$S=\lambda_1*S_1+\lambda_2*S_2 \quad (2)$$

where λ_1 and λ_2 are two parameters, which control the weight of S_1 and S_2. Here, λ_1 should approximate to 1, while λ_1 should be a small number.

2.2 Anti-texture cross-based method

Support region construction is one of the most important steps in disparity refinement. In principle, the local support region should contain only the neighboring pixels with the same disparity value or continuous disparity value. Color image are used to build the support region, which based on the assumption that pixels with similar intensity in color image in a specific area without being affected by textures are likely to be from the same image structure, thus, sharing similar disparity.

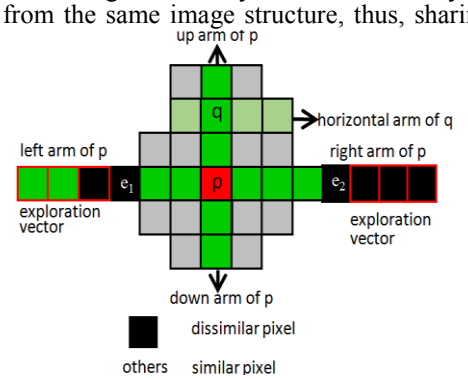

Figure2. Exploration vector in anti-texture cross.

We appreciate the cross-based support region construction method proposed by Zhang *el al.* [2], which proceeds by two steps shown in Figure 2. Here, the method does not use exploration vector. For error pixel p - also called center pixel - we find the left, right, up and down arms based on the Rule. These arms are marked in dark green in Fig.2. Then for the pixels on up arm and down arm, which is called sub-center pixels, we also find their horizontal arms to construct the whole support region. The horizontal arms are marked in light green in Fig. 3. In each arm, every pixel satisfies the following Rule:

1. Color similarity denotation:

$$\max(|I_c(p)-I_c(q)|_{c=R,G,B})<\tau_1 \quad (3)$$

p is the center pixel, q means the pixels on each arm.

2. Distance denotation, which constrains the size of the support region.

$$\Omega(p,q)<\tau_2 \quad (4)$$

where $\Omega(p,q)$ represents the Euclidean distance between p and q.

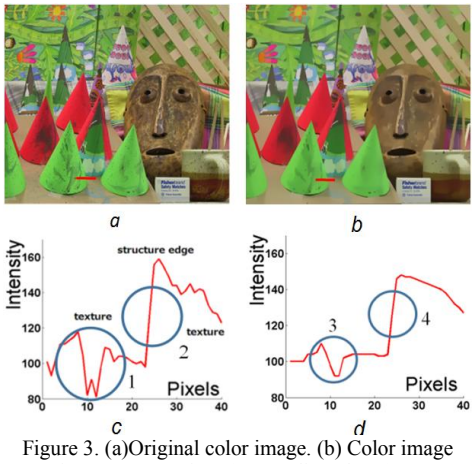

Figure 3. (a)Original color image. (b) Color image after pre-processing. (c) Scanline plots (rows indicated using red line in (a)). (d) Scanline plots (rows indicated using red line in (b)).

However, this method suffers a lot from texture in the raw color images. In order to cover the deficiency of texture, we first just use the structure part of the image S to construct the support region. In structure part of image S, most of the textures are eliminated. Then we have to find where the textural and structural edges are. We find that structural edges are like a "step", seeing circle 2 and circle 4 in Fig.3, while the texture areas are like mountains with one or more spikes, seeing circle 1 and circle 3 in Fig.3. Here, we define if a pixel satisfies the Rule, it is called similar pixel. Else, it is called dissimilar pixel, also seeing in Fig. 3. If we find one dissimilar pixel e, we use an exploration vector that could check several succession of pixels whether they are similar or not, marked in red in Fig. 2 to judge the following pixels. By analyzing pixels on exploration vector, we can get whether these pixels locate in textural areas or structural edges. If these pixels are like a "step", we treat the area as a structural edge, then, we stop to find e. If they like "mountains" with one or more spikes, we treat it as texture areas, then, we continue to find next dissimilar

Input: Color Image I.
One error pixel P(i,j).
Length of exploration vector L ($4 \le L \le 10$).
Threshold in rule 1: color threshold τ_1 and distance threshold τ_2.
Ration: λ_3 denote the sensibility of exploration vector ($\lambda_3 > 0.8$).
Output: The length of left arm k.
1: Initialize number of dissimilar pixels on exploration vector N=0 and the length of left arm k=0.
2: while ($N < \lambda_3 * L$)
3: if ($k < \tau_2$ && j-k>0)
4: Pick next pixel Q(i,j-k-1) on the left of P.
5: if (Q is a dissimilar pixel)
6: Calculate the number of dissimilar pixels on exploration vector N.
7: else N=0.
8: end if
9: else break
10: end if
11: k=k+1
12: end
13: return k

Algorithm 1. Building left arm of anti-texture cross

pixel *e*. From the Fig.2, we can see that on the left arm of *p*, we find a dissimilar pixel e_1. Then from the exploration vector, we can judge that this region is actually like a mountain. Thus, these pixels locate on texture areas. Finally, we continue to find the next dissimilar pixel. On the right arm of *p*, we find the dissimilar pixel e_2. Then from the exploration vector, we find that all the pixels are different from *p*. Based on the assumption, we judge that these pixels locate on the structural edges. At last, we stop to find right arm of *p*.

Algorithm 1 shows how to build left arm by using exploration vector. The right arm, up arm, down arms and horizontal arms are built in the same way.

2.3 Weighted cross-based update method

In order to reduce computation complexity, our updating method divided into four steps. We build cross, compute disparity cost, aggregate cost and pick optimal disparity in a specific support region constructed from anti-texture cross-based method, rather than in the whole image proposed by MST.

2.3.1 Building weighted cross

In order to build the relationship between neighboring pixels and to use the original structure of the cross, we add weights between neighboring pixels according to how the cross is built (Fig.4). The black ones are where we need to add weights. For an error pixel, we first add weights between neighboring pixels on the left arm, right arm, up arm and down arm. And for the pixels on the up arm and down arm, we also add weights between neighboring pixels on the horizontal arms. The weights are calculated by:

$$W_{mn} = \begin{cases} \max_{c\in\{R,G,B\}} \lambda_I * |I_c(m) - I_c(n)| + (1-\lambda_I) * \Delta d & \text{if } \Delta d < \tau \\ c & \text{otherwise} \end{cases} \quad (5)$$

$$\Delta d = |d(m) - d(n)| \quad (6)$$

where α, τ are thresholds. m and n are neighboring pixels. I is the intensity value of three channels as R,G,B and d represents the disparity value in raw disparity map.

2.3.2 Disparity cost computation

Let $D(p)$ denote the original input disparity value of pixel p. For each pixel in the support region, a new cost value is computed with disparity level d:

$$C_d(p) = \begin{cases} |d - D(p)|, & p \text{ is a true pixel} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

2.3.3 Cost aggregation

Cost aggregation in disparity refinement step is actually information propagation from true pixels to error pixels in a specific support region. The similarity of two neighboring pixels S denotes how much one pixel can contribute to the other during aggregation. $D(p,q)$ determine the distance between pixel p and q, which is the sum of the weight along the weighted cross. Similarity of p and q can be calculated as:

$$S(p,q) = \exp(-\frac{D(p,q)}{\sigma}) \quad (8)$$

where σ is the parameter adjusting the similarity. Then, the aggregation cost can be calculated as:

$$C_d^A(p) = \sum_q S(p,q) C_d(q) \quad (9)$$

where $C_d^A(p)$ is the aggregation cost of p, aggregated from pixels from the whole support region.

But directly applying Eq. 9 may be time-consuming. The linear time algorithm has been proposed to fasten the aggregation speed. Similar to the linear time exact algorithm proposed in [4], the progresses are divided into two steps: the upward step and downward step.

The disparity cost firstly aggregated from cross terminal to the center pixel. Then the disparity cost propagates from the center pixel to the cross terminal. Fig. 4 shows the upward step. The downward step is opposite.

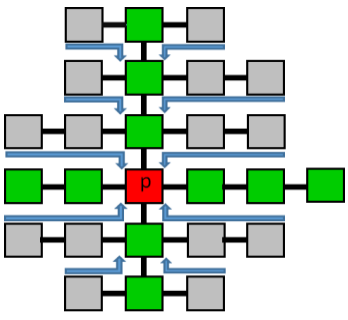

Figure 4. Weighted cross and upward step.

2.3.4 Winner take all method

We select optimal disparity value from candidates, which is described as:

$$d_{optimal} = \arg\min_{d\in D_d} f(d) \quad (10)$$

$d_{optimal}$ is the disparity for error pixels, D_d is the candidate disparity values for d and $f(d)$ is the cost after aggregation.

3. Experiment Results

We tested the proposed method and compared with other methods by using the Middleburry dataset and its benchmark.

Quantitative Evaluation

Table 1. Error rate comparison with error threshold =1

Image	Original		ST		Stability-based tree		Proposed method	
	non occ	all	non occ	all	non occ	all	non occ	all
cones	2.58	14.47	2.06	12.84	1.93	10.69	1.68	10.31
teddy	5.70	15.66	4.43	12.86	4.86	11.36	3.35	9.95
venus	0.69	2.64	0.21	1.35	0.20	0.40	0.12	0.95
tsukuba	2.22	6.27	1.06	1.56	2.53	3.11	1.87	2.39

Table 2. Error rate comparison with error threshold=2

Image	Original		ST		Stability-based tree		Proposed method	
	non occ	all	non occ	all	non occ	all	non occ	all
cones	3.16	15.5	2.91	14.29	2.69	12.31	2.68	12.85
teddy	7.16	17.71	6.12	15.69	6.42	14.79	5.32	13.17
venus	0.78	2.81	0.23	1.41	0.22	0.52	0.14	0.99
tsukuba	2.49	3.93	1.51	2.28	3.66	4.44	4.79	5.70

For evaluating the effectiveness of proposed method, ST [5] and stability-based tree algorithm [8] are compared. Table 1 and Table 2 indicate the error percentage of bad pixels before and after disparity refinement with greater disparity error than 1 disparity interval and 2 disparity intervals respectively. We list two common types of error rate used in Middleburry benchmark: errors in non-occlusion regions and in all regions. Compared to stability-based tree method, the errors are reduced about 15% in average by using our method. We also assess the result visually in removing the outliers near the boundaries. Fig.5 compares the disparity maps of ground truth, ours, ST and stability-based tree method. The proposed method could preserve the boundaries clearly compared to others.

Computation Complexity

We verify the advantage of our method in computational time complexity compared with other two tree-based methods. ST method and stability tree-based method segment image into several minimum spanning trees. But in the aggregation part, all the pixels in the image have to be considered. Our method, instead, constructs the support region for the error pixels firstly. Then, in the updating method, we just aggregated the disparity cost in each support regions. Most of the pixels far away from the error pixels need not to be considered, which will definitely save lots of time.

Support region evaluation

Results of anti-texture cross-based method are shown in (*i*)-(*l*) of Fig. 5. We can see the support regions built by our method is texture robust and could recognize the boundaries of objects.

4. Conclusion

In the paper, a new disparity refinement method based on two-stage cross is proposed. Due to texture elimination and anti-texture cross, the performance in constructing the support region is improved. Then, a new updating method is proposed based on weighted cross. Experiments show that the proposed method not only reduces running time but also achieves a high accuracy compared with other algorithm. The proposed method shows a high potential of disparity refinement in stereo matching, which could be used in many applications.

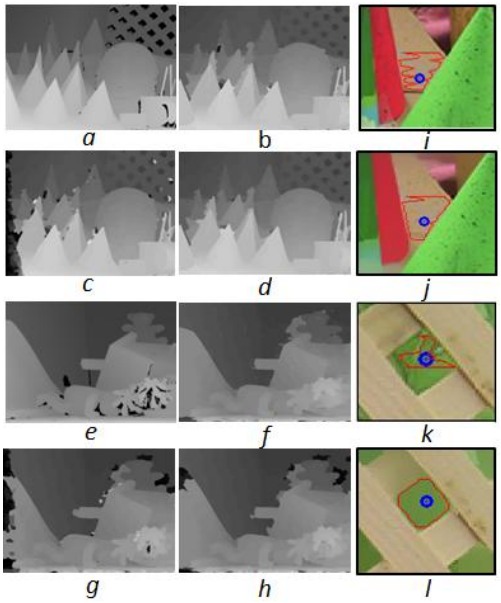

Figure 5. Visual results comparison. (*a-d*) cones left disparity map. (*e*)-(*h*) teddy left disparity map. (*a*,*e*) ground truth. (*b*)(*f*) proposed method. (*c*,*g*) ST method. (*d*,*h*) stability-based tree method. (*i*,*k*) support regions constructed by cross-based method. (*j*,*l*) support regions constructed by anti-texture cross-based method.

Reference

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,"*Int'l J. Computer Vision*, vol. 47, no. 1-3, pp. 7-42, 2002.

[2] K. Zhang *et al.*. "Cross-based local stereo matching using orthogonal integral images."*IEEE Trans. on Circuits and Systems for Video Technology*, vol.19, no.7, pp. 1073-1079, 2009.

[3] X. Mei *et al*. "On building an accurate stereo matching system on graphics hardware."*IEEE Int'l Conf. on Computer Vision Workshops*. pp. 467-474, 2011.

[4] Q. Yang, "A non-local cost aggregation method for stereo matching," *Int'l. Conf. Computer Vision and Pattern Recognition*, pp. 1402-1409, 2012.

[5] X. Mei *et al.*, "Segment-tree based cost aggregation for stereo matching,"*IEEE Int'l. Conf. Computer Vision and Pattern Recognition*, pp. 313-320, 2013.

[6] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images,"*IEEE Int'l Conf. on Computer Vision,* 1998, pp. 839-846

[7] L. Xu *et al.*, "Structure Extraction from Texture via Relative Total Variation,"*ACM Trans. on Graphics*, vol. 31, no. 6, pp.439-445, 2012.

[8] Y. Ji *et al.*, "Disparity Refinement with Stability-based Tree for Stereo Matching,"*IEEE Intelligent Vehicles Symposium*,pp. 469-474,2015.