Continuous Stereo Matching using Local Expansion Moves

Tatsunori Taniai, Yasuyuki Matsushita, Yoichi Sato, and Takeshi Naemura

Abstract—We present an accurate and efficient stereo matching method using *local expansion moves*, a new move making scheme using graph cuts. The local expansion moves are presented as many α-expansions defined for small grid regions. The local expansion moves extend the traditional expansion moves by two ways: localization and spatial propagation. By localization, we use different candidate α-labels according to the locations of local α-expansions. By spatial propagation, we design our local α-expansions to propagate currently assigned labels for nearby regions. With this localization and spatial propagation, our method can efficiently infer Markov random field models with a huge or continuous label space using a randomized search scheme. Our local expansion move method has several advantages over previous approaches that are based on fusion moves or belief propagation; it produces *submodular moves* deriving a *subproblem optimality*; it helps find good, smooth, piecewise linear disparity maps; it is suitable for parallelization; it can use cost-volume filtering techniques for accelerating the matching cost computations. Our method is evaluated using the Middlebury stereo benchmark and shown to have the best performance in sub-pixel accuracy.

Index Terms—Stereo Vision, 3D Reconstruction, Graph Cuts, Markov Random Fields, Discrete-Continuous Optimization.

1 Introduction

Recent years have seen significant progress in accuracy of stereo vision. One of the breakthroughs is the use of slanted patch matching [3], [4], [11], [24], [25]. In this approach, the disparity d_p of each pixel p is over-parameterized by a local disparity plane

$$d_p = a_p u + b_p v + c_p \tag{1}$$

defined on the image domain (u, v), and rather than directly estimating d_p, the triplet (a_p, b_p, c_p) is estimated for each pixel p. The matching window is then transformed according to this disparity plane, which produces linearly-varying disparities within the window and thereby achieves accurate photo-consistency measures between matching pixels even with large matching windows. While stereo with standard 1D discrete disparity labels [6], [18], [19], [32] can be directly solved by discrete optimizers such as graph cuts (GC) [5], [20] and belief propagation (BP) [7], [36], such approaches cannot be directly used for continuous 3D labels due to the huge or infinite label space $(a, b, c) \in \mathbb{R}^3$.

Recent successful methods [3], [4], [11], [24] use PatchMatch inference [1], [2] to efficiently infer correct 3D planes using spatial propagation; each pixel's candidate label is, in raster-scan order, refined and then propagated to next pixels. Further in [3], this sequential algorithm is combined with BP yielding an efficient optimizer PMBP for pairwise Markov random fields (MRFs) [8]. In terms of MRF optimization, however, BP is considered a *sequential optimizer*, which improves each variable individually keeping others conditioned at the current state. In contrast, GC improves all variables simultaneously by accounting for interactions across variables, and this global property helps optimization avoid local minimas [29], [35]. In order to take this advantage and efficiently infer 3D planes by GC, it is important to use spatial propagation. Nevertheless, incorporating such spatial propagation into GC-based optimization is not straightforward, because inference using GC proceeds rather *all-nodes-simultaneously*, not *one-by-one-sequentially* like PatchMatch and BP.

In this paper, we introduce a new move making scheme, *local expansion moves*, that enables spatial propagation in GC optimization. The local expansion moves are presented as many α-expansions [6] defined for a small extend of regions at different locations. Each of this small or local α-expansion tries improving the current labels in its local region in an energy minimization manner using GC. Here, those current labels are allowed to move to a candidate label α, which is given uniquely to each local α-expansion in order to achieve efficient label searching. At the same time, this procedure is designed to propagate a current label in a local region for nearby pixels. For natural scenes that often exhibit locally planar structures, the joint use of local expansion moves and GC has a useful property. It allows multiple pixels in a local region to be assigned the same disparity plane *by a single min-cut* in order to find a smooth solution. Being able to simultaneously update multiple variables also helps to avoid trapped at a bad local minima.

The advantages of our method are as follows. 1) Our local expansion move method produces *submodular moves* that guarantee the optimal labeling at each min-cut (subproblem optimal), which in contrast is not guaranteed in general fusion moves [21]. 2) This optimality property and spatial propagation allow randomized search, rather than employ external methods to generate plausible initial proposals as done in fusion approaches [21], [25], [35], which may limit the possible solutions. 3) Our method achieves greater accuracy than BP [3] thanks to the good properties of GC and local expansion moves. 4) Unlike other PatchMatch based methods [3], [4], [11], our method can effectively incorporate

- *T. Taniai, T. Naemura, and Y. Sato are with the University of Tokyo, Japan.*
- *Y. Matsushita is with Osaka University, Japan.*

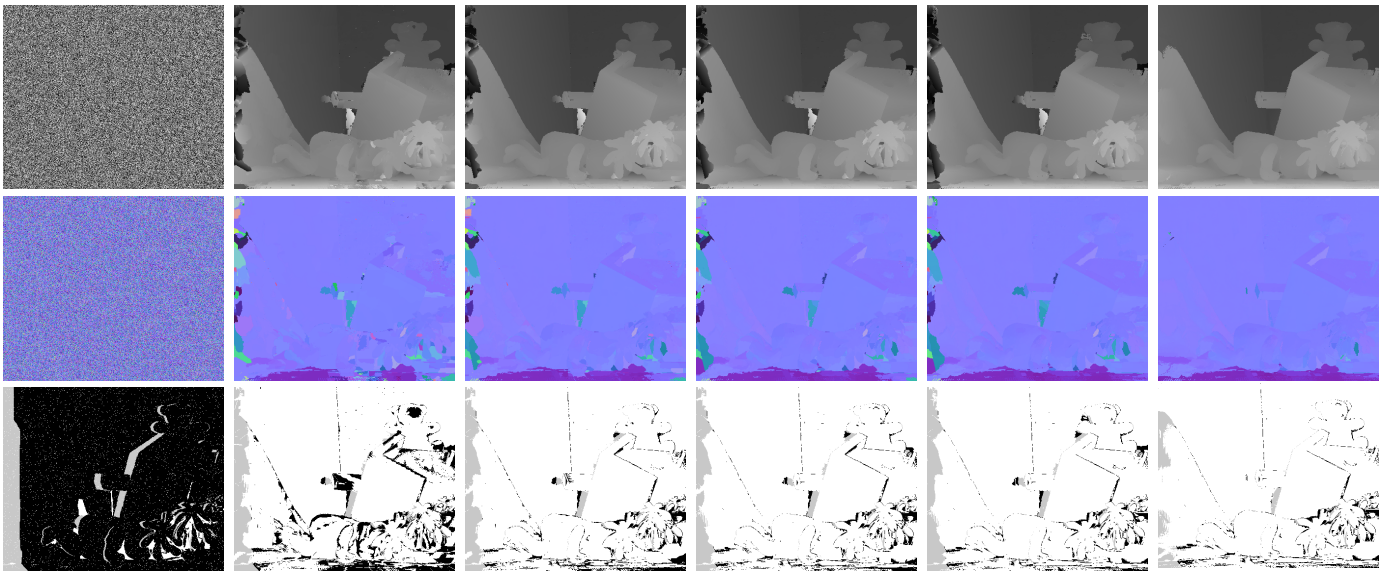


Fig. 1. Evolution of our stereo matching estimates. From top to bottom, we show disparity maps, normal maps of disparity planes, and error maps with 0.5 pixel threshold. In our framework, we start with random disparities that are represented by per-pixel 3D planes, *i.e.*, disparities (top) and normals (middle). We then iteratively apply our local expansion moves using GC (middles) to update and propagate local disparity planes. Finally, the resulting disparity map is further refined at a post-processing stage using left-right consistency check and weighted median filtering (rightmost).

the fast cost filtering technique of [24]. In this manner, we can efficiently reduce the computation complexity of unary terms from $O(|W|)$ to approximately $O(1)$, removing the dependency from the support window size $|W|$. 5) Unlike PMBP [3], our method is well suited for parallelization using both CPU and GPU.[1] With multiple CPU cores, each of our local α-expansions (*i.e.*, min-cut computations) can be individually performed in parallel. With a GPU implementation, the computation of unary terms can be efficiently performed in a parallel manner for further acceleration.

This paper is an extended version of our conference paper [30]. The extensions are summarized as follows. We add theoretical verifications on the preferability of our method for piecewise planar scenes in Sec. 3.1, and also on the subproblem optimality of our algorithm in Sec. 3.3. Furthermore, the efficiency of our algorithm is improved by two ways; In Sec. 3.3 we show the parallelizablity of our local expansion move algorithm; In Sec. 3.5 we show that the fast cost filtering technique of [24] can be used in our method. The effectiveness of both extensions is thoroughly evaluated in the experiments, and we show that even a CPU implementation of the proposed method achieves about 2.1x faster running times than our previous GPU implementation [30], with comparable or even greater accuracy.

2 Related Work

2.1 MRF stereo methods

MRF stereo methods can be categorized into three approaches: discrete stereo, segment-based stereo, and continuous stereo.

1. Although BP is usually parallelizable on GPU as well as CPU, PMBP differs from BP's standard settings in that it defines label space *uniquely and distinctively* for each pixel and *propagate* it; both make parallelization indeed non-trivial.

Discrete stereo [6], [18], [19], [32] formulates stereo matching as a discrete multi-labeling problem, where each pixel is individually assigned one of pre-defined discrete disparity values. For this problem, many powerful discrete optimizers, such as BP [7], [36], TRW [16], and GC [5], [20], can be directly used. Successful results are shown using GC with expansion moves [6], [29]. In expansion moves, the multi-labeling problem is reduced to a sequence of binary-labeling problems, each of which can be exactly solved by GC, if only pairwise potentials ψ meet the following submodularity of expansion moves [6], [17]:

$$\psi(\alpha,\alpha)+\psi(\beta,\gamma)\leq\psi(\beta,\alpha)+\psi(\alpha,\gamma). \tag{2}$$

Segment-based stereo [12], [15], [31], [33] assigns a 3D disparity plane for each of over-segmented image regions. The candidate planes are generated by fitting planes to a roughly estimated disparity map, and then the optimal assignment of the planes is estimated by, *e.g.*, GC with expansion moves [6], [12] or BP [7], [15]. Although this approach yields continuous-valued disparities, it strictly limits the reconstruction to a piecewise planar representation. Also, results are subject to the quality of the segmentation.

The last group, to which our method belongs, is continuous stereo [3], [4], [11], [24], [25], [35], where each pixel is assigned a distinct continuous disparity value. Some methods [25], [35] use fusion moves [21], an operation that combines two disparity maps to make a better one (binary fusion) by solving a non-submodular binary-labeling problem using QPBO-GC [17], [21]. In this approach, a number of continuous-valued disparity maps (or so-called *proposals* in the literature [21]) are first generated by other external methods (*e.g.*, segment-based stereo [35]), which are then combined as a sequence of binary fusions. Our method differs from this fusion-based approach in that we use spatial propagation and randomization search during inference, by which we only require a randomized initial

solution instead of those generated by external methods. More importantly, binary energies produced in our method are always submodular, *i.e.*, each binary-energy minimization is optimally solved via GC (subproblem optimal). A stereo method by Bleyer *et al.* [4] proposes accurate photo-consistency measures using 3D disparity planes that are inferred by PatchMatch [1], [2]. Heise *et al.* [11] incorporate Huber regularization into [4] using convex optimization. Besse *et al.* [3] point out a close relationship between PatchMatch and BP and present a unified method called PatchMatch BP (PMBP) for pairwise continuous MRFs. PMBP is probably the closest approach to ours in spirit, but we use GC instead of BP for the inference. Therefore, our method is able to take advantage of better convergence of GC [29] for achieving greater accuracy. In addition, our method allows efficient parallel computation of unary matching costs.

2.2 Cost-volume filtering

Patch-based stereo methods often use cost-volume filtering for fast implementations. Generally, computing a matching cost C for a patch requires $O(|W|)$ of computation, where $|W|$ is the size of the patch. However, given a cost-volume $\rho_d(p)$ that represents pixelwise raw matching costs $\|I(p) - I'(p-d)\|$ for a certain disparity label d, the patch-based matching costs can be efficiently computed by applying a filtering to the cost-volume as $C_d(p) = \sum_q \omega_{pq}\rho_d(q)$. Here, the filter kernel ω_{pq} represents the matching window at p. If we use a constant-time filtering, each matching cost $C_d(p)$ is efficiently computed in $O(1)$.

The box filtering can achieve $O(1)$ by using integral image but results in flattening object boundaries. For this boundary issue, Yoon *et al.* [37] propose an adaptive support-window technique that uses the joint bilateral filtering [26] for cost filtering. Although this adaptive window technique successfully deals with the boundary issue [13], it involves $O(|W|)$ of computation because of the complexity of the bilateral filtering. Recently, He *et al.* [9], [10] propose a constant-time edge-aware filtering named the *guided image filtering*. This filtering is employed in a cost-volume filtering method of [14], [27], achieving both edge-awareness and $O(1)$ of matching-cost computation.

In principle, stereo methods using PatchMatch inference [3], [4], [11] cannot take advantage of the cost filtering acceleration, since in those methods the candidate disparity labels are given uniquely to each pixel and we cannot make a constant-label cost-volume $\rho_d(p)$. To this issue, Lu *et al.* [24] extend [4] to use superpixels as a unit of cost calculations. In their method, called PatchMatch filter (PMF), fast cost-volume filtering of [14], [27] is applied in small subregions and that approximately achieves $O(1)$ of complexity. We will show in Sec. 3.5 that their subregion filtering technique can be effectively incorporated into our method, and we achieve greater accuracy than their local stereo method [24].

3 Proposed Method

This section describes our proposed method. Given two input images I_L and I_R, our purpose is to estimate the disparities of both images.

In Sec. 3.1, we first analyze the formulation of disparity planes used in the slanted patch matching approach [4], and show its relationship to homographies. We then define our energy function in Sec. 3.2, and describe the fundamental idea of our optimizing strategy and its properties in Sec. 3.3. The whole optimization procedure is presented in Sec. 3.4, and we further discuss a fast implementation in Sec. 3.5.

3.1 Geometric interpretation to slanted patch matching

Slanted patch matching [3], [4], [11], [24], [30] is an important technique for patch-based stereo matching, where a patch is warped and transformed in the other view using a disparity plane in Eq. (1). More specifically, each patch is warped by the following form of affine transformations

$$\mathbf{u}' = \begin{bmatrix} 1-a & -b & -c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}, \tag{3}$$

where $\mathbf{u}$ and $\mathbf{u}'$ are homogeneous pixel coordinates in the left and right view images, respectively. As illustrated in Fig. 2, this formulation produces linearly-varying disparities within a matching window, and can avoid a bias toward front-parallel surfaces that appears when using a constant disparity within a matching window [4]. This approach implicitly assumes that the true disparity maps are piecewise linear. In this section we show the validity of this piecewise linear disparity assumption for the scenes with *piecewise planar surfaces*. As we will discuss later, the result of this section also supports our method design, since our model and inference both prefer piecewise linear disparity maps.

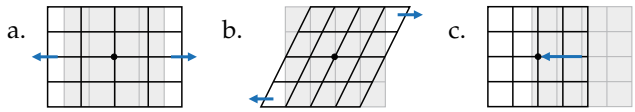


Fig. 2. Effects of (a, b, c) in slanted patch matching [4], [11]. Square windows (gray) in a reference view are warped and transformed in the other view by disparity planes $d = au + bv + c$.

Let us assume a rectified stereo setting, *i.e.*, two cameras are placed at $\mathbf{x} = \mathbf{0}$ (left) and $\mathbf{x} = (B, 0, 0)^T$ (right), respectively, in the 3D world coordinates $\mathbf{x} = (x, y, z)^T \in \mathbb{R}^3$. The two cameras have the same focal length $\{f_x, f_y\}$ and the identity rotation $R = I$.

We then assume there exists a planar surface in the scene

$$a'_p x + b'_p y + c'_p z = h'_p \tag{4}$$

that is parameterized by (a'_p, b'_p, c'_p, h'_p) and observed at the pixel p in the left view. Using the perspective model

$$x = uz/f_x, \quad y = vz/f_y, \tag{5}$$

the 3D points (x, y, z) on the geometry plane of Eq. (4) are projected at (u, v, z) on the left image domain (u, v) as

$$z(u, v) = h'_p / \left(\frac{a'_p}{f_x} u + \frac{b'_p}{f_y} v + c'_p \right). \tag{6}$$

Here, $z(u, v)$ is so-called a depth map representing depth values z of the geometry plane at the left image coordinates

(u, v). In the rectified stereo, it is well known that depth z and disparity d are related by

$$z = Bf_x/d. \tag{7}$$

By plugging this equation into Eq. (6), we finally obtain the geometry plane that is transformed into the form of disparity plane of Eq. (1):

$$d(u,v) = \frac{Bf_x}{h'_p}\left(\frac{a'_p}{f_x}u + \frac{b'_p}{f_y}v + c'_p\right). \tag{8}$$

This result shows that, in the rectified stereo setting, the warping induced by a planar surface or a *homography* can be described by a single disparity plane with three degrees of freedom. Therefore, the piecewise linearity of disparity maps in the image coordinates is equivalent to the piecewise planarity of object surfaces in the world coordinates.

3.2 Formulation

We follow the slanted patch matching approach of [4]. Here, each pixel p's disparity d_p is over-parameterized by a 3D plane $d_p = a_p u + b_p v + c_p$. Therefore, we seek a mapping $f_p = f(p) : \Omega \to \mathcal{L}$ that assigns a disparity plane $f_p = (a_p, b_p, c_p) \in \mathcal{L}$ for every pixel p in the left and right images. To estimate f, we use a pairwise MRF formulation by following conventional stereo matching methods [6], [18], [19], [25], [32]. In the MRF framework, we seek f such that minimizes

$$E(f) = \sum_{p\in\Omega} \phi_p(f_p) + \lambda \sum_{(p,q)\in\mathcal{N}} \psi_{pq}(f_p, f_q). \tag{9}$$

The first term, called the *data term* or *unary term*, measures the photo-consistency between matching pixels. The disparity plane f_p defines a warp from a pixel p in one image to its correspondence in the other image. The second term is called the *smoothness term* or *pairwise term*, which penalizes discontinuity of disparities between neighboring pixel pairs $(p, q) \in \mathcal{N}$. We define these terms as below.

Data term

To measure photo-consistencies, we use a data term that has been recently proposed by [4]. The data term of p in the left image is defined as

$$\phi_p(f_p) = \sum_{s\in W_p} \omega_{ps}\, \rho(s|f_p). \tag{10}$$

Here, W_p is a square window centered at p. The weight ω_{ps} implements the adaptive support window proposed in [37], and is defined as

$$\omega_{ps} = e^{-\|I_L(p) - I_L(s)\|_1/\gamma}, \tag{11}$$

where γ is a user-defined parameter, and $\|\cdot\|_1$ represents the ℓ_1-norm.[2] Here, we assume RGB color intensities $I \in [0, 255]^3$. Note that this weight ω_{ps} will be re-defined in Sec. 3.5 for a fast implementation. Given a disparity plane $f_p = (a_p, b_p, c_p)$, the function $\rho(s|f_p)$ in Eq. (10) measures the pixel dissimilarity between a support pixel $s = (s_u, s_v)$ in the window W_p and its matching point in the right image

$$s' = s - (a_p s_u + b_p s_v + c_p, 0) \tag{12}$$

as

$$\begin{aligned}\rho(s|f_p) = (1-\alpha)\ &\min(\|I_L(s) - I_R(s')\|_1, \tau_{col}) \\ &+ \alpha\ \min(|\nabla_x I_L(s) - \nabla_x I_R(s')|, \tau_{grad}).\end{aligned} \tag{13}$$

Here, $\nabla_x I$ represents the x-component of the gray-value gradient of image I, and α is a factor that balances the weights of color and gradient terms. The two terms are truncated by τ_{col} and τ_{grad} to increase the robustness for occluded regions. We use linear interpolation for $I_R(s')$, and a sobel filter kernel of $[-0.5 \;\; 0 \;\; 0.5]$ for ∇_x. When the data term is defined on the right image, we swap I_L and I_R in Eqs. (11) and (13), and add the disparity value in Eq. (12).

Smoothness term

For the smoothness term, we use a curvature-based, second-order smoothness regularization term [25] defined as

$$\psi_{pq}(f_p, f_q) = \max(\omega_{pq}, \epsilon)\ \min(\bar{\psi}_{pq}(f_p, f_q), \tau_{dis}), \tag{14}$$

where ϵ is a small constant value that gives a lower bound to the weight ω_{pq} to increase the robustness for image noises. The function $\bar{\psi}_{pq}(f_p, f_q)$ penalizes the discontinuity between f_p and f_q in terms of disparity as

$$\bar{\psi}_{pq}(f_p, f_q) = |d_p(f_p) - d_p(f_q)| + |d_q(f_q) - d_q(f_p)|, \tag{15}$$

where $d_p(f_q) = a_q p_u + b_q p_v + c_q$. The first term in Eq. (15) measures the difference between f_p and f_q by their disparity values at p, and the second term is defined similarly at q. We visualize $\bar{\psi}_{pq}(f_p, f_q)$ as red arrows in Fig. 3 (a). The $\bar{\psi}_{pq}(f_p, f_q)$ is truncated by τ_{dis} to allow sharp jumps in disparity at depth edges.

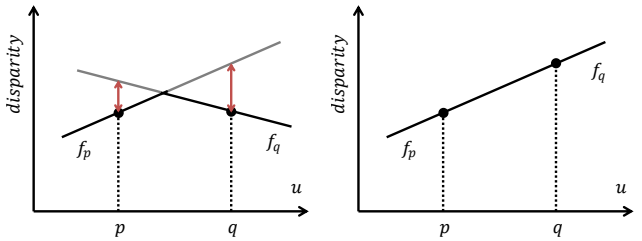


Fig. 3. Illustration of the smoothness term proposed in [25]. (a) The smoothness term penalizes the deviations of neighboring disparity planes shown as red arrows. (b) When neighboring pixels are assigned the same disparity plane, it gives no penalty; thus, it enforces second order smoothness for the disparity maps.

Notice that $\bar{\psi}_{pq}(f_p, f_q) = 2|c_p - c_q|$ when $a = b = 0$ is forced; therefore, the smoothness term $\psi_{pq}(f_p, f_q)$ naturally extends the traditional truncated linear model [6], although it has a front-parallel bias and should be avoided [25], [35]. Also, as shown in Fig. 3 (b), this term becomes zero when $f_p = f_q$. This enforces piecewise linear disparity maps and so piecewise planar object surfaces. Furthermore, this term satisfies the following property for taking advantage of GC.

Lemma 1. *The term $\psi_{pq}(f_p, f_q)$ in Eq. (14) satisfies the submodularity of expansion moves in Eq. (2).*

Proof. See [25] and also Appendix A. □

2. We have removed the spatial range weight of [37] that compares pixel positions. As mentioned in [4], [13], improvement due to this term is minor.

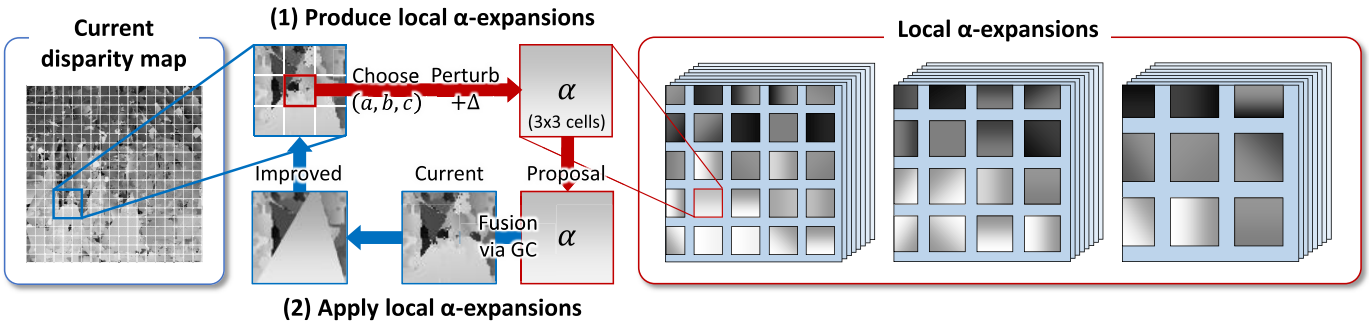


Fig. 4. Illustration of the proposed local expansion moves. The local expansion moves consist of many small α-expansions (or *local α-expansions*), which are defined using grid structures such shown in the left figure. These local α-expansions are defined at each grid-cell and applied for 3×3 cells of regions (or *expansion regions*). In the middle part, we illustrate how each of local α-expansions works. (1) The candidate label α (*i.e.*, $\alpha = (a, b, c)$ representing a disparity plane $d = au + bv + c$) is produced by randomly choosing and perturbing one of the currently assigned labels in its center cell. (2) The current labels in the expansion region are updated by α in an energy minimization manner using GC. Consequently, a current label in the center cell of each local α-expansion can be propagated for its surrounding cells. In the right part, local α-expansions are visualized as small patches on stacked layers with three different sizes of grid structures. As shown here, using local α-expansions we can localize the scopes of label searching by their locations. Each layer represents a group of mutually-disjoint local α-expansions, which are efficiently performed individually in a parallel manner.

3.3 Local expansion moves

In this section, we describe the fundamental idea of our method, local expansion moves, as the main contribution of this paper. We first briefly review the original expansion move algorithm [6], and then describe how we extend it for efficiently optimizing continuous MRFs.

The expansion move algorithm [6] is a discrete optimization method for pairwise MRFs of Eq. (9), which iteratively solves a sequence of the following binary labeling problems

$$f^{(t+1)} = \underset{f'}{\operatorname{argmin}}\, E(f' \mid f'_p \in \{f_p^{(t)}, \alpha\}) \tag{16}$$

for all possible candidate labels $^{\forall}\alpha \in \mathcal{L}$. Here, the binary variable f'_p for each pixel p is assigned either its current label $f_p^{(t)}$ or a candidate label α. If all the pairwise terms in $E(f)$ meet the condition of Eq. (2), then the binary energies $E(f')$ in Eq. (16) are submodular and this minimization can thus be exactly solved via GC [6] (subproblem optimal). Here, it is guaranteed that the energy does not increase: $E(f^{(t+1)}) \leq E(f^{(t)})$. However, the label space $\mathcal{L}$ in our setting is a three dimensional continuous space (a, b, c); therefore, such an exhaustive approach cannot be employed.

Our local expansion moves extend traditional expansion moves by two ways; *localization* and *spatial propagation*. By localization, we use different candidate labels α depending on the locations of pixels p in Eq. (16), rather than using the same α label for all pixels. This is reasonable because the distributions of disparities should be different from location to location, therefore the selection of candidate labels α should be accordingly different. By spatial propagation, we incorporate label propagation similar to the PatchMatch inference [1], [2], [4] into GC optimization, and propagate currently assigned labels to nearby pixels via GC. The assumption behind this propagation is that, if a good label is assigned to a pixel, this label is likely a good estimate for nearby pixels as well. The localization and spatial propagation together make it possible for us to use a powerful randomized search scheme, where we no longer need to produce initial solution proposals as usually done in the fusion based approach [21]. Below we provide the detailed descriptions of our algorithm.

Local α-expansions for spatial propagation

We first define a grid structure that divides the image domain Ω into grid regions $C_{ij} \subset \Omega$, which are indexed by 2D integer coordinates $(i, j) \in \mathbb{Z}^2$. We refer to each of these grid regions as a *cell*. We assume a regular square cell and its size can be as small as 1×1 pixel. At a high level, the size of cells balances between the level of localization and the range of spatial propagation. Smaller sizes of cells can achieve finer localization but result in shorter ranges of spatial propagation. Later in Sec 3.4 we introduce different sizes of multiple grid structures for well balancing these two factors, but for now let us focus on using one grid structure.

Given a grid structure, we define a *local α-expansion* at each cell (i, j), which we specifically denote as α_{ij}-*expansion*. We further define two types of regions for each α_{ij}-expansion: its *center region* C_{ij} and *expansion region*

$$R_{ij} = C_{ij} \cup \left\{ \bigcup_{(m,n)\in\mathcal{N}(i,j)} C_{mn} \right\}, \tag{17}$$

i.e., 3×3 cells consisting of the center region C_{ij} and its eight neighbor cells.

In the middle part of Fig. 4, we focus on an expansion region and illustrate how an α_{ij}-expansion works. We first randomly select a pixel r from the center region C_{ij}, and take its currently assigned label as $(a, b, c) = f_r$. We then make a candidate label α_{ij} by perturbing this current label as $\alpha_{ij} = (a, b, c)+\Delta$. Finally, we update the current labels of pixels p in the expansion region R_{ij}, by choosing either their current labels f_p or the candidate label α_{ij}. Here, similarly to Eq. (16), we update the partial labeling by minimizing $E(f')$ with binary variables: $f'_p \in \{f_p, \alpha_{ij}\}$ for $p \in R_{ij}$, and $f'_p = f_p$ for $p \notin R_{ij}$. Consequently, we obtain an improved solution as its minimizer with a lower or equal energy.

Notice that making the expansion region R_{ij} larger than the label selection region C_{ij} is the key idea for achieving spatial propagation. We can see this since, in an α_{ij}-expansion without perturbation ($\Delta = \mathbf{0}$), a current label f_p

Algorithm 1 ITERATIVE α_{ij}-EXPANSION

INPUTS: current f, target cell (i, j), perturbation size $|\Delta'|$.
$\diamondsuit$ **Propagation:**
repeat
  $\alpha_{ij} \leftarrow f_r$ with randomly chosen $r \in C_{ij}$.
  $f \leftarrow \mathrm{argmin}\, E(f' \mid f'_p \in \{f_p, \alpha_{ij}\}, p \in R_{ij})$
until K_{prop} times
$\diamondsuit$ **Randomization:**
repeat
  $\alpha_{ij} \leftarrow f_r$ with randomly chosen $r \in C_{ij}$.
  $\alpha_{ij} \leftarrow \alpha_{ij} + \Delta'$
  $f \leftarrow \mathrm{argmin}\, E(f' \mid f'_p \in \{f_p, \alpha_{ij}\}, p \in R_{ij})$
  $|\Delta'| \leftarrow |\Delta'|/2$
until K_{rand} times (or $|\Delta'|$ is sufficiently small)

in the center region C_{ij} can be propagated for its nearby pixels in R_{ij} as the candidate label α_{ij}.

We use this α_{ij}-expansion iteratively as shown in Algorithm 1. Similarity to the PatchMatch algorithm [1], this iterative algorithm consists of two steps: In the propagation step, we apply α_{ij}-expansions without perturbation to propagate labels from C_{ij} for R_{ij}; In the randomization step, we apply α_{ij}-expansions with an exponentially decreasing perturbation-size to refine the labels. We perform this iterative α_{ij}-expansion at every cell (i, j).

This local α-expansion method has the following useful properties. **Piecewise linearity**: It helps to obtain smooth solutions. In each α_{ij}-expansion, multiple pixels in the expansion region R_{ij} are allowed to move-at-once to the same candidate label α_{ij} at one binary-energy minimization, which contrasts to BP that updates only one pixel at once. Since a label represents a disparity plane here, our method helps to obtain piecewise linear disparity maps and thus piecewise planar surfaces as we have shown in Sec 3.1. **Cost filtering acceleration**: We can accelerate the computation of matching costs $\phi_p(f_p)$ in Eq. (10) by using cost-volume filtering techniques. We discuss this more in Sec 3.5. **Optimality and parallelizability**: With our energy formulation, it is guaranteed that each binary-energy minimization in Algorithm 1 can be optimally solved via GC. In addition, we can efficiently perform many α_{ij}-expansions in a parallel manner. We discuss these matters in the following sections.

Mutually-disjoint local α-expansions

While the previous section shows how each local α-expansion behaves, here we discuss the scheduling of local α-expansions. We need a proper scheduling, because local α-expansions cannot be simultaneously performed due to the overlapping expansion regions.

To efficiently perform local α-expansions, we divide them into groups such that the local α-expansions in each group are mutually disjoint. Specifically, we assign each α_{ij}-expansion a group index k given by

$$k = 4\,(j \bmod 4) + (i \bmod 4), \tag{18}$$

and perform the iterative α_{ij}-expansions in one group and another. As illustrated in Fig. 5, this grouping rule picks α_{ij}-expansions at every four vertical and horizontal cells into the same group, and it amounts to 16 groups of mutually-disjoint local α-expansions. We visualize a group of disjoint local α-expansions as orange regions in Fig. 6, and also as a single layer of stacks in the right part of Fig. 4 with three different grid structures.

Notice that in each group, we leave *gaps* of one cell width between neighboring local α-expansions. These gaps are to guarantee submodularity and independency for local α-expansions. By the submodularity, we can show that our local α-expansions always produce submodular energies and can thus be optimally solved via GC. Because of this submodularity, we can use a standard GC algorithm [5] instead of an expensive QPBO-GC algorithm [17], which is usually required in the fusion based approach. By the independency, we can show that the local α-expansions in the same group do not interfere with each other. Hence, we can perform them simultaneously in a parallel manner. The parallelization of GC algorithms are of interests in computer vision [22], [28]. Our scheme is simple and can use existing GC implementations. The proof of this submodularity and independency is presented in the next section.

Submodularity and independency

To formally address the submodularity and independency of local α-expansions, we discuss it using the form of fusion moves [21]. Let us assume a current solution f and a group of mutually-disjoint α_{ij}-expansions to be applied. Simultaneously applying these α_{ij}-expansions is equivalent to the following fusion-energy minimization:

$$f^* = \mathrm{argmin}\, E(f' \mid f'_p \in \{f_p, g_p\}), \tag{19}$$

where the proposal solution g is set to $g_p = \alpha_{ij}$ if p belongs to any of expansion regions R_{ij} in this group; otherwise, p is in gaps so we assign $\phi_p(g_p)$ an infinite unary cost for forcing $f'_p = f_p$. This proposal disparity map g is visualized as a single layer of stacks in the right part of Fig. 4. We prove the following lemmas:

Lemma 2. *Submodularity: the binary energies in Eq. (19) are submodular,* i.e.*, all the pairwise interactions in Eq. (19) meet the following submodularity of fusion moves [17], [21]:*

$$\psi_{pq}(g_p, g_q) + \psi_{pq}(f_p, f_q) \leq \psi_{pq}(f_p, g_q) + \psi_{pq}(g_p, f_q). \tag{20}$$

Proof. All the pairwise terms $\psi_{pq}(f'_p, f'_q)$ in Eq. (19) can be summarized into three types: pairwise terms inside expansion regions $\psi_{ab}(f'_a, f'_b)$, inside gap regions $\psi_{cd}(f'_c, f'_d)$, and between expansion and gap regions $\psi_{ac}(f'_a, f'_c)$. Examples of the four pixels (a, b, c, d) are shown in Fig. 6, where the cell size is shown as a 1×1 pixel here for simplicity. Notice that there is no $\psi_{pq}(f'_p, f'_q)$ that directly connects p and q in different expansion regions, since we use the eight-neighbor pairwise terms and their neighbor ranges cannot be longer than the gap width.

For $\psi_{ab}(f'_a, f'_b)$: It holds that $g_a = g_b$ because g_p is constant within an expansion region. By substituting $g_p = g_q = \alpha$, $f_p = \beta$, and $f_q = \gamma$ into Eq. (20), we obtain a relaxed condition known as the submodularity of expansion moves shown in Eq. (2). This condition holds for our pairwise term ψ_{pq}, as proved in Appendix A.

For $\psi_{cd}(f'_c, f'_d)$ and $\psi_{ac}(f'_a, f'_c)$: Because of the infinite unary costs of g_c and g_d, the binary variables f'_c and f'_d are forced to take their current labels f_c and f_d, respectively. Thus, $\psi_{ac}(f'_a, f'_c)$ becomes an a's unary poten-

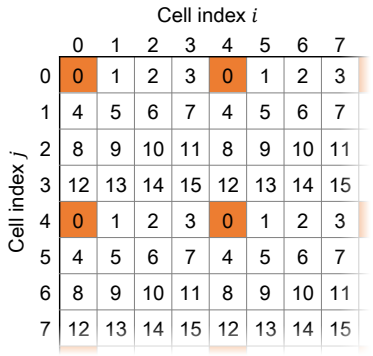


Fig. 5. Group index for α_{ij}-expansions. We perform α_{ij}-expansions in the same group in parallel.

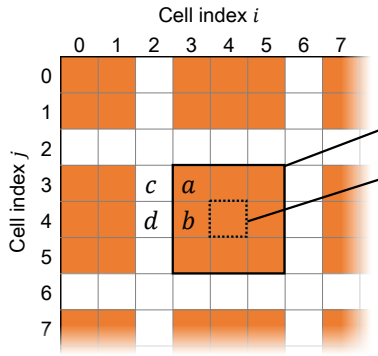


Fig. 6. Expansion regions of mutually disjoint α_{ij}-expansions (group index $k = 0$). We leave white gaps between neighbors.

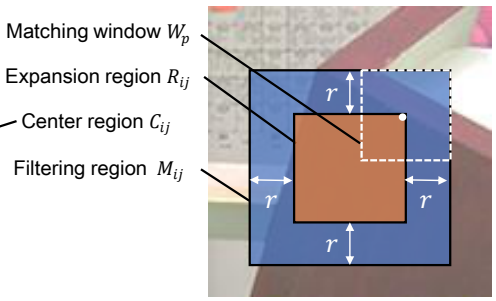


Fig. 7. Filtering region M_{ij}. The margin width r corresponds with the radius of the matching window W_p.

Algorithm 2 OVERVIEW OF OPTIMIZATION PROCEDURE

1: Define three levels of grid structures: $H = \{h_1, h_2, h_3\}$.
2: Initialize the current solution f randomly.
3: Initialize the perturbation size $|\Delta|$.
4: **repeat**
5: **for** each grid structure $h \in H$ **do**
6: **for** each disjoint group $k = 0, 1, ..., 15$ **do**
7: // apply disjoint α_{ij}-expansions (parallelizable)
8: **for** each cell (i, j) in the disjoint group k **do**
9: Apply iterative α_{ij} expansion $(f, (i, j), |\Delta|)$.
10: **end for**
11: **end for**
12: **end for**
13: $|\Delta| \leftarrow |\Delta|/2$
14: **until** convergence
15: Post processing

tial $\psi_{ac}(f'_a, f_c)$, and $\psi_{cd}(f'_c, f'_d)$ becomes a constant energy $\psi_{cd}(f_c, f_d)$, both are submodular.[3] □

Lemma 3. *Independency: the assignments to f'_p and f'_q do not influence each other, if p and q are in different expansion regions.*

Proof. The f'_p and f'_q have interactions if and only if there exists a chain of pairwise interactions $C = \{\psi(f'_{s_0}, f'_{s_1}), \psi(f'_{s_1}, f'_{s_2}), \cdots, \psi(f'_{s_{n-1}}, f'_{s_n})\}$ that connects $p = s_0$ and $q = s_n$.

As discussed in the above proof of submodularity, there is no direct interaction $\psi(f'_p, f'_q)$. Therefore, such a chain must contain two types of pairwise terms $\psi_{cd}(f'_a, f'_c)$ and $\psi_{ac}(f'_c, f'_d)$ described above, which become a unary potential and a constant, respectively. Therefore, there is no such chain of pairwise interactions connecting p and q. □

3.4 Optimization procedure

Using the local expansion moves shown in the previous section, we present the overall optimization procedure in this section and summarize it in Algorithm 2.

This algorithm begins with defining grid structures. Here, we use thee different sizes of grid structures for better balancing localization and spatial propagation.

3. Therefore, when implementing an α_{ij}-expansion using min-cut in a subregion, we create nodes for $^{\forall}p \in R_{ij}$, and add $\psi_{ac}(f'_a, f_c)$ as unary potentials of nodes a at the inner edge of R_{ij} as well as the unary and pairwise terms defined inside R_{ij}. See also [6] for the conversion of pairwise terms into edge capacities under expansion moves.

At line 2 of Algorithm 2, the solution f is randomly initialized. To evenly sample the allowed solution space, we take the initialization strategy described in [4]. Specifically, for each $f_p = (a_p, b_p, c_p)$ we select a random disparity z_0 in the allowed disparity range $[0, \mathrm{dispmax}]$. Then, a random unit vector $n = (n_x, n_y, n_z)$ and z_0 are converted to the plane representation by $a_p = -n_x/n_z$, $b_p = -n_y/n_z$, and $c_p = -(n_x p_u + n_y p_v + n_z z_0)/n_z$.

In the main loop through lines 4–14, we select one grid level h from the pre-defined grid structures (line 5), and apply the iterative α_{ij} expansions of Algorithm 1 for each cell of the selected structure h (lines 6–11). As discussed in Sec 3.3, we perform the iterative α_{ij} expansions by dividing into disjoint groups defined by Eq. (18). Because of this grouping, the α_{ij} expansions in the loop at lines 8–10 are mutually independent and can be efficiently performed in parallel.

The perturbation at line 10 of Algorithm 1 is implemented as described in [4]. Namely, each candidate label $\alpha_{ij} = (a, b, c)$ is converted to the form of a disparity d and normal vector n. We then add a random disparity $\Delta'_d \in [-r_d, r_d]$ and a random unit vector Δ'_n to them, respectively, as $d' = d_p + \Delta'_d$ and $n' = n + r_n \Delta'_n$. Finally, d' and $n'/|n'|$ are converted to the plane representation $\alpha_{ij} \leftarrow (a', b', c')$ to obtain a perturbed candidate label. The values r_d and r_n define an allowed change of disparity planes. We initialize them by setting $r_d \leftarrow \mathrm{dispmax}/2$ and $r_n \leftarrow 1$ at line 3 of Algorithm 2, and update them by $r_d \leftarrow r_d/2$ and $r_n \leftarrow r_n/2$ at line 13 of Algorithm 2 and line 12 of Algorithm 1.

Finally, after the whole process, we perform post-processing using left-right consistency check and weighted median filtering as described in [4] for further improving the results. This step is widely employed in recent methods [3], [4], [11], [24].

Note that there are mainly two differences between this algorithm and our previous version [30]. For one thing, we have removed a per-pixel label refinement step of [30]. This step is required for updating a special label space structure named *locally shared labels* (LSL) used in [30], but can be removed in our new algorithm by using local α-expansions instead. For the other, the previous algorithm proceeds rather in a batch cycle; *i.e.*, it produces all local α-expansions with all grid structures at once and then applies

them to the current solution f in one iteration. This way we can minimize the overhead of data transferring between GPU in unary cost computation, but results in slower convergence than our new algorithm that produces each local α-expansion always from the latest current solution f. Our new algorithm rather concedes the increased overhead of GPU because it is also intended for a fast CPU implementation as we describe below.

3.5 Fast implementation

Our method has two major computation parts: the calculations of matching costs $\phi_p(f_p)$ of Eq. (10), and application of GC in Algorithm 1. In Sec 3.3 and Sec. 3.4, we have shown that the latter part can be accelerated by performing disjoint local α expansions in parallel. In this section, we discuss the former part.

The calculations of the matching costs $\phi_p(f_p)$ are very expensive, since they require $O(|W|)$ of computation for each term, where $|W|$ is the size of the matching window. In our previous algorithm [30], we accelerate this part by using GPU. The use of GPU is reasonable for our method because $\phi_p(f_p)$ of all pixels can be individually computed in parallel during the inference, which contrasts to PMBP [3] that can only sequentially process each pixel. Still, the computation complexity is $O(|W|)$ and it becomes very inefficient if we only use CPU [30].

In the following part, we show that we can approximately achieve $O(1)$ of complexity for computing each $\phi_p(f_p)$. The key observation here is that, we only need to compute this matching cost $\phi_p(f_p)$ during the α_{ij}-expansions in Algorithm 1, where $\phi_p(f_p)$ is computed as $\phi_p(\alpha_{ij})$ for all $p \in R_{ij}$. With this constant-label property, we can effectively incorporate the fast subregion cost-filtering technique used in [24], by which $\phi_p(\alpha_{ij})$ for all $p \in R_{ij}$ are computed altogether.

To more specifically discuss it, we first separate the computation of $\phi_p(f_p)$ into two steps: the calculations of raw matching costs

$$\rho_{f_p}(s) = \rho(s|f_p) \tag{21}$$

for the support pixels $s \in W_p$, and the aggregation of the raw matching costs using an edge-aware filter kernel ω_{ps}

$$\phi_{f_p}(p) = \sum_{s \in W_p} \omega_{ps}\, \rho_{f_p}(s). \tag{22}$$

We also define a filtering region M_{ij} as the joint matching windows in R_{ij} as

$$M_{ij} = \bigcup_{p \in R_{ij}} W_p. \tag{23}$$

As shown in Fig. 7, this M_{ij} is typically a square region margining R_{ij} with r pixels of width around R_{ij}, where r is the radius of the matching windows.

In the raw matching part of Eq. (21), the calculations of the raw costs $\rho_{f_p}(s)$, $^\forall s \in W_p$ for all $p \in R_{ij}$ generally require $O(|W||R_{ij}|)$ of total computation. However, with the constant-label property (*i.e.*, $f_p = \alpha_{ij}$ for all $p \in R_{ij}$), they can be computed at once in $O(|M_{ij}|)$ by computing $\rho_{\alpha_{ij}}(s)$ for $^\forall s \in M_{ij}$. Here, the computation complexity per each unary term is $O(|M_{ij}|/|R_{ij}|)$. Therefore, if $|M_{ij}| \simeq |R_{ij}|$, we can approximately achieve $O(|M_{ij}|/|R_{ij}|) \simeq O(1)$ [24].

Similarly, the cost aggregation part of Eq. (22) can be done in approximately $O(1)$, if we apply a constant-time edge-aware filtering ω_{ps} to $\rho_{\alpha_{ij}}(s)$, $^\forall s \in M_{ij}$ [24]. Unfortunately, our weight function w_{ps} in Eq. (11) represents a variant of the joint bilateral filtering [26], which generally requires $O(|W|)$ of computation. However, there are several constant-time edge-aware filterings that work similarly to or even better than the bilateral filtering, such as the guided image filtering [10] and the cross-based local multipoint filterings [23]. Therefore, if we replace the adaptive window weight w_{ps} of Eq. (11) with those filter kernels, the overall computation complexity for each unary cost $\phi_p(f_p)$ becomes approximately $O(1)$. Formally, we use the following filter kernel of the guided image filtering [10]:

$$\omega_{ps} = \frac{1}{|W'|^2} \sum_{k:(p,s)\in W'_k} \Big(1 + (I_p - \mu_k)^T (\Sigma_k + e)^{-1} (I_s - \mu_k)\Big) \tag{24}$$

Here, $I_p = I_L(p)/255$ is a normalized color vector, μ_k and Σ_k are the mean and co-variance matrix of I_p in a local regression window W'_k, and ϵ is an identity matrix with a small positive coefficient for avoiding over-fitting. This filtering can be computed in $O(1)$ using integral image.

Note that as the constant-label property is the key to being able to use the filtering techniques, this scheme cannot be used in other PatchMatch based methods [3], [4], [11] except for [24] that uses superpixels as computation units.

4 Experiments

In the experiments, we first evaluate our method on the Middlebury benchmark. We further assess the effect of sizes of grid-cells and also the effectiveness of the proposed acceleration schemes in comparison to our previous method [30]. Our method is further compared with the PMBP [3] and PMF methods [24] that are closely related to our approach.

Settings

We use the following settings throughout the experiments. We use a desktop computer with a Core i7 CPU (3.5 GHz $\times$ 4 physical cores) and NVIDIA GeForce GTX Titan Black GPU. All methods are implemented using C++ and OpenCV.

The parameters of our data term are set as $\{\tau_{col}, \tau_{grad}, \gamma, \alpha\} = \{10, 2, 10, 0.9\}$ as specified in [4]. The size of matching windows W_p is set to 41×41, which is the same setting with PMBP [3] and [4], [11]. For the smoothness term, we use $\{\lambda, \tau_{dis}, \epsilon\} = \{20, 1, 0.01\}$ and eight neighbors for $\mathcal{N}$. All of these model parameters are the same with the ones we used in [30].

For our algorithm, we use three grid structures with cell sizes of 5×5, 15×15, and 25×25 pixels. The iteration numbers $\{K_{prop}, K_{rand}\}$ in Algorithm 1 are set to $\{1, 7\}$ for the first grid structure, and $\{2, 0\}$ (only propagation step) for the other two. We iterate the main loop ten times. We use a GC implementation of [5].

We use two variants of our method. **LE-BF** uses the bilateral filtering weight ω_{ps} of Eq. (11) for the adaptive windows. Therefore, the computation of matching costs is as slow as $O(|W|)$. In a GPU implementation, we accelerate

this calculation by computing each unary term individually in parallel on GPU. In a CPU implementation, we compute them as filtering to raw matching costs. This way we can still accelerate the computation at the first step of cost filtering. **LE-GF** uses the guided image filtering [10] of Eq. (24) for w_{ps}. The size of local regression windows W'_k is set to 21×21 so that the size of actual matching windows becomes 41×41. We use $e = 0.01^2$ as specified in [24], [27], and $\lambda = 1$. We only use a CPU implementation for this method. For both LE-BF and LE-GF, we perform disjoint local α-expansions in parallel using four CPU cores.

4.1 Evaluation on the Middlebury benchmark

We show in Table 1 selected rankings on the Middlebury stereo benchmark for 0.5-pixel accuracy. The proposed LE-GF method achieves the current best average rank (3.9) and bad-pixel-rate (5.97%) amongst more than 150 stereo methods including our previous method (GC+LSL) [30]. Even without post-processing, our LE-GF method still outperforms the other methods in average rank, despite that methods [3], [4], [11], [24], [30] use the post-processing. On the other hand, the proposed LE-BF method achieves comparable accuracy with our previous algorithm [30], since both methods optimize the same energy function in a very similar way.

Compared with closely related approaches (PMBP [3] and PatchMatch stereo [4]), which are ranked eleventh and fourteenth in Table 1, although their results of Cones are slightly better than ours in some evaluations, our LE-GF and LE-BF methods consistently outperform the two methods in the other evaluations. We summarize the results of our LE-GF method in Fig. 8.

4.2 Effect of grid-cell sizes

To observe the effect of grid-cell sizes, we use the three sizes of grid-cells, 5×5 pixels (denoted as "S"mall), 15×15 pixels (denoted as "M"edium), 25×25 pixels (denoted as "L"arge), in different combinations and assess the performance using the following five different settings; (S, S, S): the small-size cells for all grid-structures; (M, M, M): the medium-size cells for all grid-structures; (L, L, L): the large-size cells for all grid-structures; (S, M, M): the small and medium-size cells for the first and the other two grid-structures, respectively; (S, M, L): the small, medium, and large-size cells for the first, second, and third grid-structures, respectively (the default setting for our method described above). Here, the iteration numbers for the first to third grid-structures are kept as default. We use the LE-GF method so as to access the effect on cost filtering acceleration as well. We use $\lambda = 0.5$ but keep the other parameters as default. Using these settings, we observe the performance variations by estimating the disparities of only the left image of the *Reindeer* dataset without post-processing.

The plots in Fig. 9 (a) show the transitions of the energy function values over iterations. Here, energies are evaluated by the re-defined energy function using Eq. (24). The plots in Fig. 9 (b) show the temporal transitions of error rates with subpixel accuracy. As shown, the joint use of multiple cell sizes improves the performance in both energy reduction and accuracy. Although (S, M, M) and (S, M, L) show almost the same energy transitions, the proposed combination (S, M, L) shows faster and better convergence in accuracy.

Figure 10 compares the results of the five settings with error maps for (S, M, M) and (S, M, L). The use of larger grid-cells helps to obtain smoother disparities, and it is especially effective for occluded regions.

Comparing the running times in Fig. 9 (b), the use of the small-size cells is inefficient due to the increased overhead in cost filtering, whereas the use of the medium and large-size cells achieves almost the same efficiency.

4.3 Efficiency evaluation in comparison to our previous algorithm [30]

In this section, we evaluate the effectiveness of the three acceleration techniques: 1) parallelization of disjoint α-expansions, and 2) acceleration of unary cost computation by GPU and 3) by cost filtering. For evaluation, we compare following six variants of our method: LE-BF and LE-GF using one or four CPU cores (denoted as CPUx1 and CPUx4), and LE-BF using GPU and one or four CPU cores (denoted as GPU+CPUx1 and GPU+CPUx4). Additionally, we compare with our previous algorithm of [30] with CPU and GPU implementations (denoted as LSL with CPUx1 and GPU+CPUx4). We use the *Rocks1* dataset by estimating the disparities of only the left image without post-processing. Figures 11 (a)–(c) show the temporal transitions of the energy values in the full and zoomed scales, and the subpixel error rates, respectively.

Parallelization of local α-expansions on CPU: Comparing CPUx1 and CPUx4, we observe about 3.5x of speed-up for both LE-BF and LE-GF. Note that the converged energy values of LE-GF are relatively higher than the others because it optimizes a different energy function. However, if we see Fig. 11 (c), it actually finds better solutions in this example. On the other hand, speed-up for LE-BF from GPU+CPUx1 to GPU+CPUx4 is limited to about 1.7x. This is because the unary cost computation is already parallelized by GPU and this speed-up is accounted for the parallelization of other parts, *e.g.*, the computation of pairwise terms and min-cut.

Parallelization of unary cost computation on GPU: By comparing GPU+CPUx1 and CPUx1 of LE-BF, we obtain about 19x of speed-up. This is relatively smaller than 32x of speed-up in our previous LSL method, mainly due to the larger overhead of GPU data transferring.

Fast cost filtering: By comparing LE-GF and LE-BF methods, we observe about 5.3x speed-up for both CPUx1 and CPUx4 versions. It is also worth noting that even a CPU implementation of LE-GF (CPUx4) achieves almost comparable efficiency with a GPU implementation of LE-BF (GPU+CPUx1).

Comparison to our previous method [30]: In comparison to our previous LSL method [30], our LE-BF shows much faster convergence than LSL in the same single-core results (CPUx1). We consider there are mainly three factors contributing to this speed-up; First, the batch-cycle algorithm adopted in [30] makes its convergence slower; Second, we use cost filtering and compute its first step (*i.e.*, raw image matching) in $O(1)$, while [30] computes each unary term individually in $O(|W|)$; Finally, we have removed a per-pixel label refinement step of [30] which requires additional unary-cost computations. Similar speed-up can be

TABLE 1
Middlebury benchmark for 0.5-pixel accuracy. Our method using the guided image filtering achieves the current best average rank 3.9. In *all*, results are evaluated for all pixels where the ground truth is given, while only for non-occluded pixels in *nonocc*, and around depth discontinuities in *disc*.

Algorithm	Avg. Rank	Tsukuba			Venus			Teddy			Cones			Average Percent Bad Pixels
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
1. PROPOSED (GF)	3.9	4.08 2	4.71 2	9.71 4	0.35 2	0.56 2	3.30 2	5.16 5	7.73 3	14.2 5	3.46 5	8.65 6	9.72 9	5.97
2. GC+LSL [30]	6.2	5.04 3	5.56 3	14.0 13	0.66 6	0.88 6	5.82 8	**4.20** 1	7.12 2	12.9 3	3.77 8	9.16 9	10.4 13	6.63
3. PM-Huber [11]	8.8	7.12 11	7.80 13	13.7 11	1.00 12	1.40 13	7.80 19	5.53 8	9.36 5	15.9 9	**2.70** 1	7.90 2	**7.77** 1	7.33
7. PMF [24]	12.5	11.0 39	11.4 36	16.0 32	0.72 8	0.92 7	5.27 7	4.45 3	9.44 7	13.7 4	2.89 2	8.31 3	8.22 2	7.69
11. PMBP [3]	19.7	11.9 52	12.3 48	17.8 60	0.85 10	1.10 8	6.45 11	5.60 9	12.0 12	15.5 6	3.48 6	8.88 8	9.41 6	8.77
14. PatchMatch [4]	28.2	15.0 74	15.4 73	20.3 89	1.00 13	1.34 12	7.75 17	5.66 10	11.8 10	16.5 10	3.80 9	10.2 11	10.2 11	9.91
Reference Evaluations														
BF w/ post-proc	6.6	5.48 3	6.07 3	14.5 15	0.82 9	1.08 7	6.32 10	**4.05** 1	7.24 3	12.4 3	3.69 7	9.12 8	9.96 10	6.73
BF w/o post-proc	8.4	5.36 3	5.99 3	14.1 14	0.82 9	1.11 9	6.33 10	4.59 4	10.8 8	14.0 5	3.90 9	10.3 13	10.6 14	7.32
GF w/ post-proc	3.9	4.08 2	4.71 2	9.71 4	0.35 2	0.56 2	3.30 2	5.16 5	7.73 3	14.2 5	3.46 5	8.65 6	9.72 9	5.97
GF w/o post-proc	3.9	4.07 2	4.79 2	9.78 4	0.41 2	0.78 2	4.01 2	4.26 2	9.19 4	13.6 4	3.37 5	9.63 9	9.59 9	6.13

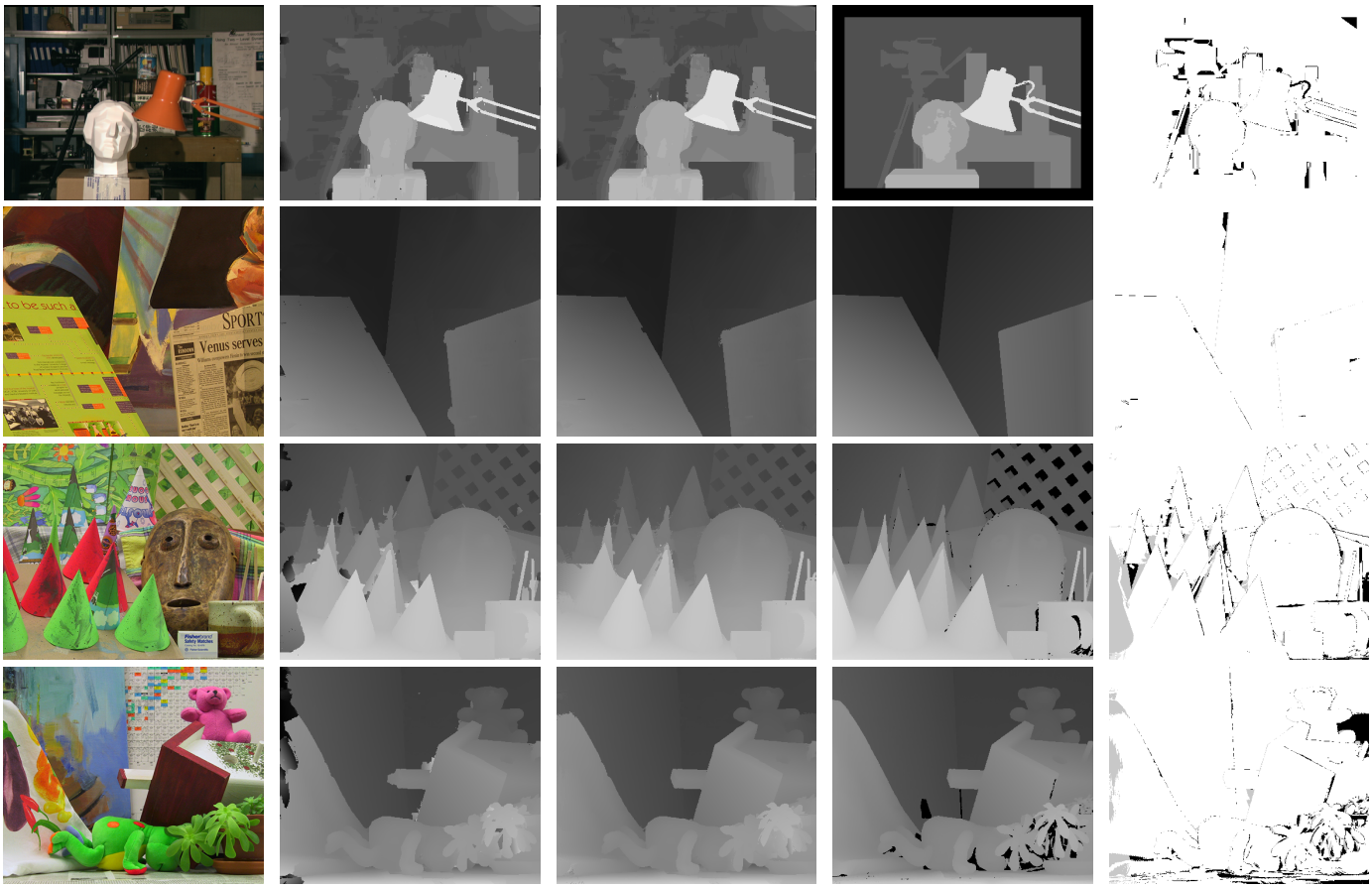
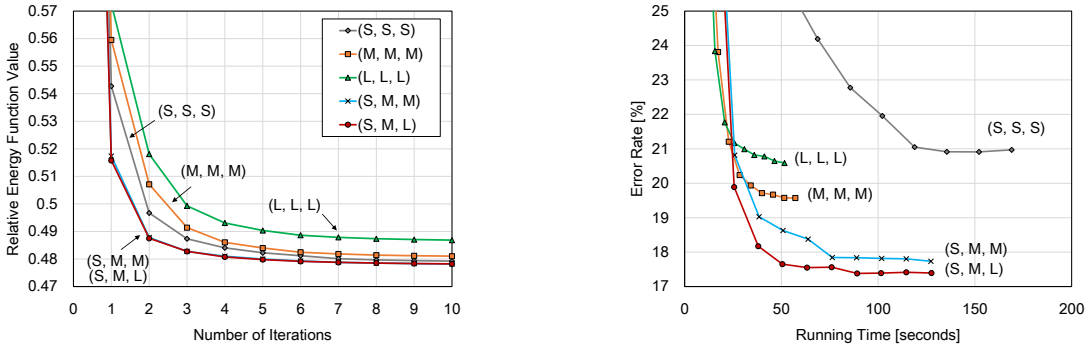


Fig. 8. The results of the proposed LE-GF method on the Middlebury benchmark. From left to right, one of the input images, our results without post-processing, after post-processing, the ground truth, and 0.5-pixel error maps of the results after post-processing are shown. In the error maps, white and black pixels indicate correct and incorrect disparities, while gray indicates incorrect but occluded pixels.

observed from LSL (GPU+CPUx4) to LE-BF (GPU+CPUx4). Note that we obtain only a few percents of speed-up by CPU parallelization in [30], since it does not perform min-cut computation in parallel as disjoint local α-expansions.

4.4 Comparison with PMBP [3]

We compare our method with PMBP [3] that is the closest method to ours. For a fair comparison, we use four neighbors for $\mathcal{N}$ in Eq. (9), which is the same setting with PMBP. For a comparable smoothness weight with the



(a) Energy function values *w.r.t.* iterations (b) Temporal error-rate transitions

Fig. 9. Effect of grid-cell sizes. We use LE-GF with different combinations of grid structures. The S, M, and L denote small, medium, and large grid-cells, respectively. The joint use of different sizes of grid-cells improves the performance. See also Fig. 10 for visual comparison.

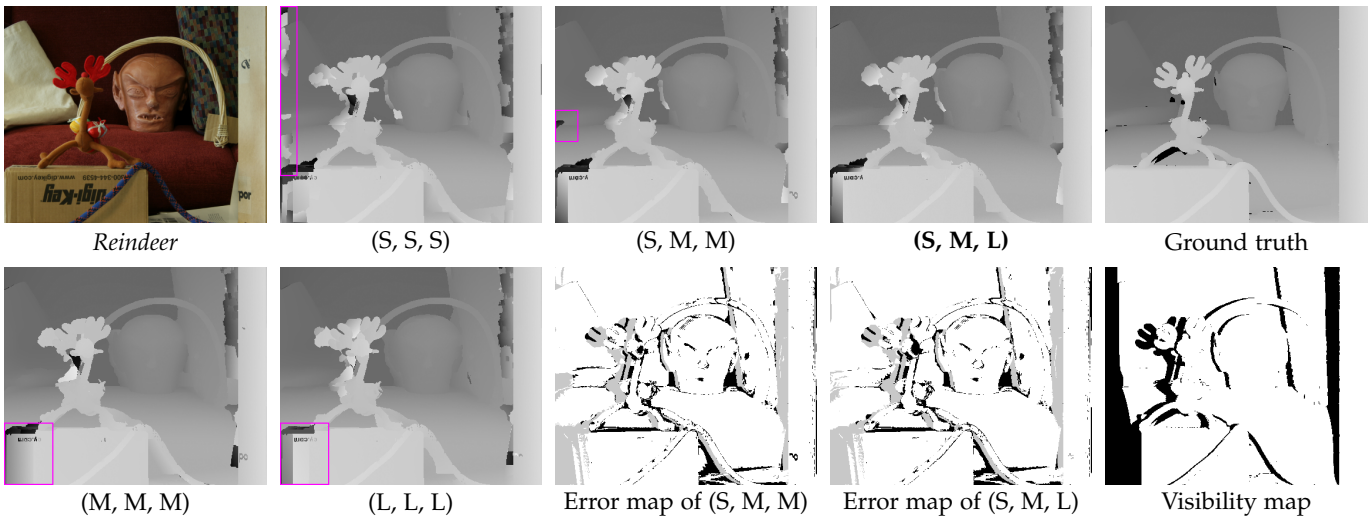
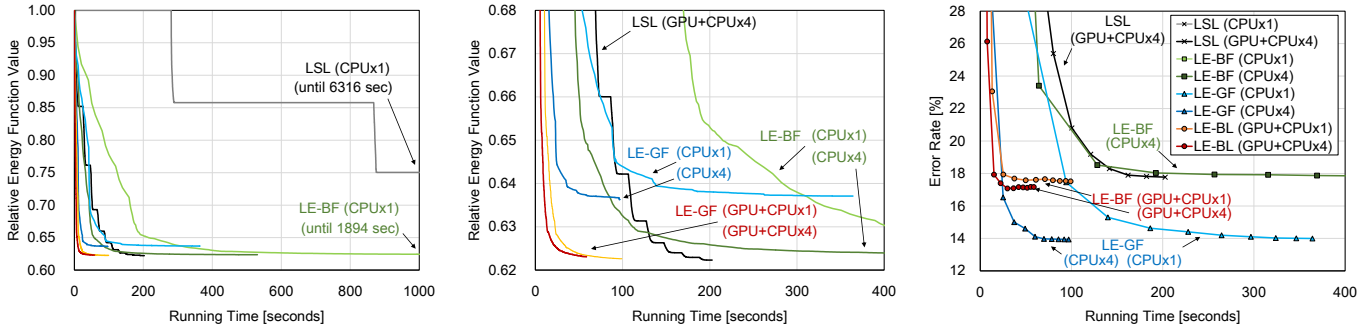


Fig. 10. Visual effect of grid-cell sizes. The use of larger grid-cells leads to smoother solutions and effective for occluded regions. The proposed combination (S, M, L) well balances localization and spatial propagation and performs best. These are all raw results without post-processing.



(a) Time-energy transitions (full) (b) Time-energy transitions (zoomed) (c) Error rate transistions

Fig. 11. Efficiency evaluation in comparison to our previous algorithm (LSL) [30]. Accuracies are evaluated for all-regions at each iteration.

default setting (eight-neighbor $\mathcal{N}$), we use $\lambda = 40$ for LE-BF and $\lambda = 4$ for LE-GF, and keep the other parameters as default. For PMBP, we use the same model as ours; the only difference from the original PMBP is the smoothness term, which does not satisfy the submodularity of Eq. (2). In PMBP, it defines K candidate labels for each pixel, for which we set $K = 1$ and $K = 5$ (original paper uses $K = 5$). We show the comparison using the *Cones* dataset by estimating the disparities of only the left image without post-processing.

Figures 12 (a)–(c) show the temporal transitions of the energy values in the full and zoomed scales, and the sub-pixel error rates, respectively. We show the performance of our method using its GPU and CPU (one or four CPU cores) implementations. For PMBP, we also implemented the unary cost computation on GPU, but it became rather

slow, due to the overhead of data transfer. Efficient GPU implementations for PMBP are not available in literature.[4] Therefore, the plots show PMBP results that use a single CPU core. Figures 12 (a) and (b) show that, even with a single CPU-core implementation, our LE-BF and LE-GF show comparable or even faster convergence than PMBP. With CPU and GPU parallelization, our methods achieve much faster convergence than PMBP. Furthermore, our methods reach lower energies with greater accuracies than PMBP at the convergence.

In Fig. 13 we compare the results of our LE-GF method and PMBP with $K = 5$. While PMBP yields noisy disparities, our method finds smoother and better disparities at around edges and occluded regions.

4.5 Comparison with PMF [24]

We also compare our LE-GF method with PMF [24] using the same data term as ours. For PMF, the number of superpixels K is set to 300, 500, and 700 as used in [24] ($K = 500$ is the default in [24]), and we sufficiently iterate 30 times. Both LE-GF and PMF are run using a single CPU core.

Figures 14 (a)–(c) show the temporal transitions of the energy values in the full and zoomed scales, and the subpixel error rates, respectively. Here, the energy values are evaluated by the re-defined weights using Eq. (24). As shown in Figs. 14 (a) and (b), PMF converges at higher energies than ours, since it cannot explicitly optimize pairwise smoothness terms as being a local method. Furthermore, although energies are reduced almost monotonically in PMF, the transitions of accuracies are not stable and even degrade in same cases. This is also because of the lack of explicit smoothness regularizer, and PMF converges at a bad local minima. Figure 15 compares the results of our method and PMF with $K = 500$. Again, our methods find smoother and better disparities at around edges and occluded regions.

5 Conclusions

In this paper, we have presented an accurate and efficient stereo matching method for continuous disparity estimation. Unlike previous approaches that use fusion moves [21], [25], [35], our method is subproblem optimal and only requires a randomized initial solution. By comparing with a recent continuous MRF stereo method, PMBP [3], our method has shown an advantage in efficiency and comparable or greater accuracy. The use of a GC-based optimizer makes our method advantageous.

Furthermore, by using the subregion cost-filtering scheme developed in a local stereo method PMF [24], we achieve a fast CPU implementation of our algorithm and greater accuracy than PMF. As shown in [24], our method will be even faster by using a more light-weight constant-time filtering such as [23].

We believe that our optimization strategy can be applied for more general corresponding field estimation such as optical flow. We also believe that some occlusion handling schemes based on GC optimization [18], [19], [34] can be incorporated into our framework, which may yield even greater accuracy.

4. GPU-parallelization schemes of BP are not directly applicable due to PMBP's unique settings. The "jump flooding" used in the original PatchMatch [1] reports 7x speed-ups by GPU. However, because it propagates candidate labels to distant pixels, it is not applicable to PMBP that must propagate messages to *neighbors*, and is not as efficient as our 19x, either.

Appendix A

Proof. For the completeness, we repeat the original proof given in [25] with our notations. Obviously, $\bar{\psi}_{pq}(\alpha, \alpha) = 0$. Therefore,

$$
\begin{aligned}
&\bar{\psi}_{pq}(\alpha, \alpha) + \bar{\psi}_{pq}(\beta, \gamma) \\
&\quad = \bar{\psi}_{pq}(\beta, \gamma) && (25)\\
&\quad = |d_p(\beta) - d_p(\gamma)| + |d_q(\beta) - d_q(\gamma)| && (26)\\
&\quad = |\,(d_p(\beta) - d_p(\alpha)) - (d_p(\gamma) - d_p(\alpha))\,| + \\
&\qquad |\,(d_q(\beta) - d_q(\alpha)) - (d_q(\gamma) - d_q(\alpha))\,| && (27)\\
&\quad \leq |d_p(\beta) - d_p(\alpha)| + |d_p(\gamma) - d_p(\alpha)| + \\
&\qquad |d_q(\beta) - d_q(\alpha)| + |d_q(\gamma) - d_q(\alpha)| && (28)\\
&\quad = \bar{\psi}_{pq}(\beta, \alpha) + \bar{\psi}_{pq}(\alpha, \gamma). && (29)
\end{aligned}
$$

Thus, $\bar{\psi}_{pq}(f_p, f_q)$ satisfies the submodularity of expansion moves. For its truncated function,

$$
\begin{aligned}
&\min\left(\bar{\psi}_{pq}(\alpha, \alpha), \tau\right) + \min\left(\bar{\psi}_{pq}(\beta, \gamma), \tau\right) \\
&\quad = \min\left(\bar{\psi}_{pq}(\beta, \gamma), \tau\right) && (30)\\
&\quad \leq \min\left(\bar{\psi}_{pq}(\beta, \alpha) + \bar{\psi}_{pq}(\alpha, \gamma), \tau\right) && (31)\\
&\quad \leq \min\left(\bar{\psi}_{pq}(\beta, \alpha), \tau\right) + \min\left(\bar{\psi}_{pq}(\alpha, \gamma), \tau\right). && (32)
\end{aligned}
$$

Therefore, the truncated function of $\bar{\psi}_{pq}(f_p, f_q)$ also satisfies the submodularity, and so does $\psi_{pq}(f_p, f_q)$. □

Acknowledgments

The authors would thank to anonymous sighted reviewers of our conference paper for giving us helpful comments needed to improve the method representation. This work was supported by JSPS KAKENHI Grant Number 14J09001.

References

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. *Proc. of SIGGRAPH (ACM Trans. on Graph.)*, 28(3):24:1–24:11, 2009.

[2] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 29–43, 2010.

[3] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz. PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation. In *Proc. of British Machine Vision Conf. (BMVC)*, pages 132.1–132.11, 2012.

[4] M. Bleyer, C. Rhemann, and C. Rother. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *Proc. of British Machine Vision Conf. (BMVC)*, pages 14.1–14.11, 2011.

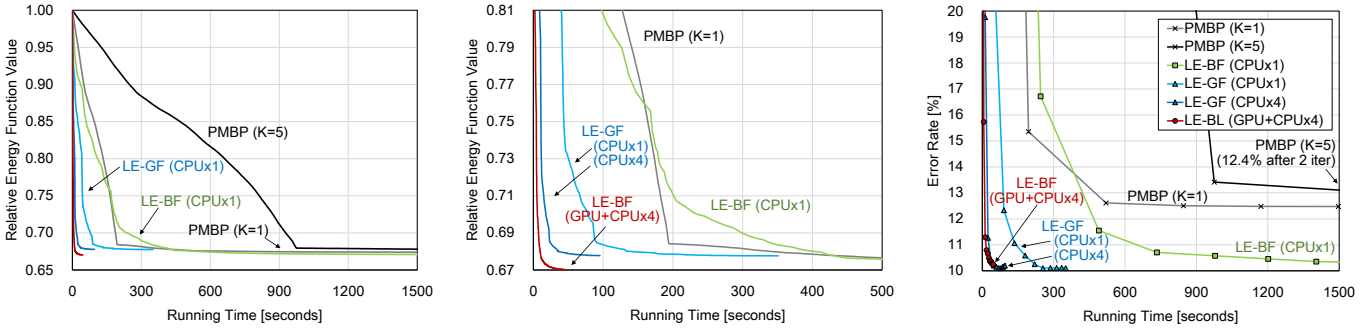
[5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[7] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 261–268, 2004.

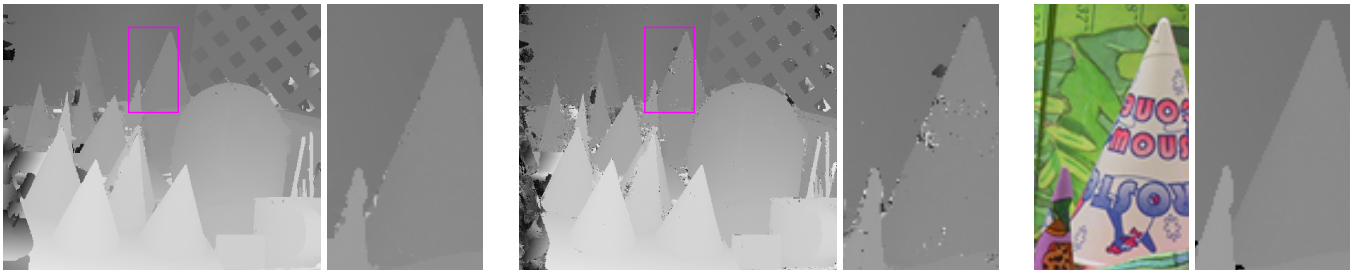
[8] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, 1984.

[9] K. He, J. Sun, and X. Tang. Guided image filtering. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 1–14, 2010.



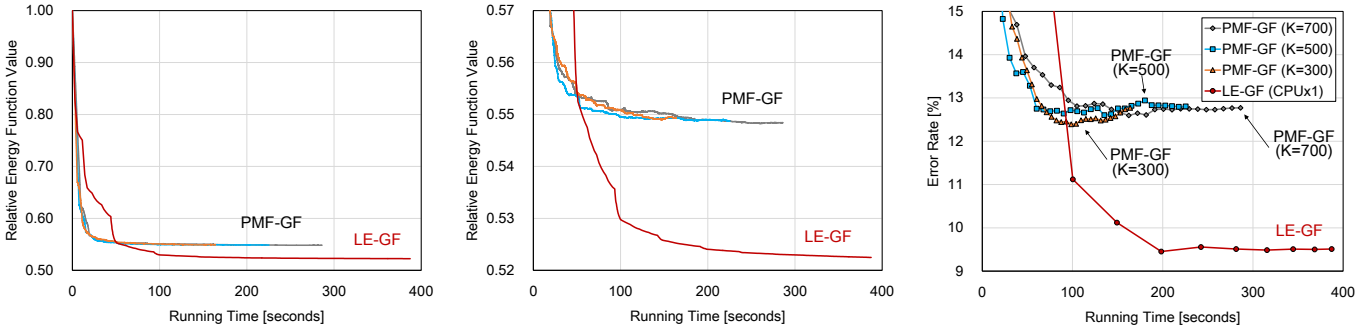
(a) Time-energy transitions (full) (b) Time-energy transitions (zoomed) (c) Error rate transistions

Fig. 12. Efficiency and accuracy comparison with PMBP [3]. Our methods achieve much faster convergence, reaching lower energies and better accuracies at the convergence. Accuracies are evaluated for all-regions at each iteration. See also Fig. 13 for visual comparison.



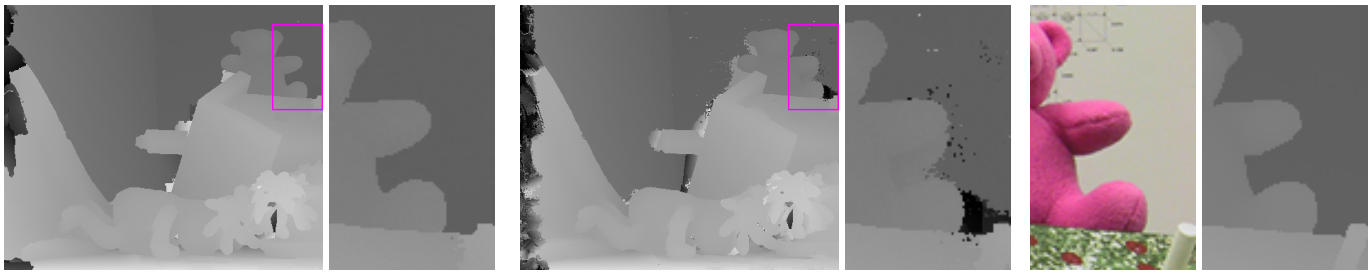
(a) Our LE-BF (b) PMBP [3] (c) Ground truth

Fig. 13. Visual comparison with PMBP [3]. Our method finds smother and better disparities at around edges and occluded regions.



(a) Time-energy transitions (full) (b) Time-energy transitions (zoomed) (c) Error rate transistions

Fig. 14. Efficiency and accuracy comparison with PMF [24]. Our method stably improves the solution and reaches a lower energy with greater accuracy at the convergence. Accuracies are evaluated for all-regions at each iteration. See also Fig. 15 for visual comparison.



(a) Our LE-GF (b) PMF [24] (c) Ground truth

Fig. 15. Visual comparison with PMF [24]. With explicit smoothness regularization, our method finds smoother and better disparities at around edges and occluded regions.

[10] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013.
[11] P. Heise, S. Klose, B. Jensen, and A. Knoll. PM-Huber: PatchMatch with Huber Regularization for Stereo Matching. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, pages 2360–2367, 2013.
[12] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 74–81, 2004.

[13] A. Hosni, M. Gelautz, and M. Bleyer. Accuracy-Efficiency Evaluation of Adaptive Support Weight Techniques for Local Stereo Matching. In *Proc. DAGM/OAGM Symposium*, pages 337–346, 2012.
[14] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2):504–511, 2013.
[15] A. Klaus, M. Sormann, and K. Karner. Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. In *Proc. of Int'l Conf. on Pattern Recognition (ICPR)*, volume 3, pages 15–18, 2006.
[16] V. Kolmogorov. Convergent Tree-Reweighted Message Passing for Energy Minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006.
[17] V. Kolmogorov and C. Rother. Minimizing Nonsubmodular Functions with Graph Cuts – A Review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(7):1274–1279, 2007.
[18] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, volume 2, pages 508–515, 2001.
[19] V. Kolmogorov and R. Zabih. Multi-camera Scene Reconstruction via Graph Cuts. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 82–96, 2002.
[20] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004.
[21] V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion Moves for Markov Random Field Optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1392–1405, 2010.
[22] J. Liu and J. Sun. Parallel graph-cuts by adaptive bottom-up merging. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2181–2188, 2010.
[23] J. Lu, K. Shi, D. Min, L. Lin, and M. Do. Cross-based local multipoint filtering. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 430–437, 2012.
[24] J. Lu, H. Yang, D. Min, and M. N. Do. Patch Match Filter: Efficient Edge-Aware Filtering Meets Randomized Search for Fast Correspondence Field Estimation. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1854–1861, 2013.
[25] C. Olsson, J. Ulen, and Y. Boykov. In Defense of 3D-Label Stereo. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1730–1737, 2013.
[26] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Trans. on Graph.*, 23(3):664–672, 2004.
[27] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3017–3024, 2011.
[28] P. Strandmark and F. Kahl. Parallel and distributed graph cuts by dual decomposition. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2085–2092, 2010.
[29] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):1068–1080, 2008.
[30] T. Taniai, Y. Matsushita, and T. Naemura. Graph Cut based Continuous Stereo Matching using Locally Shared Labels. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1613–1620, 2014.
[31] H. Tao, H. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, volume 1, pages 532–539, 2001.
[32] L. Wang and R. Yang. Global stereo matching leveraged by sparse ground control points. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3033–3040, 2011.
[33] Z.-F. Wang and Z.-G. Zheng. A region based stereo matching algorithm using cooperative optimization. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.
[34] Y. Wei and L. Quan. Asymmetrical Occlusion Handling Using Graph Cut for Multi-View Stereo. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 902–909, 2005.
[35] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global Stereo Reconstruction under Second-Order Smoothness Priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2115–2128, 2009.
[36] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized Belief Propagation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, pages 689–695, 2000.
[37] K.-J. Yoon and I.-S. Kweon. Locally adaptive support-weight approach for visual correspondence search. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 924–931, 2005.