A Fast Line Segment Based Dense Stereo Algorithm Using Tree Dynamic Programming

Yi Deng and Xueyin Lin

Department of Computer Science,
Intitute of HCI and Media Integration, Key Lab of Pervasive Computing(MOE),
3-524, Fit building, Tsinghua University, Beijing 100084, P.R. China
dengyi00@mails.tsinghua.edu.cn,
lxy-dcs@mail.tsinghua.edu.cn

Abstract. Many traditional stereo correspondence methods emphasized on utilizing epipolar constraint and ignored the information embedded in inter-epipolar lines. Actually some researchers have already proposed several grid-based algorithms for fully utilizing information embodied in both intra- and inter-epipolar lines. Though their performances are greatly improved, they are very time-consuming. The new graph-cut and believe-propagation methods have made the grid-based algorithms more efficient, but time-consuming still remains a hard problem for many applications. Recently, a tree dynamic programming algorithm is proposed. Though the computation speed is much higher than that of grid-based methods, the performance is degraded apparently. We think that the problem stems from the pixel-based tree construction. Many edges in the original grid are forced to be cut out, and much information embedded in these edges is thus lost. In this paper, a novel line segment based stereo correspondence algorithm using tree dynamic programming (LSTDP) is presented. Each epipolar line of the reference image is segmented into segments first, and a tree is then constructed with these line segments as its vertexes. The tree dynamic programming is adopted to compute the correspondence of each line segment. By using line segments as the vertexes instead of pixels, the connection between neighboring pixels within the same region can be reserved as completely as possible. Experimental results show that our algorithm can obtain comparable performance with state-of-the-art algorithms but is much more time-efficient.

1 Introduction

Stereo correspondence has been one of the most important problems in computer vision, and still remains a hard problem that needs more efforts. It is used in many areas like robot navigation, 3D reconstruction, tracking and so on. Introduction of different stereo correspondence algorithms can be found in the survey by Scharstern and Szeliski [1] and the one by Brown *et al.* [2].

Because of the noise and ambiguity, stereo correspondence problem is considered to be greatly ill-posed. To achieve a reasonable result, people use some

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part III, LNCS 3953, pp. 201–212, 2006.

assumptions on the scene, one of which is the *smoothness assumption*. This assumption supposes that the disparity map is almost smooth everywhere except at the borders of the objects, or equivalently that the scene is composed of several smooth structures. We formulate stereo algorithms as an energy minimization framework, and impose the smoothness assumption in a smoothness energy function. The optimal disparity map f will minimize the energy function as follow:

$$E(f) = \sum_{p} E^{p}_{data}(f_p) + \sum_{\langle p,q\rangle \in \mathcal{N}} E^{p,q}_{smooth}(f_p, f_q) \ , \tag{1}$$

where p and q are some points in the image, f_p and f_q are the disparities assigned to them, $E^{p}_{data}(f_p)$ is the matching energy (error) for point p if assigned with disparity f_p, and $E^{p,q}_{smooth}(f_p, f_q)$ is the smoothness energy that imposes punishment if disparities of two neighboring points are not *smooth*. $\mathcal{N}$ is a neighboring system that contains the pairs of points which need to be imposed with smoothness assumption. The choice of $\mathcal{N}$ is essential because it will affect both the accuracy and efficiency of the algorithm.

In traditional algorithms, e.g. classic dynamic programming methods [3] [4], $\mathcal{N}$ is often chosen within the same scanline (without lost of generality, from now on we use scanline as rectified epipolar line) for imposing the disparity inconsistency punishment. The inter-scanline smoothness is usually ignored or considered in the post-processing procedure. The equivalent neighboring system graph is shown in Fig. 1.b. It is obvious that such asymmetric manner is unnatural and can not receive good performance. Based on this observation, graph-based *global* method (we use the terminology of [1]) has been proposed. In a global method, $\mathcal{N}$ is chosen as a four-connected grid in the image (shown in Fig. 1.a). Except the points on the image borders and corners, each point is connected with its four neighbors. This structure fully uses the correlation between neighboring points, and leads to the state-of-the-art performance [1] [5] [6]. But except for some special cases [7], the four-connected grid structure makes the minimization of the energy function generally NP-hard, and even using approximation methods are still very time-consuming. The traditional simulated annealing [8] algorithm usually takes hours to run, and the recent fast minimization methods, e.g. graph-cuts [6] and belief propagation[5], still need several minutes. They are still far from being in real-time.

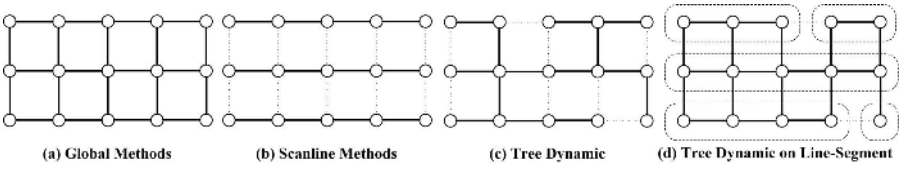


Fig. 1. Effective edges (marked by solid lines) for difference algorithms. In (d), points of each line segment are encircled by a dashed line.

Recently, Veksler [9] proposed a novel approach that connected all the pixels with a tree, and performed the dynamic programming on that tree (see Fig. 1.c). Since more edges are remained, and more importantly, horizontal and vertical edges are chosen in a symmetric style, better performance than classic dynamic programming methods is obtained. When using some special smoothness functin, the complexity of dynamic programming becomes as low as $O(hn)$ [9], supposing h is the number of possible disparities and n is the number of points.

Nevertheless, the performance of dynamic programming methods is still not comparable with that of global methods. We consider this problem by analyzing how much information has been lost in dynamic programming compared with global methods. Suppose the image is in the size of $N \times N$. We can see that the number of edges in Fig. 1.a is about $2N^2$, and in Fig. 1.b and Fig. 1.c the number of *effective*[1] edges has been reduced to about N^2. That is to say, half of the edges are discarded in dynamic programming methods, and much information embodied in these edges is lost. This is the main reason why their performance is apparently worse than global methods. Then our new approach is motivated by how to remain as many effective edges as possible while still utilizing the time efficiency of dynamic programming. This is achieved with the help of color segmentation.

Color segmentation is used in recent years to improve the performance of stereo correspondence in several publications [10] [11] [12] [13], called segment-based approaches. In the surfaces in the scene can be approximated by several slanted planes, better performance is achieved especially on textureless and discontinuity areas. The main assumption they use is that discontinuity may happen at the boundary of a segmented area. All the pixels within a segment are assigned with the same label, which means they must belong to the same plane in the scene. At the same time, we only need one vertex for all the pixels in one segment, which means the scale of the graph is decreased. Besides, segment-based methods commonly use a 3-parameter linear transform label space which can well model slanted planes in the scene.

In our approach, we segment each scanline into several line segments according to the colors of pixels. Pixels in one line segment are assigned with the same label, or we use the line segment as the matching unit. A tree is constructed to connect all segments, and smoothness is imposed in a line segment level. In this way, when the edge connecting two line segments in different scanlines are remained, it is equivalent to remain a number of edges in pixel level. The number of effective edges removed is greatly reduced, as shown in Fig. 1.d. Therefor our algorithm gives a much better approximation to the four-connected grid, and better correspondence result can be achieved. Our experimental results also show that the accuracy of our algorithm is comparable to the global methods, while the algorithm is still very time-efficient. Besides, using the 3-parameter linear transform space, we can well model the slanted plane and give a sub-pixel disparity map as the results. Disparities of the half-occluded area are given a good guess which will be shown in our experimental results in Sect. 4.1.

[1] The effective edges mentioned here means the information embodied in those edges are used.

The rest of the paper is organized as follow: Section 2 introduces our formulation of the stereo correspondence problem and how to compose a tree on line segments that can mostly estimate the grid structure. In Sect. 3, we discuss some implementation issues which are also essential to the performance of our algorithm. Experimental results and analysis are given in Sect. 4 and Sect. 5 is the conclusion.

2 Tree Dynamic Programming on Line Segments

In this section, we firstly formulate the stereo correspondence problem into a labelling problem in the line segment level. Then the construction of the tree for dynamic programming, which is the key of our algorithm, is introduced.

2.1 Problem Formulation

We denote the left and right images as I_L and I_R, and choose the left image as the reference image. The color segmentation algorithm, (described in Sect. 3.1 in detail), will segment the scanlines of the image into a set of line segments, denoted as S.Our goal is to assign each line segment $s \in S$ a label $f_s \in \mathcal{L}$, where $\mathcal{L}$ is the set of all possible labels (the label space). Each label in $\mathcal{L}$ represents a correspondence between points in left and right image respectively.

In order to model the slanted plane in the scene, the label space $\mathcal{L}$ is chosen to be a 3-parameter linear transform space:

$$f_s = \langle c_1, c_2, c_3 \rangle \Leftrightarrow$$
$$\forall p \in s, p \overset{\langle c_1,c_2,c_3 \rangle}{\leftrightarrow} p', \text{ with } p'_x = c_1 p_x + c_2 p_y + c_3, \;\; p'_y = p_y \;\; ,$$

where p' is a point in the right image, and $p \overset{\langle c_1,c_2,c_3 \rangle}{\leftrightarrow} p'$ means p and p' are corresponding points if assigned by a label $\langle c_1, c_2, c_3 \rangle$.

We formulate the correspondence problem in an energy minimization framework, and the optimal label configuration f_{opt} for line segments S is:

$$f_{opt}(S) = \arg\min_{D(S)} \sum_s E^s_{data}(f_s) + \sum_{\langle s,t \rangle \in \mathcal{N}} E^{s,t}_{smooth}(f_s, f_t) \;\; , \tag{2}$$

where $f(S)$ is the disparity map represented in the line segment level, and $\mathcal{N}$ is the neighboring system in the line segment level. $E^s_{data}(f_s)$ is the data term that measures how well the label f_s agrees with the input image pairs. One simple choice (which is used in our experiment in this paper) is to use the summation of the matching costs of all the points in the segment, i.e.:

$$E^s_{data}(f_s) = \sum_{p \in s} C(p, p'), \;\; p \overset{f_s}{\leftrightarrow} p', \;\; p \in I_L, p' \in I_R \;\; . \tag{3}$$

We use the combination of trimmed linear function and Potts model as our smoothness energy function $E^{s,t}_{smooth}$:

$$E^{s,t}_{smooth}(f_s, f_t) = v_{st} L_c(s,t) \cdot \begin{cases} s_T^{\lambda,\tau}(f_s, f_t) & FRNT(f_s) \text{ and } FRNT(f_t) \\ s_P(f_s, f_t) & otherwise \end{cases} , \quad (4)$$

where v_{st} is a coefficient which is a descending function of the color difference between s and t, and $L_c(s,t)$ is the length of the boundary shared by s and t. $FRNT(f_s)$ returns whether f_s represents a fronto plane, i.e.:

$$FRNT(\langle c_1, c_2, c_3 \rangle) = \begin{cases} true & c_1 = c_2 = 0 \\ false & otherwise \end{cases} .$$

s_T is the trimmed linear function defined as:

$$s_T^{\lambda,\tau}(\langle 0,0,c_3^s \rangle, \langle 0,0,c_3^t \rangle) = \min\{\lambda | c_3^s - c_3^t |, \tau\} .$$

s_P is the Potts smoothness function:

$$s_P(f_s, f_t) = \begin{cases} 0 & f_s = f_t \\ 1 & otherwise \end{cases} .$$

2.2 Constructing the Tree

Selecting the neighboring system $\mathcal{N}$ or constructing the tree is the key of our algorithm.

Let $G(V, E)$ be a graph with vertices V and edges E. Each vertex in V represents a line segment in S. All possible edges in E reflects the connection between two neighboring line segments. In general, G is a graph with many loops inside. Our goal is to find a spanning tree of G, denoted as G^T, to best estimate the full grid graph.

Two criteria for the selection of the optimal tree among all possible ones are used:

1. The line segments connected by a remained edge in the G^T are likely with similar disparities, they are probably belonging to the same region in the image, and
2. The connected line segment pair should have as many neighboring pixels as possible from each other.

The first criterion is similar to the strategy used in [9], which means the neighboring segments with similar color attribution values more likely share the same disparity. The second one assures that the edge that connects line segment pair sharing the longer boundary are preferred to remain in G^T.

Combining above two criteria, we define a weight function w_{st} between two neighboring line segments $\langle s,t \rangle$ as follows:

$$w_{st} = L_{max} - \sigma(\bar{I}_s, \bar{I}_t) L_c(s,t) ,$$

where L_{max} is the length of the longest segment of S in pixels, $\bar{I}_s$ and $\bar{I}_t$ are average colors of the segments s and t respectively, σ is a similarity function which returns a real value between 0 and 1 representing how similar the two colors are. For consecutive segments within the same scanline, $L_c(s,t)$ is 1, and for segments in neighboring scanlines $L_c(s,t) = \min\{s_{max}, t_{max}\} - \max\{s_{min}, t_{min}\}$, where s_{min} and t_{min} are horizontal coordinates of the left ends of segment s and t, and s_{max} and t_{max} are those of the right ends.

After defining the weights for each neighboring line segment pair, we use standard minimum-spanning tree (MST) algorithm, which can be found in any data-structure book, to choose the optimal tree. The complexity is almost linear to the number of segments $|S|$. It can be seen that the MID tree construction algorithm in [9] can be considered as a special case of ours, in which line segments have degenerated to individual points. In their situation, L_{max} and L_c are always 1, and then $w_{pq} = 1 - \sigma(I_p, I_q)$ is proportional to the intensity (or color) difference between two neighboring pixels.

3 Implementation

The flowchart of our algorithm is shown in Fig. 2. Each part is described in detail in the sub-sections.

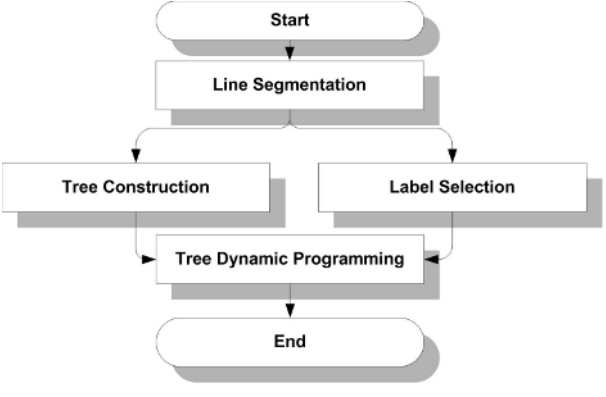


Fig. 2. The flowchart of our LSTDP algorithm

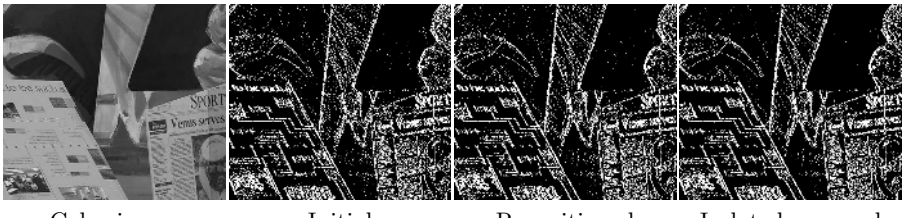
3.1 Line Segmentation

The line segmentation algorithm segments each scanline into several small parts, each of which contains pixels with similar colors. We do not choose some complicated segmentation algorithms, such as mean-shift [14] or normalized cuts [15], because they are not efficient and may become the bottleneck of the whole algorithm. Instead, we design a simple and fast scanline segmentation algorithm.

Our algorithm contains 3 steps as follows:

1. Computing Initialization Marks
   For each image line, we scan the pixels from left to right. Two registers stores the minimum and maximum intensities of the current segment. For color images, the registers are both vectors with three channels. If the difference between the minimum and maximum intensities are greater than a threshold T_{seg}, a mark is put at the current position and two registers are reset. After processing, the points between two marks are considered as one line segment. The maximum intensity difference between pixels within a segment is no more than T_{seg}.
2. Repositioning Marks
   The marks made in the first step may not lay at the accurate edge. So a repositioning procedure is performed. Each mark is moved to the near local maximum of intensity gradients without changing their orders.
3. Removing Isolated Marks
   The image noise often leads to some isolated marks in the image, and makes the image being wrongly segmented. We check each mark and remove those who do not have enough close neighbors in 2D area.

This segmentation method works fast and produces good segmentation in our algorithm. We show the results of segmentation results in Fig. 3.



Color image Initial Repositioned Isolated removed

Fig. 3. Results of different steps of the segmentation on the "Venus" image

3.2 Label Selection

The label set $\mathcal{L}$ is first initialized with all possible fronto linear transforms, i.e. $\{\langle 0, 0, -d\rangle | d = 0, \ldots, D_{max}\}$.

Then we need to estimate some possible 3-parameter linear transform labels. To do this, we first segment both left and right images. Line segments on two images are matched locally according to their average colors. For each matched line segment pair, whose colors are similar enough, we obtain two matched point pairs(the corresponding ends). This matching is rough and may contains many errors. A robust estimation method, like M-estimators [16], is then used to extract the linear planes by fitting on the sparse correspondences robustly.

3.3 Tree Construction

The algorithm described in Sect. 2.2 is used to construct a tree on the reference image.

3.4 Dynamic Programming

Dynamic programming is performed on the constructed tree to minimize the energy function defined in (2). Readers can find more details in [9]. Using the technology introduced in [17] and [9], our energy function with smoothness energy defined in (4) can be minimized with the complexity of $O(hn)$.

4 Experiments

Our experiments include two parts. First, we perform our algorithm on the testbed of Middlebury University [1], and performance is compared with other algorithms submitted to that testbed. To further test the accuracy and efficiency on the real-time system, we embed our algorithm into a realtime automatic navigation system, in which outdoor image series are processed.

4.1 Experiments on Middlebury dataset

We adopted Birchfield and Tomasi's matching cost [18] which is insensitive to image sampling as $C(p, p')$ in (3). v_{st} in (4) is defined as

$$v_{st} = C_1 + \sigma(\bar{I}_s, \bar{I}_t)C_2$$

All parameters are listed in Table 1, and are used for all image pairs. Computed disparity maps are shown in Fig. 4 accompany with the results from [9]. We also listed the time (in milliseconds) of the different parts of our algorithm, i.e. DSI (Disparity Space Image[1]) computing, line segmentation, label selection, and tree dynamic programming, and the total time in Table 3. They are measured on a computer with an Intel Pentium IV 2.4 GHz processor. We submit the results into Middlebury test-bed and show the accuracy evaluations in Table 2. Three criteria are used in the evaluation table which are percentages of: bad points in *non-occluded* area, in *all* area, and *near discontinuities*. A *bad* point is a point whose absolute disparity error is greater than one [1].

From the evaluation table we can see that our algorithm can achieve overall accuracy comparable with the state-of-the-art global methods (4 out of 13). The result of "venus" is almost equal to the best one. For all the four images, the rank of "all" column of our algorithm, which includes the guessing for half-occluded areas, is better than the other two. That is because we use the line segment as the matching unit, and the disparities of some occluded pixels can be inferred by the disparity of the segment where the occluded pixels belong to. Besides the good performance, our algorithm runs very fast. Processing time for "tsukuba" is only about 160ms, and the other three can be processed within one

Table 1. Parameter values set for experiments for Middlebury image pairs

Parameter	C_1	C_2	λ	τ	T_{seg}
Value	5	75	0.5	1.0	20

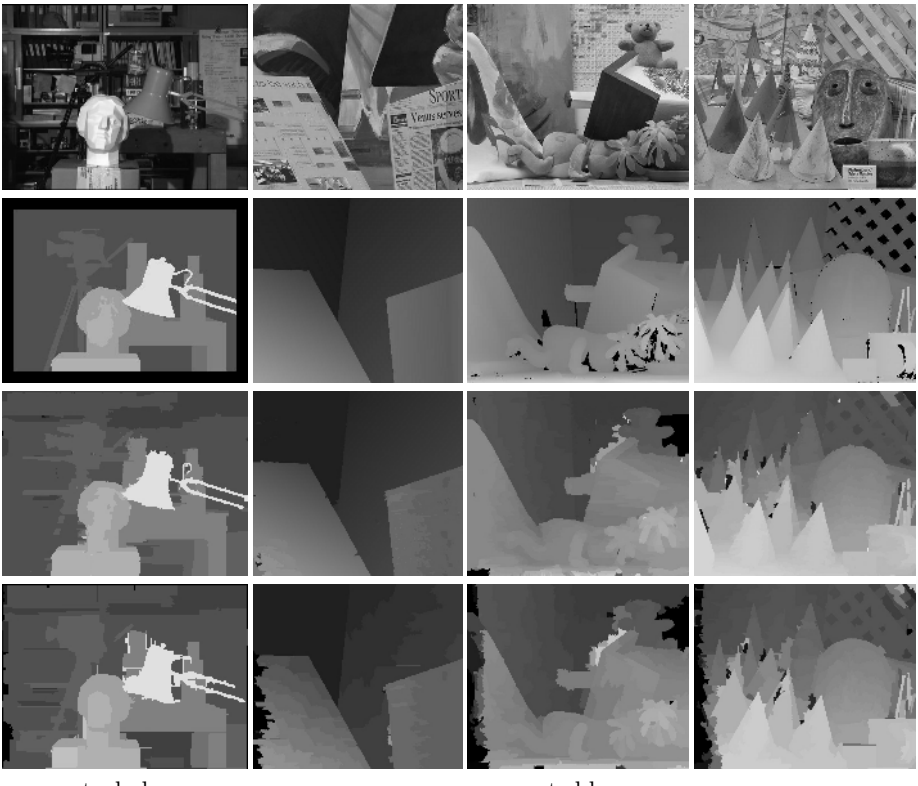


Fig. 4. Experimental results for Middlebury database. The first row is left images, the second row is ground truth of disparity map, the third row is results by our LSTDP algorithm, and the last row is the results of pixel-based Tree DP method from [9].

Table 2. Accuracy Evaluation Results on Middlebury Stereo Test-bed

Algorithm	**Tsukuba**			**Venus**			**Teddy**			**Cones**		
	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
Sym.BP+occl	**0.97** $_{1}$	1.75 $_{2}$	**5.09** $_{1}$	**0.16** $_{1}$	0.33 $_{2}$	**2.19** $_{1}$	6.47 $_{3}$	10.7 $_{2}$	17.0 $_{3}$	4.79 $_{4}$	10.7 $_{3}$	10.9 $_{3}$
Segm+visb	1.30 $_{4}$	**1.57** $_{1}$	6.92 $_{4}$	0.79 $_{3}$	1.06 $_{3}$	6.76 $_{5}$	**5.00** $_{1}$	**6.54** $_{1}$	**12.3** $_{1}$	3.72 $_{2}$	**8.62** $_{1}$	10.2 $_{2}$
SemiGlob	3.26 $_{9}$	3.96 $_{8}$	12.8$_{12}$	1.00 $_{4}$	1.57 $_{4}$	11.3 $_{9}$	6.02 $_{2}$	12.2 $_{3}$	16.3 $_{2}$	**3.06** $_{1}$	9.75 $_{2}$	**8.90** $_{1}$
LSTDP	*1.93* $_{6}$	*2.59* $_{6}$	*9.70* $_{8}$	*0.19* $_{2}$	***0.26*** $_{1}$	*2.49* $_{2}$	*11.1* $_{6}$	*16.4* $_{5}$	*23.4* $_{8}$	*6.39* $_{7}$	*11.8* $_{5}$	*13.5* $_{7}$
Layered	1.57 $_{5}$	1.87 $_{3}$	8.28 $_{5}$	1.34 $_{6}$	1.85 $_{5}$	6.85 $_{6}$	8.64 $_{4}$	14.3 $_{4}$	18.5 $_{4}$	6.59 $_{8}$	14.7 $_{8}$	14.4 $_{8}$
GC+occ	1.19 $_{2}$	2.01 $_{5}$	6.24 $_{2}$	1.64 $_{8}$	2.19 $_{8}$	6.75 $_{4}$	11.2 $_{7}$	17.4 $_{7}$	19.8 $_{5}$	5.36 $_{6}$	12.4 $_{7}$	13.0 $_{6}$
MultiCamGC	1.27 $_{3}$	1.99 $_{4}$	6.48 $_{3}$	2.79$_{10}$	3.13 $_{9}$	3.60 $_{3}$	12.0 $_{8}$	17.6 $_{8}$	22.0 $_{7}$	4.89 $_{5}$	11.8 $_{6}$	12.1 $_{4}$
TensorVoting	3.79$_{10}$	4.79$_{10}$	8.86 $_{6}$	1.23 $_{5}$	1.88 $_{6}$	11.5$_{10}$	9.76 $_{5}$	17.0 $_{6}$	24.0 $_{9}$	4.38 $_{3}$	11.4 $_{4}$	12.2 $_{5}$
TreeDP	1.99 $_{8}$	2.84 $_{7}$	9.96 $_{9}$	1.41 $_{7}$	2.10 $_{7}$	7.74 $_{7}$	15.9$_{10}$	23.9$_{10}$	27.1$_{12}$	10.0$_{10}$	18.3$_{10}$	18.9$_{10}$
⋮							⋮					
SO[1c]	5.08$_{12}$	7.22$_{13}$	12.2$_{11}$	9.44$_{12}$	10.9$_{12}$	21.9$_{13}$	19.9$_{13}$	28.2$_{13}$	26.3$_{11}$	13.0$_{13}$	22.8$_{13}$	22.3$_{12}$

second. From Table 3, we can see that besides the dynamic programming modula, half of the processing time is spent on preprocessing modules, and they can be greatly accelerated with special hardware if necessary. Like other segment-based methods, some artifacts caused by segmentation can be found in the disparity

Table 3. Time Analysis of Our Algorithm on Middlebury Dataset

	Size	$\|S\|$	Disp. Range	DSI	Line-Segm.	Lab-Sel	Tree-DP	Total
tsukuba	384×288	19621	0..15	30	8	37	88	**163**
venus	434×384	29664	0..19	76	12	89	143	**320**
teddy	450×375	37435	0..59	195	10	359	299	**863**
cones	450×375	50780	0..59	194	16	170	370	**750**

† Unit for all the time (the last 5 columns) in this table is millisecond.

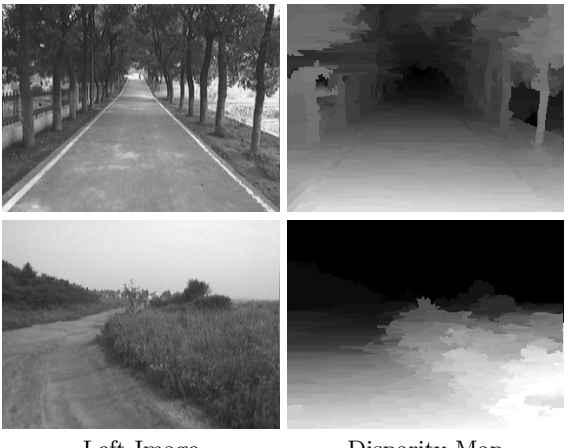
Table 4. Effective edges of three kinds of algorithms

	Size	$\|S\|$	Global	Pixel-TDP	LSTDP		
					Total	Hard	Soft
tsukuba	384×288	19621	220512	110591 (50.1%)	192517 (87.3%)	90971	101546
venus	434×384	29664	332494	166655 (50.1%)	283241 (85.2%)	136558	146683
teddy	450×375	37435	336675	168749 (50.1%)	274557 (81.6%)	131315	143242
cones	450×375	50780	336675	168749 (50.1%)	259205 (77.0%)	117970	141235

† The percentages of equivalent edges of *Pixel-TDP* and *LSTDP* over full grid(*Global*) are listed in brackets.
‡ In the *LSTDP* columns, *Hard* means edges connecting pixels within a line segment, and *Soft* means the equivalent edges crossing line segments.

map. But this only happens along the scanline direction, because we do not perform a hard constraint on inter-scanlines.

Moreover, we give the statistics on the numbers of effective edges in Table 4. Note that the effective edges here are not the edges in the tree on the line segment level, but the equivalent edges in pixel level. Our algorithm remains much more edges than pixel-based dynamic programming method (*Pixel-TDP*). Less than a quarter of the edges are discarded, and for images with less texture, e.g. "tsukuba", almost 90% of edges are remained.



Left Image Disparity Map

Fig. 5. Disparity and elevation results in a real-time outdoor automatic navigation system. The upper row is one of the frame captured on an avenue, and the lower row is from a country road.

4.2 Results on a Real-Time System

Our algorithm is used in a real-time outdoor stereo system. Because the outdoor images are of relatively higher contrast and for obtaining higher efficiency, the input images are first converted into gray-level images. The dynamic histogram warping algorithm by Cox *et al.* [19] is used to rectify the difference of image capturing. We only use fronto labels and hence label selection is not performed. The size of the input images is 320×240, and disparity ranges from 0 to 40. No acceleration hardware is used. Two frames of results are shown in Fig. 5. One is from an avenue environment and the other is from a country road. We can see that our matching results are rather accurate. The system is running on a Dual Intel Xeron 2.4 GHz processor, and the processing time for each frame is only 60–70ms.

5 Conclusion

In this paper, we proposed a fast stereo correspondence algorithm based on line segments using tree dynamic programming. From our preliminary experimental results on both standard image pairs and real image sequences, it can be seen that the performance of our algorithm is comparable to those of state-of-the-art algorithms while our algorithm runs much faster. It can be used in different real-time systems providing high accuracy disparity map.

We will continue our work on this proposed method to further improve the performance of our method. Our future work includes occlusion modelling, new construction rules for the tree, and parallel algorithm for the tree dynamic programming.

References

1. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int'l J. Comput. Vision **47**(1) (2002) 7–42 http://cat.middlebury.edu/stereo/.
2. Brown, M.Z., Burschka, D., Hager, G.D.: Advances in computational stereo. IEEE Trans. Pattern Anal. Machine Intell. **25**(8) (2003) 993–1008
3. Baker, H., Binford, T.: Depth from edge and intensity based stereo. In: Int'l Joint Conf. on Artificial Intell. Volume 2 of 20-26. (1981) 384–390
4. Cox, I., Hingorani, S., Rao, S., Maggs, B.: A maximum likelyhood stereo algorithm. Computer Vision, Graphics and Image Processing **25**(8) (2003) 993–1008
5. Sun, J., Li, Y., Kang, S.B., Shum, H.Y.: Symmetric stereo matching for occlusion handling. In: Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition. Volume 2. (2005) 399–406
6. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. In: Proc. IEEE Int'l Conf. on Computer Vision. Volume 2. (2001) 508–515
7. Roy, S.: Stereo without epipolar lines: A maximum-flow formulation. Int'l J. Comput. Vision **24**(2/3) (1999) 147–161

8. Geman, S., Geman, D.: Gibbs distributions, and the baysian restoration of images. IEEE Trans. Pattern Anal. Machine Intell. **6** (1984) 721–741
9. Veksler, O.: Stereo correspondenc by dynamic programming on a tree. In: Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition. Volume 2 of 20-26. (2005) 384–390
10. Tao, H., Sawhney, H.S., Kumar, R.: A global matching framework for stereo computation. In: Proc. IEEE Int'l Conf. on Computer Vision. Volume 1. (2001) 532–539
11. Wei, Y., Quan, L.: Region-based progressive stereo matching. In: Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition. Volume 1. (2004) 106–113
12. Hong, L., Chen, G.: Segment-based stereo matching using graph cuts. In: Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition. Volume 1. (2004) 74–81
13. Deng, Y., Yang, Q., Lin, X., Tang, X.: A symmetric patch-based correspondence model for occlusion handling. In: Proc. IEEE Int'l Conf. on Computer Vision. Volume II., Beijing, China, 2005 (2005) 1316–1322
14. Comaniciu, D., Meer, P.: Robust analysis of feature spaces: Color image segmentation. In: Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition, Puerto Rico (1997) 750–755
15. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Machine Intell. **22**(8) (2000) 888–905
16. Stewart, C.V.: Robust parameter estimation in computer vision. SIAM Reviews **41**(3) (1999) 513–537
17. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. In: Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition. Volume 1. (2004) 261–268
18. Birchfield, S., Tomasi, C.: A pixel dissimilarity measure that is insensitive to image sampling. IEEE Trans. Pattern Anal. Machine Intell. **20**(4) (1998) 401–406
19. Cox, I.J., Roy, S., Hingorani, S.L.: Dynamic histogram warping of image pairs for constant image brightness. In: Proc. Int'l Conf. on Image Processing. Volume II. (1995) 366–369