Fast Cost-Volume Filtering for Visual Correspondence and Beyond

Christoph Rhemann[1], Asmaa Hosni[1], Michael Bleyer[1], Carsten Rother[2], Margrit Gelautz[1]
[1]Vienna University of Technology, Vienna, Austria [2]Microsoft Research Cambridge, Cambridge, UK

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-take-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 with 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.

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 (i.e. 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. [26]) usually come at the price of loss in quality, due to less-global optimization schema.

Continuous counterparts to discrete labeling methods are based on convex energy functionals which can be efficiently optimized on the GPU, e.g. [17, 13, 10]. 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, still 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 a local filtering method. 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 this and previous work [31] is that "local smoothing" is able to achieve high quality results. We believe that the reason is the dominance of the 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. 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 [31, 19, 8]. Above all, 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.

This work was supported in part by the Vienna Science and Technology Fund (WWTF) under project ICT08-019.

[1]For some applications it may be possible to show that the smoothness term of a learned energy propagates information only locally. 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.

Let us briefly review the existing ideas of filter-based methods, which are the motivation of our work (details in sec. 2). Apart from [8] all work has concentrated on the application of stereo matching. Yoon and Kweon [31] showed that an edge-preserving bilateral filter on the cost volume can achieve high-accuracy results. Note that the authors of [31] 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 runtime advantage of local over global methods. Richard et al. [19] 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 concept can be leveraged to general labeling tasks, outside stereo matching. Recently, [8] 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.

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* [11], which has the edge-preserving property and a runtime 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 [21]. Figure 1(b) shows an (x, l) slice through this volume for the scanline in figure 1(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(b)). The pixels with the lowest costs are marked red in figure 1(b). The result is noisy, because the solution is not regularized.

To regularize the solution we can aggregate (smooth) the costs over a support window (known as window-based methods in stereo matching). It is known that this aggregation is equivalent to filtering the (x, y) dimensions of the cost volume [21] with a box filter. The result is shown in figure 1(c), where we filtered the cost volume in figure 1(b). The solution with the minimum costs (marked red in figure 1(c)) is smooth but not aligned with the image edges. This is because the box filter overlaps depth discontinuities, which are illustrated with green dashed lines in figure 1. This leads to the well-known "edge-fattening effect" in stereo.

To overcome this problem, we smooth the cost volume with a *weighted* box filter. The weights are chosen such that they preserve edges in the input image. For instance,

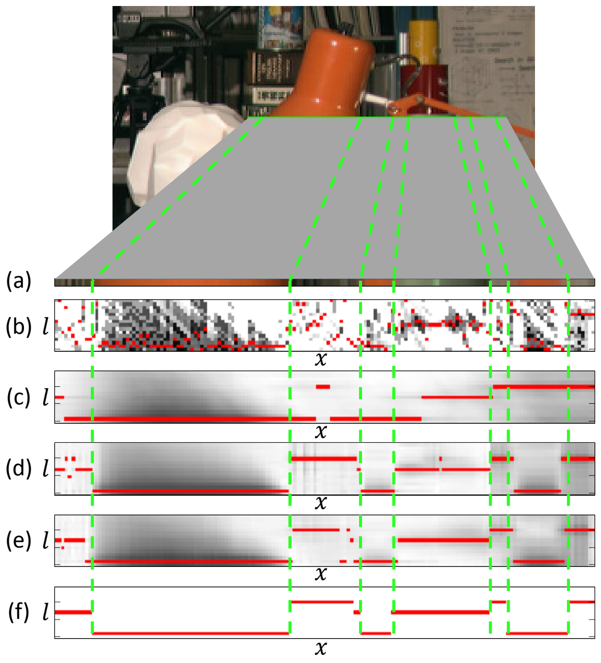

Figure 1. **Cost volume filtering.** (a) Zoom of the green line in the input image. (b) Slice of cost volume (white/black/red: high/low/lowest costs) for line in (a). (c-e) Cost slice smoothed along x and y-axes (y is not shown here) with box filter, bilateral filter and guided filter [11], respectively. (f) Ground truth labeling.

smoothing the cost volume with the bilateral filter (figure 1(d)) gives a spatially smooth solution, which is also aligned with the image edges. Since fast approximations of the bilateral filter degrade the quality, we use the guided filter [11]. Figure 1(e) shows its edge-preserving properties.

Our generic and fast cost-filtering framework, is widely applicable, which we demonstrate 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 an CRF by filtering the label costs. We review those now.

Criminisi et al. [8, 7] 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 if this approach could be extended to multi-label problems such as stereo.[2] Also, the relationship between their morphological operator and e.g. the bilateral filter is not discussed in detail.

In the continuous domain, Tschumperlè 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. [29], the Gaussian kernels were replaced by the bilateral filter. In this line of research, Werlberger et al. [28] and Sun et al. [24] incorporated the adaptive support weights of [31] into a variational approach. In contrast, the goal of our work is to apply local filtering in a discrete framework.

In the real-time stereo approach of Richardt et al. [19] the cost volume is smoothed with a fast approximation of the bilateral filter. Due to this approximation, huge amounts of memory would be required ([19] reported 764 GB) when applied to full-color (e.g. RGB) stereo images. Therefore, [19] is limited to grayscale input images, giving poor results at disparity boundaries. This is also reflected in the Middlebury stereo benchmark [2], where their method is on the 76th rank out of over 90 methods. In contrast, our real-time implementation uses color images and ranks 9th. To increase the quality, [19] proposed an alternative that uses two color channels. However, this method is 13 times slower than their grayscale approach (non-real-time) and still inferior to their re-implementation of [31].

Since we also apply our labeling framework to optical flow and interactive segmentation, we now briefly review those methods most relevant in the context of this work.

For optical flow, a popular approach is to use a variational coarse-to-fine framework that cannot handle large displacements of small objects, as shown in [5, 23]. To overcome this problem, discrete data terms can be integrated into a variational framework (see [23, 5, 30]). 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 subpixel accuracy further increases the label space). Due to these difficulties, discrete approaches, e.g. [4, 16, 15], usually have to trade off search space (quality) against speed. In contrast, our filter based method efficiently deals with the search space and handles both, large displacements and fine small-scale structures.

Related to interactive segmentation, [11] adopted the guided filter to compute a soft-segmentation, so called alpha matte [18]. This is done by filtering a binary segmentation mask, as opposed to the cost volume as in our approach.

3. Cost-Volume Filtering

In this section we describe our labeling framework and apply it to three different vision applications in section 4.

[2]To generate a smooth real-valued output for image editing tasks, such as denoising or cartoonization, [8] smoothes the *input image* (as opposed to a multi-labeled likelihood map).

We consider a general labeling problem, where the goal is to assign each pixel i with coordinates (x, y) in the image I to a label l from the set $\mathcal{L} = \{1, \ldots, L\}$. 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: constructing the cost-volume, filtering the cost volume and 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 at label l is a weighted average of all 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 indexes. 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-take-all manner as

$$f_i = \arg\min_l C'_{i,l}. \quad (2)$$

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 [11], 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 the variance of I in a squared window ω_k with dimensions $r \times r$, centered at pixel k.[3] 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 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 is controlled by the parameter ϵ in eq. (3). If $\sigma^2 \ll \epsilon$ (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: $W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} 1$.

[3]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 indexes i and j.

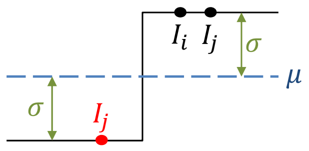

Figure 2. **1D step edge.** We shown μ and σ for a kernel centered exactly at the edge. See text for details. Figure courtesy from [11].

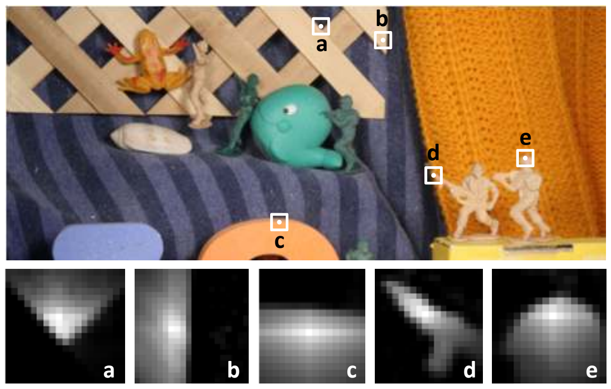

Figure 3. **Filter kernels.** We show kernels of the guided filter with $r = 9$ and $\epsilon = 0.01^2$, at different locations in an image of [1].

The 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 [11] that a weighted average with weights in eq. (3) or (4) can be implemented efficiently on the CPU as a sequence of box filters using the integral imaging technique [9]. We apply the same technique to obtain an even more efficient GPU implementation.

4. Applications

We implemented three different vision applications in our framework. Notice 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 direction. In the x direction, the displacement corresponds to the disparity d ($u = d$) and there is no shift in 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 [5, 6]:

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

Here, ∇_x is the gradient in x direction, α balances the color and gradient terms and τ_1, τ_2 are truncation values.[4]

We then filter the cost volume according to eq. (1) with weights in (4), using I as guidance image. We then compute the disparity map f for image I as per eq. (2).
Occlusion detection and filling: To detect occlusions, we additionally compute the disparity map f' for the right image I' in a similar manner. We mark a pixel in the left disparity map as occluded if the disparity of its matching pixel differs. The occluded pixels are then assigned to the lowest disparity value of the spatially closest non-occluded pixels which lie on the same scanline (pixel row).
Post-processing: This simple occlusion filling strategy can generate streak-like artifacts in the disparity map. To remove them, while preserving the object boundaries, we apply a weighted median filter to the filled pixels. 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 windows sizes thus immense memory and time is required for computing this matrix [5]. Thus we resort to the bilateral filter weights:

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

where σ_s and σ_c adjust spatial and color similarity, K_i is a normalization factor and we use filter dimensions $r_b \times r_b$.
Alternative - symmetric stereo: Our stereo approach can be extended to filter the cost volume, while preserving edges in both input images simultaneously. 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 I_i and I'_{i+l}. The dimensions of I_j, Σ_k, U and μ_k change similarly. We tested this approach but found the average improvement negligible. Thus we do not report results for this approach.

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 direction, respectively.

[4]One could also use a spatially varying α as it was recently proposed in [30]. However, this approach gave worse results on the stereo test images.

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

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

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

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

Occlusion detection and filling: We apply the same left-right cross checking procedure as in stereo to find occluded pixels. For occlusion filling we cannot simply assign the flow vector with the lowest magnitude of the spatially closest pixels. 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 with weights as in eq. (6) to the occluded pixels. The windows of the median filter overlap the non-occluded regions thus can propagate the flow vectors into the occluded image parts.

Subpixel precision: To find sub-pixel accurate flow vectors, we follow [23] and simply upscale the input images using bicubic interpolation. This increases the size of the cost volume in the label dimension (but not in the x and y dimensions) and hence raises the running time. In practice, we found that smoothing the final flow vectors with the guided filter can compensate for a lower upscaling factor.[6] We empirically found that an upscaling factor of 4 gives visually pleasing results, but in this paper we upscale by a factor of 8 to demonstrate the best possible performance.

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 [27]. Then the cost volume is given by:

$$C_{i,l} = 1 - \theta^l_{b(i)}. \quad (8)$$

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:

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

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 [20]. In practice, we achieved good results with 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 [11] to approximate an alpha matting method.

5. Experimental Results

We use the following, *same* constant parameter settings for optical flow and stereo to generate all results: $\{r, \epsilon, \alpha, \sigma_s, \sigma_c, r_b, \tau_1\} = \{9, 0.01^2, 0.9, 9, 0.1, 19, 0.0028\}$. 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 [21] 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: $\{r, \epsilon\} = \{11, 0.2^2\}$. Also, we use $K = 32$ bins for the color-model histogram.

We implemented our method on the graphics card using CUDA. All experiments were conducted on a 2.4GHZ processor and an NVIDIA GeForce GTX480 graphics card with 1.5GB of memory. Our approach takes about 5ms to filter a 1Mpix image. Thus we can process about 200 labels per second in a 1Mpix image. Problem specific timings are reported below (we report runtimes of the full methods, including postprocessing). A Matlab implementation of our stereo method is available on our project website [7].

5.1. Stereo Matching

We evaluated our approach on the Middlebury stereo benchmark [2] and list the results in table 1. Our approach gives excellent results (see figure 4 and supp. material) ranking 9th out of over 90 methods at the time of submission. Even more importantly, we are the best performing local stereo method outperforming even the original implementation of [31] (rank 32). To understand why our method performs better than [31], we plugged their weights into our method. Hence, we use the same matching costs and occlusion handling as in our method. This approach is about 230 times slower than ours but ranks closely behind it on rank 15 (we tuned the parameters of this approach to give best

[6] An alternative to achieve subpixel precision is to upscale the final flow vectors with the joint bilateral filter or the guided filter.

[7] http://www.ims.tuwien.ac.at/research/costFilter/

Method	Rank	Avg. Error (%)	Avg. Runtime (ms)
Ours	9	5.55	65
GeoSup [12]	12	5.80	16000
Plane-fit BP	13	5.78	650
Ours using AdaptWeight [31]	15	5.86	15000
AdaptWeight [31]	32	6.67	8550
Real-time GPU	66	9.82	122.5
Reliability DP	69	10.7	187.8
DCB Grid [19]	76	10.9	95.4*

Table 1. **Stereo evaluation on Middlebury.** Rankings for selected stereo methods. We are the best-performing local approach. *The runtime in [19] was reported before left-right consistency check. For fairness, we report the approx. total runtime here.

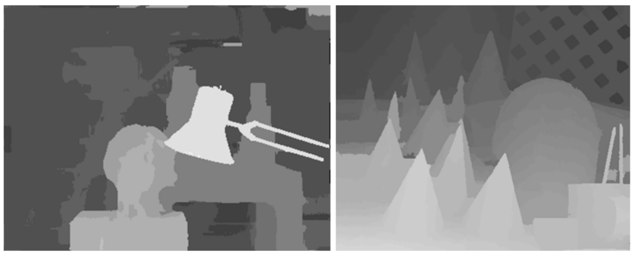
Figure 4. **Stereo results on Middlebury.** Disparity maps for the "Tsukuba" and "Cones" scenes using constant parameters.

possible results). This suggests that the guided filter and the bilateral filter are both well suited for stereo matching.

The average runtime (we report times of the full method including e.g. left-right post-processing) of our method and its competitors are shown in table 1. Our approach works in real-time (approx. $23fps$ on average) and is the fastest on the Middlebury test set. Runtimes of the competing methods are taken from [19].[8]

5.2. Optical Flow

We evaluate our approach on the Middlebury flow benchmark [1] and report the results in table 2. Overall, our approach ranks on the 4^{th} and 6^{th} rank with respect to the angle error and endpoint error out of almost 40 methods at the time of submission. This performance is comparable to the method of Werlberger et al. (NL-TV-NCC) [28] that uses adaptive support weights in a variational method. Our method has several advantages over its competitors. First, our approach outperforms most other methods on scenes with fine details and strong motion discontinuities such as "Schefflera", "Grove" and "Teddy", where we achieve an average rank of 2.2 and rank 1^{st} on the "Teddy" scene (see details in table 2 and figure 5). Second, our approach (using identical parameter settings) can handle scenes with large displacements, which is more difficult for approaches like [28] that are restricted by their coarse-to-fine framework.

[8]Times were measured on different machines but still give a good indication of the computational complexity.

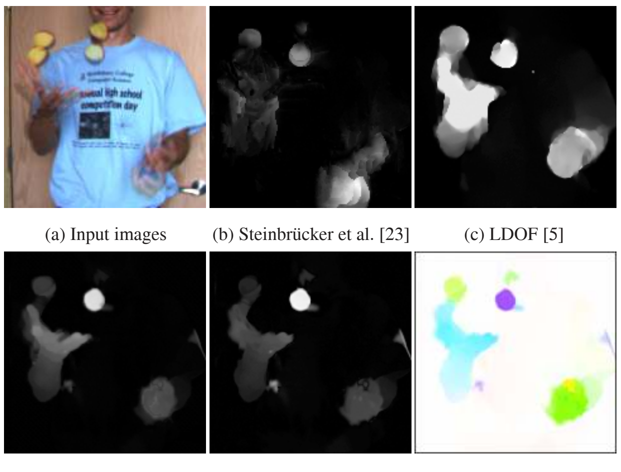

Figure 6. **Large displacement flow (Beanbags).** (b-e) Motion magnitude for different methods. (f) Flow vectors with the color coding as in [1]. Our method nicely recovers the shape of the hand.

Finally, the simplicity of our method is another advantage over many approaches that 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, it has a larger effect in the ranking. In the future, other matching cost functions could be used to overcome this problem. 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 640×480 "Urban" sequence (about $30,000$ labels when using a subsampling factor of 8) was about 90 seconds. In practice we found that much smaller subampling factors give visually comparable results at considerably lower runtimes.

Large displacement flow results: An important advantage of our method is that it also handles large displacements without changing any parameters (see figures 6 and 7 for a comparison). Our approach generates results that are visually comparable or better than methods which are mostly specialized on large displacement flow and do not perform best for small displacements (exceptions are [30, 14]).

5.3. Interactive Image Segmentation

To show that our approach also performs well for image segmentation, we visually compare it to GrabCut [20]

Method	Angle Error				Endpoint Error				Time
	Rank	Schefflera	Grove	Teddy	Rank	Schefflera	Grove	Teddy	(sec)
Layers++	1	(1,1,9)	(1,1,1)	(2,1,3)	1	(1,1,6)	(1,1,2)	(1,1,6)	18206
Classic+NL [24]	2	(6,5,12)	(3,3,3)	(3,3,6)	2	(7,7,10)	(3,3,4)	(3,2,7)	972
MDP-Flow [30]	3	(4,4,15)	(4,4,9)	(19,20,23)	3	(3,4,11)	(5,5,8)	(20,22,19)	188
Ours	4	(2,2,3)	(2,2,2)	(1,2,1)	6	(2,2,6)	(2,2,1)	(1,3,3)	90
OFH	5	(15,16,4)	(6,6,14)	(6,10,4)	4	(13,16,3)	(11,10,14)	(6,10,7)	620
NL-TV-NCC [28]	6	(11,11,1)	(18,21,6)	(5,7,5)	5	(11,11,1)	(7,8,5)	(5,5,2)	20
DPOF [15]	9	(3,3,10)	(8,9,13)	(4,4,2)	9	(3,3,11)	(4,4,3)	(4,3,1)	287
ACK-Prior [14]	10	(5,6,2)	(12,7,18)	(12,6,9)	12	(5,5,1)	(8,7,11)	(18,11,16)	5872

Table 2. **Optical flow evaluation on Middlebury.** Our approach works well for the challenging "Schefflera", "Grove" and "Teddy" sequences. The 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 [1]), which has the largest label space.

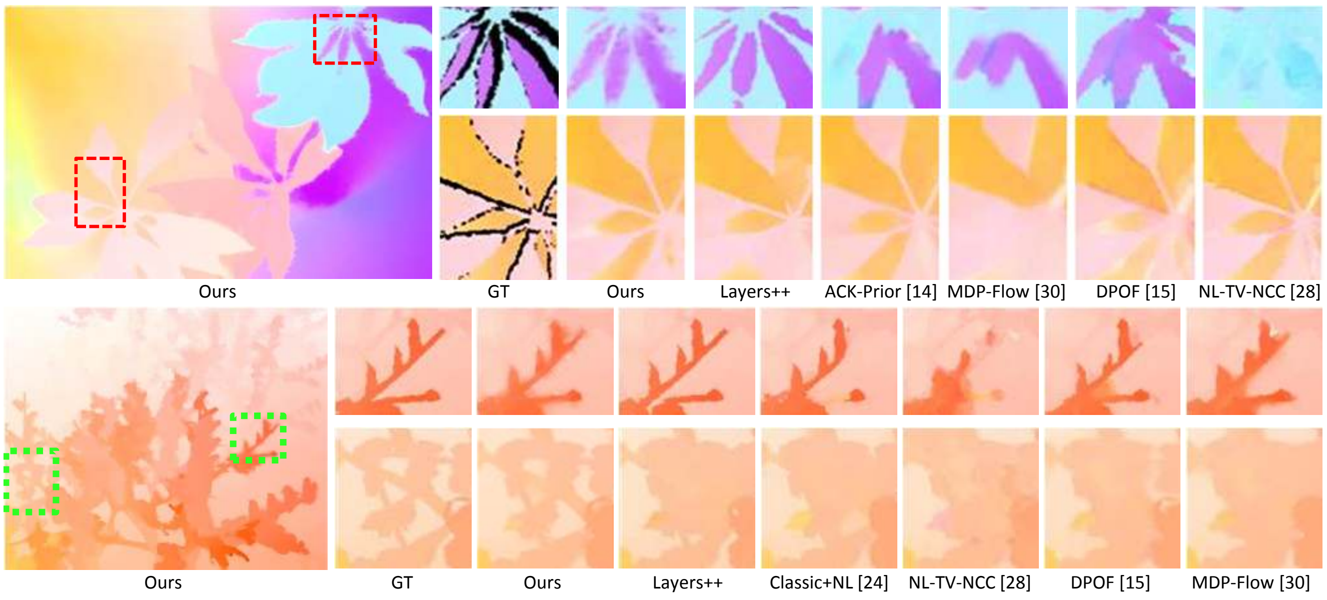

Figure 5. **Detailed flow results.** Comparison for two fine structured sequences (upper part: "Schefflera" scene; lower part: "Grove" scene), where many competitors fail to preserve flow discontinuities. (We boosted the colors in the second upper row for better visualization.)

in figure 8. As user input we either use coarse scribbles or a single bounding box. The results are visually comparable at lower runtimes (5ms vs. about 300ms (425ms) using the graph cut implementation of [7] ([3]) on a 1Mpixel image). Furthermore, our method gives comparable results to GrabCut [20] on a ground truth database of 50 images [20]. The error (percentage of misclassified pixels in the regions not marked by the user) using trimap input 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.

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 continuous and discrete models, such as belief propagation are, to the best of our knowledge, not fully known. One relationship, given in [11], is that the guided filter is one step of a conjugate gradient solver of a particular linear system. We believe that a better understanding can lead to fast and even better (iterative) filtering approaches.

References

[1] http://vision.middlebury.edu/flow.
[2] http://vision.middlebury.edu/stereo.
[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 2004.
[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2004.
[5] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, to appear.
[6] A. Bruhn and J. Weickert. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *ICCV*, 2005.
[7] A. Criminisi, T. Sharp, and A. Blake. GeoS: Geodesic image segmentation. In *ECCV*, 2008.

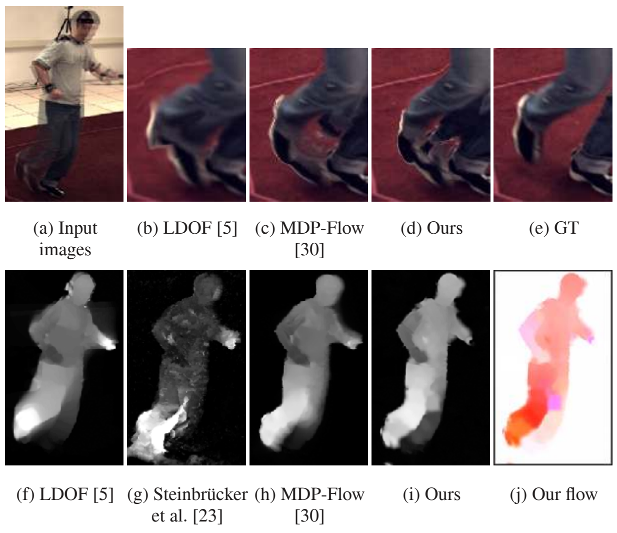

Figure 7. **Large displacement flow (HumanEva) [22].** (b-e) Backward warping results using flow of different methods. The tip of the foot is correctly recovered by our method. Note that the occluded portions cannot be correctly recovered by any method. (f-i) The motion magnitude of different methods. (j) Flow vectors with the color coding as in [1].

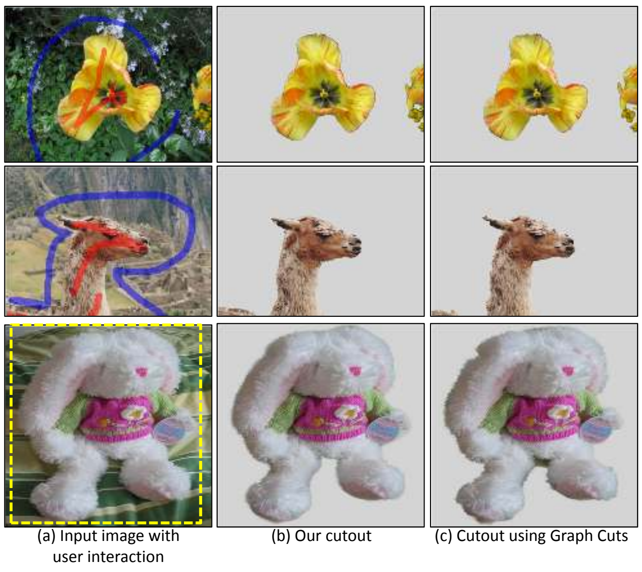
(a) Input image with user interaction
(b) Our cutout
(c) Cutout using Graph Cuts

Figure 8. **Segmentation results.** (b) Binary segmentation from user input in (a). (c) Result corresponding to a single iteration of GrabCut [20]. 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.

[8] A. Criminisi, T. Sharp, and C. Rother. Geodesic image and video editing. *ACM Trans. Graphics*, 2010.

[9] F. Crow. Summed-area tables for texture mapping. *SIGGRAPH*, 1984.

[10] 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.

[11] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV*, 2010.

[12] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann. Local stereo matching using geodesic support weights. In *ICIP*, 2009.

[13] M. Klodt, T. Schoenemann, K. Kolev, M. Schikora, and D. Cremers. An experimental comparison of discrete and continuous shape optimization methods. In *ECCV*, 2008.

[14] K. Lee, D. Kwon, I. Yun, and S. Lee. Optical flow estimation with adaptive convolution kernel prior on discrete framework. In *CVPR*, 2008.

[15] C. Lei and Y. Yang. Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *ICCV '09*.

[16] V. Lempitsky, C. Rother, and A. Blake. Logcut - efficient graph cut optimization for markov random fields. In *ICCV*, 2007.

[17] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers. A convex formulation of continuous multi-label problems. In *CVPR*, 2008.

[18] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *CVPR*, 2009.

[19] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In *ECCV'10*.

[20] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004.

[21] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[22] L. Sigal, A. Balan, and M. Black. HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion. *IJCV*, 2010.

[23] F. Steinbrücker, T. Pock, and D. Cremers. Large displacement optical flow computation without warping. In *ICCV*, 2009.

[24] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.

[25] D. Tschumperlè and R. Deriche. Vector-valued image regularization with PDE's : A common framework for different applications. In *CVPR*, 2003.

[26] O. Veksler. Stereo correspondence by dynamic programming on a tree. In *CVPR*, 2005.

[27] S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. In *ICCV*, 2009.

[28] M. Werlberger, T. Pock, and H. Bischof. Motion estimation with non-local total variation regularization. In *CVPR*, 2010.

[29] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *ECCV*, 2006.

[30] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. In *CVPR*, 2010.

[31] K. Yoon and S. Kweon. Adaptive support-weight approach for correspondence search. *PAMI*, 2006.