Fast Estimation of Large Displacement Optical Flow Using Dominant Motion Patterns & Sub-Volume PatchMatch Filtering

M.A. Helala and F.Z. Qureshi
Faculty of Science
University of Ontario Institute of Technology
Oshawa, Canada
{*mohamed.helala, faisal.qureshi*}*@uoit.ca*

Figure 1: (Left) the first image from the RubberWhale dataset; and (2nd from left) the ground truth optical flow. (3rd from left) the optical flow estimated using [1] with average End-Point Error (EPE) = 0.3026 and average Angular Error (AE) = 8.1951; (Right) the optical flow computed by the proposed method with EPE = 0.099 and AE = 3.07.

***Abstract*—This paper presents a new method for efficiently computing large-displacement optical flow. The method uses dominant motion patterns to identify a sparse set of sub-volumes within the cost volume and restricts subsequent Edge-Aware Filtering (EAF) to these sub-volumes. The method uses an extension of PatchMatch to filter these sub-volumes. The fact that our method only applies EAF to a small fraction of the entire cost volume boosts runtime performance. We also show that computational complexity is linear in the size of the images and does not depend upon the size of the label space. We evaluate the proposed technique on MPI Sintel, Middlebury and KITTI benchmarks and show that our method achieves accuracy comparable to those of several recent state-of-the-art methods, while posting significantly faster runtimes.**

***Keywords*-optical flow; large-displacement optical flow; motion patterns; patch match filtering;**

I. Introduction

Optical flow estimation given an image pair is a fundamental problem in computer vision. Optical flow describes two-dimensional motions of objects (usually individual pixels) between the two images, and it is often the first step in many computer vision techniques, such as motion detection, object segmentation, video encoding and compression, scene analysis, activity recognition, etc. Despite great advances in optical flow estimation since early seminal works by Horn and Schunck [2] and Lucas and Kanade [3], we still need better methods for dealing with large-displacement optical flow estimation. Most existing methods, for example, ignore higher-order terms in the optical flow constraint equation, which leads to poor performance when dealing with large motions [4]. Given the current trend to pack more pixels per image, we also need more efficient methods for optical flow estimation. We envision that these methods will be able to trade accuracy versus speed, adapting to imaging artifacts—such as, large motions, motion blur, occlusions, etc.—and leveraging the available hardware in the best way possible. Motivated by this vision, we present a new method for large-displacement optical flow estimation. Our method relies upon sparse processing, which can be tuned to achieve the desired accuracy versus speed balance.

Many recent techniques for optical flow estimation treat it as a pixel labeling problem, where each label is a 2D vector describing a pixel's motion between the two images [5], [1], [6], [7]. These methods start with an initial motion field and set up a cost volume that represents the costs of assigning differential motion (from the initial field) to individual pixels. Local or global energy minimization techniques process the cost volume to refine optical flow estimation. Within this setting, dealing with large-displacement optical flow estimation requires that we expand the label space, increasing the size of the cost volume. This leads to increased memory requirements and longer processing times. It may also results in a more noisy final optical flow [8].

The last few years have seen increased focus in improving both accuracy and speed of cost volume filtering based optical flow estimation methods [5], [6], [1]. Our method falls into this category. The proposed method is able to achieve accuracy comparable to the state-of-the-art methods at a fraction of their computational costs [5], [9], [6]. We are able to achieve this feat by restricting filtering to a sparse set of salient sub-volumes within the cost volume. Helala and Qureshi [10] demonstrate that it is possible to filter a sparse set of sub-volumes within the cost volume and still achieve acceptable accuracy, albeit in the context of depth estimation. Building upon this idea, we develop a new method for efficiently identifying salient sub-volumes within the cost volume. Subsequent filtering is restricted to these salient sub-volumes. Salient sub-volumes are selected as follows: 1) Approximate Nearest Neighbor Fields (ANNFs) search computes an initial flow field; 2) superpixel segmentation divides images into homogeneous regions; and 3) using the initial flow, we assign each superpixel, a salient region in the cost volume that bounds the dominant motion vectors of that superpixel. Step (2) above is motivated by the observation that images can often be viewed as a set of homogeneous regions having coherent dominant motions. Hence we avoid the computationally expensive motion segmentation [5], which would otherwise be needed to identify salient sub-volumes.

We evaluate the proposed method on three standard optical flow benchmarks: 1) Middlebury benchmark; 2)

MPI Sintel benchmark and 3) KITTI. Our method outperforms [9], [11], [12], [13], [1] methods on the Middlebury benchmark, and it achieves comparable performance to those of [14], [6], [5] methods on this benchmark. For MPI Sintel benchmark, our method outperforms [14], [15], [16], [1], [17] and achieves comparable results to those of [9], [13]. For local cost volume filtering, our method provides orders of magnitude speedup over [7] and [10], and upto 5 times speedup over [6].

A. Contributions

The contributions of this work are: (1) the proposed method leverages sparse cost volume filtering to achieve state-of-the-art runtime performance numbers on two standard benchmarks; (2) we present a technique for efficiently identifying salient sub-volumes within cost volume based on superpixel segmentation and ANNFs. To our knowledge, this technique is novel and has not been used before for optical flow. The computational complexity of our method is linear in the image size and independent of the size of the label space.

II. Related Work

We briefly summarize optical flow estimation techniques that are relevant to our method. For a comprehensive survey on computation of optical flow, we refer the reader to [4].

Coarse-to-fine approaches are often chosen to deal with large-displacement optical flow estimation [18]. These approaches, however, have trouble dealing with small and thin structures. Steinbruecker *et al.* [19] deal with this issue by identifying pixel-wise correspondences between the two images, however finding correspondences is expensive, rendering their method unable to deal with large motion ranges. Brox and Malik [20] addresses the issue of small and thin structures via keypoint matching. Their method is not affected by large motion ranges. [14] improves the method proposed in [20] by incorporating a series of discrete fusion moves. [9] proposes DeepFlow that relies upon dense feature matching to deal with large-displacement optical flow. DeepFlow; however, uses DeepMatching that has $O(M^2)$ space and time complexity, where M is the number of pixels [21]. It therefore requires several orders of magnitude more memory then other state-of-the-art methods. The PCA-Layers method developed by Wulff *et al.* [15] provided a much efficient algorithm that relies on an offline learning stage to estimate a PCA-model from sparse feature matches. A layered flow model is then used to estimate optical flow. Yang *et al.* [22] also provided a more accurate algorithm that uses piecewise homography models to estimate optical flow, however, at a large computational cost of about 500 seconds.

PatchMatch [23] and its variants [23], [24], [25] can efficiently compute ANNFs, which can be used to set up coarse correspondences between images. [5], [1], [26] employ ANNFs to compute initial optical flow. ANNFs based approaches attempt to minimize dissimilarity between patches without enforcing spatial coherence, i.e., patches in one image may have corresponding patches at arbitrary locations in the other image. Chen *et al.* [5], who use ANNFs to set up the initial optical flow, address this issue through motion segmentation. They set up a global optimization problem whose solution gives state-of-the-art large-displacement optical flow results. This method, however, has very high computational costs. Besse *et al.* [26] also proposed a method that relies upon ANNFs to compute the initial optical flow. Here belief propagation is used to refine the initial optical flow.

Edge-Aware Filtering (EAF) methods [27], [28] have been shown to provide a fast alternate to the global energy minimization techniques for solving pixel labeling problems [7], [6], [1]. These methods are often referred to as cost volume filtering. Hosni *et al.* [7] successfully implemented a cost volume filtering framework that applies the guided image filter [28] for optical flow and stereo disparity estimation. Their framework has fared well on the Middlebury benchmarks.

Cost volume filtering methods scale linearly with the size of the label space. This renders these methods inefficient for dealing with high-resolution images or computing motion detail preserving optical flow. SimpleFlow [29] attempts to accelerate the filtering process by providing sublinear solution, albeit at the cost of reduced accuracy. Lu *et al.* [6], on the other hand, reduce computational cost by randomly picking out candidate regions in the cost volume. The regions are picked using PatchMatch [23] search over the entire cost volume. Computational savings are minimal as the entire cost volume is searched. More recently, Bao *et al.* [1] proposed an algorithm that also integrates EAF with PatchMatch. [1] is able to achieve this speed up by 1) applying a hierarchical matching step that downsamples the input images and 2) for each pixel in one downsampled image, search is restricted to "similar" pixels in the other image during ANNFs setup, finally 3) labels are propagated from the downsampled image to the original image. Both steps (1) and (2) adversely effect the accuracy of this method. Approach presented in [6] is computationally expensive as it sets up ANNFs by random search over the entire cost volume. Method in [1] trades off speed for accuracy and provides a GPU implementation that is about 100 times faster than the method in [6].

Within the context of stereo depth estimation, Helala *et al.* [10], proposed a method that locates salient sub-volumes within the cost volume. Filtering is restricted to these sub-volumes, boosting runtime performance. Their stereo depth estimation method has ranked high on the Middlebury benchmark, outperforming [7], suggesting that it is possible to restrict filtering to a carefully selected set of sub-volumes in the cost volume and still achieve reasonable accuracy. This serves as the motivation for this work.

Optical flow estimation algorithms that use ANNFs to setup correspondences suffer from the spatial coherence problem. This stems from the fact that ANNFs search does not constrain the search radius for finding correspondences. To address this issue and to enforce spatial coherence, [5] uses motion segmentation, [6] and [1] rely upon EAF. Our method deals with spatial coherence via superpixel segmentation, assuming that homogeneous image regions exhibit coherent dominant motions.

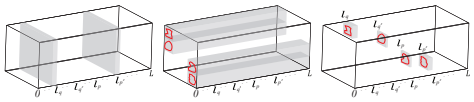

Figure 2: Different approaches for constructing sub-volumes. The volume of the entire cost volume is $I_{\text{height}} \times I_{\text{width}} \times |L|$, where the first two terms refer to the height and the width of the input images, respectively, and $|L|$ refers to label space size. (Left) [5] searches sub-volumes around dominant motions. (Middle) [6] randomly searches the entire label range for each superpixel (shown in red). (Right) our method randomly searches around the dominant motion for each superpixel.

III. Methodology

Given an input image pair (I_1, I_2), our goal is to assign a label $l = (u, v) \in L$ to each pixel $(x, y) \in I_1$. The label represents the displacement of pixel $(x, y) \in I_1$ to $(x + u, y + v) \in I_2$, and L is the label space. It is easy to extend this idea to multiple images $(I_1, I_2, \cdots, I_n)$. In this case, without the loss of generality, the first image I_1 is usually referred to the reference image I_{r} and we deal with image pairs of the form (I_{r}, I_k), where $I_k \in \{I_2, \cdots, I_n\}$.

A. Cost Volume Filtering

Our method follows the general framework of local correspondence search for computing optical flow [7]. The framework consists of three steps: 1) the first step uses pixel correspondences to set up a cost volume $C(x, y, l)$, which stores the cost of assigning a label $l \in L$ to a pixel (x, y); 2) the second step aggregates costs at each cost slice using EAF; and 3) the last step picks the "optimal" label assignment for each pixel, so as to minimize the overall cost of label assignment. A final step is often used to further refine label assignments, including assigning missing labels. In case of optical flow estimation, the desired solution to this label assignment problem is (1) spatially coherent, (2) follows label assignment costs, and (3) preserves edge discontinuities. From [7], [6], the cost volume is computed as follows:

$$C(x, y, l) = (1 - \beta) \min(d_1, \gamma_1) + \beta \min(d_2, \gamma_2), \quad (1)$$

where $d_1 = |I_{\text{r}}(x, y) - I_k(x_k, y_k)|$ and $d_2 = |\nabla_x I_{\text{r}}(x, y) - \nabla_x I_k(x_k, y_k)| + |\nabla_y I_{\text{r}}(x, y) - \nabla_y I_k(x_k, y_k)|$. γ_1 and γ_2 are user defined parameters. During filtering, each slice l of the cost volume is processed as follows:

$$C'(x, y, l) = W_{(x,y)} \otimes C(x, y, l), \quad (2)$$

where the kernel weights $W_{(x,y)} \in \mathbb{R}^{(2r+1)\times(2r+1)}$ for window $\omega_{(x,y)}$ centered on (x, y), depend upon the reference image and are computed for every (x, y). r is the kernel radius. Symbol $\otimes$ denotes convolution operator. The reference image is oftentimes called the *guidance* image. Many schemes exist to select $W_{(x,y)}$, so as to maintain the intensity changes and preserve the edges of the guidance image [28], [8], [30]. In this work, we use the method proposed in [28] to compute $W_{(x,y)}$. Given $i = (x, y)$ and $j = (x', y')$ such that j are neighbors of i in ω_i, we have the weight $W_i(j) = \frac{1}{(2r+1)^2} \sum_{k:i,j\in\omega_k} \left(1 + \frac{(I_{\text{r}}(i)-\mu_k)(I_{\text{r}}(j)-\mu_k)}{\sigma_k^2+\epsilon}\right)$, where μ_k and σ_k^2 are the mean and variance of I_{r} in ω_k, and ϵ is a regularization parameter. A selection step applies *winner-takes-all* strategy to pick for each pixel p at location (x, y) a label in L that minimizes the assignment cost:

$$l_p = \arg\min_l C'(x, y, l). \quad (3)$$

Cost volume filtering is linear in the size of label space, which makes these techniques slow for large L [7], [8]. An obvious strategy is to restrict filtering to sub-volumes within the cost volume [31], [10]. The method proposed in [10], for example, divides each slice in the cost volume into visible and non-visible regions, where visible regions denote areas of minimum costs. The method identifies visible regions via keypoint matching. The sub-volumes are constructed around the keypoint locations in the cost volume, and the parameters r_s and ε_s control the width (within slice) and the depth (across adjacent slices), respectively, of the sub-volume. Helala and Qureshi method achieves accuracy comparable to filtering the entire cost volume on stereo depth estimation. Their approach has several limitations, including the choice of r_s and ε_s and the dependency on the performance of the keypoint detector. Here we address these limitations and present a new method for dynamically selecting sub-volumes within the cost volume. The filtering is restricted to these sub-volumes, improving runtime performance while maintaining acceptable accuracy.

B. Approximate Nearest Neighbor Fields

Our method uses the fast edge preserving algorithm of [1] to compute ANNF using the input image pair. As mentioned earlier, [1] achieves speedup by downsampling the input images, which degrades accuracy. See the presence of noise in a sample output of this technique on the RubberWhale Middlebury training dataset (Figure 1). Still the ANNF computed by this method closely matches the ground truth (optical flow) at a large number of pixel locations. [5] performed an empirical study that also supports this observation. It highlights the advantage of using ANNFs for computing optical flow. ANNFs computation does not constrain the search radius between corresponding patches, which allows it to capture large motions between two images. Unlike traditional optical flow estimation schemes, ANNFs computation also preserves small and thin structures. The primary issue then with ANNFs based methods for computing optical flow is the existence of noise and missing labels for certain pixel locations. We now discuss how to resolve this issue.

C. Selecting Sub-Volumes

Chen *et al.* use dominant motions to handle noise present in the initial optical flow computed using ANNF [5]. Here dominant motions are extracted using a costly motion segmentation step. Local perturbations around each dominant motion capture local deformation and together define a salient range within the label space. We extend this idea to cost volume filtering by observing that dominant motions can divide cost volume slices into visible and non-visible

regions. This is akin to using keypoint matching for stereo disparity estimation in [10]. Consequently, it is possible to identify salient sub-volumes in the cost volume using dominant motions.

We are interested in a "fast" algorithm for computing optical flow. Therefore, using a costly motion segmentation step to compute dominant motions to identify the salient sub-volumes within the cost volume does not serve our purpose. Instead we rely upon superpixel segmentation to identify the salient sub-volumes [30], [6], [10]. Our choice stems from the following three observations: 1) The desired solution for optical flow should be spatially smooth and preserve intensity changes of image edges; 2) compact and spatially uniform superpixels respect image boundaries and include pixels that have a higher chance of sharing similar labels; 3) processing superpixels reduces the computation complexity and boosts the performance. We use the SLIC superpixel segmentation algorithm [32] which scales linearly with the image size and can be efficiently computed in real-time.

Specifically, using SLIC superpixel segmentation algorithm, the input image I is decomposed into K non-overlapping superpixels $\mathcal{S} = \{S_i | i \in [1, K]\}$. ANNF computed earlier is then used to compute dominant motion μ and motion variance σ^2 for each superpixel S. Specifically, (u, v) for pixels (x, y) belonging to each superpixel S_i are used to estimate μ_i and σ_i^2; $\mu_i = (\widehat{u}, \widehat{v})$ is computed using meanshift algorithm and $\sigma_i^2 = (\sum_u (u - \widehat{u})^2, \sum_v (v - \widehat{v})^2)/n$ for $n = |S_i|$. Set $\Omega_\mu = \{\mu_i | i \in [1, K]\}$ defines K dominant motions and the set $\Omega_{\sigma^2} = \{\sigma_i^2 | i \in [1, K]\}$ defines the motion variances corresponding to K superpixels. Together these sets are used to define a sparse set of K sub-volumes $\mathcal{V} = \{V_i | i \in [1, K]\}$. The width and height (defined in image space) of sub-volume V_i is determined by the minimum bounding rectangle B_i of superpixel S_i and its depth (defined in label space) is set such that it contains all labels l that satisfy $\|l - \mu_i\| < \beta_s \sigma_i$. For the results presented here, β_s, the expansion factor, is set to 1.9. In practice, we bound the expansion by γ_s to avoid situations involving very large σ_i by ensuring that $\beta_s \sigma_i \leq \gamma_s$.

Figure 2 shows a comparison between our method of constructing sub-volumes against other methods. Figure 2 (left) shows the sub-volumes defined by [5] around the dominant motion patterns. Here the sub-volumes span the entire image space and very small sections of the label space. Figure 2 (middle) shows the searched sub-volumes constructed by [6] for each superpixel, where the image space is partitioned but the entire label space is randomly searched. Figure 2 (right) shows our way of constructing sub-volumes that captures the benefits of both [6] and [5] by partitioning both the label and image spaces.

D. Coarse-to-Fine Sub-Volumes

In PatchMatch, the patch size affects both the runtime complexity and the accuracy. Larger patch sizes are better at enhancing edge-aware filtering and resolving ambiguities of matching costs; thereby, producing higher quality ANNFs. So, larger patch sizes are desirable for our method to produce a good set of sub-volumes. However, it is often not possible to specify the optimal patch size without knowing the object scales *a priori*. Multi-scale approaches are typically employed to solve such issues.

	CF [7]	PM [23]	PMF [6]	DM [21], [9]	Ours
Com.	$O(M\|L\|)$	$O(mM \log \|L\|)$	$O(M \log \|L\|)$	$O(M^2)$	$O(M)$
Mem.	$O(M)$	$O(M)$	$O(M)$	$O(M^2)$	$O(M)$

Table I: Computational (Comp.) and Memory (Mem.) complexities for Cost Filter (CF) [7], PatchMatch (PM) [23], PatchMatch Filter (PMF) [6], Deep Matching (DM) [21], [9] and our method.

We propose a multi-scale (coarse-to-fine) method for identifying a set of sub-volumes for each superpixel. Our method begins by constructing an n level image pyramid from the input image—starting with the input image and downsampling it by half for successive levels. For all our experiments, we set $n = 3$.

We perform superpixel segmentation only on the input image resolution. Then, we downsample each superpixel to find its boundary at every pyramid level. For each pyramid level s, we calculate an initial flow map [1]. and define a sub-volume V_i^s for every superpixel S_i. Then, for each superpixel, we scale up its sub-volumes from lower pyramid levels to the input image resolution. So, each superpixel S_i at the input image resolution receives n sub-volumes $\{V_i^s | s \in [1, n]\}$. These sub-volumes are subsequently filtered in the original image resolution (i.e., level 1), ensuring that small and thin structures are not lost during filtering. Downsampling the image while keeping patch size constant has an unexpected benefit. It effectively increases patch size for computing the initial flow maps (using [1]), improving the quality of sub-volumes.

E. PatchMatch for Sub-Volume Filtering

The sub-volumes defined in the previous step are filtered using an extended version of randomized PatchMatch filter [6] that considers both dominant motions of and local deformations within superpixels. Let $G = (\mathcal{S}, E)$ denotes an adjacency graph over the set of nodes $\mathcal{S}$ and with the set of edges E. In G, the nodes represent superpixels and the edges encode neighborhood relationship. Two superpixels are considered neighbors if they share a boundary. For each superpixel S, the set $\mathcal{N}(S)$ denotes its neighbors. PatchMatch is an iterative technique and each iteration consists of two steps: 1) label propagation and (2) random search. Graph G supports fast superpixel neighbor queries, allowing efficient label propagation and random searches used by PatchMatch. Our algorithm initializes PatchMatch using the ANNF computed earlier, sets up initial costs $\tilde{C}(x, y, l)$ using equations 1 and 2. $\tilde{C}(x, y, l)$ stores the current best (minimal) costs of assigning label l to pixel (x, y). Algorithm 1 summarizes our approach. Notice that the Aggregate function performs cost aggregation on superpixel S_i using the given label l'. This function sets up a cost slice $C(x, y, l')$ using Equation 1, where pixel $(x, y) \in B_i$, and filter it to compute $C'(x, y, l')$ using Equation 2. For each pixel $(x, y) \in B_i$, the function updates its (current best) label to l' if $C'(x, y, l') < \tilde{C}(x, y, l)$.

Complexity. Our algorithm has $O(M)$ time and space complexity, where M is the size of the input image I. A

Algorithm 1 PatchMatch for Sub-Volume Filtering

Input: $\tilde{C}(x,y,l), \mathcal{S}, \{\mathcal{V}^s\}_{s=1}^n$
Output: Updated $\tilde{C}(x,y,l)$

/* Propagating Local Deformations */
for $i = 0$ to $|\mathcal{S}|$ **do**
  for all $V \in \{V_i^s\}_{s=1}^n$ **do**
    Pick a random label $l' \in V$
    Aggregate$(\tilde{C}, i, l')$
  end for
end for
/* Propagating Motions Across Neighbors */
for $i = 0$ to $|\mathcal{S}|$ **do**
  for all $S_j \in \mathcal{N}(S_i)$ **do**
    Pick random$(x,y) \in B_j$with best label l^*
    Aggregate$(\tilde{C}, i, l^*)$
  end for
end for
/* Random Search */
for $i = 0$ to $|\mathcal{S}|$ **do**
  Pick a random (x,y) from S_i with label l^*
  Select $V \in \{V_i^s\}_{s=1}^n$ such that $l^* \in V$
  For all $l' \in V$ at exponentially decreasing distance from l^* **do**
  Aggregate$(\tilde{C}, i, l^*)$
  end for
end for

detailed proof of this linear time complexity is given in the supplementary material.

F. Occlusion Handling & Subpixel Accuracy

Occluded regions are detected by computing both forward and backward optical flow. It is easy to compute both forward and backward flow at the same time using the symmetric kernel [7], [1] in Equation 2. Pixels having inconsistent motion in forward and backward optical flow are considered occluded; this is often termed as the *forward-backward consistency check* [7], [1], [6]. We use the fast occlusion handling approach presented in [10] to handle occluded pixels. Outliers are dealt with using the fast Weighted Median Filtering (WMF) algorithm presented in [33]. A second round of forward-backward consistency check fixes any inconsistent label assignments [1].

Subpixel accurate optical flow is computed by upsampling the input images. Similar to the scheme presented in [7], [6], our method upsamples the input images by a factor of 8 using bi-cubic interpolation. Fortunately, this only causes a small runtime performance hit, since matching costs are computed at resolutions of the input images.

IV. Experimental Results

We evaluate our method on three standard optical flow benchmarks: (1) MPI Sintel benchmark [34]; (2) Middlebury benchmark [35] and 3) KITTI benchmark [36]. We also evaluate the runtime performance of the proposed method on a 2880×1620 high-resolution image [37] (we assume zero motion and focus only on processing times).

For these evaluations, we employ the guided filter proposed in [28], [6] for EAF. The parameter setting for guided filter are taken from [28], [6]: $\sigma_r = 0.1$, $\beta = 0.9$, $\gamma_1 = 0.039$, $\gamma_2 = 0.016$, and $\epsilon = 0.0001$. The kernel radius r is set to 5. The sub-volume selection parameters that control sparsity are: $\beta_s = 1.9$ and $\gamma_s = 2$. We fix the number of superpixels K equal to 1500, and the number of PatchMatch iterations is set to 7. These parameters values are fixed for all our evaluations.

The algorithm is implemented in C++ using CUDA, and all experiments were carried out using NVIDIA GeForce GTX 780 GPU. On standard size images, ANNF computation takes around 0.32 seconds, the preprocessing step takes about 0.4 seconds and our sub-volume PatchMatch filtering takes 0.84 seconds. On average, our algorithm takes around 1.56 seconds to process standard resolution images from Middlebury benchmark. The proposed algorithm also scales well to medium resolution images, taking on average 1.95 and 2 seconds to compute optical flow from MPI Sintel and KITTI benchmarks, respectively. A single PatchMatch filtering iteration takes about 0.14 seconds on medium resolution images.

Filtering time. Table II compares the filtering times (using both CPU and GPU) of our method against PMF, CF, and ACF schemes. These runtimes include both cost volume computation and aggregation. Notice that PMF randomly filters the entire cost volume, while CF performs an exhaustive search. The table also provides a comparison of the total runtime of our method against PCA-Layers, EPPM, and NNF-Local. We perform all comparisons on the same machine except for NNF-Local that we report from benchmarks due to current unavailability of code.

The filtering time comparison is performed on three different images sizes: a) 640×480 (Middlebury dataset), b) 1024×436 (from MPI Sintel dataset), and c) 2880×1620 [37]. For PMF, CF, and ACF schemes, the label space size is 410000 for 640×480 (see PMF [6]). The label space sizes are 1600000 and 102000 for 1024×436 and 2880×1620 images, respectively. Subpixel accuracy computation is turned off for 2880×1620 images because of memory constraints. Notice that [6] has $O(M \log |L|)$ time complexity; whereas, our method has $O(M)$ time complexity. Here M represent the number of pixels (i.e., image resolution) and $|L|$ is the size of the label space. Both CF and ACF post large runtimes, which renders these methods infeasible for higher resolution images. Even ACF, which introduced sub-volume filtering, is able to achieve only 4 times speedup over CF.

Multi-GPUs. The filtering time comparison are also reported on a multi-GPU setup consisting of 3 GPUs. For this setup, we perform loop unrolling for parallelizing the filtering iterations of our method and PMF. CF and ACF are parallelized on the 3 GPUs. Table II shows a clear performance improvement of our method on this multi-GPU setup. For example, the total runtime of our method reduces to 0.5 seconds on the Sintel dataset, while using an extra GPU for preprocessing.

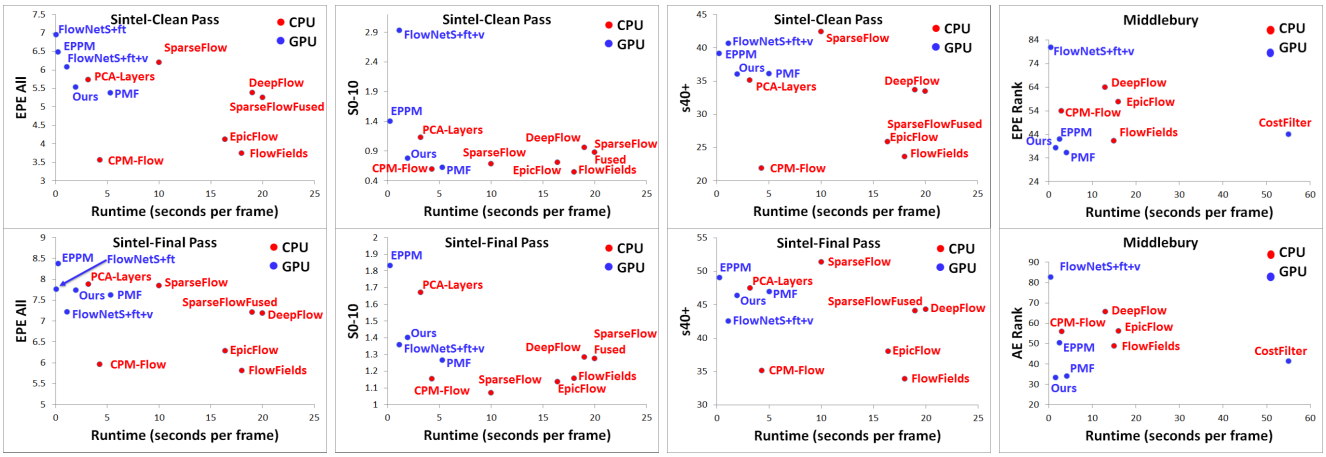

Figure 3: Accuracy versus runtime evaluation (summarization) recorded on 24 January 2017, on MPI Sintel and Middlebury benchmarks. We focus on the top-ranked techniques with run-time ≤ 40 seconds. For Middlebury, EPPM is computed by [1] without hierarchical matching. Detailed results are available online and in the supplementary material.

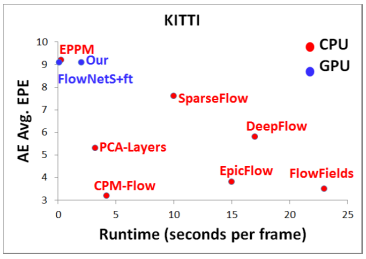

Figure 4: Accuracy versus runtime evaluation recorded on 24 January 2017, on KITTI benchmarks. We focus on the top-ranked techniques with run-time ≤ 40 seconds.

Algorithm	Middlebury-Standard(640x480)			Sintel-Medium(1024x463)			Cat-High(2880x1620)		
	CPU	GPU	Multi-GPUs	CPU	GPU	Multi-GPUs	CPU	GPU	Multi-GPUs
Ours	9.38	0.84	0.38	10.6	0.98	0.45	98.4	5.53	2.4
PMF [6]	35	3.4	1.38	44	4.2	1.7	437.5	29	11.23
CF [7]	59526.9	750	251	-	-	-	-	-	-
ACF [10]	35015.8	410.4	137.8	-	-	-	-	-	-
Ours	11	1.56	0.4	14	1.98	0.5	110	9.53	3.2
EPPM [1]	-	0.17	-	-	0.22	-	-	0.32	-
PCA-Layers [15]	10.1	-	-	15.2	-	-	-	-	-
NNF-Local [5]	1073	-	-	-	-	-	-	-	-

Table II: (Top four rows) Comparing filtering times for our method against that of PMF, CF, and ACF schemes. We report these times on a single CPU at 2.8GHz, a single GPU, and multiple GPUs. Our algorithm and PMF are pipelined on 3 GPUs; CF and ACF are parallelized on the 3 GPUs. (Bottom four rows) Comparing the total runtime of our algorithm against competing methods. All runtimes are computed on the same machine except for NNF-Local that we report from benchmarks.

A. *MPI Sintel Benchmark*

MPI Sintel benchmark contains two rendering passes: (1) clean pass and (2) final pass. The clean pass exhibits large motions and illumination, reflectance and shading effects; whereas, the final pass adds color correction, defocus, motion blur, and atmospheric effects to the images from the clean pass. The dataset contains 12 sequences for each pass and the total number of images is 564.

Figure 3 summarizes the MPI Sintel benchmark results of our method against other schemes (with runtimes no greater than 40 seconds per frame). The figure shows average End-Point Error (EPE) versus runtime on the entire image (*all*), on regions with displacements more than 40 pixels (*s40+*), and on regions with displacements less than 10 pixels (*s0-10*). Our method achieves accuracy that is comparable to those obtained by PMF [6] and DeepFlow [9]—two state-of-the-art methods—however, our method posts significantly faster runtimes. E.g., our method is nearly 3 times faster than PMF and 10 times faster than DeepFlow. Table II shows that our method achieves around 5 times speedup over PMF for higher resolution images. Method in DeepFlow has trouble dealing with higher resolution images due to its large memory requirements (see [21]). See Table I for a comparison of space/time complexity of our method vs. PMF and DeepFlow.

Figure 3 also shows that the proposed method outperforms several other recent methods—PCA-Layers [15], and SparseFlow [16]—on both accuracy and speed. Furthermore, our method outperforms the method of [1] on nearly all error measures (see results on clean passes). As expected, however, [1] posts faster runtimes (see Table II).

The FlowNetS+ft+v [13] is the only method that imposes faster runtime over our single GPU implementation on the MPI final pass. Our method, however, outperforms it on accuracy by a large margin on the clean pass. Our method also gives better results on the Middlebury datasets (see Figure 3) than those of FlowNetS+ft+v. The runtime of FlowNetS+ft+v is 1.12 seconds compared to 1.95 seconds for our method. FlowNetS+ft+v requires a large set of training images with ground truth, which as indicated by [13] is difficult to obtain in practice. Perhaps, it is its inability to generalize why FlowNetS+ft+v performs poorly on Middlebury dataset.

CPM-Flow [38] outperforms our method on accuracy, while having a slower runtime. This is because it relies on

EpicFlow which involves both variational energy minimization and dense interpolation and takes around 3 seconds for post-processing results. Our method is much simpler and can be further accelerated on multi-GPUs (see Table II).

B. Middlebury Benchmark

Figure 3 also shows the Middlebury quantitative benchmark results and compares our method against several other techniques. Middlebury benchmark contains images exhibiting small displacements. Some image pairs, such as "Basketball" and "Backyard" do show large displacements though. Figure 3 (top of last column) shows average EPE rank values versus runtime for all image regions. Figure 3 (bottom of last column) shows average Angle Error (AE) rank versus runtime for all image regions. Our method beats every other method in terms of runtime performance, while achieving comparable (in some cases, only marginally worse) accuracy. Our method outperforms several methods, including EPPM [1], CF [7], EpicFlow [12], FlowFields [11], and DeepFlow [9]. Our method also achieves around 3 times speedup over PMF, while getting better accuracy on AE rank and a comparable accuracy on EPE rank. This suggests that our method is able to achieve good performance on both MPI Sintel and Middlebury benchmarks. These evaluations also strengthen the key contribution of our work: a fast method for computing optical flow that is able to achieve good accuracy on both small- and large-displacement optical flow estimation problems.

C. KITTI Benchmark

The KITTI benchmark [36] was obtained by a moving vehicle capturing images of city traffic streets. So, the recorded flow is due to camera motion, which results in scaling and rotation of objects with large smooth areas and few motion boundaries. In addition, the camera creates large distortions along image boundaries. This limits our local method from finding accurate sub-volumes. We share this limitation with other local methods [1], [7], [6]. We think that methods employing global energy minimization [12] will perform better than our local method on KITTI. Figure 4 shows the average EPE versus runtime of our method against other state-of-the-art methods. It is worth noting that we have a similar accuracy to FlowNetS+ft [13] on KITTI.

D. Time-Accuracy Tradeoff

Figure 5 (Left column) shows a time-accuracy tradeoff evaluation of our method for different values of K, where K is the number of superpixels. The EPE values are averaged over the entire MPI Sintel testing datasets, and the filtering times are reported on a single GPU. We observe that our method is robust to the choice of K. Furthermore, larger values of K yield better optical flow estimation results. This, however, comes at the cost of increased processing costs. We found that setting K to 1500 gives a reasonable tradeoff between accuracy and time. Similar results have been obtained on the Middlebury training datasets and are available in the supplementary material. Notice that K, β_s and γ_s control the sub-volume size, however, we show by experiments that the chosen values of β_s and γ_s allow our

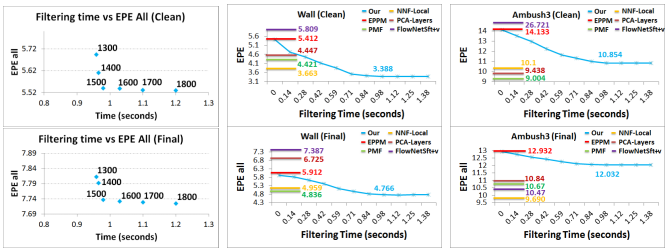

Figure 5: (Left column) Time-Accuracy tradeoff study for different values of K (number of superpixels), on the clean and final passes of the MPI Sintel testing dataset. (Middle and right columns) The convergence for 10 iterations (baseline is 7) of our algorithm on Wall and Ambush3 datasets compared to other methods.

method to achieve similar accuracy to PMF that randomly filters the entire cost volume. Increasing γ_s will not carry much benefits. Reducing γ_s, however, reduces accuracy as we will be setting more on the mean motion of each superpixel.

The middle and right columns of Figure 5 shows the convergence of our method on two datasets: Wall and Ambush3, on both the clean and final passes. Notice that our method typically achieves convergence after 7 iterations. PMF [6], on the other hand, requires 10 iterations for convergence. A closer look at the figures shows that our method outperforms all compared methods on the Wall dataset. This is true for most datasets, however our method fails on the Ambush1 and Ambush3 datasets as they have large textureless regions. we share this limitation with other local methods [6], [1], [7].

V. Conclusion

This paper develops a new method for fast optical flow computation. The proposed method is evaluated on MPI Sintel, Middlebury and KITTI benchmarks and it achieves performance comparable to that of several state-of-the-art methods, while posting significantly faster runtimes. The results indicate that our method does well on both small- and large-displacement optical flow estimation. We also show that the computational complexity of our method is linear in the size of image space and does not depend upon the size of the label space. Consequently, our method appears particularly suited to optical flow estimation problems having large label spaces, such as when one needs to resolve fine motions in large-displacement settings.

The method belongs to the class of cost filtering based optical flow estimation algorithms and it leverages sparse processing of cost volume to achieve its runtime performance. It seems possible that the proposed method can trade accuracy versus speed or *vice versa* by controlling sparsity. In the future we want to study this aspect of our algorithm in more detail. We also wish to apply this technique to other domains, such as depth estimation.

References

[1] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving patchmatch for large displacement optical flow," *TIP*, vol. 23, no. 12, pp. 4996–5006, 2014. 1, 2, 3, 4, 5, 6, 7

[2] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981. 1

[3] D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, vol. 2, 1981, pp. 674–679. 1

[4] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: A survey," *CVIU*, vol. 134, pp. 1–21, May 2015. 1, 2

[5] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in *CVPR*, June 2013, pp. 2443–2450. 1, 2, 3, 4, 6

[6] J. Lu, H. Yang, D. Min, and M. N. Do, "Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," *IEEE PAMI*, vol. PP, no. 99, pp. 1–1, jul 2016. 1, 2, 3, 4, 5, 6, 7

[7] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," *TPAMI*, vol. 25, no. 2, pp. 504–511, 2013. 1, 2, 3, 4, 5, 6, 7

[8] K. Zhang, Y. Fang, D. Min, L. Sun, S. Yang, S. Yan, and Q. Tian, "Cross-scale cost aggregation for stereo matching," in *CVPR*, June 2014, pp. 1590–1597. 1, 3

[9] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *ICCV*, Dec 2013, pp. 1385–1392. 1, 2, 4, 6, 7

[10] M. A. Helala and F. Z. Qureshi, "Accelerating cost volume filtering using salient subvolumes and robust occlusion handling," in *ACCV*, Nov 2014. 1, 2, 3, 4, 5, 6

[11] C. Bailer, B. Taetz, and D. Stricker, "Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation," in *ICCV*, December 2015. 2, 7

[12] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," in *CVPR*, Boston, MA, June 2015. 2, 7

[13] P. Fischer, A. Dosovitskiy, E. Ilg, C. H. Philip Hausser, and V. Golkov, "Flownet: Learning optical flow with convolutional networks," in *ICCV*, December 2015. 2, 6, 7

[14] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *TPAMI*, vol. 34, no. 9, 2012. 2

[15] J. Wulff and M. Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers," in *CVPR*, June 2015, pp. 120–130. 2, 6

[16] R. Timofte and L. Gool, "Sparse flow: Sparse matching for small to large displacement optical flow," in *WACV*, Jan 2015, pp. 1100–1106. 2, 6

[17] D. Sun, C. Liu, and H. Pfister, "Local layering for joint motion estimation and occlusion detection," in *CVPR*, June 2014, pp. 1098–1105. 2

[18] L. Alvarez, J. Weickert, and J. Snchez, "Reliable estimation of dense optical flow fields with large displacements," *IJCV*, vol. 39, no. 1, pp. 41–56, 2000. 2

[19] F. Steinbrücker, T. Pock, and D. Cremers, "Large displacement optical flow computation without warping," in *ICCV*, Sep. 2009, pp. 1609–1614. 2

[20] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *TPAMI*, vol. 33, no. 3, pp. 500–513, 2011. 2

[21] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Deep Convolutional Matching," INRIA, Tech. Rep., 2015. 2, 4, 6

[22] H. L. J. Yang, "Dense, accurate optical flow estimation with piecewise parametric model," in *CVPR*, 2015. 2

[23] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 24:1–24:11, jul 2009. 2, 4

[24] J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," in *CVPR*, 2012, pp. 111–118. 2

[25] S. Korman and S. Avidan, "Coherency sensitive hashing," in *ICCV*, 2011, pp. 1607–1614. 2

[26] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz, "Pmbp: Patchmatch belief propagation for correspondence field estimation," *IJCV*, vol. 110, no. 1, pp. 2–13, 2014. 2

[27] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, "Bilateral filtering: Theory and applications," *FnT CGV*, vol. 4, no. 1, pp. 1–73, 2009. 2

[28] K. He, J. Sun, and X. Tang, "Guided image filtering," *TPAMI*, vol. 35, no. 6, pp. 1397–1409, 2013. 2, 3, 5

[29] M. Tao, J. Bai, P. Kohli, and S. Paris, "Simpleflow: A non-iterative, sublinear optical flow algorithm," *Comp. Graph. Forum*, vol. 31, pp. 345–353, May 2012. 2

[30] X. Mei, X. Sun, W. Dong, H. Wang, and X. Zhang, "Segment-tree based cost aggregation for stereo matching," in *CVPR*, June 2013, pp. 313–320. 3, 4

[31] S. N. Sinha, D. Scharstein, and R. Szeliski, "Efficient high-resolution stereo matching using local plane sweeps," in *CVPR*, June 2014, pp. 1582–1589. 3

[32] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *TPAMI*, vol. 34, no. 11, pp. 2274–2282, 2012. 4

[33] Q. Zhang, L. Xu, and J. Jia, "100+ times faster Weighted Median Filter (WMF)," in *CVPR*, June 2014, pp. 2830–2837. 5

[34] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *ECCV*, Oct. 2012, pp. 611–625. 5

[35] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *IJCV*, vol. 92, no. 1, pp. 1–31, 2011. 5

[36] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE CVPR*, June 2012, pp. 3354–3361. 5, 7

[37] ultrahdwallpapers, "Cat Staring Wallpaper," http://www.ultrahdwallpapers.net/animals/cat_staring-wallpaper-3840x2160, 2015, [Online; accessed 11-July-2015]. 5

[38] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine patchmatch for large displacement optical flow," in *IEEE CVPR*, 2016. 6