Fast Cost-Volume Filtering for Visual Correspondence and Beyond

A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz

Abstract—Many computer vision tasks can be formulated as labeling problems. The desired solution is often a spatially smooth labeling where label transitions are aligned with color edges of the input image. We show that such solutions can be efficiently achieved by smoothing the label costs with a very fast edge-preserving filter. In this paper we propose a generic and simple framework comprising three steps: (i) Constructing a cost volume; (ii) Fast cost volume filtering; and (iii) Winner-Takes-All label selection. Our main contribution is to show that with such a simple framework state-of-the-art results can be achieved for several computer vision applications. In particular, we achieve (i) disparity maps in real-time, whose quality exceeds those of all other fast (local) approaches on the Middlebury stereo benchmark, and (ii) optical flow fields which contain very fine structures as well as large displacements. To demonstrate robustness, the few parameters of our framework are set to nearly identical values for both applications. Also, competitive results for interactive image segmentation are presented. With this work, we hope to inspire other researchers to leverage this framework to other application areas.

Index Terms—Stereo matching, Optical flow, Interactive image segmentation.

1 Introduction

Discrete label-based approaches have been successfully applied to many computer vision problems such as stereo, optical flow, interactive image segmentation or object recognition. In a typical labeling approach, the input data is used to construct a three-dimensional cost volume, which stores the costs for choosing a label l (e.g. disparities in stereo) at image coordinates x and y. For stereo, these costs are given by pixel-wise correlation (e.g. absolute differences of the intensities) between corresponding pixels.

Then the goal is to find a solution which (i) obeys the label costs, (ii) is spatially smooth; and (iii) label changes are aligned with edges in the image. To this end, a popular approach is to utilize a Conditional (Markov) Random Field model (CRF). This means that an energy function is formulated, where the label costs are encoded in a data term and the spatially smooth edge-aligned solution is enforced by an e.g. pairwise smoothness term. This cost function can then be minimized using global energy minimization approaches such as graph cut or belief propagation. A drawback is that such global methods are often relatively slow and do not scale well to high-resolution images or large label spaces. Fast approximations (e.g. [1]) usually come at the price of loss in quality, due to less-global optimization schema.

- *A. Hosni, C. Rhemann, M. Bleyer and M. Gelautz are with the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Vienna, Austria.*
  E-mail: [asmaa, Rhemann, Bleyer, Gelautz]@ims.tuwien.ac.at
- *C. Rother is with Microsoft Research, Cambridge, UK.*
  E-mail: carrot@microsoft.com

Continuous counterparts to discrete labeling methods are based on convex energy functions which can be efficiently optimized on the GPU, e.g. [2], [3], [4]. A drawback is that many of these approaches have a restricted form of the data and smoothness term. For instance, the brightness constancy assumption in optical flow is usually linearized and thus only valid for small displacements. To overcome this problem, a coarse-to-fine framework is commonly used which, however, cannot handle objects whose scale is much smaller than their motion. Another problem is posed by the convexity of the smoothness term, which might over-smooth the solution. This may be the reason why convex models have not reported state-of-the-art stereo results yet.

An interesting alternative to an energy-based approach is to apply local filtering techniques. The filtering operation achieves a form of spatially-local smoothing of the label space, in contrast to a potential spatially-global smoothing of a CRF. Despite this conceptual drawback, an observation of our work, and previous work [5], is that "local smoothing" is able to achieve high-quality results. We believe that one of the reasons may be the dominance of the (pixel-wise) data term with respect to the smoothness term.[1] An important observation is that the data term will play an even more dominant role in the future, since both video and still-picture cameras are consistently growing in terms of frame-resolution and also dynamic range.

1. As part of future work, it may be possible to show that for some applications the smoothness term of a learned energy propagates information only in a small neighborhood. Note that for some applications global constraints exist such as the occlusion constraint in stereo matching and optical flow. In our approach we model the occlusion constraint with a fast additional operation.

Note, a detailed comparison between energy-based and filtering-based methods is beyond the scope of this paper, and we will only briefly discuss them in sec. 6.

In general, relatively little work has been done in the domain of filter-based methods for discrete labeling problems [5], [6], [7]. Most importantly, there is no filter-based approach for *general multi-labeling* problems which is both *fast (real-time)* and achieves *high-quality* results. The key contribution of this paper is to present such a framework.

1.1 Previous Filter-Based Methods

Let us briefly review the existing ideas of filter-based methods, which are the motivation of our work (details in sec. 2). Apart from [7] all works have concentrated on the application of stereo matching. Yoon and Kweon [5] showed that an edge-preserving bilateral filter on the cost volume can achieve high-accuracy results. Note that the authors of [5] did not use the term "filtering" to describe their method, but called it weighted support window aggregation scheme. This means that they use a naive implementation of the bilateral filter, which is slow and diminishes the run time advantage of local over global methods. Richard et al. [6] realized this shortcoming and suggested an approximate but fast (real-time) implementation of the filter. However, their solution could not even get close to the state-of-the-art results in stereo matching. Also, their approach is specifically tailored to stereo matching and hence does not convey the important insight that this filtering technique can be leveraged to general labeling tasks, outside stereo matching. Recently, [7] suggested edge-sensitive smoothing of label costs for image editing tasks different from stereo. However, their approach, based on fast geodesic filter operations, is inherently limited to problems with two labels only.

1.2 Our Approach

In this work, we overcome the above limitations and present a filter-framework which efficiently achieves high-quality solutions for general multi-label problems, hence is competitive with energy-based methods. This is possible due to the recently proposed *guided filter* [8], which has the edge-preserving property and a run time independent of the filter size. Thus, state-of-the-art results can be achieved without the need to trade off accuracy against efficiency.

Let us now detail our method from a stereo perspective. We first construct a cost volume with axes (x, y, l), which is known as disparity space image (DSI) in stereo [9]. Figure 1(i)(b) shows an (x, l) slice through this volume for the scanline in figure 1(i)(a). We can obtain a solution to the labeling problem by choosing the label of the lowest cost at each pixel (i.e. $\arg\min$ over the columns of figure 1(i)(b)). The pixels with the lowest costs are marked red in figure 1(i)(b). The resulting disparity map (figure 1(ii)(b)) is very noisy, because the solution is not regularized. However, this method is very fast.

To regularize the solution we can aggregate (smooth) the costs within a support window before applying the Winner-Takes-All label selection. This is known as window-based matching in stereo literature. It is known that this aggregation step is equivalent to filtering the (x, y) dimensions of the cost volume with a box filter [9]. The result of filtering the cost volume in figure 1(i)(b) using a box filter is shown in figure 1(i)(c). The disparity solution of minimum costs (marked red in figure 1(i)(c)) is smooth but not aligned with image edges. This is because the box filter overlaps depth discontinuities that are illustrated with green dashed lines in figure 1. This leads to the well-known "edge-fattening effect" in stereo (see disparity map in figure 1(ii)(c)). While the quality of the disparity map is poor, the advantage is that the filtering process is very fast because it can be speeded up via a sliding window technique.

To overcome the "edge-fattening problem", we can smooth the cost volume with a *weighted* box filter where weights are chosen such that the filter preserves edges of the input color image. For instance, one can apply a joint bilateral filter. Here, weights are computed from the color image and weighted averaging is performed on the cost values. In this case, the approach becomes very similar to the one proposed in the original adaptive support weight paper [5].[2] Figure 1(d) shows that applying the joint bilateral filter on the cost volume leads to a spatially smooth disparity solution that is also aligned with image edges. While this method leads to high-quality results, its computational speed represents a problem. The runtime of an exact implementation of the joint bilateral filter depends on the size of the support window (i.e. is slow for applications like stereo matching that require a large support window) and fast approximations degrade the quality considerably (see section 2).

In our work, we propose an algorithm that produces high-quality results at real-time frame rates and hence combines the advantages of all filtering approaches outlined above. We use the recently proposed guided filter [8] to smooth the cost volume in a way that color edges are preserved (see figure 1(e)). In our experiments, this filtering technique can even outperform the joint bilateral filter in terms of quality of results. Even more importantly, it can be implemented as a sequence of box filters so that runtime is independent of the filter window size. Moving away from the stereo perspective, we show that the concept of filtering the cost volume leads to a

2. The approach in [5] is motivated from a different perspective and does not show its relations to filtering explicitly.

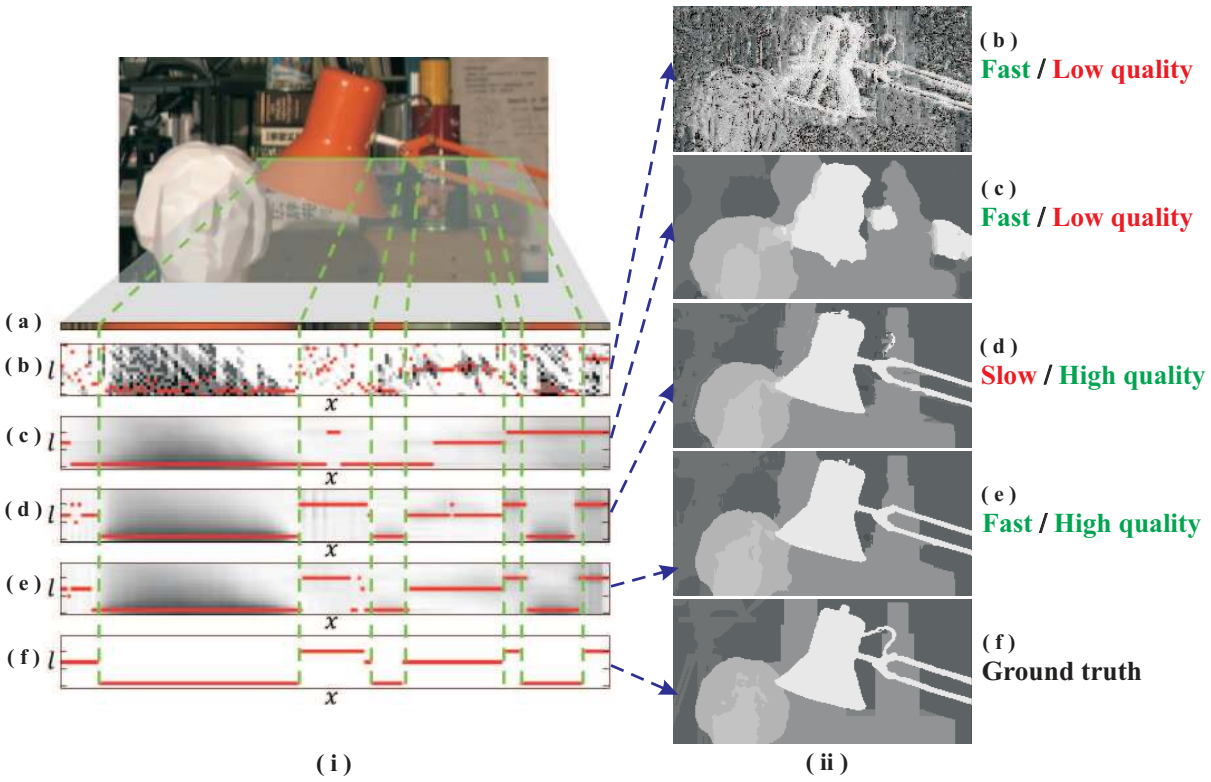


Fig. 1: **Our approach applied on the stereo matching problem.** (i) Cost volume filtering. (a) Zoom of the green line in the input image. (b) Slice of the cost volume (white/black/red: high/low/lowest costs) for the line in (a). (c-e) Cost slice smoothed along x and y-axes (y is not shown here) using (c) the box filter, (d) the joint bilateral filter and (e) the guided filter [8], respectively. (f) Ground truth labeling. (ii) Crops of the "Tsukuba" disparity maps that are computed using different methods of filtering in (i). In contrast to the other illustrated methods, our approach shown in (e) produces high-quality disparity maps *and* is computationally efficient (real-time).

generic and fast framework that is widely applicable to other computer vision problems. In particular, we instantiate our framework for three applications:

- A real-time stereo approach that outperforms all other local methods on the Middlebury benchmark both in terms of speed and accuracy.
- A discrete optical flow approach that handles both fine (small scale) motion structure and large displacements. We tackle the huge label space by fast cost filtering.
- A fast and high-quality interactive image segmentation method.

2 Related Work

As mentioned above, there have only been a few attempts to simulate the edge-preserving smoothness properties of a CRF by filtering the label costs. We review those now. We also review three application areas related to this work (i.e. stereo, optical flow and interactive image segmentation). In each application area a large number of methods have been proposed, and in the following we will only focus on the most relevant ones.

2.1 Stereo

In local stereo matching, Yoon and Kweon [5] have proposed a weighted support aggregation scheme. The main idea is to choose an appropriate local support region for each pixel adaptively. This is done via adjusting the support weights of the pixels in a typically squared support window by comparing the pixels' color and spatial position against the ones of the center pixel. Several subsequent works, which are based on the same idea, have been proposed since then (e.g. [10], [11]), and are summarized in [12], [13]. These methods give results comparable to those based on global energy minimization. However, adaptive support weight approaches are usually computationally slow, which cancels out the runtime advantage of local over global methods. The problem is that, when using adaptive weights, the computation of aggregated costs can no longer be accomplished in an incremental way (using a sliding window technique). Consequently, the computational complexity depends on the size of the match window. This is a major disadvantage, because adaptive support weight methods typically use large windows (e.g., 35×35 pixels). In the

following, we review previous work that concentrates on overcoming the dependency on the match window size, i.e., so-called O(1) approaches.

Some O(1) algorithms for adaptive support weights stereo [14], [15], [6] concentrate on the bilateral filter. However, an exact implementation of the bilateral filter is slow, because its runtime heavily depends on the filter size. Thus these methods resort to approximations of the bilateral filter in order to achieve real-time performance. In [14] and [15] the bilateral filter is approximated using integral histograms [16], while in [6] the bilateral grid [17] is used. These approximations cannot be easily applied to color images due to runtime limitations or high memory requirements. For instance, the authors of [6] report a memory consumption of 764 GB when applying their method on full-color (e.g. RGB) stereo images. As a consequence, these approaches are limited to grayscale input images, giving poor results at disparity boundaries. This is also reflected in the Middlebury stereo benchmark [18], where the method of [6] is on the 90^{th} rank out of over 110 methods [3]. In contrast, our real-time implementation ranks 16^{th}. To increase the quality, the authors of [6] proposed an alternative that uses two color channels. However, this method is 13 times slower than their monochromatic approach (no real-time performance) and still inferior to their re-implementation of [5]. In contrast, we use the guided filter [8] that enables real-time performance on grayscale and color images, and leads to results that even outperform the exact implementation of [5] in terms of quality.

Zhang et al. [19] also proposed an O(1) method for local stereo matching. The speed-up comes from using (i) a cross-shaped support region and (ii) only binary support weights. The problem of the applied cross-shaped support region is that the algorithm fails for thin structures that are neither horizontal nor vertical. The problem of using only binary support weights is that slanted surfaces cannot be reconstructed well, which has also been noted in [20]. The reason is that pixels at the borders of the match window may have the same influence on the aggregated costs as pixels close to the center point, since a spatial weighting is missing. Binary support weights have also been used in [20] and [21]. The idea is to compute a color segmentation in a preprocessing step and then apply a modified version of the sliding window technique which leads to a runtime independent of the match window size. Apart from binary support weights being a suboptimal choice, computing the color segmentation becomes a new bottleneck in the algorithm.

After publication of the conference version of this paper [22], De-Maeztu et al. [23] proposed a stereo matching approach that is based on a cost aggregation strategy very similar to ours. In more detail, [23] presented a symmetric stereo approach that aggregates costs according to both input images simultaneously. It is worth noting that we have also presented a symmetric formulation in [22] that is in fact identical to the symmetric stereo method in [23]. However, in general, our approach aggregates costs based on a single image. This enables a generalization of our strategy to other vision problems such as interactive image segmentation where only a single input image is available, whereas [23] only works for stereo matching.

It is also interesting to note that we recently extended the proposed stereo matching technique to the spatio-temporal domain, see Hosni et al. [24]. The key idea is to filter the cost volume in the spatio-temporal domain in order to achieve temporally consistent disparity maps, where disparity changes are aligned with spatio-temporal edges of the video cube.

2.2 Optical Flow

Many optical flow methods rely on continuous optimization strategies. In the continuous domain, Tschumperl'e and Deriche [25] showed that a diffusion tensor-based smoothness regularizer can be transformed into local convolutions with oriented Gaussians. In the optical flow approach of Xiao et al. [26], the Gaussian kernels were replaced by the bilateral filter. In this line of research, Werlberger et al. [27] and Sun et al. [28] incorporated the adaptive support weights of [5] into a variational approach. In contrast, the goal of our work is to apply local filtering in a discrete framework.

All aforementioned approaches are based on the popular variational coarse-to-fine framework that cannot handle large displacements of small objects, as shown in [29], [30]. To overcome this problem, a discretized data term can be integrated into a variational framework as a soft constraint (see [29], [30], [31]). Recently, Xu et al. [31] presented a method that handles both large and small displacements in a coarse-to-fine framework by using a discrete optimization approach to refine the flow estimates of the coarser level.

Purely discrete label-based approaches do not suffer from this problem, but a major challenge is the huge label space (each flow vector is a label and sub-pixel accuracy further increases the label space). Due to these difficulties, discrete approaches [32] could not report state-of-the-art performance for a long time and usually have to trade off search space (quality) against speed. A discrete-continuous approach was proposed by Lempitsky et al. [33] which fuse continuous proposal solutions in a discrete optimization framework. A similar approach has been taken in [34]. In [35] an energy function is defined over a tree of over-segmentations and minimized using dynamic programming. Recently, [36] developed a discrete ana-

3. The methods of [14] and [15] are not ranked in the Middlebury table, but the qualitative results are clearly inferior to our method.

log to the work of [26] (see above), based on highly-connected graph structures. While all aforementioned discrete optical flow methods trade off search space (quality) against speed, our filter based method efficiently deals with the search space and handles both large displacements and fine small-scale structures.

2.3 Interactive Image Segmentation

Interactive image segmentation is a binary labeling problem. It aims to separate the image into foreground and background regions given some hints by the user. In the seminal work of Boykov and Jolly [37], the user assigns a few pixels in the image to foreground and background. The assigned pixels are used to build appearance models (i.e. histograms of intensities), which define the costs for labeling a pixel as foreground or background. The optimal segmentation is computed using graph cuts, regarding the user-assigned pixels as hard constraints. Based on [37], the GrabCut approach of Rother et al. [38] proposes an even simpler user interface that requires only a bounding box around the object of interest. The appearance models are then iteratively refined in the optimization process. Criminisi et al. [7], [39] showed an approach which is applicable to problems with two labels, like binary image segmentation and panoramic stitching of two overlapping images. The idea is to filter a likelihood ratio mask with a fast geodesic morphological operator. It remains unclear whether this approach could be extended to multi-label problems such as stereo. Also, the relationship between their morphological operator and e.g. the bilateral filter is not discussed in detail.

Related to interactive segmentation, [8] adopted the guided filter to compute a soft-segmentation, i.e., a so-called alpha matte [40]. This is done by filtering a binary segmentation mask, as opposed to the cost volume as in our approach. We close the loop by estimating the binary segmentation via filtering the cost volume constructed from coarse user input. This results in a purely filter-based segmentation and matting pipeline.

3 Cost-Volume Filtering

We now describe our labeling framework. Let us consider a general labeling problem, where the goal is to assign each pixel i with coordinates (x, y) in the image I a label l from the set $\mathcal{L} = \{1, \dots, L\}$. The label assigned to pixel i is denoted by f_i and f is the collection of all label assignments. Our approach consists of three steps: (i) constructing the cost volume, (ii) filtering the cost volume and (iii) label selection. The cost volume C is a three dimensional array which stores the costs for choosing label l at pixel $i = (x, y)$.

The L slices of the cost volume are now filtered. To be more precise, the output of the filtering at pixel index i with label l is a weighted average of neighboring pixels in the same slice:

$$C'_{i,l} = \sum_j W_{i,j}(I) C_{j,l}. \quad (1)$$

Here, C' is the filtered cost volume and i and j are pixel indices. The filter weights $W_{i,j}$ depend on the guidance image I, which is in the case of e.g. stereo the reference image.

Once the cost volume is filtered, the label at pixel i is simply chosen in a Winner-Takes-All manner as

$$f_i = \arg\min_{l \in \mathcal{L}} C'_{i,l}. \quad (2)$$

The filter weights $W_{i,j}$ in eq. (1) should be chosen such that intensity changes in the guidance image are maintained in the filter output. In this work we use the weights of the guided filter [8], which we briefly review now (but other weights are also possible).

For simplicity, we start by using a grayscale guidance image I. Then the weights $W_{i,j}$ are given by:

$$W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right), \quad (3)$$

where μ_k and σ_k are the mean and variance of I in a squared window ω_k with dimensions $2r+1 \times 2r+1$, centered at pixel k.[4] We denote the number of pixels in this window with $|\omega|$ and ϵ is a smoothness parameter explained below.

To see why the filter weights preserve edges of I in the filter output, let us consider figure 2 which shows a 1-D step edge. The numerator $(I_i - \mu_k)(I_j - \mu_k)$ in eq. (3) has a positive sign if I_j is located on the same side of the edge as I_i, and has a negative sign otherwise. Thus the term $1 + \frac{(I_i-\mu_k)(I_j-\mu_k)}{\sigma_k^2+\epsilon}$ in eq. (3), is large for pixel pairs which are on the same side of the edge and small otherwise. Hence, pixels are not averaged if they are separated by an image edge. The strength of the averaging effect is controlled by the parameter ϵ in eq. (3). If $\sigma_k^2 \ll \epsilon$ (then μ_k is similar to I_i and I_j), then the numerator in eq. (3) is much smaller than the denominator. Hence, the kernel converges to an (unweighed) low-pass filter: $W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} 1$. The

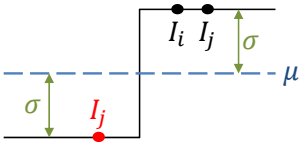


Fig. 2: **1D step edge.** We show μ and σ for a kernel centered exactly at the edge. See text for details. Figure courtesy from [8].

4. The size of the filter kernel itself is $(4r+1)^2$, because the sum in eq. (3) is defined over all windows which include pixel indices i and j.

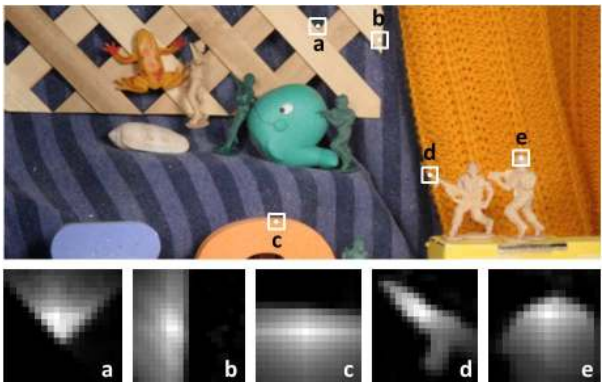


Fig. 3: **Filter kernels.** We show kernels of the guided filter with $r = 9$ and $\epsilon = 0.01^2$ at different locations in an image taken from the Middlebury optical flow evaluation webpage [41].

filter weights are similarly defined for color images:

$$W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} (1+(I_i-\mu_k)^T(\Sigma_k+\epsilon U)^{-1}(I_j-\mu_k)). \tag{4}$$

Here, I_i, I_j and μ_k are 3×1 (color) vectors and the covariance matrix Σ_k and identity matrix U are of size 3×3. The filter weights for some image regions are visualized in figure 3. The weights are high in regions which are self-similar to the central pixel and low otherwise. It has been shown [8] that a weighted average with weights as defined in eq. (3) or (4) can be implemented efficiently on the CPU as a sequence of box filters using the integral imaging technique [42]. We apply the same technique to obtain an even more efficient GPU implementation.

4 Applications

We now apply our cost filtering framework to three different vision applications. Note that the method for stereo and optical flow is almost identical and only one parameter is set differently in the experiments (see explanation in sec. 5).

4.1 Stereo Matching

For stereo matching the labels l correspond to vectors (u, v) which define the displacement in x and y directions. In the x direction, the displacement corresponds to the disparity d ($u = d$) and there is no shift in the y direction ($v = 0$).

Cost computation: The cost volume expresses how well a pixel i in image I matches the same pixel in the second image I' shifted by vector l. We choose our pixel-based matching costs to be a truncated absolute difference of the color and the gradient at the matching points. Such a model has been shown to be robust to illumination changes and is commonly used in optical flow estimation [29], [43]:

$$C_{i,l} = (1-\alpha) \cdot \min\left[||I_i - I'_{i-l}||, \tau_1\right] + \alpha \cdot \min\left[||\nabla_x I_i - \nabla_x I'_{i-l}||, \tau_2\right]. \tag{5}$$

Here, ∇_x is the gradient in x direction, α balances the color and gradient terms and τ_1, τ_2 are truncation values.[5]

We then filter the cost volume according to eq. (1) with weights in eq. (4) using I as guidance image. Finally, we compute the disparity map f for image I as per eq. (2).

Occlusion detection and filling: To detect pixels with unreliable disparity values, which are mainly caused by occlusions, we apply a left-right cross checking procedure. To this end, we additionally compute the disparity map f' for the right image I' in the same way as described above. The disparity map f, computed from the left input image is shown in figure 4(a). We mark a pixel in the left disparity map as invalid if the disparity of its matching pixel differs, i.e. if $f_i \neq f'_{i-l}$. This process detects the occluded pixels as well as mismatched ones. The invalid pixels are marked red in figure 4(b). Each invalid pixel is then assigned to the lowest disparity value of the spatially closest non-occluded pixel which lies on the same scanline (pixel row). This scanline-based filling process is illustrated in figure 5, where the invalid pixel i (outlined in yellow) is assigned to the disparity of pixel j. The disparity map after filling is shown in figure 4(c).

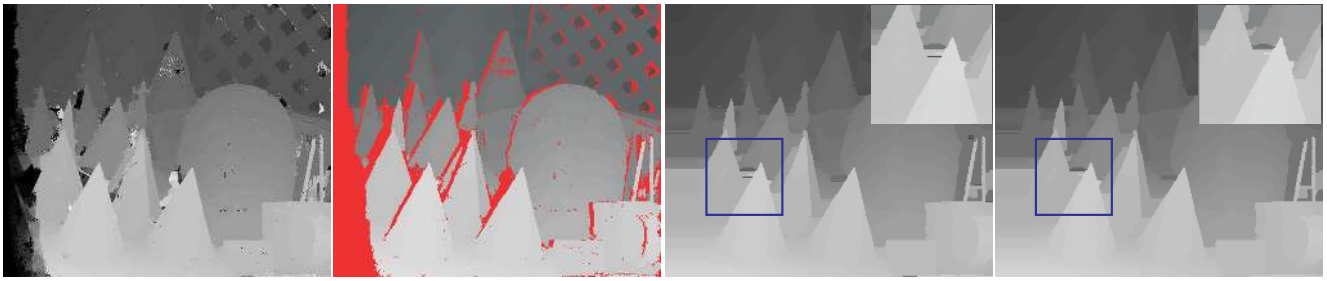
Post-processing: Our simple strategy for filling invalidated pixels generates streak-like artifacts in the disparity map (see inlet in figure 4(c)). To remove these artifacts, while preserving the object boundaries, we apply a weighted median filter for the filled-in pixels. Note that the weighted median filter is applied *only* to invalid pixels, i.e. pixels which fail the left-right cross checking (red pixels in figure 4(b)). As filter weights, we would ideally like to choose those of the guided filter defined in eq. (4). However, computing these weights involves building a sparse matrix of size $N \times N$, where N is the number of image pixels. The non-zero entries of this matrix increase tremendously for large window sizes, thus immense memory and time are required for computing this matrix[6]. Thus we resort to the bilateral filter weights:

$$W^{bf}_{i,j} = \frac{1}{K_i} \exp(-\frac{|i-j|^2}{\sigma_s^2}) \exp(-\frac{|I_i - I_j|^2}{\sigma_c^2}), \tag{6}$$

where σ_s and σ_c adjust the spatial and color similarity and K_i is a normalization factor. We truncate the

5. One could also use a spatially varying α as it was recently proposed in [31]. However, this approach gave worse results on the stereo test images.

6. The filter weights do not have to be computed explicitly when using a weighted average filter, which we use to smooth the cost volume.



(a) Computed left disparity map | (b) Disparity map of (a) with invalidated pixels marked in red | (c) Invalidated pixels of (b) filled with scan-line procedure | (d) Disparity map in (c) after applying a joint bilateral median filter for invalidated pixels

Fig. 4: **Occlusion detection, filling and post-processing.** (a) Disparity map of the left image. (b) Occlusions and mismatches are detected by applying left-right consistency checking. Pixels that fail this check are marked in red. (c) Scan-line based occlusion filling. (d) A joint bilateral median filter is applied on the invalidated pixels (red pixels in (b)) to smooth the filled-in regions. Note that pixels that passed the left-right check (non-red pixels in (b)) are not affected by this operation.

filter weights at a radius of 7 pixels. We discuss the influence of the post-processing on the quality of the disparity map in section 5.1.

Alternative - symmetric stereo: Let us consider figure 6(a). It shows a crop of the left input image of the "Cones" image pair. The image regions in figure 6(a) that are occluded in the right view are marked by a red band. The pixel marked by a yellow cross in figure 6(a) is visible in both images but is very close to an occluded region. The filter weights, defined in eq. (4), for a kernel centered at the marked pixel are shown in figure 6(b). The kernel weights are high in the occluded region (outlined in red), hence the disparity for the yellow pixel will be influenced by matching costs in the occluded regions. The matching costs in the occluded regions are unreliable because there are no matching points for these pixels in the right stereo image. As a consequence, the unreliable matching costs are propagated into the visible regions and lead to artifacts as shown in figure 6(c).

To avoid propagating mismatches from the occluded regions, the filter kernel centered at the yellow pixel must not overlap the occluded region. This can be achieved by computing the filter weights based on *both* input images. In particular, we compute the filter weights for pixel i in eq. (4) based on the color of the pixel in the left image I_i and its matching pixel in the right image I'_{i-l} that can be computed using disparity hypothesis l. To this end, we replace the 3×1 vector I_i in eq. (4) with a 6×1 vector whose entries are given by the RGB color channels of both I_i and I'_{i-l}. The dimensions of I_j, Σ_k, U and μ_k change accordingly.

The filter weights computed with this symmetric approach are visualized in figure 6(d). We see that the filter weights are low in the occluded regions. As a consequence, disparity artifacts do not leak out of the occluded region (see figure 6(e)). We found that the symmetric stereo approach gives visible improvements near occlusion boundaries, which results in a slight quality gain over the asymmetric approach. We compare the asymmetric with the symmetric approach in section 5.1.

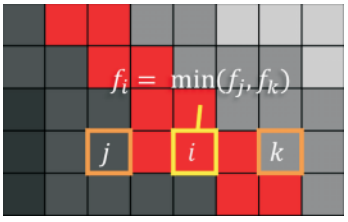


Fig. 5: **Filling strategy.** A pixel grid is shown. The disparity values are indicated by gray-scale values (dark means low disparity). Pixels invalidated by the left-right cross check are shown in red. For the invalid pixel i marked in yellow, we find the closest valid pixels (here j and k) which lie on the same scanline. Pixel i is then assigned to the lower of the two disparities f_j and f_k, which is f_j in this example.

4.2 Optical Flow

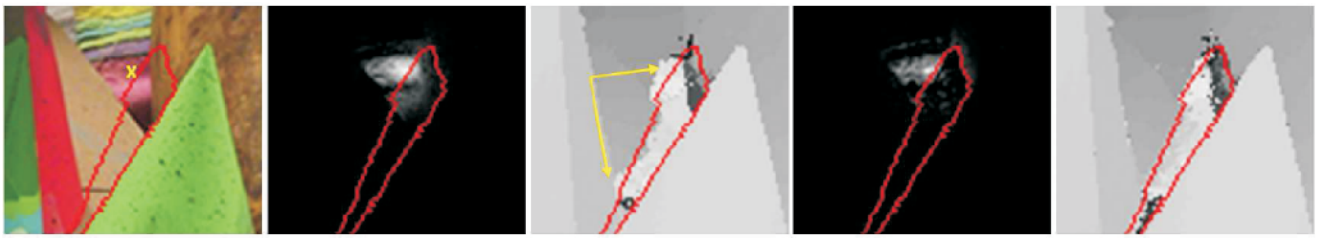
Our optical flow approach is almost identical to stereo. Here, the labels l correspond to vectors (u, v) which define the flow in x and y directions, respectively.

Cost computation: We compute the matching costs of corresponding pixels as in stereo, but additionally use the gradient ∇_y in y direction:

$$C_{i,l} = (1-\alpha) \cdot \min\left[||I_i - I'_{i-l}||, \tau_1\right] + \quad (7)$$
$$\alpha \cdot \min\left[||\nabla_x I_i - \nabla_x I'_{i-l}|| + ||\nabla_y I_i - \nabla_y I'_{i-l}||, \tau_2\right].$$

As for stereo, we filter C using I as guidance image and obtain the flow field in a Winner-Takes-all manner.

Occlusion detection and filling: We apply the same left-right cross checking procedure as in stereo to find



(a) Zoom of left stereo image (b) Asymmetric filter weights for yellow marked pixel in (a) (c) Asymmetric disparity map (d) Symmetric filter weights for yellow marked pixel in (a) (e) Symmetric disparity map

Fig. 6: **Symmetric stereo matching.** The pixel, which is marked in (a) by a yellow cross, is close to an occlusion region, which is marked by a red band. When using the asymmetric approach, occluded pixels obtain high support weights as illustrated in (b). (Bright intensities correspond to high weights.) Hence matching costs computed in the occluded region influence the disparity assignment of the yellow pixel. This leads to disparity artifacts as shown in (c). Image (d) shows the support mask of our symmetric approach where we use both color images to compute the support weights. Note that pixels of the occluded region now have very small influence in the matching cost aggregation for the yellow pixel. Hence, our symmetric approach improves disparity reconstruction for pixels close to occluded regions as shown in (e).

unreliable flow estimates caused mainly due to occlusions[7]. To fill the invalidated pixels with meaningful flow vectors, we cannot simply assign the flow vector with the lowest magnitude of the spatially closest pixels (as we have done for stereo). This is because objects with a smaller flow magnitude can occlude objects with higher flow magnitude. Therefore, we use a weighted median filter to fill the occluded regions based on their color similarity to the visible flow regions. In detail, we apply a weighted median filtering with weights as in eq. (6) to the occluded pixels. The windows of the median filter overlap the valid regions. Hence, we can propagate the flow vectors into the invalid image parts. If the invalidated region is larger than the size of the window used in weighted median filtering, some pixels will not be assigned to any flow vector. In such situations, we iterate the median filtering procedure to incrementally fill invalid pixels.

Sub-pixel precision: To find sub-pixel accurate flow vectors, we simply upscale the cost volume in label dimension and derive the image colors at sub-pixel positions via bicubic interpolation. Hence, this process increases the runtime. In practice, we found that smoothing the final flow vectors with the guided filter can compensate for a lower upscaling factor[8]. We empirically found that an upscaling factor of 4 gives visually pleasing results. However, in this paper, we apply an upscaling factor of 8 to demonstrate the best possible performance.

Alternative - symmetric flow: Similar to stereo, it is possible to formulate a symmetric approach for optical flow computation. We tested this approach and report the results in section 5.2. Practically, we found a slight overall performance improvement over the asymmetric approach.

7. In optical flow literature this process is called forward-backward consistency check.

8. An alternative to achieve sub-pixel precision is to upscale the final flow vectors with the joint bilateral filter or the guided filter.

4.3 Interactive Image Segmentation

In interactive image segmentation the labels encode whether a pixel belongs to the foreground F or the background B, thus $\mathcal{L} = \{F, B\}$. For initialization, the user assigns parts of the image to foreground and background.

Cost computation: From the user assignments, we build fore- and background color histograms denoted as θ^F and θ^B, which sum up to 1. Each histogram has K bins and we denote the bin into which pixel i falls with $b(i)$. We can also use a bounding box as input, where the pixels outside the box build θ^B and all pixels inside the box build θ^F as in [44]. Then the cost volume is given by:

$$C_{i,l} = 1 - \theta^l_{b(i)}; l \in \mathcal{L}. \tag{8}$$

For binary labeling problems we can reduce the cost volume $C_{i,l}$ to a two-dimensional cost surface C_i which denotes the costs of a pixel to belong to foreground:

$$C_i = 1 - \theta^F_{b(i)}/(\theta^F_{b(i)} + \theta^B_{b(i)}). \tag{9}$$

If a pixel i has been assigned to fore- or background by the user, C_i is set to 0 or 1, respectively. After filtering the cost surface, a pixel i is assigned to foreground if $C_i < 0.5$ and assigned to background otherwise. When using bounding boxes, we iteratively update the color models as in [38]. In practice, we achieved good results using 5 iterations.

To account for semi-transparent pixels along the object boundary in an efficient manner, we filter the computed binary mask with the guided filter. This has been shown [8] to approximate an alpha matting method.

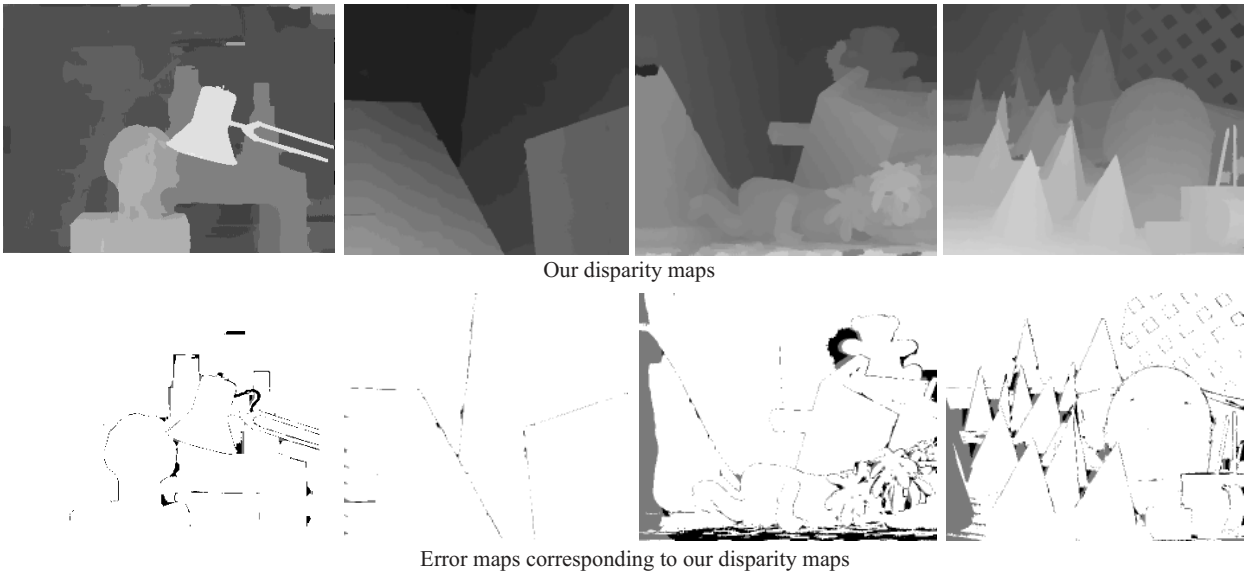


Fig. 7: **Our results on the four evaluation Middlebury images.** All results are generated using constant parameter settings. 1^{st} row shows the disparity maps computed by our algorithm. 2^{nd} row shows a comparison against the ground truth maps. Disparity errors larger than one pixel in non-occluded regions are plotted as black pixels, while gray pixels correspond to errors in occluded areas.

5 Experimental Results

In this section, we experimentally demonstrate that our method is capable to generate state-of-the-art results for stereo matching, optical flow and interactive image segmentation. We use the following, *same* constant parameter settings for stereo and optical flow to generate all of our results: $\{\mathrm{r}, \epsilon, \alpha, \sigma_s, \sigma_c, \tau_1\} = \{9, 0.01^2, 0.9, 9, 0.1, 0.028\}$. This demonstrates the robustness of our method. The only exception is the truncation value τ_2 of the matching costs in eq. (6) and (8). This value depends on the signal-to-noise ratio of an image [9] as well as on the size of the occluded regions. Thus we use $\tau_2 = 0.008$ for stereo and $\tau_2 = 0.016$ for optical flow.

Interactive image segmentation is a very different problem; hence we found different, constant parameter settings (i.e. more smoothing) work well: $\{r, \epsilon\} = \{11, 0.2^2\}$. Note that for interactive segmentation the only further parameter we use is $K = 32$, which defines the number of bins of the color histogram.

We implemented our method on the graphics card using CUDA. All experiments were conducted on an Intel Core 2 Quad, 2.4GHZ processor and an NVIDIA GeForce GTX480 graphics card with 1.5GB of memory. Our approach takes about 2.85ms to filter a 1Mpix image. Thus we can process about 351 labels per second in a 1Mpix image. Problem specific timings are reported below. A (slower) Matlab implementation of our stereo method is available on our webpage.[9]

9. http://www.ims.tuwien.ac.at/research/costFilter/index.html

5.1 Stereo Matching

To evaluate our stereo approach we conducted experiments on (i) the Middlebury stereo benchmark [18], (ii) real outdoor images recorded by ourselves and (iii) a live video stream in order to show that our approach works well for real-time application scenarios.

First, we discuss results on the Middlebury stereo benchmark [18] which provides 35 image pairs with known ground truth. A subset of these images (four images) are used to compare over 110 stereo matching methods in the Middlebury online evaluation table. The disparity maps generated by our approach on these four test images are shown in figure 7. We can see that our method generates high quality disparity maps and preserves the boundaries of thin objects. We report quantitative results that we have taken from the Middlebury online evaluation system for the four test images in table 1.[10] The table plots Middlebury ranks and average percentages of bad pixels (rightmost column of the Middlebury online table) using the Middlebury default error threshold of 1. Our approach (CostFilter) takes rank 16 out of more than 110 methods.

We also tested our alternative symmetric stereo approach described in section 4.1 (see entry "CostFilter (sym.)" in table 1). We found that the average error for the four test images is reduced from 5.55% to 5.35% leading to an improvement in the ranking from position 16 to position 12. The entry "CostFilter(w/o

10. Note that table 1 shows only selected approaches from the Middlebury table that compete with our algorithm in terms of method (local techniques) and quality, similar to the comparison in [6].

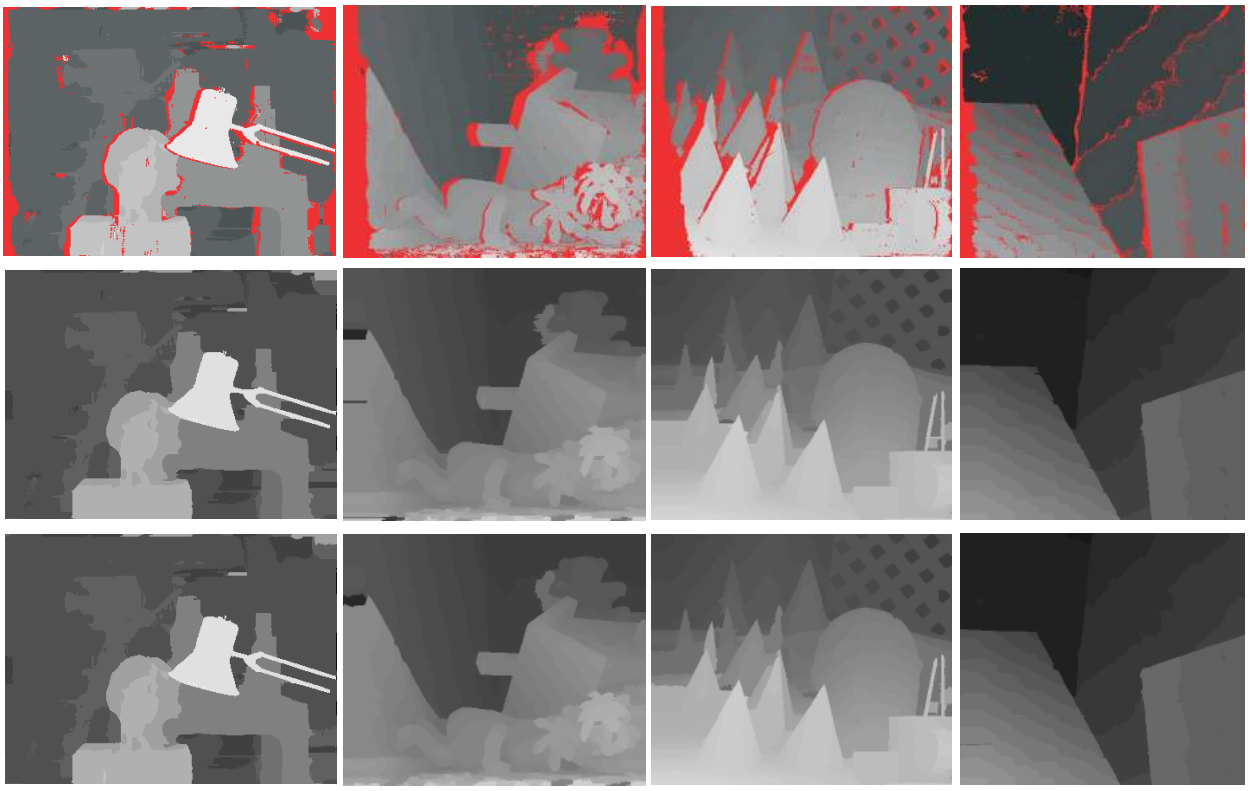


Fig. 8: **Effect of post-processing.** Post-processing was applied only in invalidated regions (i.e. red regions that fail the left-right cross checking in the top row). 1^{st} row shows our disparity maps with invalidated pixels marked in red. 2^{nd} row shows disparity maps after scanline-based filling. 3^{rd} row shows disparity maps after applying weighted median filtering for invalid pixels marked red in top row.

post-processing)" in table 1 shows the error rate of our method without post-processing (i.e. without filtering the invalid regions by a weighted median filter). We see that the average error slightly increases, which results in a slightly worse overall rank 23. This can largely be attributed to a higher error percentage in occluded regions.

In figure 8, we show the effect of our post-processing method. The top row highlights those pixels in red which were invalidated by the left-right checking. We observe that most invalid pixels are located in the proximity of object boundaries and hence seem to correspond well to the occluded areas in these image pairs. The middle row shows the result of our simple scanline-based disparity filling procedure. Scanline streaking artifacts produced by this filling scheme are then removed using the bilateral median filter. We show the results of this step in the bottom row. We would like to emphasize again that our post-processing filter only affects invalid pixels (red pixels in the top row of figure 8) and hence influences the results only marginally. We note that it would be counter-productive to apply the filter on the whole disparity map, as the resulting smoothing may lead to a loss of disparity details.

An important observation from the Middlebury online table is that, to our knowledge, our stereo algorithm is top-performer among all *local* stereo methods, which are listed in the ranking. In particular, as can be seen in table 1, it can outperform the geodesic approach (GeoSup), the original adaptive support weight algorithm of [5] (AdaptWeight) and a fast approximation of the latter technique (DCB-Grid). To understand why our method performs better than [5], we plugged their support weights into our algorithm. Hence, we use the same matching costs, occlusion handling and post-processing methods as in our algorithm. We tuned the parameters of the resulting algorithm to optimize the Middlebury ranking. This approach (CostFilter (Y & K Weights [5])) ranks behind our guided filter-based algorithm, i.e. on rank 24 in contrast to rank 16. This suggests that the guided filter and the joint bilateral filter are both well suited for stereo matching. However, our guided filter-based approach seems to perform slightly better in terms of quality and considerably better in terms of computational efficiency, as we discuss later.

In table 2 we compare the performance of local stereo matching methods on all 35 Middlebury ground truth images. This test is important not only because it is a richer test set, but also because the four Middlebury evaluation images are close to being "solved", which means that a minor error in the disparity map leads to a large difference in the

Algorithm	Rank	Avg. Error (%)
CostFilter (sym.)	**12**	**5.35**
CostFilter	**16**	**5.55**
GeoSup [10]	21	5.80
CostFilter (w/o post-processing)	23	5.77
CostFilter (Y & K Weights [5])	24	5.86
AdaptWeight [5]	48	6.67
DCBGrid [6]	90	10.9

TABLE 1: **Rankings for selected local stereo methods.** We are the best performing local method and overall in the top 15.

Algorithm	Rank	Avg. Error (%)
CostFilter (sym.)	**1**	**8.16**
CostFilter	**2**	**8.36**
CostFilter (w/o post-processing)	3	8.51
GeoSup [10] (our GPU impl.)	4	9.21
CostFilter (Y & K Weights [5])	5	9.28
AdaptWeight [5] (our GPU impl.)	6	15.69
DCBGrid [6]	7	16.73

TABLE 2: **Evaluation for selected local stereo methods on all 35 Middlebury stereo image pairs.** The second column represents the rank of each method according to its average error measured on all 35 image pairs.

ranking. Since only four of the 35 test scenes are used for the online evaluation, no results are available for competitors to our algorithm. To obtain results of competing techniques, we have performed a GPU-based reimplementation of [5] (AdaptWeight [5](our GPU impl.)) as well as [10] (GeoSup [10] (our GPU impl.)) and used the GPU-based implementation of [6] obtained from the authors of the respective paper (DCBGrid). To measure matching performance, we compute the percentage of pixels that have a disparity error larger than one pixel in non-occluded regions and build the average of this measure over all 35 test images. We plot the corresponding values in table 2. It can be seen that the ranking of the selected methods in table 1 is almost the same as in table 2, i.e. our symmetrical method is the winner and slightly outperforms our asymmetrical technique. Note that in this test, the parameters for each method were first tuned to give best results for the four images which are used in the online Middlebury evaluation. These settings were then used for all 35 test pairs, shown in table 2, where the 4 evaluation images are a subset.

Let us now focus on computational efficiency of the methods in table 2. As stated above, we have implemented all of these methods on the GPU. To keep the runtime comparison fair, we have run them on the same computer, whose specifications were given at the beginning of this section. We measure computational performance in a unit of Million Disparity Estimations per second (MDE/s), as done previously in e.g. [45]. The MDE/s measure of an algorithm for a stereo test pair is computed as:

$$\mathrm{MDE/s} = (imgW \times imgH \times \mathcal{D} \times \mathrm{FPS})/10^6. \quad (10)$$

Here, imgW and imgH are the image width and height in pixels, $\mathcal{D}$ represents the number of allowed disparities and FPS is the number of frames per second.[11] Hence, a larger MDE number means a better performing system.

In figure 9 we plot the MDE/s performance of various methods with respect to different sizes of the support window. As can be seen, our method is the only one for which the runtime is independent of the support window size. The runtime performance of all other algorithms, except of DCBgrid (see discussion below), increases with larger window sizes, which is a disadvantage considering that large window sizes are needed to give good results. We have marked optimal window sizes for each method using black arrows in figure 9.[12] Note that for a window size of 35×35 our algorithm "CostFilter" is more than eight times faster than our GPU implementation of the asymmetric joint bilateral filter "CostFilter (Y & K Weights)".

In figure 9 it is worth to note that there is no considerable difference in runtime between "CostFilter (Y & K Weights)" and "GeoSup (our GPU impl.)", although computation of geodesic weights is more expensive than those of the joint bilateral filter. This is for the following reason. To optimize runtime performance, we precompute the weight masks (i.e., geodesic or joint bilateral filter masks) at the beginning of the algorithm. When aggregating the matching costs at each disparity level, we can then look-up the precomputed mask at each pixel. Since the aggregation process, which happens for each possible disparity, is considerably more expensive than the precomputation, which happens only once, the difference in runtime between "CostFilter (Y & K Weights)" and "GeoSup (our GPU impl.)" is negligible. Note that this trick does not work for "AdaptWeight (our GPU impl.)", because it is a symmetrical formulation in which weight masks change depending on the disparity under consideration. Hence our reimplementation of [5] spends a considerable amount of time on computing the support weights and hence is the slowest algorithm among those evaluated. For DCBGrid [6] there is no explicit window size. We plot the average MDE/s for varying σ_s.[13] DCBGrid is the algorithm that is closest to ours in terms of runtime. However,

11. Note, MDE/s can be converted into frames per second by: $\mathrm{FPS} = (\mathrm{MDE/s} \times 10^6)/(imgW \times imgH \times \mathcal{D})$.

12. Here, optimal means the window size reported in the corresponding paper.

13. σ_s in the DCBGrid [6] is the spatial sampling rate. It can be considered as being equivalent to the window size in naive aggregation methods because it controls the amount of smoothing. The number of grid cells is inversely proportional to the sampling rate: a larger σ_s yields a smaller number of grid cells and requires less memory. This explains why the average MDE/s for this method increases with the increasing values of σ_s.

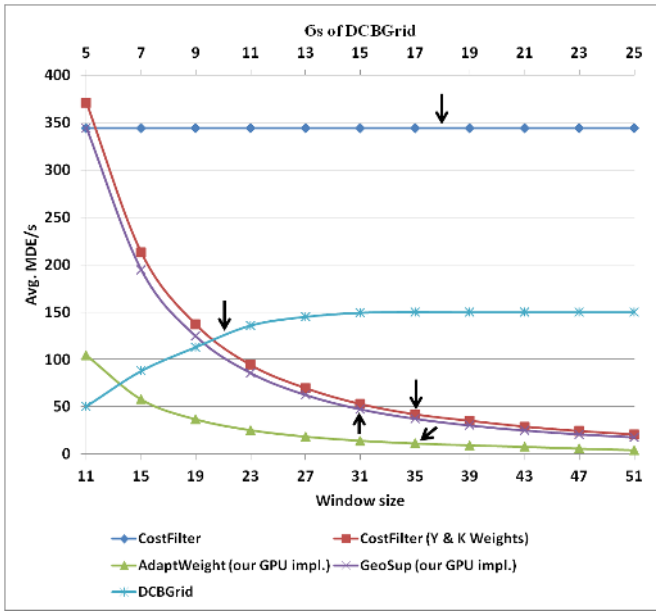


Fig. 9: **Efficiency comparison of our method in comparison to competitors.** Note that large MDE/s score means better performance. The optimal window size for each method is marked by an arrow. Our algorithm is independent of the window size.

Algorithm	Rank	Avg. Error [%]	Avg. MDE/s
CostFilter	**16**	**5.55**	**343.9**
PlaneFitBP	20	5.78	9.35
RT-ColorAW	46	6.55	36.87
RealTimeABW	54	7.9	4.42
RealtimeBFV	59	7.65	112.47
RealtimeBP	61	7.69	20.89
FastAggreg	65	8.24	18.90
OptimizedDP	74	8.83	10.62
RealtimeVar	74	9.05	21.28
RTCensus	79	9.73	660.90
RealTimeGPU	80	9.82	53.91
ReliabilityDP	83	10.7	35.03
DCBGrid	90	10.9	130.64

TABLE 3: **Rankings of all real-time algorithms of the Middlebury online database.** Our algorithm is the best-performing real-time algorithm in terms of quality and the second best-performing real-time algorithm in terms of computational efficiency. Note that large MDE/s score means better performance.

it is considerably inferior in terms of matching quality (see tables 1 and 2) and results get worse with larger σ_s, in which case the algorithm would be attractive in terms of runtime.

Table 3 shows the ranking of our algorithm compared to all other real-time algorithms in the Middlebury online ranking. The right-most column in the table shows the average MDE/s computed for all algorithms. The values are taken from the respective papers.[14] Note that performance values were obtained using different hardware and hence table 3 can only give a rough idea about the computational efficiency of the listed methods. From this table we observe that there is only one method (RTCensus [46]) which has a larger MDE/s value compared to ours (hence, is faster than our method). However, there is a large gap in terms of ranking of this method and our algorithm.[15] To summarize, our method is currently the best performing real-time method in the Middlebury table and shows an excellent tradeoff between speed and quality of results.

We also show results of our algorithm when applied to real outdoor scenes. We captured outdoor images with a consumer stereo camera (Fujifilm FinePix Real3D W1). The images were rectified manually after recording. The input images and corresponding disparity maps generated by our method for two complex outdoor scenes are shown in figure 10. Note that we neither applied occlusion filling nor post-processing on the disparity maps. Instead, invalidated pixels were set to disparity zero (visualized in black). Our method seems to work very well although the input images contain many challenges including untextured regions (e.g. walls), thin structures (e.g. bicycles) or different illumination conditions in left and right image.

Finally, we show results of our system when processing live video streams that we acquired using a Point Grey Bumblebee stereo camera. We match the 640 $\times$ 480 pixel images of the video stream with allowed disparities ranging from 0 to 40. Our live system runs at 33.3 fps (excluding the computational overhead for rendering and rectification). Figure 11 shows two disparity maps computed for selected frames captured by our live system. We believe that the quality of calculated disparity maps is high, especially when considering the speed of our method. We also found that although we do not enforce temporal consistency, the disparity maps appear temporally smooth, i.e. there is only little disparity flickering visible. This is a good indicator that our method is robust. Please also see the video in the supplementary material for a demonstration.

5.2 Optical Flow

We evaluate our approach on the Middlebury flow benchmark [41]. The benchmark comprises an evaluation dataset of 8 images with hidden ground truth flow as well as 8 training scenes with publically available ground truth flow. Table 4 shows the results of our method on the evaluation dataset[16]. To be more

14. If different implementations are published, the fastest is reported.

15. This method mainly relies on a very simple way for cost aggregation without any process for weight computation, which usually requires large processing time. Moreover, this method is limited to grayscale input images.

16. Note that table 4 shows only selected approaches from the Middlebury table that compete with our algorithm in terms of method and quality.

Left input image (1) Our disparity map Left input image (2) Our disparity map

Fig. 10: **Stereo results for real outdoor scenes.** Two stereo images captured by ourselves with a consumer stereo camera. Here we only show the disparity maps without occlusion filling and post-processing (invalidated pixels are shown in black).

Left input image (1) Our disparity map Left input image (2) Our disparity map

Fig. 11: **Two sample frames captured by our live system and their corresponding disparity maps.** We see that thin structures marked by arrows are correctly preserved.

precise, we show the rank of our method computed over all 8 evaluation datasets, as well as detailed results for three challenging datasets "Schefflera", "Grove" and "Teddy". We can see that overall, our approach ranks on the 10^{th} and 13^{th} rank, with respect to the Average Angular Error (AAE) and Average Endpoint Error (AEE), out of almost 60 methods. This performance is comparable to the method of Werlberger et al. (NL-TV-NCC) [27] that uses adaptive support weights in a variational framework. However, our method has several advantages over its competitors including [27]. Firstly, our approach outperforms most other methods on scenes with thin structures and strong motion discontinuities such as in "Schefflera", "Grove" and "Teddy", where we achieve an average rank of 2.2 (not shown in the table) and rank 1 on the "Teddy" scene (see details in table 4 and figure 14). Secondly, our approach, which uses fixed parameter settings, can handle scenes with large displacements, which is difficult for approaches such as [27] that are restricted by their coarse-to-fine framework. Finally, the simplicity of our method is another advantage over many other approaches, which often require a large number of parameters to be tuned, e.g. number of pyramid levels and interpolation strategy.

Our approach performs less well on the "Wooden" and "Yosemite" sequence. This is because in the "Wooden" sequence our algorithm assigns wrong flow values to a shadowed region. Although the difference to the top performers in terms of error appears to be small, this has a large effect in the ranking. The artificial "Yosemite" sequence contains many untextured regions where the data term is unreliable. Variational methods smoothly interpolate over these regions while our method misinterprets them as motion discontinuities. We observed that this is less of a problem in natural high-resolution images where the data term gives useful information even in regions that appear homogeneous at a first glance.

The total runtime of our method for the "Urban" scene, which has the largest label space (640×480 pixel with $30,000$ labels at a sub-sampling factor of 8) is approximately 55.31 seconds. In figure 12 we study the effect of using different sub-sampling factors. As can be seen from this plot, smaller sub-sampling factors give results of comparable quality at considerably lower runtimes. For example, an upscale factor of four has a runtime of 13.52 seconds for the "Urban" sequence.

For completeness, we show the performance of our optical flow algorithm for the Middlebury training images in figure 13. We use the Middlebury color coding to visualize the computed optical flow field. The measured AEE and AAE are shown at the bottom of each image. Similar to the evaluation dataset our method recovers fine structures very well and preserves motion discontinuities. The reader is referred to the supplementary material in order to see the complete result.

Large displacement flow results. An important ad-

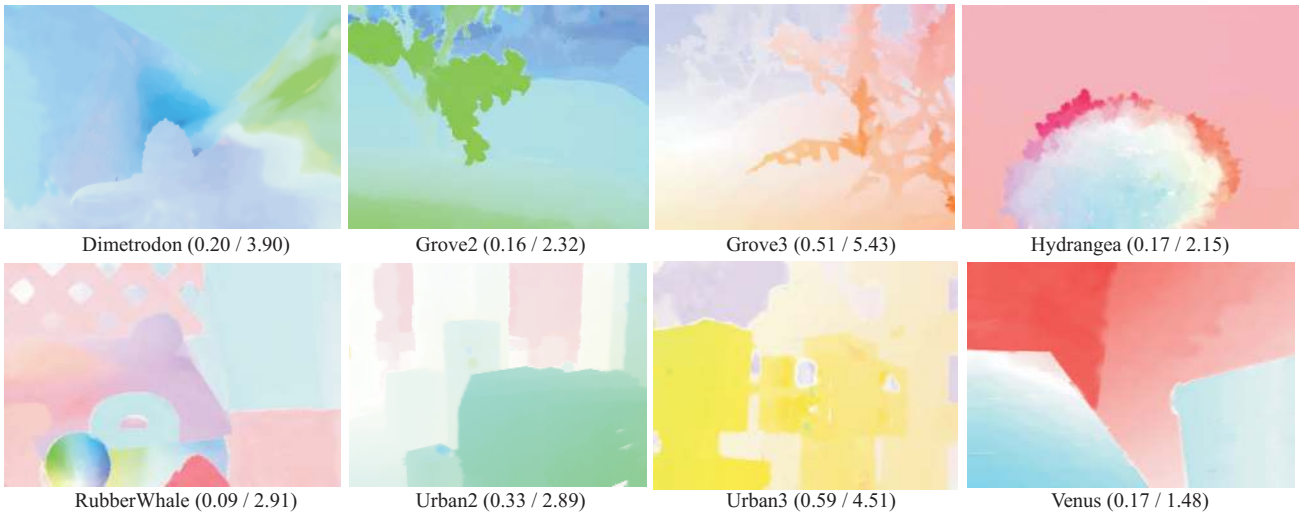


Fig. 13: **Optical flow results.** Results for the training sequences of the Middlebury benchmark dataset. The respective AEE and AAE are given in parentheses (AEE / AAE). Color coding as in [41].

Method	Angular Error				Endpoint Error				Time
	Rank	Schefflera	Grove	Teddy	Rank	Schefflera	Grove	Teddy	(sec)
Layers++	2	(1,1,12)	(1,1,1)	(2,1,6)	2	(1,1,8)	(1,1,2)	(1,1,7)	18206
Classic+NL [28]	5	(9,7,16)	(5,3,4)	(4,4,11)	5	(10,10,13)	(3,3,5)	(3,2,12)	972
MDP-Flow [31]	8	(5,5,22)	(9,8,15)	(27,29,30)	7	(4,5,17)	(9,9,15)	(29,32,28)	188
CostFilter	**10**	**(2,2,4)**	**(2,2,3)**	**(1,2,1)**	**13**	**(2,2,8)**	**(2,2,1)**	**(1,4,3)**	**55.31**
CostFilter (sym.)	**11**	**(4,3,2)**	**(2,3,1)**	**(1,4,1)**	**12**	**(4,4,4)**	**(3,3,1)**	**(1,4,1)**	**110.65**
OFH	12	(23,25,5)	(11,11,21)	(12,17,8)	8	(20,25,4)	(18,17,21)	(12,17,12)	620
NL-TV-NCC [27]	13	(16,16,2)	(26,31,11)	(11,13,10)	10	(16,16,2)	(14,15,9)	(10,11,2)	20
DPOF [35]	17	(4,4,13)	(13,15,20)	(9,6,5)	16	(4,4,17)	(5,5,3)	(7,4,1)	287
ACK-Prior [36]	18	(6,9,3)	(19,13,27)	(19,12,14)	20	(6,6,2)	(15,14,18)	(27,18,23)	5872

TABLE 4: **Optical flow evaluation on Middlebury.** Overall our method gives rank 10 and 13 with respect to the angular error and endpoint error, respectively. Our approach works particularly well for the challenging "Schefflera", "Grove" and "Teddy" sequence. Note that fine structures and strong motion discontinuities cannot be handled by many competitors. We report the ranks for these sequences in brackets (all, disc, untext). Runtime is given for the "Urban" sequence (as requested by [41]), which has the largest label space.

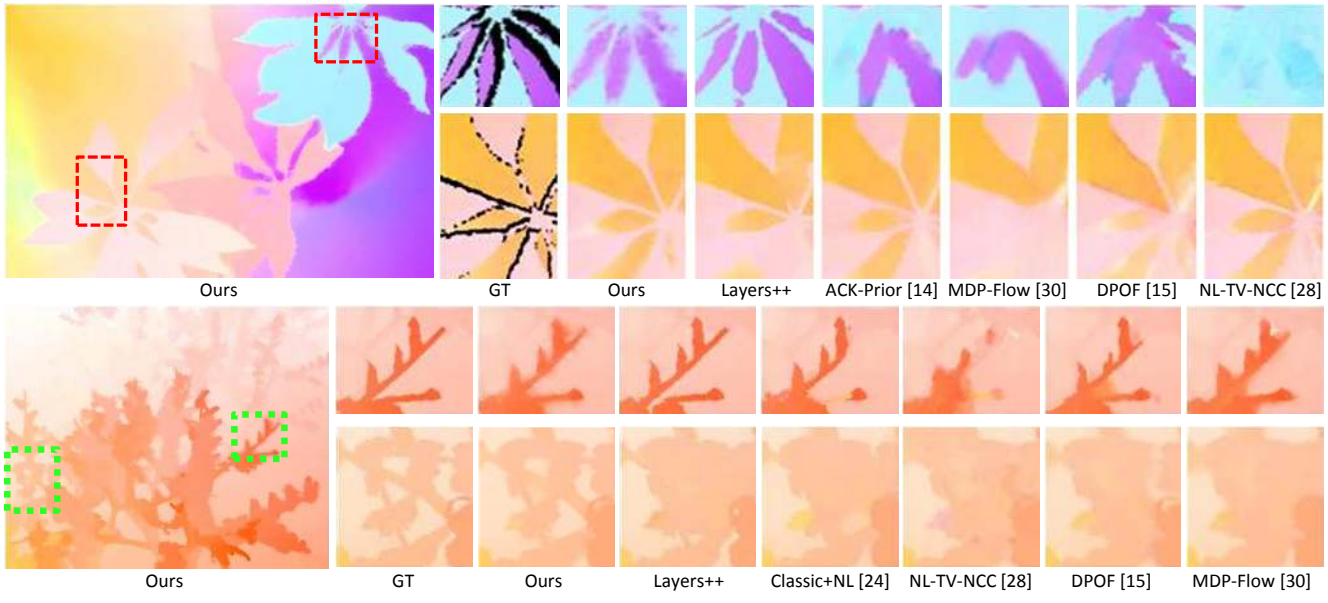


Fig. 14: **Detailed flow results.** Comparison of two sequences with thin structure (upper part: "Schefflera" scene; lower part: "Grove" scene), where many competitors fail to preserve flow discontinuities. (We boosted the colors in the second row from the top for better visualization.) Color coding as in [41].

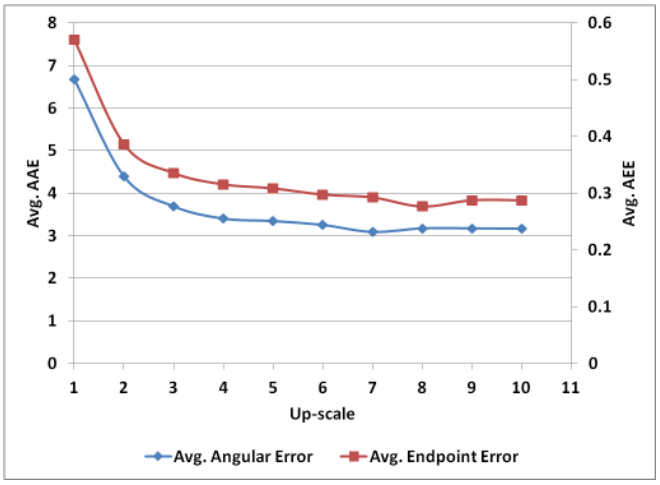
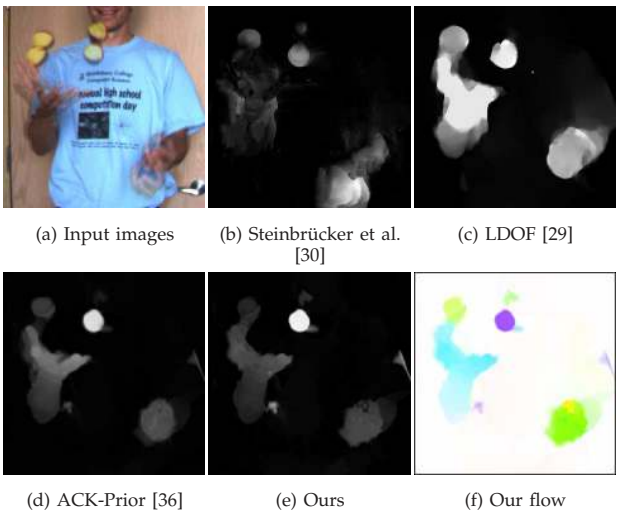


Fig. 12: **Effect of subsampling factor on the average AAE and AEE errors.** Here we used the Middlebury training images with their ground truth flow values as reference.



(a) Input images (b) Steinbrücker et al. [30] (c) LDOF [29]

(d) ACK-Prior [36] (e) Ours (f) Our flow

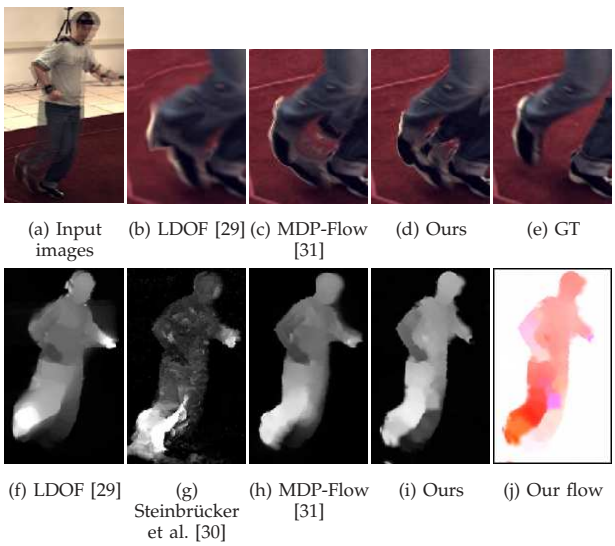
Fig. 15: **Large displacement flow (Beanbags).** (b-d) Motion magnitude for different methods specialized on large displacement flow. (e) Motion magnitude for our method. (f) Our flow vectors with the color coding as in [41]. Our method nicely recovers the shape of the hand.

vantage of our method is that it can also handle large displacements without the need of changing any parameters (see figures 15 and 16 for a comparison to other methods). Our approach generates results that are visually comparable, or better, than methods which are specifically tailored on large displacement flow and are not top performing for small displacements (exceptions are [31], [36]).

Symmetric Approach. We also tested a symmetric optical flow formulation of our approach, see table 4. In contrast to stereo matching, the symmetric optical flow formulation does not seem to have a big effect on the results. For the AAE measure, the symmetric approach even performs worse than our asymmetric formulation (rank 11 in comparison to rank 10). However, the AEE error is slightly improved (rank 12 in comparison to rank 13). We also tested the symmetric optical flow approach on the training sequences from Middlebury. We found that the overall improvement is negligible. The average measured AEE error is reduced from 0.28 to 0.27 and the average measured AAE is reduced from 3.2 to 3.1.

5.3 Interactive Image Segmentation

To show that our approach also performs well for image segmentation, we visually compare our results to those of GrabCut [38], figure 17. As user input we either use coarse scribbles/trimaps or a single bounding box. The results are visually comparable at lower runtimes (2.85ms versus approximately 300ms using the graph cut implementation of [39] and 425ms using the graph cut implementation of [47] on a 1Mpixel image). Furthermore, our method gives comparable results to GrabCut [38] on a ground truth database of 50 images [38]. Here, error is measured as the percentage of misclassified pixels in the area not marked by the user. Given the trimap input of



(a) Input images (b) LDOF [29] (c) MDP-Flow [31] (d) Ours (e) GT

(f) LDOF [29] (g) Steinbrücker et al. [30] (h) MDP-Flow [31] (i) Ours (j) Our flow

Fig. 16: **Large displacement flow (HumanEva) [48].** (b-e) Backward warping results using flow of different methods. The tip of the foot is correctly recovered by our method. Note that the occluded portions cannot be correctly recovered by any method. (f-i) The motion magnitude of different methods. (j) Flow vectors with the color coding as in [41].

[38], the error is 5.3% for GrabCut and 6.2% for our method. This shows the potential of our approach to be successfully applied to other vision applications. A video of our real-time segmentation tool is shown in the supplementary material.

(a) Input image with user interaction (b) Our cutout (c) Cutout using Graph Cuts

Fig. 17: **Segmentation results.** (b) Binary segmentation from user input in (a). (c) Result corresponding to a single iteration of GrabCut [38]. For the "Bunny" image (last row), we additionally filtered the cutout masks in (b,c) with the guided filter to obtain a soft alpha matte.

6 Discussion and Future Work

This paper presented a simple, yet powerful filtering approach for solving discrete labeling problems. As mentioned in the introduction, the relationship between filtering-based operations and energy-based optimization schema, for both continuous and discrete label spaces, are to the best of our knowledge not fully understood. One relationship, given in [8], is that the guided filter is one step of a conjugate gradient solver of a particular linear system. We believe that a better understanding of this relationship can lead to fast and even better (iterative) filtering approaches.

Finally, we note that all aforementioned stereo algorithms, as well as ours, assume that pixels within a support window have a constant disparity value. This assumption is violated for slanted surfaces where the support window comprises of pixels that lie on different disparities. Recently, Bleyer et al. [49] proposed to overcome this problem by estimating a 3D plane at each pixel onto which the support region is projected. To find the optimal 3D plane among all possible planes, whose number is infinite, a randomized nearest neighbor search strategy, called PatchMatch [50], was adopted. The work uses the adaptive support weights of [5] for matching cost aggregation to improve results at disparity borders. The PatchMatch algorithm [49] achieves excellent results, however, has the major drawback that it does not operate in real-time. An interesting future research direction is to accelerate this technique by levering the insights of our proposed approach.

Acknowledgments

Asmaa Hosni was supported by the Vienna PhD School of Informatics. Christoph Rhemann and Michael Bleyer received financial support from the Vienna Science and Technology Fund (WWTF) under project ICT08-019. The authors would also like to thank Georg Braun for implementing the proposed interactive segmentation algorithm.

References

[1] O. Veksler, "Stereo correspondence by dynamic programming on a tree," in *CVPR*, 2005.
[2] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers, "A convex formulation of continuous multi-label problems," in *CVPR*, 2008.
[3] M. Klodt, T. Schoenemann, K. Kolev, M. Schikora, and D. Cremers, "An experimental comparison of discrete and continuous shape optimization methods," in *ECCV*, 2008.
[4] P. Gwosdek, H. Zimmer, S. Grewenig, A. Bruhn, and J. Weickert, "A highly efficient GPU implementation for variational optic flow based on the Euler-Lagrange framework," in *CVGPU Workshop*, 2010.
[5] K. Yoon and S. Kweon, "Adaptive support-weight approach for correspondence search," *PAMI*, 2006.
[6] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. Dodgson, "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid," in *ECCV*, 2010.
[7] A. Criminisi, T. Sharp, and C. Rother, "Geodesic image and video editing," *ACM Trans. Graphics*, 2010.
[8] K. He, J. Sun, and X. Tang, "Guided image filtering," in *ECCV*, 2010.
[9] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, 2002.
[10] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann, "Local stereo matching using geodesic support weights," in *ICIP*, 2009.
[11] F. Tombari, S. Mattoccia, and L. D. Stefano, "Segmentation-based adaptive support for accurate stereo correspondence," in *PSIVT*, 2007.
[12] M. Gong, R. Yang, L. Wang, and M. Gong, "A performance study on different cost aggregation approaches used in real-time stereo matching," *IJCV*, 2007.
[13] F. Tombari, S. Mattoccia, L. Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *CVPR*, 2008.
[14] M. Ju and H. Kang, "Constant time stereo matching," in *MVIP*, 2009.
[15] K. Zhang, G. Lafruit, R. Lauwereins, and L. Gool, "Joint integral histograms and its application in stereo matching," in *ICIP*, 2010.
[16] F. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *CVPR*, 2005.
[17] S. Paris and F. Durandi, "A fast approximation of the bilateral filter using a signal processing approach," in *IJCV*, 2009.

[18] "http://vision.middlebury.edu/stereo."
[19] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," in *TCSVT*, 2009.
[20] A. Hosni, M. Bleyer, and M. Gelautz, "Near real-time stereo with adaptive support weight approaches," in *3DPVT*, 2010.
[21] M. Gerrits and P. Bekaert, "Local stereo matching with segmentation-based outlier rejection," in *CRV*, 2006.
[22] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *CVPR*, 2011.
[23] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, "Linear stereo matching," in *ICCV*, 2011.
[24] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz, "Temporally consistent disparity and optical flow via efficient spatio-temporal filtering," in *PSIVT*, 2011.
[25] D. Tschumperlè and R. Deriche, "Vector-valued image regularization with PDE's : A common framework for different applications," in *CVPR*, 2003.
[26] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *ECCV*, 2006.
[27] M. Werlberger, T. Pock, and H. Bischof, "Motion estimation with non-local total variation regularization," in *CVPR*, 2010.
[28] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *CVPR*, 2010.
[29] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *PAMI*, 2010.
[30] F. Steinbrücker, T. Pock, and D. Cremers, "Large displacement optical flow computation without warping," in *ICCV*, 2009.
[31] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," in *CVPR*, 2010.
[32] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *PAMI*, 2004.
[33] V. Lempitsky, C. Rother, and A. Blake, "Logcut - efficient graph cut optimization for Markov Random Fields," in *ICCV*, 2007.
[34] W. Trobin, T. Pock, D. Cremers, and H. Bishof, "Continuous energy minimization via repeated binary fusion," in *ECCV*, 2008.
[35] C. Lei and Y. Yang, "Optical flow estimation on coarse-to-fine region-trees using discrete optimization," in *ICCV '09*.
[36] K. Lee, D. Kwon, I. Yun, and S. Lee, "Optical flow estimation with adaptive convolution kernel prior on discrete framework," in *CVPR*, 2008.
[37] V. Aurich and J. Weule, "Non-linear Gaussian filters performing edge preserving diffusion," in *DAGM*, 1995.
[38] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut - interactive foreground extraction using iterated graph cuts," *SIGGRAPH*, 2004.
[39] A. Criminisi, T. Sharp, and A. Blake, "GeoS: Geodesic image segmentation," in *ECCV*, 2008.
[40] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," in *CVPR*, 2009.
[41] "http://vision.middlebury.edu/flow."
[42] F. Crow, "Summed-area tables for texture mapping," *SIGGRAPH*, 1984.
[43] A. Bruhn and J. Weickert, "Optical flow estimation on coarse-to-fine region-trees using discrete optimization," in *ICCV*, 2005.
[44] S. Vicente, V. Kolmogorov, and C. Rother, "Joint optimization of segmentation and appearance models," in *ICCV*, 2009.
[45] R. Yang and M. Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware," in *CVPR*, 2003.
[46] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, vol. 114, pp. 1180 – 1202, 2010.
[47] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *PAMI*, 2004.
[48] L. Sigal, A. Balan, and M. Black, "HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion," *IJCV*, 2010.
[49] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo - stereo matching with slanted support windows," in *BMVC*, 2011.
[50] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *ACM Transactions on Graphics (SIGGRAPH)*, 2009.