Stereo Matching based on Multi-Direction Polynomial Model

Xiao Tan[a,b], Changming Sun[b], Tuan D. Pham[c]

[a] *The University of New South Wales, Canberra, ACT 2600, Australia.*
[b] *CSIRO Digital Productivity, PO Box 52, North Ryde, NSW 1670, Australia.*
[c] *Aizu Research Cluster for Medical Engineering and Informatics, The University of Aizu, Fukushima 965-8580, Japan.*

Abstract

This paper presents a segmentation based stereo matching algorithm. For the purposes of both preserving the shape of object surfaces and being robust to under segmentations, we introduce a new scene formulation where the reference image is divided into overlapping lines. The disparity value and the index of pixels on lines are modeled by polynomial functions. Polynomial functions are propagated among lines to obtain smooth surfaces via solving energy minimizing problems. Finally, the disparity of pixels is estimated from the disparity fields provided by lines. Because lines in multiple directions implicitly segment different objects in an under segmentation region, our method is robust for under segmented regions where it is usually difficult for conventional region based methods to produce satisfactory results. Experimental results demonstrate that the proposed method has an outstanding performance compared with the current state-of-the-art methods. The scene representation method in this work is also a powerful approach to surface based scene representations.

Keywords:
Stereo matching, Multi-direction polynomial model, Energy optimization

1. Introduction

Stereo matching is the process of finding the correspondences between pixels in two or more images which are captured of the same scene. Matching algorithms can be roughly categorized into pixel based and region based methods. Algorithms in the former category assign each pixel a disparity

value according to a similarity measure between two matching pixels such as sum-of-squared-differences or mutual information [1, 2] and some constraints such as smoothness constraint and uniqueness constraint. Finding correspondences is then formulated as a task of minimizing an energy function which can be solved by global optimization methods, such as Graph Cuts (GC) based [3], maximum-surface based [4], and Belief Propagation (BP) based methods [5, 6, 7] or formulated as a filtering like tasks such as cost volume filtering [8], soft aggregation [9], and information permeability [10]. In the region based algorithms, a region segmentation is carried out beforehand, then a fitting process is conducted on each segment according to a specified model such as the planar model [11, 12, 13], the B-splines model [14], and the plane-plus-parallax model [15].

Compared with the pixel based methods, the region based algorithms usually provide good results at textureless regions and at the disparity boundaries which usually coincide with region boundaries. Moreover, they perform well at highly slanted surfaces. However, the region based algorithms are subject to region segmentations results, particularly under segmentations where the assumption that the disparity in a segment follows a particular model is broken. A soft segmentation based method proposed in [14] decides if a pixel belongs to a surface model by minimizing an energy function. A similar method described in [15] updates the relationship between pixels and segments in the object level. Using the soft segmentation and the object information, these two methods are robust to under segmentations. However, as noticed in their work these types of methods are computationally expensive.

This paper proposes a stereo matching algorithm based on a new formulation: multi-direction polynomial model (MDPM) to represent the scene. In this model, the scene is decomposed into intersected lines in multi-directions using the boundaries from region segments. The disparity of pixels on these lines is fitted by a polynomial function. The intersected lines give disparity estimations to the intersecting pixel according to their polynomial functions. The disparities of pixels are chosen from the disparity values of all these estimation values so that the scene is both piecewise smooth and consistent with the local observation. As will be shown in the later parts of this paper, our model takes advantage of the region based methods in that it has good performance at highly slanted surfaces; another advantage over the conventional segmentation based methods is that our model is robust to under segmentations. Compared with scanline based methods [16, 17, 18], our method

has the following advantages: (1) Using the polynomial model gives better representation than using the linear model [18]; (2) By choosing the disparity value provided by lines rather than all disparity values within the search ranges, the assignment of pixel disparities is carried out efficiently.

2. Overview of the Idea and Notations

Unlike previous region based methods, this study models the relationship between the disparity of a pixel and its pixel index by a polynomial function on 1D lines. The lines are bounded by region boundaries from region segmentation results (see Fig. 1). Due to the existence of noises in cost aggregation results, the polynomial regression results on lines at some pixels may be incorrect. Fortunately, we find that by alternating the direction of lines at these incorrect pixels, and carrying out polynomial regression once again, the regression results will then provide correct results for many of them. An interesting question that follows is whether we can obtain better results by "fusing" results from polynomial functions in different directions. This paper answers this question by proposing a method to obtain a good disparity map. The proposed method consists of the following steps: (1) Polynomial regression: calculating a polynomial function along each line; (2) Polynomial propagation: obtaining disparity fields where object surfaces are smooth while different objects in under segmented regions are divided; (3) Disparity fusion: combing multiple disparity fields to obtain the final disparity results. Each disparity field corresponds to polynomial propagation results on lines in one specified direction.

Let I be the reference image of the input image pairs, and $L^{(\theta)}$ be a line at direction θ to the horizontal axis. In the proposed algorithm, four directions are used, where θ is equal to $0, \pi/4, \pi/2$, and $3\pi/4$, respectively (see Fig. 2(a)). Lines passing through a pixel are bounded by the region boundaries containing this pixel (see Fig. 2 (b)). In this paper, a region segment is called a region for short.

We model the disparity value of pixels and the index of pixels by a polynomial function: $d_p^{(\theta)} = \mathcal{F}^{(\theta)}(C_p)$, where C_p is the pixel index of p; $\mathcal{F}^{(\theta)}$ is the polynomial function of p; and θ indicates the direction of the line. The polynomial function is explicitly expressed as $\mathcal{F}^{(\theta)}(C_p) = \sum_{i=0}^{\Theta} \alpha_i C_p^i$ where α_i is the coefficient of the ith order term, and Θ is the highest order of the

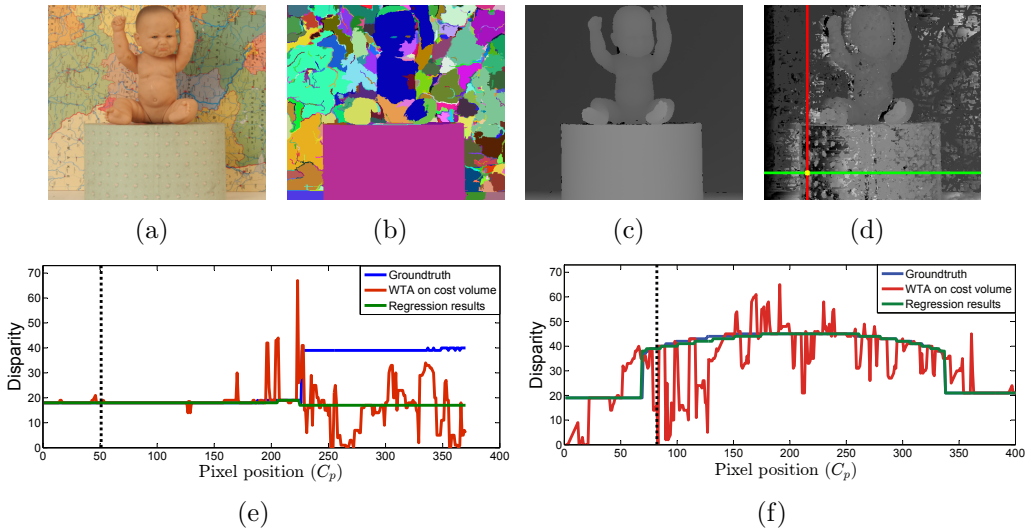


Figure 1: Multi-direction polynomial regression. (a) Reference image. (b) Region segmentation results. (c) Groundtruth. (d) Polynomial regression along vertical and horizontal direction at the pixel colored in yellow. (e) Polynomial regression results in vertical direction (along the red line in (d)). (f) Polynomial regression results in horizontal direction (along the green line in (d)). The disparity of the yellow pixel is denoted in the vertical dash line in (e) and (f). Regression results along vertical direction is incorrect. After altering the line to the horizontal direction, the regression result at the yellow pixel becomes correct.

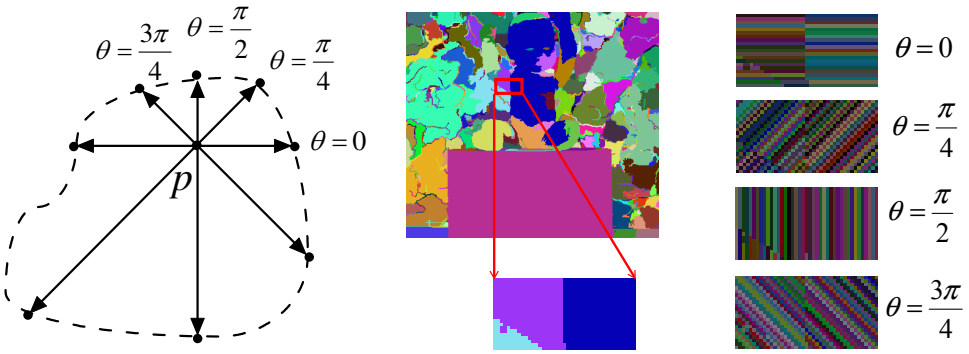
polynomial function. In the rest of the paper, we call polynomial function as polynomial for simplicity.

3. Polynomial Regression and Polynomial Propagation

Polynomials are calculated separately for lines in each of the four directions. We take horizontal direction ($\theta = 0$) as an example to present polynomial regression and the propagation. Methods shown in this section are similarly applied to the other directions. In this research, we assume that the stereo image pairs are rectified and take the left image as the reference image.

3.1. Polynomial Regression

Polynomials are computed on individual lines using least squares regression with a random sampling method. More specifically, we randomly sample



(a) (b)

Figure 2: Illustration of the MDPM model. (a) Lines in four directions at pixel p. Ending pixels of lines are located on region boundaries. (b) A close-up view of lines in four directions at a certain region

$n+1$ non-occluded pixels and solve a linear system between their image coordinate and initial disparity values to calculate a n-th order polynomial function. The initial disparity values can be obtained by directly applying the winner-take-all (WTA) method to a cost volume D or by using low level processing methods such as the box-filtering, the edge-aware filtering [8], or the hierarchical belief propagation approach [6] on D to suppress noises. The occluded pixels are detected before performing the polynomial regression. To this end, a cross checking process is carried out on the initial disparity maps of the left and the right images. We distinguish a pixel as an occluded pixel or a non-occluded pixel because the cost volume of an occluded pixel contains the mis-leading information instead of information about true disparity value of this pixel, which will degrade the performance of polynomial regression especially in regions where occluded pixels dominate. For the sake of robustness in high order regression, we scale pixel coordinates to the range of $[0, 1]$ and use the scaled coordinates $\hat{C}_p$ for regression. After regression we adjust the obtained coefficients $\hat{\alpha}_i$ to α_i so that $\sum_{i=0}^{\Theta} \hat{\alpha}_i \hat{C}_p = \sum_{i=0}^{\Theta} \alpha_i C_p$ is satisfied for all pixels on this line. We perform several trials for random sampling. The number of total trails and the implementation details will be discussed in Section 3.3.1. The polynomial of line L we are seeking for is the one that has the minimum regression score among all trials. The regression score is

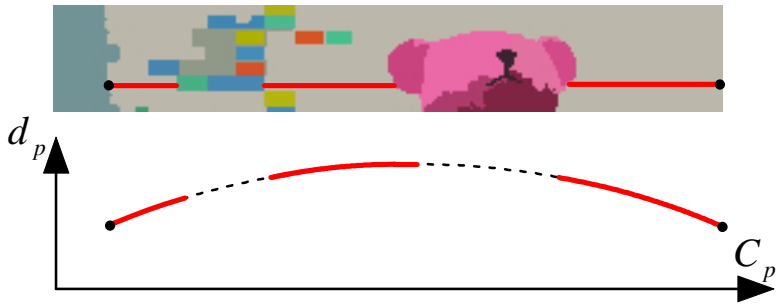


Figure 3: Polynomial regression for an unhinged line. Top: A line denoted in the red line on the textureless wall is unhinged into several parts of the "Teddy" bear. Bottom: An identical polynomial is assigned to different parts of this unhinged line.

defined as:

$$E_{\text{reg}}(\mathcal{F}) = \sum_{p \in L} D(d_p) O(p) + \sum_i \lambda_i |\alpha_i| \tag{1}$$

where $D(d_p)$ is the cost of pixel p at disparity d_p which is calculated using polynomial $\mathcal{F}$ at p, and O is a boolean field where occluded pixel is flagged by 0. We add a penalty λ_i to the absolute value of coefficients to make the polynomial as simple as possible, i.e., a constant value is the best choice and a linear model is preferred than a higher order polynomial. This is achieved by setting: $\lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_\Theta$. Very small lines are discarded as there is no sufficient data for reliable regression.

On a concave region, a line may be broken into several parts. We refer to this kind of line as an unhinged line. For the unhinged line, we treat all parts (line segments) as a whole (see Fig. 3 for detailed description). Treating the unhinged line as a whole benefits the polynomial regression for backgrounds which are occluded by foreground objects such as the textureless wall behind the Teddy bear in the "Teddy" dataset. But it is clearly incorrect in under segmented regions which contain different objects. However, as shown later in Fig. 7, our method can also handle this situation.

3.2. Polynomial Propagation

The regression results obtained from polynomial regression are not usually satisfactory. Because polynomials are calculated by considering pixels on only one line. Neighboring pixels of different lines on a smooth surface may have

large disparity jumps. To produce results which are both piecewise smooth and consistent with the observation from the cost volume, the polynomial models are propagated among lines. The first stage of propagation is carried out along the direction of lines. Then a second stage propagation is carried out among lines in the same region. These two stages could work in an iterative way; however, we find the results from the iterative manner are not very different compared with results from the straight-forward manner.

3.2.1. Propagation along Line Direction

We formulate the propagation as a discrete labeling problem in the line domain. In this propagation, we take polynomial regression results as inputs. Taking $\theta = 0$ for example, the propagation is carried out on horizontal lines. Let $\mathbb{L}_y$ be the set of the lines in the reference image having the same y coordinate, $\mathbb{M}_y$ be the regression results of lines in $\mathbb{L}_y$. Our goal is to find a labeling f that assigns each line $L \in \mathbb{L}_y$ a polynomial $\mathcal{F}_L$, $\mathcal{F}_L \in \mathbb{M}_y$. Introduce a trinary function: $\nabla(\mathcal{F}_L, \mathcal{F}_{L'}, p) = |d_p(\mathcal{F}_L) - d_p(\mathcal{F}_{L'})|$ to denote the disparity jump between polynomials $\mathcal{F}_L$ and $\mathcal{F}_{L'}$ at pixel p. The labeling is found by minimizing the following energy function:

$$E_y(f) = E_{\text{data}}(f) + E^{(1)}_{\text{smooth}}(f) \tag{2}$$

For an image with height M, the propagation is individually carried out from $y = 1$ to $y = M$. The energy terms are defined as:

- $E_{\text{data}}(f)$: The data term is the summation to the cost volume of all visible pixels:

$$E_{\text{data}}(f) = \sum_{L \in \mathbb{L}_y} \sum_{p \in L} D\left(d_p(\mathcal{F}_L)\right) O(p) \tag{3}$$

  where $d_p(\mathcal{F}_L)$ is the disparity value calculated from the polynomial $\mathcal{F}_L$ at pixel p.

- $E^{(1)}_{\text{smooth}}(f)$: The smoothness term encourages piecewise smoothness of two neighboring lines:

$$E^{(1)}_{\text{smooth}}(f) = \mu \sum_{p \in \mathbb{B}} \nabla(\mathcal{F}_L, \mathcal{F}_{L'}, p) \tag{4}$$

  where the $\mathbb{B}$ is the set of boundary pixels, $\mathcal{F}_L$ and $\mathcal{F}_{L'}$ are the polynomials assigned respectively to the two lines which meet at p (see Fig. 4); μ is a parameter regularizing the strength of the smoothness term. The larger value μ is, the stronger the smoothness term will become.

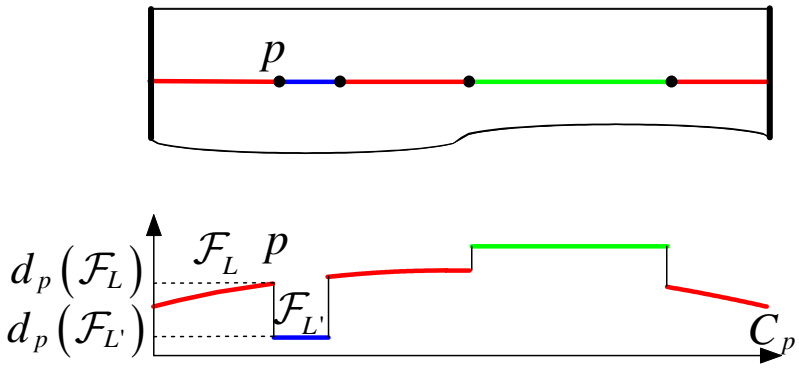


Figure 4: Disparity jump of two polynomials at a pixel. Top: Different lines are denoted in different colors. Black dots are boundary pixels in $\mathbb{B}$. Left and right image boundaries are denoted in bold lines. Bottom: A penalty is imposed to the disparity jump at the joint pixel p.

3.2.2. Propagation within Regions

This propagation is carried out among lines in the same region by taking results from the previous propagation (propagation along line direction) as inputs. Similar to propagation along line direction, this propagation is also formulated as the labeling problem in the line domain. Let $\mathbb{L}_{\mathcal{R}}$ be the set of all lines in the same region $\mathcal{R}$, $\mathbb{M}_{\mathcal{R}}$ be the previous propagation results of lines in $\mathbb{L}_{\mathcal{R}}$. The labeling we are to find is f': $\mathbb{L}_{\mathcal{R}} \rightarrow \mathbb{M}_{\mathcal{R}}$ from line to polynomials. Propagations are accomplished individually on each region. For region $\mathcal{R}$, the labeling problem is solved by minimizing the following energy function:

$$E_{\mathcal{R}}(f') = E_{\text{data}}(f') + E^{(2)}_{\text{smooth}}(f') \tag{5}$$

Energy terms are defined as follows:

- $E_{\text{data}}(f')$: The data term is identical to that used in propagation along line direction
- $E^{(2)}_{\text{smooth}}(f')$: This smoothness term penalizes the disparity jump between neighboring lines:

$$E^{(2)}_{\text{smooth}}(f') = \sum_{(L,L')\in\mathcal{N}_{\mathcal{R}}} \mathcal{P}(\mathcal{F}_L, \mathcal{F}_{L'}) \tag{6}$$

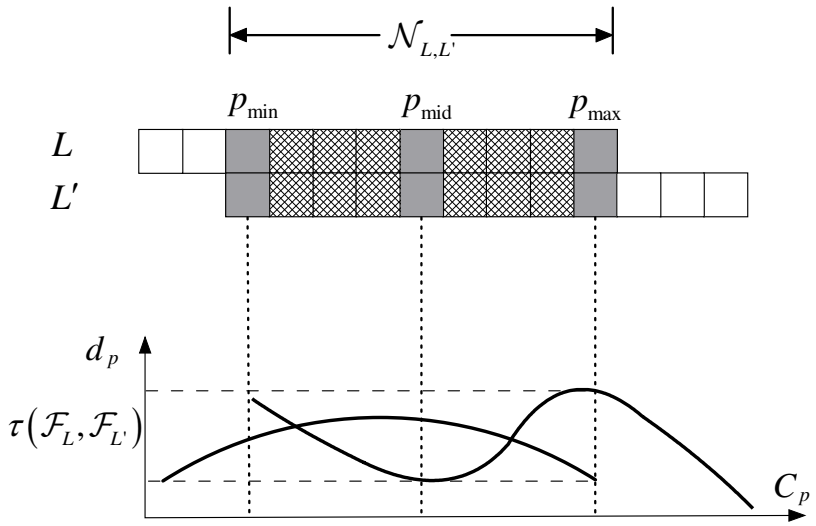


Figure 5: Disparity consistency measurement between two neighboring lines. In this example, disparity jump at pixel $p_{\max}$ is larger than that at $p_{\min}$ and p_{mid}. Therefore, $\tau(\mathcal{F}_L, \mathcal{F}_{L'}) = \nabla(\mathcal{F}_L, \mathcal{F}_{L'}, p_{\max})$.

where $\mathcal{N}_{\mathcal{R}}$ denotes the set of all neighboring line pairs in $\mathcal{R}$, $\mathcal{F}_L$ and $\mathcal{F}_{L'}$ are polynomials assigned to L and L' by f', and $\mathcal{P}$ is the penalty function based on a disparity consistency measurement $\tau(\mathcal{F}_L, \mathcal{F}_{L'})$ which is described next.

A 4-connected pixel pair (p, p') satisfying $p \in L$ and $p' \in L'$ is called a neighboring pixel pair of (L, L'). Denote pixels of all neighboring pixel pairs of (L, L') by $\mathcal{N}_{L,L'}$. Disparity consistency between L and L' requires that the difference between $d_p(\mathcal{F}_L)$ and $d_p(\mathcal{F}_{L'})$ is small for all $p \in \mathcal{N}_{L,L'}$. However, checking all pixels in $\mathcal{N}_{L,L'}$ can be computationally expensive. Therefore, we only use three values to measure the disparity consistency. More specifically, let $p_{\min}$ and $p_{\max}$ be two pixels having the minimum and maximum pixel index in $\mathcal{N}_{L,L'}$. Further, let p_{mid} be the midpoint of $p_{\min}$ and $p_{\max}$. Then $\tau(\mathcal{F}_L, \mathcal{F}_{L'})$ is defined by the maximum disparity jump at these three points: $\tau(\mathcal{F}_L, \mathcal{F}_{L'}) = \max_{p \in \{p_{\min}, p_{\max}, p_{\mathrm{mid}}\}} \nabla(\mathcal{F}_L, \mathcal{F}_{L'}, p)$ (see Fig. 5).

Note that point p_{mid} may not be in $\mathcal{N}_{L,L'}$ in the case of an unhinged line (see Fig. 3). But the disparity jump at p_{mid} gives a good measure for the disparity consistency of two polynomials. Theoretically, a small $\tau(\mathcal{F}_L, \mathcal{F}_{L'})$ is not sufficient to guarantee that disparity jump of all pixels in $\mathcal{N}_{L,L'}$ is small,

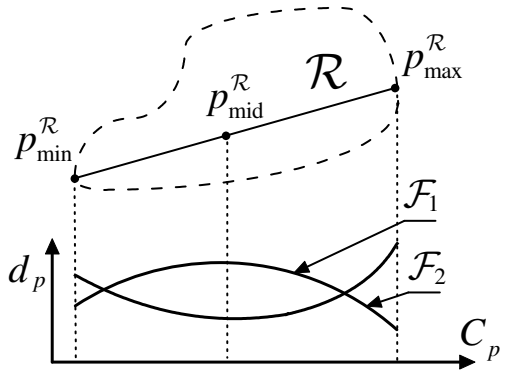


Figure 6: Grouping polynomials in $\mathbb{M}_{\mathcal{R}}$ based on the disparity jump of two polynomials $\mathcal{F}_1$ and $\mathcal{F}_2$ at points $p^{\mathcal{R}}_{\text{mid}}$, $p^{\mathcal{R}}_{\min}$, and $p^{\mathcal{R}}_{\max}$.

e.g., two polynomials having the same disparity value at these three points may be different at other pixels. However, this case is extremely rare and τ is always effective and efficient in practice. To allow small jumps which often happen on slanted surfaces and to impose a large penalty to large jumps, we define $\mathcal{P}$ as:

$$\mathcal{P}\left(\mathcal{F}_L, \mathcal{F}_{L'}\right) = \begin{cases} 0 & \text{if } \tau\left(\mathcal{F}_L, \mathcal{F}_{L'}\right) = 0 \\ P_1 & \text{if } 0 < \tau\left(\mathcal{F}_L, \mathcal{F}_{L'}\right) \leq 1 \\ P_2 & \text{otherwise} \end{cases} \tag{7}$$

where $P_2 > P_1$.

3.3. Implementation Details

This section describes implementation details in solving minimization problems as mentioned above.

3.3.1. Polynomial Regression

We prefer using low order polynomials to present the scene. But the order of a polynomial obtained by the least squares method is sensitive to noises, For example, if the disparity value and pixel index follow a linear model and the disparity value of some pixels are incorrect in the WTA results, least squares regression with the order of α_Θ will not give a linear model. To solve this problem, we carry out least squares regression with random sampling for $(\alpha_\Theta + 1)$ runs and the order of the polynomial in the kth run is set to $k - 1$. The number of trials in each run is set to 20. Thus, the total number of trials

is $20 \times (\Theta + 1)$. The final polynomial is the one having the minimum value (Eq. (1)) from all $20 \times (\Theta + 1)$ polynomials obtained by trials.

3.3.2. Propagation along Line Direction

The smoothness term in Eq. (2) is a submodular function based on the definition in [19] and the approximate solution can be efficiently found by using α-expansion [3].

3.3.3. Propagation within Regions

The penalty function as used in Eq. (7) is a non-submodular term so the energy function (5) can not be optimized by using α-expansion. Fortunately, the neighboring relation of two lines in a segment does not contain loops. Therefore we can use dynamic programming (DP) to find the minimum solution to Eq. (5). Because the complexity of DP is $|\mathbb{M}_{\mathcal{R}}|^2 |\mathbb{L}_{\mathcal{R}}|$, it is not efficient to use all polynomials in $\mathbb{M}_{\mathcal{R}}$, particularly for segments where $|\mathbb{M}_{\mathcal{R}}|$ is huge. Fortunately, polynomials in a segment are usually similar or even identical with each other, e.g., lines in a segment of a plane. Therefore, before performing DP, we group polynomials in $\mathbb{M}_{\mathcal{R}}$.

The grouping is based on the disparity jump between two polynomials at the following three points: $p^{\mathcal{R}}_{\min}$ (the pixel having the minimum coordinate value in the region $\mathcal{R}$), $p^{\mathcal{R}}_{\max}$ (the pixel having the maximum coordinate value in the region $\mathcal{R}$), and $p^{\mathcal{R}}_{\text{mid}}$ (the middle point of $p^{\mathcal{R}}_{\min}$ and $p^{\mathcal{R}}_{\max}$). Similarly to τ as used in Section 3.2, $\tau_{\mathcal{R}}(\mathcal{F}_1, \mathcal{F}_2)$ is the maximum value of the disparity jumps of two polynomials $\mathcal{F}_1$ and $\mathcal{F}_2$ at these three points (see Fig. 6). The polynomials of neighboring lines are grouped so that $\tau_{\mathcal{R}}(\mathcal{F}_1, \mathcal{F}_2)$ of every two polynomials in each group is smaller than 0.5. Then we select only one polynomial from each group and form a subset of polynomials $\hat{\mathbb{M}}_{\mathcal{R}} \subseteq \mathbb{M}_{\mathcal{R}}$. Instead of using all polynomials in $\mathbb{M}_{\mathcal{R}}$ for carrying out DP, we carry out DP with $\hat{\mathbb{M}}_{\mathcal{R}}$, i.e., finding $f : \mathbb{L}_{\mathcal{R}} \rightarrow \hat{\mathbb{M}}_{\mathcal{R}}$.

4. Disparity Computation via Fusion

The methods as described in Section 3 are applied to lines in all four directions. Consequently, for pixel p we have four disparity estimations, $d_p^{(\theta)}$ ($\theta = 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}$) or we have four disparity fields for the reference image. The final disparity map is generated by seeking a combination (or a fusion) of these four disparity fields. This problem is similar to many stereo matching algorithms which are based on energy minimization in the pixel domain. However, the main difference is that the disparity value of p is chosen only

from its four disparity candidates rather than all disparity values of the search range. The energy function for guiding the combination process is:

$$E_I(f_I) = E^I_{\text{data}}(f_I) + E^I_{\text{smooth}}(f_I) \tag{8}$$

where f_I is a mapping $f_I : I \mapsto \left\{ d_p^{(\theta)} \right\}$ that assigns a disparity from four candidates ($d_p^{(0)}$, $d_p^{\left(\frac{\pi}{4}\right)}$, $d_p^{\left(\frac{\pi}{2}\right)}$, $d_p^{\left(\frac{3\pi}{4}\right)}$) to each pixel. The energy terms are defined as follows:

- $E^I_{\text{data}}(f_I)$: The data term is the summation of cost volume over all visible pixels:

$$E^I_{\text{data}}(f_I) = \sum_{p \in I} D\left(d_p^{(\theta)}\right) O(p) \tag{9}$$

- $E^I_{\text{smooth}}(f_I)$: We reuse the penalty function in Eq. (7) to penalize disparity jumps between neighboring pixel pairs. To preserve disparity boundaries which usually coincide with color boundaries, the penalty is weighted by the color similarity between two pixels:

$$E^I_{\text{smooth}}(f_I) = \sum_{(p,q) \in \mathcal{N}_I} w_{p,q} \mathcal{P}\left(d_p^{(\theta)}, d_q^{(\theta)}\right) \tag{10}$$

  where $\mathcal{N}_I$ is the set of 4 connected pixel pairs; $w_{p,q} = \exp\left(-\frac{\Delta(p,q)}{3\sigma_c}\right)$ is the color weight based on $\Delta(p,q)$, the summation of color difference in RGB channels. Using $\mathcal{P}$ in Eq. (10), we follow the idea presented in [20] where a small penalty is imposed on the small disparity jump ($0 < \left| d_p^{(\theta)} - d_q^{(\theta)} \right| \leq 1$) and a large penalty is imposed on the large disparity jump ($\left| d_p^{(\theta)} - d_q^{(\theta)} \right| > 1$).

To minimize this function, we employ the fusion move method [21] which efficiently decomposes the multi-label problem into sequential binary label problems. In the fusion move method, four proposal solutions which are the disparity fields obtained by the line models in four directions are fused sequentially. In the first stage, a refined solution is obtained by fusing two proposals. Then the refined solution is used to generate a new refined solution by fusing with another proposal. The final result is obtained after fusing all the proposals. In each stage of fusion move, the energy minimization problem is reduced to a quadratic pseudo-boolean optimization (QPBO) problem

where the label of a node (or pixel) is chosen from two candidates. A number of alternatives are available for solving the QPBO problem [22]. In our algorithm, the QPBOP+I method [23] is employed. Our energy function, Eq. (8), contains a submodular term, and it is therefore not theoretically guaranteed that the QPBOP+I method provides a full labeling, but we find that it does label all pixels in almost all cases when minimizing our energy function. If there are some unlabeled pixels left after QPBOP, we will use BP to label them for obtaining a full labeling and then perform QPOBI as suggested in [23].

5. Experimental Results

5.1. Experiments on Middlebury Datasets

Table 1: Quantitative evaluation of our method on the Middlebury stereo datasets (the error threshold is "1" pixel).

Evaluation	Average Error	Tsukuba			Venus			Teddy			Cones		
		nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc
AdCensus [24]	3.97	1.07	1.48	5.73	0.09	0.25	1.15	4.10	6.22	10.9	2.42	7.25	6.95
LLR [25]	4.64	1.05	1.65	5.64	0.29	0.81	3.07	4.56	9.81	12.2	2.17	8.02	6.42
PMF [26]	4.06	1.74	2.04	8.07	0.33	0.49	4.16	2.52	5.87	8.30	2.13	6.80	6.32
InfP [10]	5.51	1.06	1.53	5.64	0.32	0.88	4.15	5.60	13.0	14.5	2.65	9.16	7.69
CVW-RM [13]	4.38	1.12	1.42	5.99	0.16	0.36	1.40	4.70	6.94	12.1	2.96	7.71	7.72
Cost-Filter [8]	5.55	1.51	1.85	7.61	0.20	0.39	2.42	6.16	11.8	16.0	2.71	8.24	7.66
Our results	5.29	1.41	1.82	7.50	0.35	0.64	3.60	4.50	8.97	13.1	3.37	8.71	9.58

Table 2: Quantitative evaluation of our method on the Middlebury stereo datasets (the error threshold is "0.5" pixel).

Evaluation	Average Error	Tsukuba			Venus			Teddy			Cones		
		nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc
AdCensus [24]	13.9	26.8	27.0	21.1	4.05	4.60	8.00	10.6	13.8	20.1	6.58	12.4	11.9
LLR [25]	11.2	7.55	8.63	14.1	6.25	7.13	13.0	9.02	15.8	20.8	6.46	13.0	13.0
PMF [26]	7.69	11.0	11.4	16.0	0.72	0.92	5.27	4.45	9.44	13.7	2.89	8.31	8.22
InfP [10]	17.5	25.7	26.2	21.2	8.64	9.34	15.0	15.0	22.1	29.2	7.68	15.1	15.1
CVW-RM [13]	16.5	20.7	21.0	20.0	8.19	8.64	11.1	15.0	17.9	26.2	12.6	17.9	18.6
Cost-Filter [8]	12.8	11.2	11.7	15.6	5.99	6.43	10.8	11.3	18.1	25.3	7.71	13.7	15.1
Our results	12.9	8.19	8.78	15.6	6.66	7.03	14.7	11.6	17.2	25.2	9.26	15.0	17.4

We evaluate our algorithm on the Middlebury benchmark [27]. The initial disparity maps are calculated from the WTA approach on a pixel based

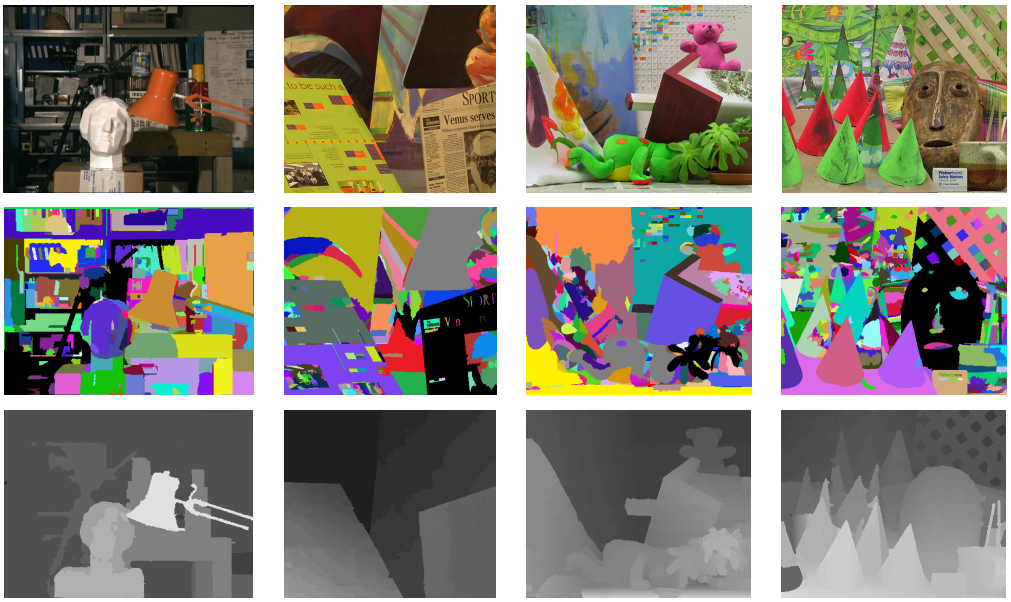


Figure 7: Our results on the four standard Middlebury image sets (from the left to the right are "Tsukuba", "Venus", "Teddy", and "Cones"). First row: the reference images. Second row: region segmentation result. Third row: our results. Note that our method produces correct disparity values for pixels in the under segmented regions.

cost volume obtained from [8]. The commonly used mean-shift segmentation method [28] is employed to generate region segments with default parameters: spatial bandwidth $\gamma_s = 7$, the color bandwidth $\gamma_c = 10$, and the minimum region size $\gamma_r = 50$. The third order polynomial is used: i.e., $\Theta = 3$. Because the linear model and the constant model are very widely observed in scenes, we therefore do not give penalty to these two cases, i.e., we set $\lambda_0 = \lambda_1 = 0$. The other parameters are found empirically to optimize the performance. The constant parameters for carrying out quantitative evaluation with all other methods on the Middlebury website are: $\mu = 1$, $\lambda_2 = \lambda_3 = 5$, $\sigma_c = 30$, $P_1 = 100$, and $P_2 = 200$ for penalty function $\mathcal{P}$ in Eq. (6); $P_1 = 0.5$, $P_2 = 5$ for penalty function $\mathcal{P}$ in Eq. (10). The disparity maps of four standard testing images are shown in Fig. 7. The intermediate results of the "Teddy" dataset are shown in Fig. 10 for a better understanding on the effect of each step. The quantitative evaluation is carried out based on the pixel accuracy (error threshold = 1 pixel) and sub-pixel accuracy (error threshold = 0.5

pixel), the results are listed in Table 1 and Table 2 respectively. We also carried out more experiments on the high definition extended Middlebury datasets. In this experiment, the WTA method on the cost volume is used to obtain initial disparity maps where the average error pixel percentage (error$>$ 1 pixel) is 49.1%. Using these initial disparity maps, our method produced much better results where the average error pixel percentage is 18.7%. The qualitative results are given in Fig. 8.

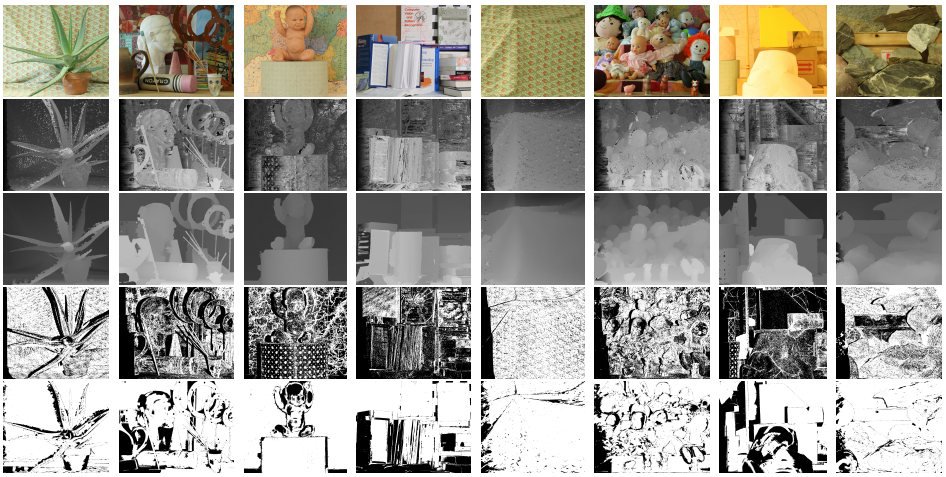
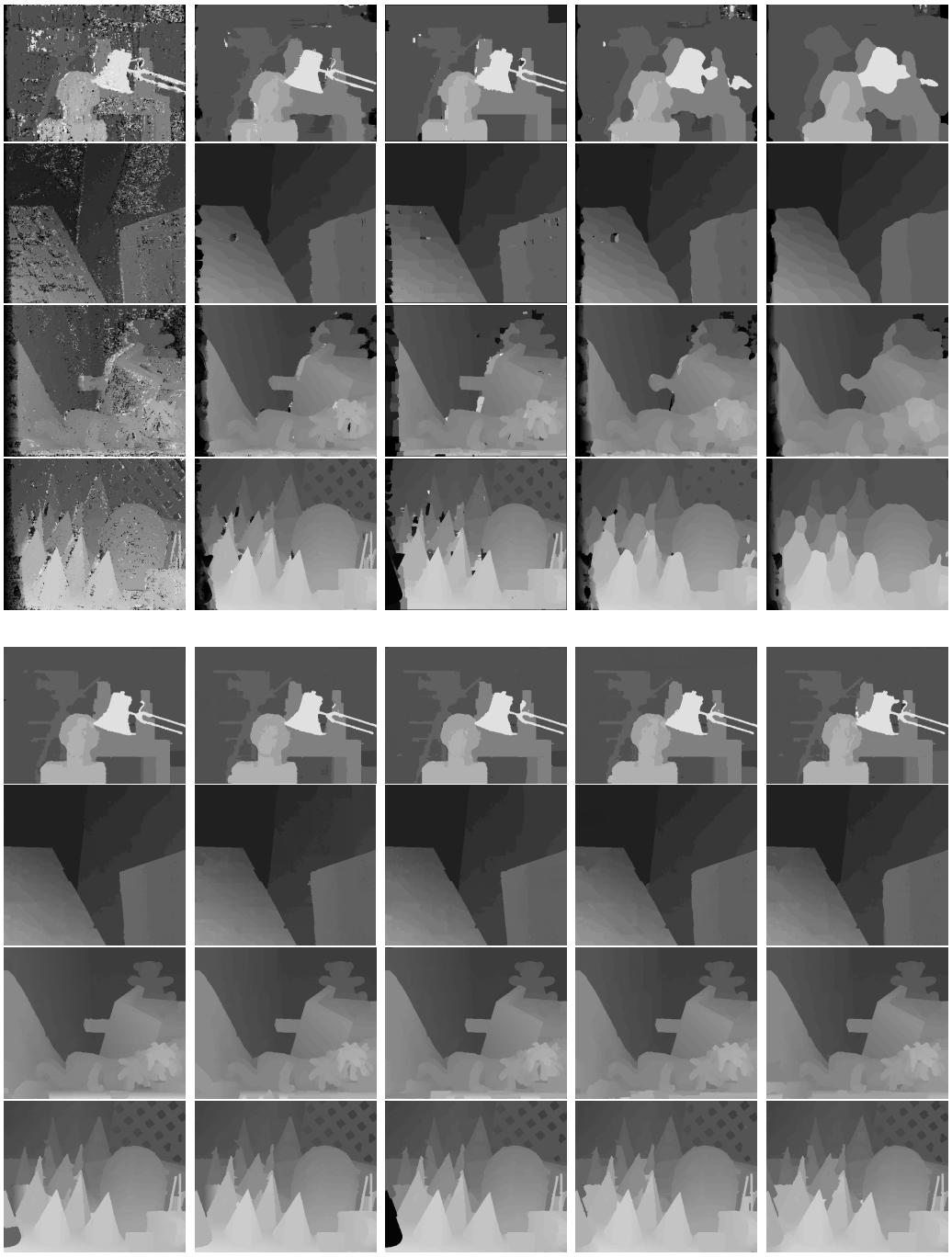


Figure 8: Results on the Middlebury stereo datasets. First row: reference images. Second row: initial disparity map. Third row: our results. Fourth row: error maps of initial disparity map. Last row: error maps of our results.

We evaluated the performance of our method against initial disparity maps. To achieve this, three alternatives are adopted to calculate the initial disparity map: cost volume filtering [8], box filtering [29], and hierarchical belief propagation approach [6] with constant parameters. The visual results results are shown in Fig. 9, and quantitative results are given in Table 3. It can be seen that although the initial disparity maps are quite different, the final results are similar, indicating that our method is robust to the initial disparity map computing methods. However, we observed slightly differences between the results. The results by using local aggregation methods to initialize disparities are not as accurate as using WTA on the cost volume in highly slanted surfaces, e.g., the bottom of the "Teddy" dataset. This is because aggregation methods bias the disparities within a local region towards fronto-parallel planes. Although some segmentations contain large under



(a) (b) (c) (d) (e)

Figure 9: Our results on four standard Middlebury image sets using the initial disparity maps from different methods. (a) WTA method on cost volume. (b) Hierarchical belief propagation [6]. (c) Cost volume filtering [8]. (d) Box filtering [29] with 9×9 windows. (d) Box filtering [29] with 15×15 windows. Top four rows are initial disparity maps, and bottom four rows are our results using these disparity maps.

Table 3: Quantitative evaluation of results using different initialization disparity map. (the error threshold is "1" pixel).

Evaluation	Average Error	Tsukuba			Venus			Teddy			Cones		
		nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc
WTA on cost volume	5.29	1.41	1.82	7.50	0.35	0.64	3.60	4.50	8.97	13.1	3.37	8.71	9.58
Hierarchical BP [6]	5.71	1.56	1.81	8.11	0.30	0.43	2.91	5.65	10.4	15.2	3.51	8.86	9.80
Cost filter [8]	5.73	1.48	1.91	7.80	0.61	0.95	4.93	5.30	10.0	14.2	3.32	8.68	9.61
Box filtering (9×9) [29]	6.62	1.51	2.09	7.80	1.00	1.61	6.29	6.59	10.8	17.6	3.72	9.91	10.6
Box filtering (15×15) [29]	6.98	1.48	2.10	7.81	1.06	1.61	7.58	6.97	11.7	17.6	4.16	10.3	11.5

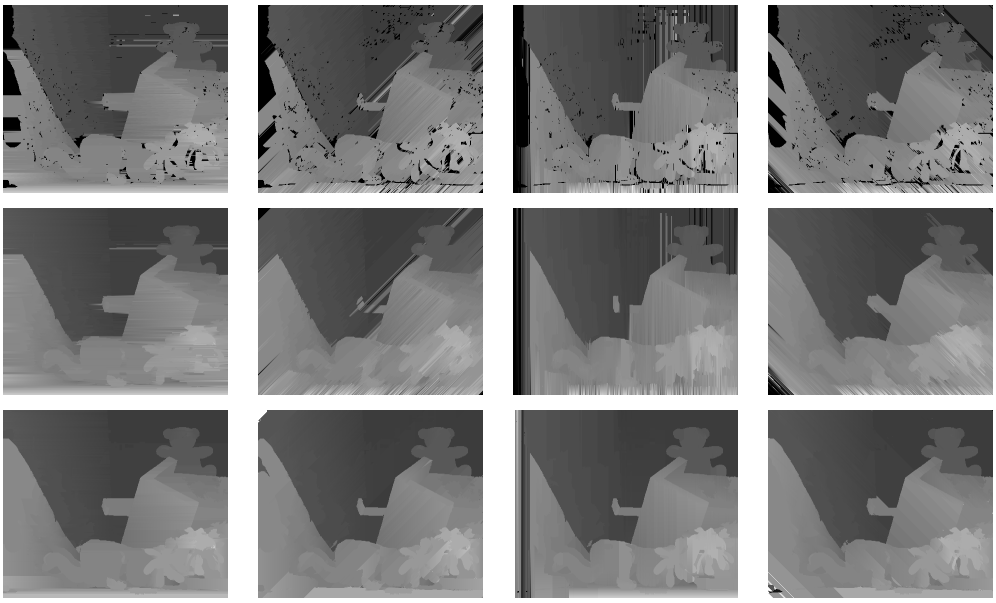


Figure 10: The intermediate results for the "Teddy" image. From the top to the bottom are results of the polynomial regression, results after propagation along the horizontal direction, and results after propagation within regions. From the left to the right are results for line segments where $\theta = 0$, $\theta = \frac{\pi}{4}$, $\theta = \frac{\pi}{2}$, and $\theta = \frac{3\pi}{4}$ respectively. The results after fusion are shown in Fig. 7.

segmented regions as shown in Fig. 7, our method still produces correct results for pixels in these regions. To better understand why our method is robust to these under segmentations, let us look at Fig. 11. From these four disparity fields, we notice that polynomial functions of lines vary smoothly in the smooth regions (e.g., surfaces of objects) but they vary dramatically near object boundaries. The variety of polynomial functions implicitly segment the under-segmented region into subregions which correspond to different objects in the disparity field. Pixels in regions where the disparity value changes a lot provide potential object boundaries in disparity fields. By considering all these disparity fields, it is expected that good object boundaries can be delineated. Our method provides correct disparity values at most parts of the highly slanted objects, such as the newspaper at the bottom of the "Teddy" dataset, with a small amount of errors. These errors can be reduced by tuning the parameters with the cost of over-smoothing other parts of the results. This is the trade-off between reducing artifacts in segments and differentiating different parts from an under segmented region.

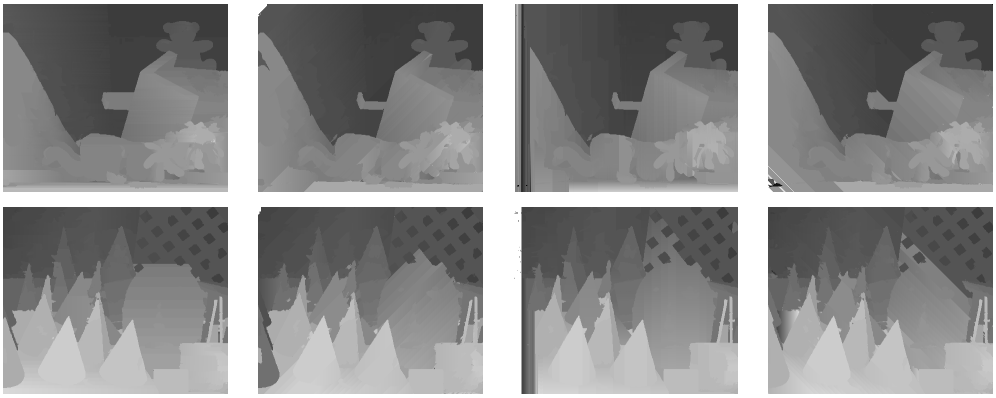


Figure 11: Four disparity fields of the "Teddy" (top) and "Cones" (bottom) datasets. From the left to the right are the disparity fields where $\theta = 0$, $\theta = \frac{\pi}{4}$, $\theta = \frac{\pi}{2}$, and $\theta = \frac{3\pi}{4}$. The fusion results of the four disparity fields in the top row and the bottom row are given in Fig. 7.

To test the performance of our method against different region segmentations, we tune parameters (γ_c, γ_s) in the mean-shift segmentation method and keep other parameters fixed. The results are fairly constant even when regions are extremely under segmented. For example when the parameters vary from $(4, 4)$ to $(18, 18)$, the percentage of error pixels vary within the

(a) (b) (c) (d)

Figure 12: Our results on extremely under segmented cases. (a) and (b) are region segmentation results. (c) and (d) are corresponding disparity results from our method.

range 7.61%-9.43% for the "Teddy" dataset and 8.43%-9.67% for the "Cones" dataset. Conventional methods which fit a region with a single model provide poor results in extremely under segmented cases. However, our method can still provide good results for those regions such as the segments colored in black in Fig. 12.

The limitation of our method is in the situation that one object is surrounded by another object and both objects are included by one region. For example, in the "Cones" dataset of Fig. 12, some parts of one chopstick are surrounded by the background which makes these parts of the chopstick lost in the results. Considering more candidate disparity fields obtained from more directions in the fusion step will help solve this problem. For example, in the case as shown in Fig. 12, considering the direction along the lost chopstick will help, but this requires more computation time.

5.2. Computational Complexity

The computational complexity of our method is the combination of each components. The cost volume calculation, initial disparity map calculation, and the cross checking based occlusion detection are all N_dWH processes where W, H are the width and height of the reference image I, and N_d is the disparity search range. QBPO and GC require recursively finding a minimum cut which is done with low-order polynomial complexities [30]. According to [3], by using the method in [31], this operation in practise is nearly linear to the problem size. Following this observation, the complexity of QPBO is linear to the size of image, and the complexity of GC based optimization long a line is linear to the product of the number of line segments and the number of models along the line. The complexity of DP based optimization within a

region is linear to the product of the number of lines and the square of the number of models within the region. The number of lines depends on the sharp of a region segment and the direction of the lines. Similarly, the number of models depends on the sharp of the objects. Therefore, the running time for two images may be different, though they have the same number of pixels and the same number of region segments. Generally, the more objects an image contains, the more segments there will be, and the more complex the surface of an object is, the more models there will be. We implemented the method in C++ on a 32bit platform with a 3.0 GHz processor without using any parallelized optimization such as GPU acceleration or SSSE instruction sets. We plot the running time of our method with respect to the number of region segments in Fig. 13 where we found the running time roughly increases with respect to the number of region segments, but in some cases fewer segments may lead to more running time. It is because when the number of segments is smaller, the size of segments becomes larger. The larger segment likely contains more models. Recall that the complexity of DP is related to the square of the number of models and hence the increase of the number of models will therefore sometimes lead to more running time.

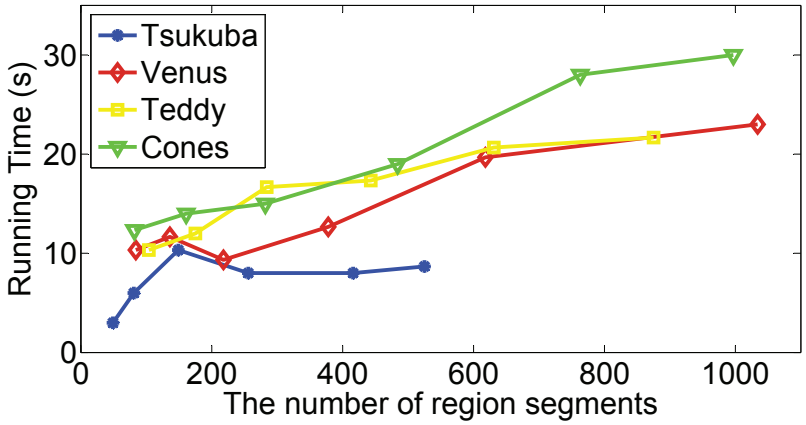


Figure 13: Running time versus the number of region segments in the image.

When our method produces the results reported in Fig. 7, the average number of region segments is 632, the average number of linear model per segment is 13, and the average running time is 18 seconds. In terms of running time of each component, the cost volume calculation and occlusion detection take 0.9s, polynomial regression takes 7.7s, DP optimization takes 4.6s, GC

optimization takes 4.1, and QPBO takes 0.7s. For comparison, we implement the cost volume filtering method [8] on the same computer, and the average runtime is 13s. Although our method consists of serval optimization steps which seem to slow down the method, the optimization steps are carried out mainly in the line segment space rather than the pixel space and the number of line segments is much smaller than the number of pixels. Hence the runtime of our method is similar to the cost volume filtering method [8]. As stated in the previous section, the polynomial regression and polynomial propagation are parallel processes. Data aggregation and occlusion detection are also parallel processes. In addition, lines in different directions are processed independently. Therefore, our method is very easy for parallel acceleration.

5.3. Order of Models

Our method employs polynomials to approximate the disparity values of pixels along line segments. The order of these polynomials is therefore a key determinant of the quality of the results. We investigated how the results vary with respect to the order of polynomials being used. To this end, we carried out experiments on the four standard datasets from the Middlebury benchmark with $\Theta = 0$ (zero order polynomials) and $\Theta = 1$ (first order polynomials) respectively and compared two results against the results with $\Theta = 3$ (third order polynomials). The qualitative results are demonstrated in Fig 14, and the quantitative comparison are given in Table 4. As noticed

Table 4: Quantitative evaluation of results with different Θ values (the error threshold is "1" pixel).

Θ value	Average Error	Tsukuba			Venus			Teddy			Cones		
		nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc
$\Theta = 0$	8.31	1.43	1.74	7.71	5.32	5.73	6.96	8.70	13.6	19.7	5.50	11.4	11.9
$\Theta = 1$	5.37	1.41	1.82	7.50	0.35	0.64	3.60	4.71	9.12	13.4	3.43	8.79	9.76
$\Theta = 3$	5.29	1.41	1.82	7.50	0.35	0.64	3.60	4.50	8.97	13.1	3.37	8.71	9.58

from Fig 14, dataset "Tsukuba" contains only frontal-plane surfaces which can be very well formulated by polynomials of any orders, and hence the results with zero order polynomials, first order polynomials and third order polynomials are quite similar for dataset "Tsukuba". In the case of dataset "Venus" which contains mainly planar surfaces where first order and third order polynomials perform better than zero order polynomials in delineating

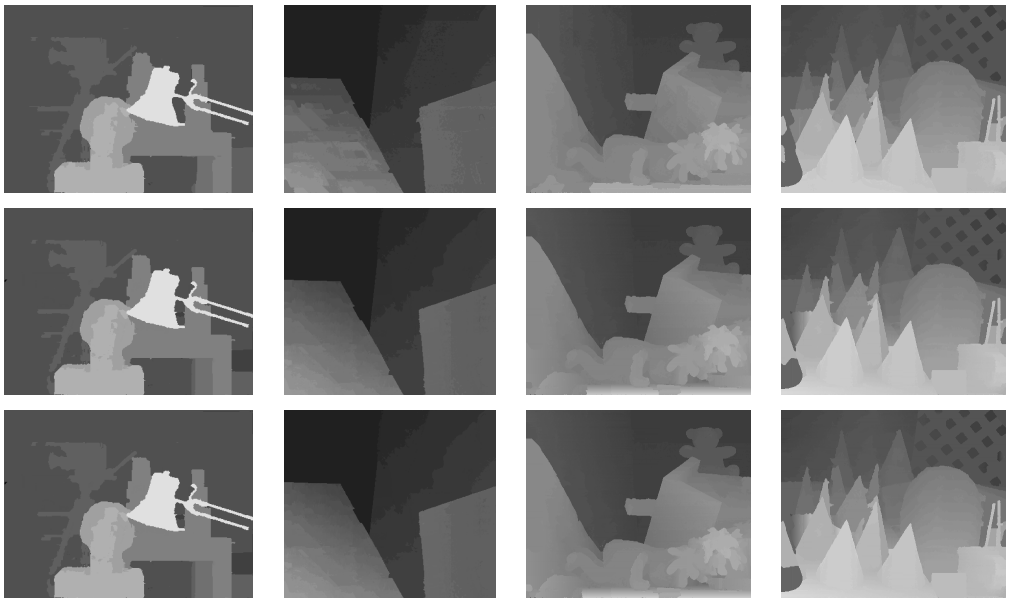
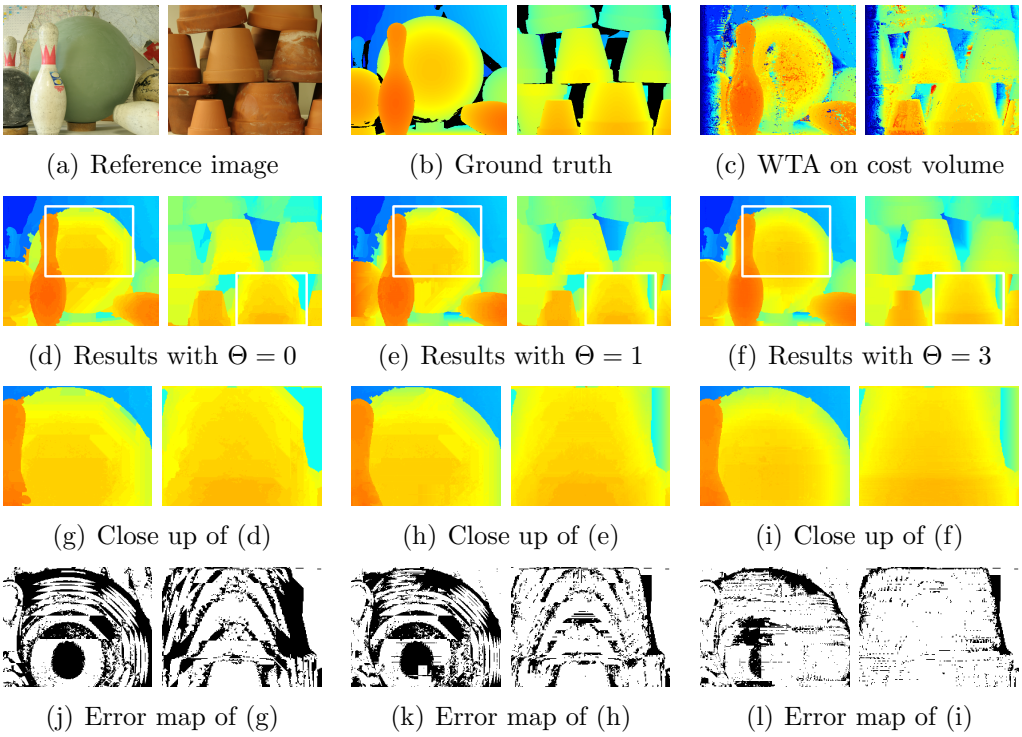


Figure 14: The results of our method vary with Θ values. First row: results with $\Theta = 0$. Second row: results with $\Theta = 1$. Third row: results with $\Theta = 3$.

the disparity variation, and therefore provide better results. We notice that results from the first and third polynomials are quite similar and they are nearly identical for datesets "Tsukuba" and "Venus". Since there are only planar surfaces in these two datasets, and also because when first order polynomials are "good" enough, the higher polynomials will not be introduced due to the penalty to higher order polynomials. Despite that the higher order polynomials provide better performance in the later two examples i.e., "Teddy" and "Cones", the improvement is not very significate, which is due to the fact that most surfaces in these two examples are comparably simple and can be well represented by the combination of line segments with the first order polynomials approximation. To better illustrate the advantage of using higher order polynomials, we show two further examples in Fig. 15. As shown in Fig 15, comparing with zero and first order polynomials, higher order polynomial approximation greatly reduces quantization artifacts when estimating disparities of curved surfaces. The error introduced by this quantization artifacts is apparent especially when sub-pixel accuracy is taken into account. The benefit of using high order polynomials over low order (zero or first order) polynomials is offered by the flexility of high order polynomials

in adapting themselves according to the shape of surfaces. Recall that lower order polynomials are preferred in the polynomial regression step if possible, Therefore setting Θ to a large value does not introduce the over-fitting problem; however, a larger Θ will increase the running time.



(a) Reference image (b) Ground truth (c) WTA on cost volume

(d) Results with $\Theta = 0$ (e) Results with $\Theta = 1$ (f) Results with $\Theta = 3$

(g) Close up of (d) (h) Close up of (e) (i) Close up of (f)

(j) Error map of (g) (k) Error map of (h) (l) Error map of (i)

Figure 15: Results for two examples which contain large regions of cured surfaces (for better comparison, disparity maps are color coded). Disparities of dark pixels are not given in the ground truth. The left dataset is "Blowings", and the right is "Flowerpots". The error threshold in the last row is 0.5 pixels. The error ratios for the "Blowings" with $\Theta = 0$, $\Theta = 1$, and $\Theta = 3$ are 41.3%, 38.7%, and 20.3% respectively. The error ratios for the "Flowerpots" with $\Theta = 0$, $\Theta = 1$, and $\Theta = 3$ are 33.4%, 19.3%, and 9.53% respectively.

6. Conclusions

We have presented a stereo matching algorithm based on a new scene representation model. The model gives a new and powerful extension to widely

used surface representations. Experiments showed that the new model and the stereo matching algorithm is very effective particularly in handling under segmented regions. Another attractive attribute of our model is that it consists of parallel processes which are very easy for GPU implementations. Comparing with existing state-of-the-art methods, our method requires more running time, which we will put effort on for reducing the complexity of the method. Our future work also includes improving the model for better preserving details in under segmented regions where one object is surrounded by another, and extending this new model to other computer vision studies such as 3D shape reconstruction from multiple views.

Acknowledgement

We thank Ryan Lagerstrom at CSIRO for his comments on this paper.

References

[1] J. Kim, V. Kolmogorov, R. Zabih, Visual correspondence using energy minimization and mutual information, in: International Conference on Computer Vision, 2003, pp. 1033–1040.

[2] H. Hirschmuller, Accurate and efficient stereo processing by semi-global matching and mutual information, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 2005, pp. 807–814.

[3] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239.

[4] C. Sun, Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques, International Journal of Computer Vision 47 (1-3) (2002) 99–117.

[5] J. Sun, N.-N. Zheng, H.-Y. Shum, Stereo matching using belief propagation, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (7) (2003) 787–800.

[6] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient belief propagation for early vision, International journal of computer vision 70 (1) (2006) 41–54.

[7] X. Tan, C. Sun, X. Sirault, R. Furbank, T. D. Pham, Cross image inference scheme for stereo matching, in: Asian Conference on Computer Vision, Springer, 2013, pp. 217–230.

[8] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, M. Gelautz, Fast cost-volume filtering for visual correspondence and beyond, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 3017–3024.

[9] X. Tan, C. Sun, D. Wang, Y. Guo, T. D. Pham, Soft cost aggregation with multi-resolution fusion, in: ECCV, Springer, 2014, pp. 17–32.

[10] C. Cigla, A. A. Alatan, Information permeability for stereo matching, Signal Processing: Image Communication 28 (9) (2013) 1072–1088.

[11] L. Hong, G. Chen, Segment-based stereo matching using graph cuts, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, 2004, pp. I–74.

[12] Z.-F. Wang, Z.-G. Zheng, A region based stereo matching algorithm using cooperative optimization, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[13] X. Tan, C. Sun, X. Sirault, R. Furbank, T. D. Pham, Stereo matching using cost volume watershed and region merging, Signal Processing: Image Communication 29 (10) (2014) 1232–1244.

[14] M. Bleyer, C. Rother, P. Kohli, Surface stereo with soft segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1570–1577.

[15] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, S. Sinha, Object stereo-joint stereo matching and object segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3081–3088.

[16] M. Gong, Y.-H. Yang, Fast unambiguous stereo matching using reliability-based dynamic programming, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (6) (2005) 998–1003.

[17] J. C. Kim, K. M. Lee, B. T. Choi, S. U. Lee, A dense stereo matching using two-pass dynamic programming with generalized ground control

points, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 2005, pp. 1075–1082.

[18] Y. Deng, X. Lin, A fast line segment based dense stereo algorithm using tree dynamic programming, in: European Conference on Computer Vision, Springer, 2006, pp. 201–212.

[19] V. Kolmogorov, C. Rother, Minimizing nonsubmodular functions with graph cuts–a review, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (7) (2007) 1274–1279.

[20] H. Hirschmuller, Stereo processing by semiglobal matching and mutual information, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (2) (2008) 328–341.

[21] V. Lempitsky, C. Rother, A. Blake, Logcut-efficient graph cut optimization for Markov random fields, in: International Conference on Computer Vision, 2007, pp. 1–8.

[22] O. Woodford, P. Torr, I. Reid, A. Fitzgibbon, Global stereo reconstruction under second-order smoothness priors, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (12) (2009) 2115–2128.

[23] C. Rother, V. Kolmogorov, V. Lempitsky, M. Szummer, Optimizing binary MRFs via extended roof duality, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[24] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, X. Zhang, On building an accurate stereo matching system on graphics hardware, in: Proc. of GPUCV, 2011, pp. 467–474.

[25] S. Zhu, L. Zhang, H. Jin, A locally linear regression model for boundary preserving regularization in stereo matching, in: European Conference on Computer Vision, 2012, pp. 101–115.

[26] J. Lu, H. Yang, D. Min, M. Do, Patchmatch filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1854–1861.

[27] http://www.vision.middlebury.edu/stereo/ (2013).

[28] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.

[29] M. McDonnell, Box-filtering techniques, Computer Graphics and Image Processing 17 (1) (1981) 65–70.

[30] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.

[31] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, in: Energy minimization methods in computer vision and pattern recognition, 2001, pp. 359–374.