Stereo reconstruction using high-order likelihoods

Ho Yub Junga, Haesol Parka, In Kyu Parkb, Kyoung Mu Leea,⁎, Sang Uk Leea

a Department of Electrical and Computer Engineering, Seoul National University, Seoul 151-742, Republic of Korea
b Department of Information and Communication Engineering, Inha University, Incheon 402-751, Republic of Korea

ABSTRACT

Under the popular Markov random field (MRF) model, low-level vision problems are usually formulated by prior and likelihood models. In recent years, the priors have been formulated from high-order cliques and have demonstrated their robustness in many problems. However, the likelihoods have remained zeroth-order clique potentials. This zeroth-order clique assumption causes inaccurate solution and gives rise to undesirable fattening effect especially when window-based matching costs are employed. In this paper, we investigate high-order likelihood modeling for the stereo matching problem which advocates the dissimilarity measure between the whole reference image and the warped non-reference image. If the dissimilarity measure is evaluated between filtered stereo images, the matching cost can be modeled as high-order clique potentials. When linear filters and nonparametric census filter are used, it is shown that the high-order clique potentials can be reduced to pairwise energy functions. Consequently, a global optimization is possible by employing efficient graph cuts algorithm. Experimental results show that the proposed high-order likelihood models produce significantly better results than the conventional zeroth-order models qualitatively as well as quantitatively.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Graphical models based on Markov random field (MRF) have become dominant in low-level computer vision problems such as image denoising, segmentation, and stereo vision. The ill-posed nature of these problems necessitates a strong presence of regularizing priors. MRF provides an effective representation of local dependency between the target variables. Efficient global optimization algorithms such as α-expansion, partial quadratic pseudo-Boolean optimization (QPBO) fusion, and belief propagation allow convincing estimates of the minimum energy states [3,7,20,26,30] of such models. The MRF framework has proven effective for incorporating smoothness priors into pixel labeling problems.

Recently, considerable progress has been made in modeling priors. The first-order priors are the most popular because these can be implemented directly in pairwise MRFs. In addition, second-order and high-order priors are introduced for the stereo and segmentation problems [14,35]. The success of the high-order models for priors has encouraged the development of high-order energy minimization algorithms, which also have become a persistent topic in computer vision [11,17,19]. The high-order models yield more accurate results than their low-order counterparts, though with an increase of computational time. To solve this problem on a conventional optimization framework, the larger cliques are first reduced to pairwise cliques before the optimization procedure.

However, high-order likelihoods have not been considered yet and the independent distribution has often been assumed so far. If we narrow our perspective to the stereo vision problem, the matching cost (likelihood) is modeled as a sum of single-variable functions, while the surface smoothness cost (prior) is modeled as the sum of two-variable or three-variable functions. Although various window-based and segment-based matching costs have been proposed, the matching costs have been nominally encased in the zeroth-order potentials until recently. Under the conventional zeroth-order clique assumption, the left and right images are filtered before matching. Note that due to this simple assumption, the conventional methods suffer from inaccuracy in matching and have noticeable undesired artifacts such as fattening effect around the disparity discontinuity. While it is desirable to filter the non-reference image after it is warped to the reference image plane, it causes high-order cliques because a pixel value in the filtered image becomes a function of many disparities inside the supporting window.
In this paper, we investigate the global matching framework and the subsequent high-order likelihood modeling for the stereo reconstruction problem. This paper differs from the previous works in proposing the elevation of the existing window-based matching costs to high order. We demonstrate how the global matching framework provides a natural interpretation for the high-order likelihoods. Furthermore, the global optimization is made possible through the pairwise clique reduction. Many of the previously introduced matching costs, such as those calculated by mean subtraction and census filters, are modeled as high-order likelihoods and optimized using graph cuts. The tests of various matching costs show that the elevation of likelihood to high order significantly eliminates the fattening effect that is inevitable when using the window-based zeroth-order likelihoods. Although the proposed method has high computational complexity due to the large neighborhood system and non-submodular edges, we show that the use of parallel implementation of the optimization algorithm on graphics processing unit (GPU) is an effective way to compensate the penalty. The early version of this paper was published in [12]. In this paper, it is expanded significantly to accommodate all high-order linear filters with L2 distances, in addition to the Census filter discussed in [12].

The rest of this paper is organized as follows. Section 2 introduces previous works related to the proposed approach. Section 3 describes the high-order likelihood from the global matching framework. In the same section, the pairwise clique reductions are provided for high-order matching costs obtained with linear and census filters. The fusion move allows asymmetrical occlusion handling by combining multiple proposal states. In Section 4, the energy minimization technique and its parallel implementation on GPU are described. Section 5 presents the experimental results, in which different high-order likelihoods are compared for various regularizing values. Section 6 gives concluding remarks.

2. Related works

Binocular stereo vision represents the visual matching problem in its simplest form. Despite the straightforwardness of the problem, the stereo matching is difficult even for images taken in well-controlled environments. Numerous matching costs and their implementation have been proposed. We explain how these can benefit from the elevation to high order.

Window-based methods involve a basic premise that nearby pixels have similar depths. More elaborate segment-based methods assume that pixels with similar color belong to the same 3D planar surface. Note that those two assumptions are also shared by prior models. In segment-based or window-based matching, nearby pixels ought to have the same depth surface, and the depth discontinuities are expected to align with color discontinuities. Thus, matching costs are aggregated for each segment by assuming the same surface. There are various dissimilarity measures, ranging from the classic sum of absolute differences to cross correlation and recent examples based on the intensity histogram [38]. However, the critical concerns of segment-based matching reside in designing the shape and size of the segment or smoothness priors, rather than the matching cost of the segment-based matching. With few exceptions, the shape and size of the segment are based on the color and distance similarity. By segmenting the stereo images, the smoothness prior is applied as hard surface constraints on the set of similar pixels. Recently, a soft segmentation has been introduced using adaptive support weights [8,36]. The soft segmentation is also based on the same assumptions as the matching costs based on hard segments, and greater support weight is given to the pixels that are closer in terms of color and position.

Alternatively, the stereo images can be preprocessed before calculating the matching costs. Instead of a color consistency assumption, the matching features of interest are transformed by filtering the stereo images before the matching cost calculation. The filtered images are subsequently matched in a pixel-wise or window-wise manner. Marr and Poggio first introduced phase and filter banks that resemble edges for stereo images [21]. In real time stereo systems, a Laplacian of Gaussian (LoG) filter is used to eliminate an intensity offset after noise smoothing [18]. Other linear filters include bilateral subtraction filters and mean subtraction filters [2,32]. Moreover, nonparametric rank and census filters output a binary map that evaluates the relative pixel intensities in a
window, which produces results more robust against outliers [37]. In a recent comparison test of stereo matching costs, the census filter method and bilateral subtraction method ranked at the top for the stereo pairs with radiometric differences [9].

Since statistical differences can be obtained, the window-based matching costs have a significant advantage over the pixel-wise matching costs. The disadvantage has been the fattening or blurring effect around disparity discontinuities. For the first group of approaches, the fattening effect is mitigated with the segmentation and adaptive weights. For the second group, the filtered stereo images are usually matched pixel-wise, which allows some alleviation of the fattening effect. However, since window-based calculations are required in obtaining the filter responses, the blurring effect cannot be eliminated. Accordingly, the current top-ranking stereo algorithms usually involve calculation of the matching costs through segmentation or adaptive weights. In this paper, however, we introduce an elevation of the filtered stereo image matches to high order, which can, in principle, completely eliminate the disadvantage of the window-based matching.

The matching costs have been limited in the zeroth-order potentials until recently. The likelihood is first viewed as a function of neighboring disparity values in [1], which allows different disparity values for the window matching. High-order likelihood is proposed for the optical flow problem [5], in which triangular regions are labeled with affine motion. However, these approaches simply approximate the high-order likelihoods by limiting the solution space. At most two disparity values in a window is assumed in [1]. For the high-order optical flow problem in [5], the matching costs are confined within the segmented regions where only a single affine motion is assumed. Note that a global matching framework has been introduced by Tao et al. [31], where the matching cost is evaluated over the whole image. The authors demonstrated how the segment-based matching cost is affected by the presence of occluded regions, which is hypothesized by the global matching framework. In [31], the global matching cost is found by iterative depth testing and greedy search over a region-based depth representation. However, although the update of a disparity value is affected by previous values of neighboring disparities during the optimization procedure, the matching term for each disparity is inherently independent of each other which consequently means the zeroth-order clique for each variable.

Fig. 2. The Middlebury dataset. In each set, the left and right input images, the ground truth disparity map, and the occlusion map are shown from left to right. (a) Tsukuba. (b) Venus. (c) Teddy. (d) Cones.
3. High-order likelihood

In this section, we propose high-order matching costs obtained with linear and census filters, where non-constant disparity values are considered in each window. After introducing the general MRF likelihood model, we interpret the high-order likelihood within the global matching framework. Then, the pairwise clique reductions are demonstrated so that the optimization by graph cuts becomes possible. Finally, the occlusion handling approaches for high-order likelihoods are examined.

3.1. Markov random field likelihood

Before reviewing the MRF stereo matching, some common notation is defined first. We are given a stereo image pair, \( I_L \) and \( I_R \), to estimate the disparity map of the left image, \( D_L \). The pixel position \( (i, j) \) is represented with \( x \in X \), where \( X \) is the set of pixel coordinates. The grayscale pixel intensity at position \( x \) is denoted by \( I(x) \). The warping function, \( \pi(x, d_x) \), projects a position \( x \) of the right image \( I_R \) to a pixel position \( I_L \) with the disparity value \( d_x \).

For the rectified stereo images, the warping function can be simplified to \( \pi(x, dx) = \left( \frac{i}{C_0 \cdot dx}, j \right) \).

The matching cost between \( I_L(x) \) and \( I_R(\pi(x, d_x)) \) is calculated for likelihood. Various distance functions are available, and the matching costs are represented by the single-variable clique potential. We can formulate the likelihood \( P(I_L, I_R|D_L) \) with a typical independent assumption and a squared difference matching cost:

\[
P(I_L, I_R|D_L) \propto \prod_{x \in X} \exp \left( -|I_L(x) - I_R(\pi(x, d_x))|^2 \right).
\]

As mentioned above, the stereo images are often preprocessed with a filter before the match. We denote a linear or nonlinear

---

**Fig. 3.** Qualitative results for the Middlebury dataset. The first set of three rows are the results of the proposed high-order (HO) matching cost, while the second set of three rows are those by the zeroth-order (ZO) matching cost for different filters. In each set, the upper, middle, and bottom rows show the estimated depth map, estimated occlusion map, and the error map, respectively. Error map is computed by comparing the estimated depth and the ground truth in unoccluded area with the disparity threshold 1. The ground truth depth and occlusion map are shown in upper-right corner. Different types of filters are used for each column: mean subtraction, LoG, bilateral, bilateral subtraction, and census filters from left to right. (a) Tsukuba. (b) Venus. (c) Teddy. (d) Cones. The figures are continued in the following pages.
Given the filtered stereo images $I^K_L$ and $I^K_R$ (i.e. $I^K_R = I^K_L \otimes K$), the likelihood is formulated as

$$P(I_L, I_R | D_L) \propto \prod_{x} \exp \left( -\frac{1}{2} \| I^K_L(x) - I^K_R(\pi(x, d)) \|^2 \right).$$

(2)

In the likelihood formulation, (1) and (2), the squared distance between filter responses is found. However, other dissimilarity measures such as the Hamming and absolute distances are available.

The model for the prior, however, disregards the independent assumption and is formulated as a product of multivariate functions. The first-order smoothness cost $\varphi_i(d_i, d_j)$ is a function of two neighboring disparity values: $P(D_L) \propto \prod_{(i,j) \in N} \exp \left( -\varphi_i(d_i, d_j) \right)$.

The posterior is proportional to the likelihood and the prior: $P(D_L | I_L, I_R) \propto P(I_L, I_R | D_L) P(D_L)$. The posterior is maximized by minimizing the equivalent energy function.

### 3.2. Global matching cost

The global matching framework offers a different perspective from that of the independently distributed likelihood. By using a depth map, the matching cost is viewed as the dissimilarity measure between the left image and the warped right image. The original left image and the left image synthesized by the right image should be similar if the correct disparity map is found. In
the warped image $I_{R\rightarrow L}(x)$, the right image pixel intensities are moved to the matching pixel positions in the left image:

$$I_{R\rightarrow L}(x) = I_R(\pi(x, d_x)) \quad \forall x \in X, \quad \forall d_x \in D_L.$$  \hspace{1cm} (3)

$I_{R\rightarrow L}$ is a function of the left disparity map and the pixel intensities of the right image. Under the global matching framework, the likelihood is defined as a single joint distribution over $D_L$, not as a product of the independent distribution at each pixel position. The likelihood can be formulated as an exponential of the matching cost between $I_{R\rightarrow L}$ and $I_L$:

$$P(I_L, I_{R\rightarrow L} | D_L) \propto \exp \left( -\frac{\| I_{R\rightarrow L} - I_L \|^2}{C_16/C_17} \right).$$  \hspace{1cm} (4)

If no filtering is applied to the stereo images, $\exp \left( -\frac{\| I_{R\rightarrow L} - I_L \|^2}{C_16/C_17} \right)$ is equivalent to the independently distributed likelihood in (1).

However, instead of directly finding the matching cost between $I_L$ and $I_{R\rightarrow L}$, both images can be filtered with a kernel $K$ before matching:

$$P(I_L, I_{R\rightarrow L} | D_L) \propto \exp \left( -\frac{\| I_{R\rightarrow L} - I_{R\rightarrow L}^K \|^2}{C_13/C_13/C_13} \right).$$  \hspace{1cm} (5)

In (5), $I_{R\rightarrow L}^K$ is calculated with only $I_L$ and $K$, but $I_{R\rightarrow L}^K$ (i.e. $I_{R\rightarrow L}^K = I_{R\rightarrow L} \circ K$) requires wrapping by $D_L$ before the convolution by $K$. The essential difference between the likelihood (2) and the global likelihood in (5) comes from the ordering of wrapping and filtering process. Note that the filtering is applied before the wrapping in the conventional approaches in (2), while the filtering is applied after the wrapping in the proposed global matching framework in (5). Fig. 1 shows the difference between conventional stereo approach and the proposed global matching approach. A uniform

---

(c) Teddy

**Fig. 3 (continued)**
Table 1

Percentage errors of different matching costs. A pixel is counted as error if its disparity difference to the ground truth is greater than one. The error is measured on the pixels in unoccluded areas. The bold characters represents the best results for each dataset. The proposed method with high-order likelihood based on Census filter shows the best performance except for the Tsukuba dataset. The result is obtained by using the best $s$ for each filter and dataset pair, which is specified in the parenthesis.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Tsukuba</th>
<th>Venus</th>
<th>Teddy</th>
<th>Cones</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High-order</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MeanSub</td>
<td>1.46 (10.)</td>
<td>0.52 (10.)</td>
<td>8.53 (3.5)</td>
<td>6.06 (10.)</td>
</tr>
<tr>
<td>LoG</td>
<td>2.68 (2.5)</td>
<td>1.57 (2.5)</td>
<td>22.2 (7.0)</td>
<td>12.3 (5.0)</td>
</tr>
<tr>
<td>Bilateral</td>
<td>2.37 (2.5)</td>
<td>0.53 (5.5)</td>
<td>22.7 (8.0)</td>
<td>18.2 (1.0)</td>
</tr>
<tr>
<td>BilateralSub</td>
<td>1.39 (10.)</td>
<td>0.36 (9.0)</td>
<td>8.70 (2.5)</td>
<td>6.06 (10.)</td>
</tr>
<tr>
<td>Census</td>
<td>1.31 (6.0)</td>
<td><strong>0.12 (8.5)</strong></td>
<td><strong>6.61 (7.0)</strong></td>
<td><strong>2.26 (4.0)</strong></td>
</tr>
<tr>
<td><strong>Zeroth-order</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MeanSub</td>
<td>4.91 (10.)</td>
<td>0.77 (8.0)</td>
<td>32.7(9.0)</td>
<td>42.4 (10.)</td>
</tr>
<tr>
<td>LoG</td>
<td>5.07 (8.0)</td>
<td>2.41 (5.0)</td>
<td>30.5(8.5)</td>
<td>54.9 (10.)</td>
</tr>
<tr>
<td>Bilateral</td>
<td>4.10 (7.5)</td>
<td>2.22 (8.5)</td>
<td>28.7(10.)</td>
<td>53.6 (10.)</td>
</tr>
<tr>
<td>BilateralSub</td>
<td>2.68 (7.5)</td>
<td>0.27 (5.0)</td>
<td>26.8(9.5)</td>
<td>34.3 (4.5)</td>
</tr>
<tr>
<td>Census</td>
<td>2.69 (10.)</td>
<td>1.06 (9.0)</td>
<td>8.43(2.5)</td>
<td>3.66 (2.5)</td>
</tr>
<tr>
<td><strong>Pixel-wise [29]</strong></td>
<td><strong>1.12</strong></td>
<td>2.23</td>
<td>7.25</td>
<td>4.46</td>
</tr>
</tbody>
</table>
depth around neighboring pixels is inherently assumed when a filtering function is applied before warping. Uniform disparity is mostly acceptable for continuous surfaces. On the other hand, this becomes the cause of a fattening effect at the pixels around disparity discontinuities.

3.3. L2 distance for linear filter responses

The optimization of (5) is a formidable task. However, certain classes of global matching cost can be reduced to pairwise energy functions and subsequently can be optimized using global
optimization algorithms. The L2 distance of linearly filtered stereo images belongs to such a class. The filter response at each pixel position is a function of disparities in window. A window is represented as an ordered tuple of pixel coordinates \( X = (x_1, x_2, \ldots, x_N), \) and \( x_i \) is the center pixel of the window.

Table 2
The high-order census matching cost is compared with a few top-performing methods in Middlebury website [13,22,34]. In this comparison, a constant parameter \( a_i = 6 \) is used.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Tsukuba</th>
<th>Venus</th>
<th>Teddy</th>
<th>Cones</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-order census</td>
<td>1.31</td>
<td>0.28</td>
<td>6.81</td>
<td>2.30</td>
</tr>
<tr>
<td>AdaptingBP</td>
<td>1.11</td>
<td>0.10</td>
<td>4.22</td>
<td>2.48</td>
</tr>
<tr>
<td>ABCensus</td>
<td>1.07</td>
<td>0.09</td>
<td>4.10</td>
<td>2.42</td>
</tr>
<tr>
<td>CoopRegion</td>
<td>0.87</td>
<td>0.11</td>
<td>5.16</td>
<td>2.79</td>
</tr>
</tbody>
</table>

Table 3
Average runtime of the proposed method with different filters for the Tsukuba dataset (in seconds).

<table>
<thead>
<tr>
<th>Filter</th>
<th>MeanSub</th>
<th>LoG</th>
<th>Bilateral</th>
<th>BilateralSub</th>
<th>Census</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>513.6</td>
<td>1365.9</td>
<td>1373.8</td>
<td>579.9</td>
<td>420.1</td>
</tr>
<tr>
<td>GPU</td>
<td>345.9</td>
<td>236.1</td>
<td>203.6</td>
<td>195.0</td>
<td>146.9</td>
</tr>
<tr>
<td>Speedup ratio</td>
<td>1.5</td>
<td>5.8</td>
<td>6.7</td>
<td>3.0</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Then, the squared distance between \( I^k \) and \( I^{k-\Delta} \) is computed as the sum of the high-order clique potentials as follows:

\[
\left\| I^k - I^{k-\Delta} \right\|^2 = \sum \phi_k(d_k),
\]

where

\[
\phi_k(d_k) = \left( \sum_{x \in X} k_i I_i(x) - \sum_{x \in X} k_i I_i(x) I_i(\pi(x, d_k)) \right)^2.
\]

Furthermore, the high-order clique (7) is simply expanded as follows.

\[
\phi_k(d_k) = \sum_{x \in X} k_i I_i(x) \sum_{x \in X} k_i I_i(x) - 2 \sum_{x \in X} k_i I_i(x) \sum_{x \in X} k_i I_i(\pi(x, d_k)) + \sum_{x \in X} k_i I_i(\pi(x, d_k)) I_i(\pi(y, d_k)).
\]

The first term in (8) is a calculation over the left image only, which is a constant in the energy function. The second and third terms are the sums of unary potentials. The last term is the sum of pairwise cliques that can be aggregated with smoothness costs. The high-order clique is finally reorganized as a sum of constant, unary, and pairwise cliques as follows:

\[
\phi_k(d_k) = \sum_{i=1}^{N} \phi_i(d_k, d_k).
\]

With the pairwise reduced \( \phi_k \), the global matching cost (6) is obtained by aggregating the potentials in each set of nodes and edges.

3.4. Census filter

Similar to the L2 distance for the linear filter responses, the high-order Hamming distance for the census filter responses can be reduced to pairwise functions. The census filter, also known as the local binary map, is a popular tool for general matching problems. A census filter outputs a binary map in which each element corresponds to a relative intensity evaluated against the center pixel [37]. The center pixel intensity \( I(x_c) \) is compared with the neighboring pixel intensity \( I(x_n) \). If \( I(x_c) > I(x_n) \), then the nth element is set to one; otherwise it is set to zero. In the conventional approach, the census filter is applied to both left and right images and computes the Hamming distance between corresponding binary maps. The census filter \( K \) is applied to the image window \( I(x) \), and the result is a binary bit string as

\[
I(x) \oplus K = (C(I(x_1), I(x_1)), C(I(x_2), I(x_2)), \ldots, C(I(x_n), I(x_n)))
\]

where

\[
C(I(x_1), I(x_1)) = \begin{cases} 1 & \text{if } I(x_1) < I(x_1) \\ 0 & \text{if } I(x_1) \geq I(x_1) \end{cases}
\]

Let \( b_i = \{b_{i_1}, b_{i_2}, \ldots, b_{i_w}\} \) and \( b_x = \{b_{x_1}, b_{x_2}, \ldots, b_{x_w}\} \) denote \( l_i(x) \oplus K \) and \( l_i(\pi(x, d_n)) \oplus K \), which are the census filter outputs from the left and warped right image windows, respectively. Then, the Hamming distance is computed by summing the absolute distances from the bits as follows: \( \phi_k(d) = \sum_{i=1}^{w} |b_i - b_x| \). The census filter bit \( b_i \) of the left image window is not a function of the disparity map and can be calculated in advance. Conversely, \( b_x \) is a function of the disparity values at pixel positions \( x_i \) and \( x_n \), which can be expressed as

\[
b_{x_i} = C(l_i(\pi(x, d_n)), l_i(\pi(x, d_n))).
\]

Both pixel positions are elements of the image window \( x \). By combining the Hamming distance and the bit response function (13), the high-order census-based matching cost is formulated as

\[
\phi_k(d_k) = \sum_{i=1}^{w} \phi_i(d_k, d_k).
\]
where
\[
\phi_x(d_x, d_y) = |b_{xy} - C(I_k(x, d_x), I_k(x, d_y))|.
\]  

Note that (14) computes the census-based matching cost from the left and warped right image windows. The high-order census-based matching cost becomes a sum of pairwise functions. The aggregated window-based high-order matching cost then becomes an MRF that is pairwise, highly connected, and non-submodular. For such MRFs, efficient global optimization is possible using QPBO-based \(x\)-expansion or fusion moves.

### 3.5. Occlusion handling and prior

For matching costs based on unary cliques, occlusions can be dealt with in the energy function through visibility constraints or one-to-one constraints [15,16]. By obtaining the disparity maps for both stereo images, a one-to-one penalty cost is assigned instead of the matching cost if a matching pair does not have the identical disparity. The occlusion is handled by pairwise cliques for MRFs with zeroth-order matching costs, which allows efficient optimization by graph cuts.

However, if the matching cost is elevated to high order, the one-to-one cost cannot replace the matching cost in the pairwise clique formulation. The matching cost is a function of disparity window, and the occlusion in the window should affect the matching cost of the window. Occlusion in different scan lines may affect window-based calculations. If we are to solve this problem in a function of the disparity map only, the clique order becomes much larger which is nontrivial to perform pairwise reduction. Therefore, the occlusion is handled with additional occlusion labeling of the pixel position. If pixel \(x\) is occluded, then \(o_x = 1\); otherwise, \(o_x = 0\).

With occlusion labeling \(o_x \in O\), we propose alternative methods for different kinds of filters to resolve how the high-order matching cost is affected by occlusion. For the case of linear filters, it is assumed that the matching pixel has the same intensity value as the reference pixel if \(x\) is occluded:
\[
I_k(\pi(x, d)) := I_k(x), \quad \text{if} \quad o_x = 1.
\]  

The problem with this approach is that color consistency is enforced. Therefore, if the radiometric difference between stereo images is large, the synthesized image is not coherent around occluded areas.

Alternatively, the high-order matching cost obtained by census filter can be disregarded for the occluded pixel:
\[
\phi_x(d_x, d_y) := 0, \quad \text{if} \quad o_x = 1 \quad \text{or} \quad 0_y = 1.
\]  

Note that this approach is suitable for stereo images with radiometric difference because it is not based on color consistency assumption. The high-order matching cost obtained by census filter is represented as the aggregation of first-order matching costs. Thus, the first-order cliques associated with the occluded pixels can be removed easily from the matching cost if the occluded areas are known. However, for high-order linear filters, the matching cost is the sum of both first-order and zeroth-order terms. Eliminating the potentials connected with the occluded pixels is equivalent to setting the coefficient \(k_x\) to zero. The convolution kernel \(K\) is severely distorted around occluded areas for linear filters.

For the prior, we employ the first-order smoothness cost \(\phi_p(d_x, d_y)\) introduced in [29]. The prior between neighboring nodes is weighted by distances in color and position. The weights are normalized for each kernel, and the whole image is multiplied by a regularizing constant, \(\lambda_c\). The normalized weights are calculated on the set of all possible edges, \(N\), as

\[
W_{xy} = \exp\left(-\frac{|x-y|}{\lambda_c}\right)\exp\left(-\frac{|r(x,y)|}{\sigma_c}\right),
\]

where \(I(x)\) and \(I(y)\) are color vectors. The associated bandwidths for position and color are chosen as \(\sigma_p = 5\) and \(\sigma_c = 10\), respectively, as in [29]. The truncated linear model is used for the first-order prior with the normalized weight as

\[
\phi_p(d_x, d_y) = \min(|d_x - d_y|, 2).
\]  

### 4. Optimization and implementation

The final energy function of the proposed method is defined on a graph structure with large neighborhood system depending on the size of filter. It is expected that it has non-submodular pairwise term, which makes it difficult to minimize the energy function using graph cuts. In the following subsections, the final energy function is formulated by incorporating the likelihood, prior, and occlusion handling terms. Its optimization techniques are discussed subsequently.

#### 4.1. Optimization

Using the occlusion handling (16), the high-order linear filter Eqs. (9) and (10) are modified with additional occlusion variables \(o_x\) and \(o_y\), at pixel positions \(x\) and \(y\) as

\[
\phi_x\{d_x, o_x\} = C_2 + \sum_{x \in X} \phi_x\{d_x, o_x\}
\]

\[
+ \sum_{(x,y) \in N_x} \varphi_{xy}\{d_x, o_x\}, \{d_y, o_y\}.
\]  

where

\[
\varphi_x\{d_x, o_x\} = \begin{cases} k_x^2 I_k^2(x) & \text{if } o_x = 1 \\ k_x^2 I_k^2(\pi(x, d)) - 2C_x k_x I_k(\pi(x, d)) & \text{otherwise} \end{cases}
\]  

and

\[
\varphi_{xy}\{d_x, o_x\}, \{d_y, o_y\} = \begin{cases} k_x k_y I_k(\pi(x, d_x)) I_k(\pi(y, d_y)) & \text{if } (o_x, o_y) = (0, 0) \\ k_x k_y I_k(\pi(x, d_x)) I_k(\pi(y, d_y)) & \text{if } (o_x, o_y) = (0, 1) \\ k_x k_y I_k(x) I_k(y) & \text{if } (o_x, o_y) = (1, 0) \\ k_x k_y I_k(x) I_k(y) & \text{if } (o_x, o_y) = (1, 1) \end{cases}
\]  

With the prior and occlusion costs, the final energy function for the left disparity map based on the high-order linear filter is

\[
E = \lambda_{oc} \sum_{x \in X} o_x + \sum_{x \in X} \phi_x\{d_x, o_x\} + \lambda_s \sum_{(x,y) \in N_x} \varphi_{xy}\{d_x, d_y\}
\]

where \(\lambda_{oc}\) is the occlusion cost and \(X\), \(N_x\) denotes the set of all pixels and possible edges in the left image, respectively.

Likewise, the energy function for the high-order census-based matching cost with occlusion handling is presented in (17). The pairwise census-based matching cost \(\phi_p\{d_x, d_y\}\) is aggregated for a set of unoccluded nodes as

\[
E = \lambda_{oc} \sum_{x \in X} o_x + \sum_{(x,y) \in N_x} (1 - o_x)(1 - o_y)\phi_p\{d_x, d_y\}
\]

\[
+ \lambda_s \sum_{(x,y) \in N_x} \varphi_{xy}\{d_x, d_y\}.
\]
The optimization of the final energy function is not trivial. The goal is to estimate the optimal disparity map and occlusion map simultaneously by optimizing the single complex potential function \( (24) \). To relax the problem, we take a heuristic alternating update scheme, in which disparity map and occlusion map are updated one by one at each iteration by simply assuming the other is fixed. The complex problem is therefore reduced to a common MRF optimization for each subproblem.

For a given occlusion map, the graph of disparity map has dense and non-submodular multi-label MRF structure which is difficult to optimize using the conventional graph cuts. However, it is known that QPBO can optimize non-submodular binary-label MRFs, although there is possibility of having unlabeled regions [26]. In this context, the proper solution of multi-label non-submodular multi-label MRF structure which is difficult to optimize using the conventional graph cuts. However, it is known that QPBO can optimize non-submodular binary-label MRFs, although there is possibility of having unlabeled regions [26]. In this context, the proper solution of multi-label non-submodular MRFs can be obtained by applying common heuristics. If the proposed approach, a traditional alpha-expansion strategy is employed, in which the disparity map with uniform \( x \) label is combined with the current disparity state using the fusion move method [20].

The overall procedure is described in Algorithm 1. For the fast convergence, we use the standard \( 3 \times 3 \) normalized cross correlation (NCC) matching cost to initialize the disparity map and to find the initial occlusion map. The maximum number of iterations, \( \text{max\_iteration} \), is set to 2, which is found empirically.

The dense connection still remains problematic during the optimization procedure. The proposed MRF has full connectivity over \( 7 \times 7 \) neighborhoods with additional edges for one-to-one constraint. The graph becomes even denser and more complex after it is transformed by the QPBO scheme. Furthermore, it contains multiple runs of graph cuts to perform the alpha-expansion (Line 5 in Algorithm 1). Consequently, the optimization procedure becomes highly complex in computation.

**Algorithm 2. Parallel Push-Relabel Algorithm**

**Require:** \( P \) is the set of all nodes, \( N(n) \) is the set of all neighboring nodes of \( n \)  
\( s(n) \) is the initial edge weight of source-to-\( n \) link for node \( n \)  
\( t(n) \) is the initial edge weight of \( n \)-to-terminal link for node \( n \)  
\( l(n,m) \) is the initial edge weight of \( n \)-to-\( m \) link for node \( n \) and \( m \in N(n) \)

1: \( h(n) = 1, e(n) = s(n) - t(n) \) for \( \forall n \)  
2: \( A = P \)  
3: \( S = \{ s \}, T = \{ t \} \)  
4: while \( |A| > \varepsilon \times |P| \) do  
5: \( \counter = 0 \)  
6: for \( \counter < \text{gap\_interval} \)  
7: Do parallel push operation for all active nodes (see Algorithm A.1)  
8: Do parallel local relabel operation for all nodes (see Algorithm A.2)  
9: \( \counter = \counter + 1 \)  
10: end for  
11: Do Gap relabel operation (see Algorithm A.3)  
12: end while

4.2. Parallelization on GPU

A possible remedy for the high computational complexity is to reduce the time required for each individual graph cuts optimization. It can be achieved by using push-relabel algorithm [6] instead of the widely-used Edmonds–Karp algorithm [3]. These two algorithms have similar complexity bound. However, the push-relabel algorithm has an advantage over Edmonds–Karp algorithm that it can be parallelized and implemented on recent GPU. Note that GPU is a massive parallel processor with 10–20 times higher performance than comparable CPU in terms of both floating point operation and memory bandwidth. GPGPU (general purpose computation on graphics processing unit) technique becomes popular in many areas including image processing and computer vision [25].

In the proposed approach, we employ the popular CUDA technology [23] which is widely used for GPGPU. Although there exist a few previous works of implementing graph cuts using CUDA [10,33], the graph structure in those papers is much simpler than ours and subsequently do not map directly to the proposed algorithm. Therefore, an improved parallelized maximum-flow algorithm is designed for the densely connected MRF with large neighborhood system. The performance of the implementation is further optimized based on the common knowledge of CUDA architecture such as shared memory and global memory coalescing [24]. In addition, we employ the heuristics to speed up the convergence of push-relabel algorithm, such as the gap relabel operation [4].

Following the same terminologies used in [6], the procedure of the proposed parallel push-relabel algorithm is summarized in Algorithm 2. The submodules of Algorithm 2 are listed in Appendix A. An important heuristic we use is the termination condition of the convergence. It is observed that, although the most of the flow is well pushed after short iteration of the algorithm, a small number of nodes remain active for a long time until the convergence. Therefore, it is desirable to terminate the algorithm if only a small portion of the nodes (1% in our implementation) remain active. This heuristic speeds up the process significantly, while sacrificing negligible degree of accuracy.

In Algorithm 2, the actual parallel processing is performed for push and local relabel operations. We generate a single thread per a single node in the target graph to maximize the throughput. The number of total parallel threads is more than 100,000 which would effectively compensate for the additional overhead of GPU thread generation and memory copy between CPU and GPU. As the related works addressed earlier [10,25,33], it is critical to use the shared memory in optimizing the performance. Since each node in the graph corresponds to a pixel in the image, we organize the threads into \( 16 \times 16 \) blocks corresponding to the same size of pixel blocks in the image. Each thread in the same block shares the information such as distance values, and we store those common values in the shared memory to enable faster access to the value and to let the threads communicate between each other.

**Algorithm 1. QPBO-x-expansion**

1: Initialize disparity map using local window-based matching  
2: Initialize occlusion map  
3: for \( i = 1 \) to \( \text{max\_iteration} \) do  
4: for \( x = 0 \) to \( \text{max\_disparity\_value} \) do  
5: \( x_s = D_i(x), \forall x \)  
6: Construct extended graph using QPBO scheme for binary problem  
\[ D_L = \arg \min (E) \text{ s.t. } D_i(x) \in \{ x_s, x \} \]  
7: Solve the maximum-flow problem and update \( D_L \)  
8: end for  
9: Update occlusion map \( O_L \)  
10: end for

4.2. Parallelization on GPU

A possible remedy for the high computational complexity is to reduce the time required for each individual graph cuts optimization. It can be achieved by using push-relabel algorithm [6] instead of the widely-used Edmonds–Karp algorithm [3]. These two algorithms have similar complexity bound. However, the push-relabel algorithm has an advantage over Edmonds–Karp algorithm that it can be parallelized and implemented on recent GPU. Note that GPU is a massive parallel processor with 10–20 times higher performance than comparable CPU in terms of both floating point operation and memory bandwidth. GPGPU (general purpose computation on graphics processing unit) technique becomes popular in many areas including image processing and computer vision [25].

In the proposed approach, we employ the popular CUDA technology [23] which is widely used for GPGPU. Although there exist a few previous works of implementing graph cuts using CUDA [10,33], the graph structure in those papers is much simpler than ours and subsequently do not map directly to the proposed algorithm. Therefore, an improved parallelized maximum-flow algorithm is designed for the densely connected MRF with large neighborhood system. The performance of the implantation is further optimized based on the common knowledge of CUDA architecture such as shared memory and global memory coalescing [24]. In addition, we employ the heuristics to speed up the convergence of push-relabel algorithm, such as the gap relabel operation [4].

Following the same terminologies used in [6], the procedure of the proposed parallel push-relabel algorithm is summarized in Algorithm 2. The submodules of Algorithm 2 are listed in Appendix A. An important heuristic we use is the termination condition of the convergence. It is observed that, although the most of the flow is well pushed after short iteration of the algorithm, a small number of nodes remain active for a long time until the convergence. Therefore, it is desirable to terminate the algorithm if only a small portion of the nodes (1% in our implementation) remain active. This heuristic speeds up the process significantly, while sacrificing negligible degree of accuracy.

In Algorithm 2, the actual parallel processing is performed for push and local relabel operations. We generate a single thread per a single node in the target graph to maximize the throughput. The number of total parallel threads is more than 100,000 which would effectively compensate for the additional overhead of GPU thread generation and memory copy between CPU and GPU. As the related works addressed earlier [10,25,33], it is critical to use the shared memory in optimizing the performance. Since each node in the graph corresponds to a pixel in the image, we organize the threads into \( 16 \times 16 \) blocks corresponding to the same size of pixel blocks in the image. Each thread in the same block shares the information such as distance values, and we store those common values in the shared memory to enable faster access to the value and to let the threads communicate between each other.
Another important observation is that many nodes actually become inactive after a few iterations. Consequently, it can reduce the runtime significantly if the inactive nodes are excluded from the further procedure. Therefore, the gap relabel operation is utilized to detect non-active nodes in advance, before they have zero excess. It is invoked after every gap interval runs of push-relabel operation. The parameter gap_interval is fixed to 10 in our implementation.

5. Experimental results

The experiments are performed on the Middlebury dataset [27,28] which is shown in Fig. 2. The algorithm is implemented on Intel Core i5-2500 K 3.30 GHz with 8 GB RAM and NVIDIA GeForce Titan with 2688 CUDA cores and 6 GB global memory. There are two important parameters in the proposed method, i.e., $\lambda_s$ and $\lambda_{occ}$. It is shown in the later part of this section that the performance of the proposed method depends on $\lambda_s$ while it is less sensitive to $\lambda_{occ}$. Through all the experiments, $\lambda_{occ}$ is fixed to 2, which has been found empirically.

Five different high-order likelihoods are compared with the corresponding zeroth-order likelihoods. The linear filters implemented are the mean subtraction [2], Laplacian of Gaussian (LoG) [18], bilateral [32], and bilateral subtraction [2] filters. The L2 distances between linear filter responses are used as the matching costs, while Hamming distances are used for high-order and zeroth-order matching costs obtained by census filter. Note that the zeroth-order likelihood utilizes the same occlusion handling and the energy function as those of the high-order likelihood cases except for the matching cost.

The goal of experimentation is to reveal performance differences between zeroth-order and high-order likelihoods. The kernel size is set large ($7 \times 7$), so that the blurring effect of the zeroth-order likelihood is better expressed while the elimination of the fattening effect is clearly demonstrated for the elevation to high order. Furthermore, a comparative evaluation of different high-order matching costs is made.

5.1. Zeroth-order and high-order matching costs

Fig. 3 shows the qualitative comparison of estimated disparity maps using zeroth-order and high-order likelihoods on the Middlebury dataset. Compared with the ground truths, the disparity maps obtained by using high-order likelihoods have less errors. Especially, a significant error reduction is observed around disparity discontinuities. The improvement is not obvious for the Venus dataset due to the simple scene structure, in which the contiguous objects have large color difference around the boundaries with smaller disparity discontinuity compared to others. It is observed that the fattening effect of zeroth-order likelihood gets worse for the area with relatively uniform color but varying disparity (e.g. the green doll and leaves in the Teddy dataset) and the thin/small objects (e.g. the necks of ramp in the Tsukuba dataset). The fattening effect is more serious when the likelihood is based on linear filters. The census filter encodes the relative intensity around the center pixel in a window as a binary value which makes the matching cost more robust to noise caused by constant disparity assumption. It is observed that there still exist performance difference between zeroth-order and high-order even in the regions with constant disparity. This is because the proposed method is based on MRF optimization structure in which the fattening effect around the disparity discontinuity may be propagated to the neighboring pixels.

Tables 1 and 2 show the quantitative comparison of different likelihoods. The performances of each likelihood is presented by percentage error for unoccluded areas. The result in this table is obtained with the best performing $\lambda_s$ value for each filter and data-set pair. The percentage error of pixel-wise matching cost with the same prior [29] is also included for comparison. For each dataset the lowest percentage error is in bold figures. Note that high-order census-based matching cost shows the best performance except for the Tsukuba dataset, where the pixel-wise matching cost outperforms the window-based ones. This is due to the fact that the pixel-wise matching cost can capture the disparity discontinuity more easily than window-based matching and much simpler to optimize if there is little ambiguity in matching. However, if the stereo images contain textureless area or repetitive textures, which is more common in real environment, the pixel-wise matching cost cannot overcome the ambiguity and it necessitates the use of the window-based matching.

The stereo performances as functions of the regularizing $\lambda_s$-value are summarized in Fig. 4. As shown in the plots, the proposed high-order matching costs generally outperform their zeroth-order counterparts, regardless of the base filter and the smoothing weight $\lambda_s$. The only exception is for the Venus dataset with the bilateral subtraction filter, where the minimum error is obtained by using zeroth-order cost. The Venus pair has the minimum disparity discontinuity among the datasets, and so the advantage of elevation to high order is exhibited the least.

Among the linear filters, the kernels that preserve high-frequency information, such as the bilateral subtraction and the mean subtraction, are observed to have lower percentage errors than their low-pass counterparts. The high-order census-based matching cost performs much better than the high-order L2-distance linear-filter MRFs. This is consistent with the zeroth-order matching cost evaluations, where the census-based MRF yields the best matching results [9].

5.2. Computational time

The runtime of the proposed algorithm is listed in Table 3 for the Tsukuba dataset. The QPBO implementation provided in [26] is used to measure the runtime of the CPU implementation. While the CPU implementation has large variation of runtime depending on the kind of filter, GPU implementation shows small difference. The GPU parallelization of the graph cuts algorithm achieves significant speedup, i.e. up to 6.7 times faster than on GPU. In addition, the runtime of high-order and zeroth-order matching cost is shown in Table 4. Both methods are implemented on GPU. The number of edges in the MRF structure is identical for the two matching costs. The difference in runtime mainly comes from the non-submodularity in high-order matching cost which results in more complex graph structure in QPBO scheme. The high-order linear filter involves more densely connected MRF than the high-order census-based MRF at the same window size, yielding longer runtime as shown in Table 4.

For a smaller window size of $3 \times 3$ or $2 \times 2$, high-order matching can be performed within significantly less time. In fact, the number of edges can be directly computed from the size of matching window and the size of input image and it mainly determines the complexity of optimization. However, we keep the window size as large as possible to demonstrate the difference of the fattening effect more clearly, which is set to $7 \times 7$ in our experiment.

6. Conclusion and discussion

A window-based matching cost can measure statistical differences between image windows. Without the smoothness prior, the window-based stereo matching methods generally outperform the pixel-by-pixel stereo matching methods. However, the inherent fattening effect limits the performance of window-based
matching costs, whereas the smoothness prior allows pixel-wise stereo matching to preserve discontinuity. In this paper, we formulated zeroth-order and high-order matching costs as the clique simplification of global matching costs. The global matching cost can be found by taking the difference between the observed image and the warped synthesized image. We reexamined the assumption behind using window-based matching costs as unary potentials. The previous window-based matching presumed that the disparities in a window are uniform. However, the projected image is a function of the disparity map, thus the matching cost is defined only with high-order cliques. For linear and census filters, the high-order clique potentials were demonstrably reduced to highly connected pairwise functions. The experiments using the Middlebury data set clearly illustrated the advantages of high-order likelihoods over zeroth-order likelihoods through elimination of the fattening effect. The proposed algorithm was parallelized and implemented on GPU using massive parallel computing, which shows significant speedup over running on CPU.

Acknowledgments

This research was supported in part by the Forensic Research Program of the National Forensic Service (NFS), Ministry of Security and Public Administration, Korea. This work was partly supported by the IT R&D program of MSIP/KEIT[10047078, 3D reconstruction technology development for scene of car accident reported by the IT R&D program of MSIP/KEIT[10047078, 3D reconstruction technology development for scene of car accident reported by the IT R&D program of MSIP/KEIT]

Appendix A. Details of GPU parallelization algorithm

Algorithm A.1. Parallel Push Operation

\[
\begin{align*}
1: & \text{for } \forall n \in A \text{ parallel do} \\
2: & \text{for } \forall m \in N(n) \text{ do} \\
3: & \text{if } e(n) > 0, h(n) = h(m) + 1 \text{ then} \\
4: & f = \min(e(n), l(n,m)) \\
5: & e(n) = e(n) - f \\
6: & l(n,m) = l(n,m) - f \\
7: & l(m,n) = l(m,n) + f \\
8: & e(m) = e(m) + f \\
9: & \text{end if} \\
10: & \text{end for} \\
11: & \text{end for}
\end{align*}
\]

Algorithm A.2. Parallel Relabel Operation

\[
\begin{align*}
1: & A = \phi \\
2: & \text{for } \forall n \in A \text{ parallel do} \\
3: & \text{if } e(n) > 0 \text{ then} \\
4: & \text{minHeight} = h(n) \\
5: & \text{for } \forall m \in N(n) \text{ do} \\
6: & \text{if } l(n,m) > 0 \text{ then} \\
7: & \text{minHeight} = \min(\text{minHeight}, h(m) + 1) \\
8: & \text{end if} \\
9: & \text{end for} \\
10: & \text{if } h(n) \neq \text{minHeight}, n \notin S \text{ then} \\
11: & h(n) = \text{minHeight} \\
12: & A = A \cup \{n\} \\
13: & \text{end if} \\
14: & \text{end if} \\
15: & \text{end for} \\
16: & S' = \{n|e(n) > 0, \forall n \in P\} \\
17: & S = S \cup S'
\end{align*}
\]

References


Algorithm A.3. Gap Relabel Operation

\[
\begin{align*}
1: & \text{for } i = 1 \text{ to } \max \{h(n)|n \in (P - S)\} \text{ do} \\
2: & G = \{n|h(n) = i, \forall n \in P\} \\
3: & \text{if } G = \phi \text{ then} \\
4: & S' = \{n|h(n) > i, \forall n \in P\} \\
5: & S = S \cup S' \\
6: & \text{end if} \\
7: & \text{end for}
\end{align*}
\]


