This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Efficient Coarse-to-Fine PatchMatch for Large Displacement Optical Flow

Yinlin Hu[1] huyinlin@gmail.com
Rui Song[1,2] rsong@xidian.edu.cn
Yunsong Li[1,*] ysli@mail.xidian.edu.cn
[1]Xidian University, China [2]Shanghai Institute of Technical Physics, China

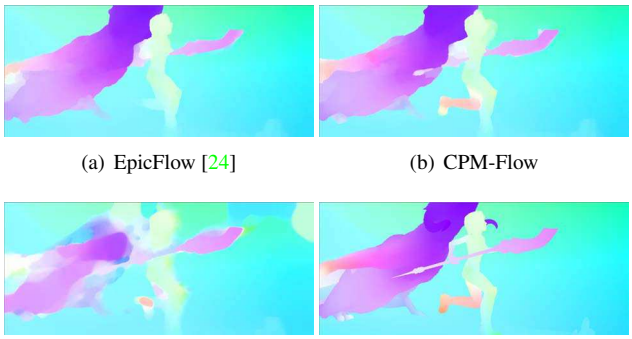
Abstract

As a key component in many computer vision systems, optical flow estimation, especially with large displacements, remains an open problem. In this paper we present a simple but powerful matching method works in a coarse-to-fine scheme for optical flow estimation. Inspired by the nearest neighbor field (NNF) algorithms, our approach, called CPM (Coarse-to-fine PatchMatch), blends an efficient random search strategy with the coarse-to-fine scheme for optical flow problem. Unlike existing NNF techniques, which is efficient but the results is often too noisy for optical flow caused by the lack of global regularization, we propose a propagation step with constrained random search radius between adjacent levels on the hierarchical architecture. The resulting correspondences enjoys a built-in smoothing effect, which is more suited for optical flow estimation than NNF techniques. Furthermore, our approach can also capture the tiny structures with large motions which is a problem for traditional coarse-to-fine optical flow algorithms. Interpolated by an edge-preserving interpolation method (EpicFlow), our method outperforms the state of the art on MPI-Sintel and KITTI, and runs much faster than the competing methods.

1. Introduction

Optical flow has traditionally been, and continues to be, one of the most fundamental components in many vision tasks. There has been abundant literature on this topic, while obtaining a reliable optical flow for real-world videos remains a challenging problem caused mainly by motion discontinuities, large displacements, and occlusions.

Since the pioneering work by Horn and Schunck [11] who formulated the optical flow estimation as a problem of energy minimization, many effective approaches have emerged for improving the performance with complex motions [5, 26, 29]. Though combined with a coarse-to-fine scheme, such approaches still often failed to estimate large displacements introduced by fast motion, which is caused by the propagation of errors from coarser levels to the finest level, especially in the case of tiny structures with large motions (see Figure 1(c)).



(a) EpicFlow [24] (b) CPM-Flow (c) MDPFlow2 [29] (d) Ground truth

Figure 1. Comparison of state-of-the-art flow methods. In addition to the sharp motion boundaries which is similar to EpicFlow, our CPM-Flow can survive in the case of tiny structures with large motions (like the leg of the character and the bar of the weapon) which is usually a problem.

The visual similarity between two image regions is the most important clue for large optical flow estimation. Recently, optical flow interpolated from sparse descriptor matching correspondences directly has shown great success for large displacements, especially with significant occlusions [24]. While the results is mainly constrained by the efficiency and accuracy of the matching techniques.

In contrast, as a core component in image editing and scene correspondence [18], the nearest neighbor field (NNF) is closely related to optical flow estimation. The objective of NNF computation is to find one or more nearest (visually similar) neighbors for every patch in a given image against another image. The main challenge in this procedure is the computational complexity. Through the seminal work called PatchMatch [4] and the improved methods [15, 10], the efficiency of computing NNF has advanced remarkably. The core idea behind this boost is random search and propagation between neighbors. While with a different objective from optical flow estimation, the computed NNF is often

*Corresponding author

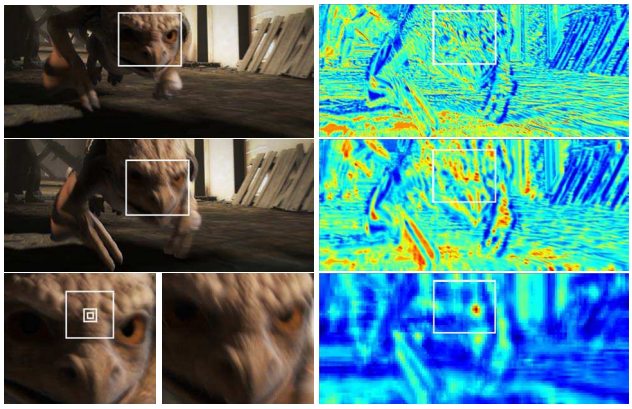


Figure 2. Response maps of some reference patches on the target image with different patch size. **Left column** consists of the two consecutive images and the zoomed version from top to bottom. **Right column** is the responses of the patches in the left-bottom with size $4 \times 4, 8 \times 8$ and 32×32 from top to bottom respectively. We can see that the response maps with larger patch size are more discriminative.

very noisy from the viewpoint of motion, especially in the criteria of edge preservation and spatial smoothness for evaluating an optical flow results, which is mainly caused by the lack of global regularization.

In this study, we propose a matching method combining a random search strategy with a coarse-to-fine scheme for optical flow with large displacements. A key observation is that matching correspondences with larger patch size are often more discriminative (see Figure 2). After the construction of the pyramids, we can use the matching correspondences from higher levels of the pyramids as a guidance for the matching process on lower levels. While as in DeepMatching[27], constructing the response maps of each reference patch on the target image at each level of the pyramid is time-consuming. We introduce a propagation procedure between the adjacent levels from top to bottom, which avoids the construction of the whole response maps. With random search strategy like in PatchMatch [4] on each level, our approach, *CPM* (Coarse-to-fine PatchMatch), interpolated using EpicFlow[24] outperforms the original EpicFlow and also most state of the art (see Figure 1).

We make the following contributions: **1)** We propose *CPM* matching, a novel coarse-to-fine matching scheme based on combination of random search and propagation between hierarchical levels. We show that it is robust to larger displacements and more suited for optical flow compared with NNF methods. **2)** We propose a fast approximate structure for the matching process, which avoids finding the matching of every pixels and leads to a significant speed-up with controllable accuracy. **3)** We show the effectiveness and efficiency of our matching approach by achieving state-of-the-art flow accuracy but running much faster than other competing methods on MPI-Sintel [6] and KITTI [9, 21].

2. Related Work

We do not review the entire literature on optical flow and only discuss the related literature. We refer to publications like [26, 2] for a detailed overview of optical flow methods based on the classic variational minimization framework first proposed in [11].

Our method is closely related to NNF estimation. The efficiency of computing NNF has advanced remarkably through the work [4, 15, 10], while the results is often too noisy from the viewpoint of motion. Nevertheless, many efforts have been made to estimate the optical flow based on the NNF: Chen *et al*. [7] used the computed NNF as a hint for motion segmentation before variational optimization. Bao *et al*. [3] adapted the PatchMatch algorithm to optical flow using an edge-preserving patch-matching measurement. However, they still often failed in the case of large displacements, especially with significant occlusions. Bailer *et al*. [1] proposed a similar pipeline to ours to handle the noisiness of NNF. However, their method relies on a dedicated hierarchical structure and many complicated techniques for ambiguity handling.

Also inspired by PatchMatch [4], Li *et al*. [17] proposed a PatchMatch belief propagation to estimate the optical flow which was formulated as a labeling problem. Yang *et al*. [30] proposed a piecewise parametric model for optical flow estimation. In contrast, we tackle the optical flow problem through a nonparametric matching.

As an important milestone regarding the integration of matching and optical flow, Brox and Malik [5] introduced a descriptor matching term to the classic variational minimization framework of optical flow. Furthermore, Xu *et al*. [29] proposed an extended coarse-to-fine framework that integrates matching to refine the flow at each level. However, due to the sparsity of the matching and the requirement of accurate initialization of the variational minimization, they still usually fail in the case of small details with motion larger than its own scale. To handle the sparsity of descriptor matching, Leordeanu *et al*. [16] extended sparse matching to dense matching with a locally affine constraint. For efficiency, Wulff *et al*. [28] introduced a basis of flow, and obtained the flow directly from the combination of the basis of the flow infered from a sparse descriptor matching. However, the quality is not satisfactory. Revaud *et al*. [24] introduced an edge-preserving interpolation framework for optical flow based on a matching algorithm termed DeepMatching [27]. However, the interpolation results is mainly constrained by the efficiency of DeepMatching. Closely related to our method, Kim *et al*. [14] and Hur *et al*. [12] also investigated hierarchical matching, but their methods requires inexact inference using loopy belief propagation.

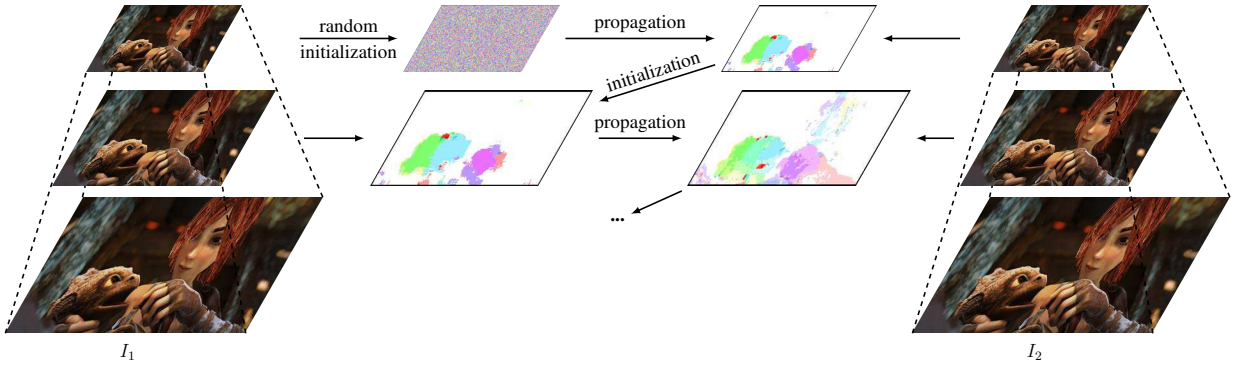


Figure 3. Overview of the proposed CPM algorithm. Given two images, we construct the pyramids and process from top to bottom. On each level, the initial matching correspondences is propagated with random search after a fixed number of times, and the results of each level is used as a initialization of the next lower level. The results after the propagation on the finest level is our matching results.

3. Coarse-to-fine PatchMatch

In this section, we present our matching framework, CPM, and discuss its main features. Our matching method builds upon a hierarchical architecture, and works in a coarse-to-fine (top-down) scheme. An overview of CPM Matching is given in Figure 3. We first detail the matching procedure on one level of the pyramid in Section 3.1, and then describe the hierarchical structures of our approach as well as the propagation step between the levels in Section 3.2.

3.1. Basic Matching

Considering the nature of smoothness of optical flow compared with the NNF, we define our goal of matching is to find the best correspondence of some *seeds* rather than every pixel of the image for efficiency. Formally, given two images $I_1, I_2 \subset R^2$ and a collection of seeds $\mathcal{S} = \{s_m\}$ at position $\{p(s_m)\}$, our goal is to determine the *flow* of each seed $f(s_m) = \mathcal{M}(p(s_m)) - p(s_m) \in R^2$, where $\mathcal{M}(p(s_m))$ is the corresponding matching position in I_2 for seed s_m in I_1. In our method, the seeds are the cross points of the regular image grid with a spacing of d pixels. Then there's only one seed in every $d \times d$ non-overlapping block. We will show that this fast approximation results in a significant speed-up with controllable accuracy.

Adopting the regular image grid, we obtain a default neighbor system according to the spatially adjacency of the seeds on the image grid. Like PatchMatch, *neighborhood propagation* and *random search* is performed iteratively in an interleaved manner after some flow initialization of each seed (detailed in Section 3.2).

Seeds are examined in *scan* order on odd iterations and in *reverse scan* order on even iterations. For a current seed s_m, we denote its set of spatially adjacent seed neighbors that is already examined in current iteration as $\mathcal{N}_m$. Flow values are propagated from neighbor seeds to current seed if they have already been examined in current iteration. That is

$$f(s_m) = \arg\min_{f(s_i)}(C(f(s_i))), s_i \in \{s_m\} \cup \mathcal{N}_m \qquad (1)$$

where $C(f(\cdot))$ denote the match cost between patch centered at $p(s_m)$ in I_1 and patch centered at $p(s_m) + f(\cdot)$ in I_2. We will discuss the computation of match cost in Section 3.3.

After the preceding propagation step, a random search as in PatchMatch [4] is performed for the current seed s_m. We attempt to improve $f(s_m)$ by testing some candidate flow around the current best flow. As in [4], a sequence of random flow sampled around the current best flow $f(s_m)$ of each seed s_m is evaluated at an exponentially decreasing scope started from a maximum search radius. The ratio α between the two consecutive search scopes is fixed to $1/2$.

Let n the number of iteration numbers. After n times of iteration, we stop our matching process. In practice a fixed iteration number works well.

3.2. Coarse-to-fine Scheme

Our basic matching is similar to PatchMatch [4] which is very noisy without global regularization. Similarly, our basic matching contains many outliers arising from the ambiguity of small patches. A common way to handle the ambiguity of small patches is increasing the size of the patches, while this often leads to less accurate matches.

We introduce a simple but powerful hierarchical architecture with propagation from top to bottom to handle this problem. First, we construct a pyramid with k levels for both I_1 and I_2 with a downsampling factor η (in our experiments, we fix $\eta = 0.5$). We denote the lth level of pyramid of I_i as $I_i^l, i \in \{1, 2\}, l \in \{0, 1, \ldots, k-1\}$. The bottom level of the pyramids I_1^0 and I_2^0 are the raw images. Our goal now is to find the matches of every seeds in I_1^0 against I_2^0.

We construct seeds on each level, and we define $\{s^l\}$ the seeds at position $\{p(s^l)\}$ on the lth level as the downscaled version from the *raw* seeds in I_1^0, that is:

$$\{p(s^l)\} = \eta \cdot \{p(s^{l-1})\}, l \geq 1 \tag{2}$$

The seeds on each level preserve the same neighboring relation as the finest level, and the number of seeds is the same on each level. Note that, in our method we do not introduce any seed with sub-pixel accuracy. The position of the seeds on each level is always truncated to the nearest integers. Then there will be some seeds with same positions on high levels when $d * \eta^{k-1} < 1$. With many seeds duplicated, the propagation with random search is performed more extensively on the coarser levels with a low resolution. This is an important feature that can guarantee the robustness of matching results on high levels.

After the construction of the pyramid and the generation of the seeds in each level, we perform the propagation with random search on each level and propagate the flow of each seed from top to bottom on the pyramid. We first set the flow of the seeds $\{s^{k-1}\}$ on the top level as random flow. Then a propagation with random search within the maximum image dimension on this level is performed iteratively. The obtained flow $\{f(s^{k-1})\}$ serve as an initialization of the seeds $\{s^{k-2}\}$ on the next level I^{k-2}, and likewise the computed flow of the seeds in each level always serves as a initialization of the seeds on the next level:

$$\{f(s^l)\} = \frac{1}{\eta} \cdot \{f(s^{l+1})\}, l < k-1 \tag{3}$$

For the seeds on level $l < k-1$, we first initialize the flow of the seeds from higher levels as Equation 3, and then a propagation with random search within a small search radius is performed iteratively to obtain the flow of the seeds on each level. We define r as the search radius of every pyramid level except the top level which has a search radius of the maximum image dimension.

The secret is to constrain r within a small range. This is the most important step in processing the lower levels. The expansive search radius on the coarsest level together with many duplicated seeds can help us to find a rough global optimal initialization. In contrast, on the lower levels, a small search radius around the propagated matching is very helpful for the smoothness of the final matching. It can also help us to avoid finding a poor local optimum far from the propagated matching. While, we find that a too small search radius on lower levels is vulnerable in the case of tiny structures with large motions. This is mainly caused by the failure of recovering the true matching of the tiny structures which is vanished on higher levels. With a proper r, we can find the matching of tiny structures as depicted in Figure 4. A quantitative analysis of the impact of search radius will be discussed in Section 4.

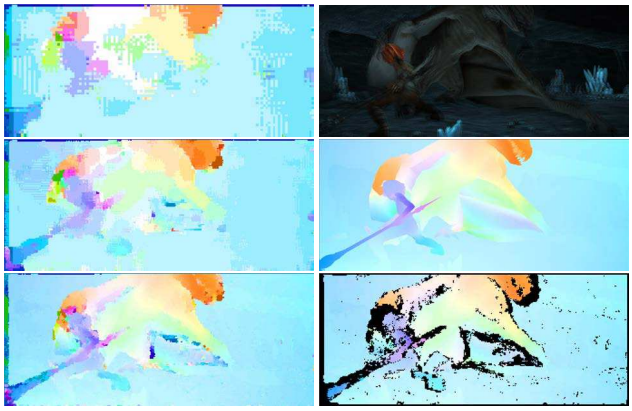


Figure 4. Example of our coarse-to-fine matching. **Left column** shows the matching results on different levels from coarse to fine (from top to bottom), and **right column** shows from top to bottom: mean of the consecutive images, ground truth and the matching results after outlier handling. We can see that, as the matching gets finer, it can recover the matching of the weapon which is vanished on higher levels. Matching correspondences are shown as small colored patches centered at the matching points (color coded as in [2]).

Algorithm 1: Coarse-to-fine PatchMatch Algorithm

Input: a pair of images I_1, I_2, a seed set $\mathcal{S} \in I_1$
Output: matching correspondences $\mathcal{M}$ of $\mathcal{S}$
Construct the image pyramids I_i^l and seeds $\{s^l\}$, $i \in \{1,2\}, l \in \{0,1,\ldots,k-1\}$
for *seeds* $\{s^l\}$ *from* $\{s^{k-1}\}$ *to* $\{s^0\}$ **do**
  if $l = k-1$ **then**
    random initialization
    search within the maximum image dimension
  else
    initialization according to Eqn. 3
    search within radius r
Outlier handling according to Sec. 3.4

3.3. Cost Computation

As an effective feature descriptor, dense SIFT flow [18] has show prominent performance demonstrated in [18, 1, 31]. We choose our patch-based matching cost to be a sum of the absolute difference over all the 128 dimensions of the SIFT flow at the matching points. We use the online code to compute the SIFT feature for every pixel on each hierarchical level[1]. Note that, we use the entire 128 dimensions for matching, and the SIFT feature is computed for every pixel on all hierarchical levels with the same patch size 8×8 in our experiments.

[1] http://people.csail.mit.edu/celiu/SIFTflow/

3.4. Outlier Handling

As in [3, 19, 25], a forward-backward consistency check is performed to detect the occlusions and remove the outliers. While, considering the effect that the matching results is more discriminative and also more robust on higher levels, we perform the consistency check on multi levels of the pyramid simultaneously other than the only check on the finest level.

Similar to [1], only the validation of the matching correspondences on the two finest levels is checked. With backward flow interpolated from matching correspondences linearly, we let the error threshold ϵ of the consistency check equal to the grid spacing d, and the coarser matches are all upscaled to the finest resolution before the consistency check. The matches larger than 400 pixels are also removed. The overall procedure is summarized in Algorithm 1.

4. Experiments

In this section, we evaluate our matching method on three optical flow datasets: MPI-Sintel [6], Middlebury [2] and KITTI [9, 21]. To fill the gaps created by outlier handling we use the EpicFlow [24] to interpolate our matching correspondences to a dense optical flow (termed as *CPM-Flow*).

We optimize the parameters of our method on a subset of the training set of the MPI-Sintel dataset. Then we use the *same* constant parameter settings to generate our matching results for all datasets for evaluation: $\{d, r, k, n\} = \{3, 4, 5, 6\}$. This demonstrates the robustness of our method. We use the online code with default parameters used in EpicFlow[24] for interpolation[2].

All algorithms were run on an Intel Core i7 3.5GHz CPU with a single-core implementation. On average, our matching method including interpolation (EpicFlow) requires only 4.3 seconds for one color image pair (1024×436) from the MPI-Sintel training set. In detail, SIFT flow computation takes 0.7s, coarse-to-fine matching (including forward-backward consistency check) 0.6s, and interpolation (EpicFlow) 3s. We can observe that 70% of the time is spent on interpolation.

4.1. MPI-Sintel Database Experiments

MPI-Sintel dataset [6] is a challenging evaluation benchmark based on an animated movie and contains many large motions. It consists of two versions: *clean* and *final*. Compared to the *clean* version with realistic illuminations and reflections, the *final* version adds rendering effects like motion, defocus blurs and atmospheric effects.

On the MPI-Sintel test set, CPM-Flow currently ranks 1st on the clean version and 2nd on the final version measured in the average endpoint error (AEE) [2]. Table 1 summarizes the main results, and our method outperforms all published methods on the clean version, and we just missed the best approach by 0.15 on the final version but runs much faster. Note that it performs especially well on the occluded areas, thanks to the use of multi-level consistency check in the outlier handling stage and the edge-preserving interpolation.

	Method	AEE All	AEE Noc	AEE Occ	Time
Clean Set	**CPM-Flow**	**3.557**	1.189	**22.889**	4.3s
	DiscreteFlow[22]	3.567	1.108	23.626	∼180s
	FlowFields[1]	3.748	**1.056**	25.700	18s
	EpicFlow[24]	4.115	1.360	26.595	16.4s
	PH-Flow[30]	4.388	1.714	26.202	∼800s
	DeepFlow[27]	5.377	1.771	34.751	19s
	PCALayers[28]	5.730	2.455	32.468	**3.2s**
Final Set	FlowFields[1]	**5.810**	**2.621**	31.799	18s
	CPM-Flow	5.960	2.990	**30.177**	**4.3s**
	DiscreteFlow[22]	6.077	2.937	31.685	∼180s
	EpicFlow[24]	6.285	3.060	32.564	16.4s
	TF+OFM[13]	6.727	3.388	33.929	∼400s
	Classic+NLP[26]	8.291	4.287	40.925	∼800s
	MDPFlow2[29]	8.445	4.150	43.430	∼700s

Table 1. Results on MPI-Sintel test set. AEE-Noc (resp. AEE-Occ) is the AEE on non-occluded areas (resp. occluded areas).

Figure 7 shows a comparison to two state-of-the-art methods. One using an extended coarse-to-fine scheme (MDPFlow2 [29]) and the other is a modern method interpolated from matching correspondences (EpicFlow[24]). Similar to EpicFlow [24], CPM-Flow benefits from the interpolation to produce sharp motion boundaries and correct estimation on occluded areas. While, note how tiny structures are captured by CPM-Flow. Even small details, like the tail of the monster in the middle column and the limbs of the character in the right column, are captured.

Comparison of different matching methods. We compare our matching method with some state-of-the-art matching techniques, and also evaluate the performance of dense optical flow interpolated from these matches (using EpicFlow [24]).

We first compare our matching method with the following matching algorithms: sparse SIFT keypoints [8] matched with FLANN [20] (referred to as SIFT-NN)[3], Kd-tree PatchMatch [10] (KPM)[4] and DeepMatching [27] (DM)[5]. Considering different matching methods often produce matches at different locations, we use a comparison method similar to [27] for fair matching comparison. After assign-

[2] http://lear.inrialpes.fr/src/epicflow/

[3] https://github.com/Itseez/opencv

[4] http://j0sh.github.io/thesis/kdtree/

[5] http://lear.inrialpes.fr/src/deepmatching/

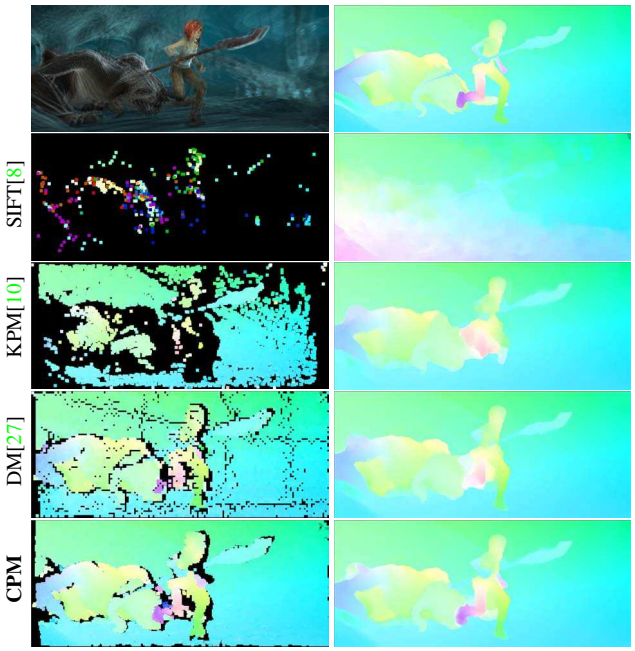
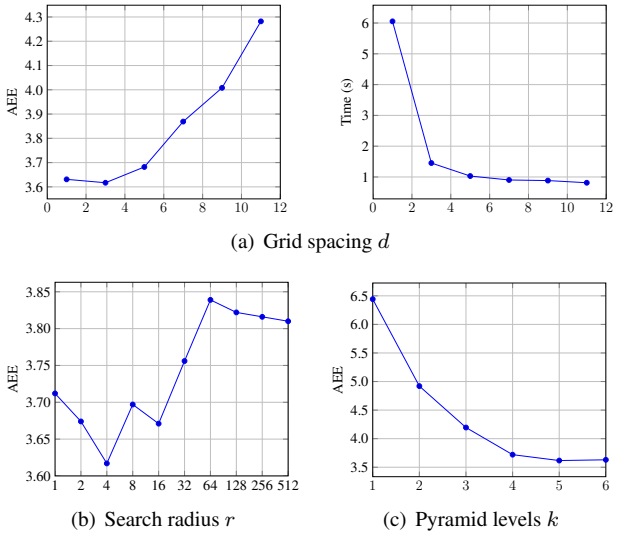


Figure 5. Comparison of different matching methods. The first row is the mean of two consecutive images (left) and the ground truth (right). From the second row, each row shows the matching results (left) and the dense flow results (right) from top to bottom: SIFT-NN [8, 20], KPM [10], DM [27] and our CPM method. The dense flow results are all interpolated from the matching correspondences using EpicFlow[24].

Method	#	Density	Precision	AEE	Time
SIFT-NN[8]	1K	0.175	0.581	29.835	0.5s
KPM[10]	**446K**	**1.000**	0.595	6.961	**0.4s**
DM[27]	5K	0.892	0.945	3.774	15s
CPM	40K	0.886	**0.975**	**3.617**	1.3s

Table 2. Comparison of different matching methods. The "#" column refers to the average number of matches per image. AEE is reported on the final version of the MPI-Sintel training set after interpolation, and interpolation time is not included in the reported time.



(a) Grid spacing d

(b) Search radius r

(c) Pyramid levels k

Figure 6. Parameter sensitivity analysis. Note that, AEE is reported on the final version of the MPI-Sintel training set, and interpolation time is not included in the reported time.

ing each point a fixed grid with a spacing of 10 pixels the nearest neighbor match, *density* is defined as the percentage of points with at least one match in the neighborhood of 10×10, and *precision* is the percentage of those matches with an error below 10 pixels.

Quantitative results are listed in Table 2, and qualitative results in Figures 5. As we can see that, the matching results produced by SIFT is too sparse to obtain a reasonable dense flow after interpolation. To handle the noisiness of KPM which is a dense NNF technique, we perform a forward-backward consistency check before the interpolation. Note that, our matching method can produce comparable matching results with DeepMatching [27] but runs much faster. Furthermore, as Table 2 shows, after interpolation, the obtained dense flow from our matching method is more accurate than that from DeepMatching.

The main reason why our CPM can outperform DeepMatching is that CPM can produce more matches than DeepMatching by default. Note that DeepMatching using the default parameters produces only about 5K matches for one 1024×436 image pair, while CPM can produce about 40K matches (grid spacing $d = 3$). Furthermore, the final flow is interpolated from these matches, and more matches are preferable under the condition of similar density and precision. We think DeepMatching can obtain similar quality as CPM if it can produce more matches by adjusting its default parameters, but at the expense of even worse efficiency.

Parameter sensitivity analysis. In order to get a better understanding of the robustness and scalability of CPM-Flow, we evaluate the impact of different parameter settings of our matching approach to the flow interpolated from the matching correspondences. After obtaining the optimized parameters as reported in Section 4 on a subset (20%) of the clean version training set of the MPI-Sintel dataset, we systematically vary the parameter setting each by one and report the AEE on the final version of the training set. Figure 6 summarizes the results.

We also report the average running time of our matching method with different grid spacing d. As Figure 6(a) shows, a small grid spacing clearly improves the performance, which is contributed by the more extensive search on each level of the pyramid. While a small grid spacing leads to a high computation complexity, especially when there is no spacing ($d = 1$). We find that $d = 3$ hardly

Figure 7. Example of our results on MPI-Sintel. Each column shows from top to bottom: mean of two consecutive images, ground truth, CPM-Flow and 2 state-of-the-art methods (EpicFlow [24] and MDPFlow2 [29]). CPM-Flow is able to capture thin moving objects like the tail of the monster (middle column) and the limbs of the character (right column).

impairs the quality but leads to a significant speed-up.

Figure 6(b) studies the effect of our method with different search radius r. We can see that with the increase of r, the flow shows a clear trend of quality deterioration, which is caused by the more outliers introduced by larger search radius. While we find that a too small $r(< 4)$ also results in quality deterioration. This is mainly caused by that the too small r will cause the match correspondences to be too smooth, and make it fails to recover the matches of small parts which is vanished on higher levels.

Next, we evaluate the effect of our hierarchical method with different number of hierarchical levels k . As we can see in Figure 6(c), when $k = 1$ (single-scale PatchMatch), the method performs poorly, mainly caused by the local ambiguity of patches without any guidance. As k increase, the quality rises contributed by the propagation from high levels which is more discriminative. This confirms the importance of the multi-scale matching strategy. However, the quality will be impaired by too more levels ($k > 5$), due to the failure of recovering small details.

For the iteration numbers n, a larger n always leads to a more accurate matching, while at the price of more running time. We find that after $n = 6$ times of iterations our matching method has almost converged, and it obtains limited gain in accuracy with even larger n.

4.2. Middlebury Database Experiments

Middlebury dataset [2] is a classical optical flow benchmark. It contains complex motions for accurate optical flow estimation, while no large displacements. On Middlebury, our method obtains an average AEE of 0.368 and performs slightly better than EpicFlow (0.393). Like EpicFlow [24], the benefits of our matching method on Middlebury are limited due to the absence of large displacements.

4.3. KITTI Database Experiments

KITTI dataset [9, 21] was created from a driving platform and contains images of city streets. It contains complex lighting conditions and large displacements. Note that the

Method	Out Noc3	Out All3	AEE Noc	AEE All	Time
PH-Flow[30]	**5.76%**	**10.57%**	**1.3**	**2.9**	800s
FlowFields[1]	5.77%	14.01%	1.4	3.5	23s
CPM-Flow	5.79%	13.70%	**1.3**	3.2	**4.2s**
NLTGV-SC[23]	5.93%	11.96%	1.6	3.8	16s*
DiscreteFlow[22]	6.23%	16.63%	**1.3**	3.6	180s
DeepFlow[27]	7.22%	17.79%	1.5	5.8	17s
EpicFlow[24]	7.88%	17.08%	1.5	3.8	15s
TF+OFM[13]	10.22%	18.46%	2.0	5.0	350s

Table 3. Results on KITTI2012 test set. AEE-Noc is the AEE on non-occluded areas. Out-Noc3 (resp. Out-All3) is the percentage of pixels with flow error above 3 pixels on non-occluded areas (resp. all pixels). *Runtime measured on a powerful GPU.

Method	Fl-all	Fl-bg	Fl-fg	Time
DiscreteFlow[22]	**22.38%**	**21.53%**	**26.68%**	180s
CPM-Flow	23.23%	22.32%	27.79%	**4.2s**
EpicFlow[24]	27.10%	25.81%	33.56%	15s
DeepFlow[27]	29.18%	27.96%	35.28%	17s

Table 4. Results on KITTI2015 test set. Fl-bg (resp. Fl-fg) is the percentage of outliers averaged only over background regions (resp. foreground regions).

KITTI dataset consists of two versions: KITTI2012 [9] and KITTI2015 [21].

Table 3 compares our method to state-of-the-art methods that do not use epipolar geometry or stereo vison on the KITTI2012 dataset. Note that we use the *same* matching parameters for both KITTI and MPI-Sintel. Our method clearly outperforms the original EpicFlow [24] as well as most other methods. As can be seen, in addition to the efficiency, our method just missed the best approach by 0.03% in *Out-Noc3* and performs best in terms of AEE on non-occluded areas. We also evaluate our method on the newly emerged KITTI2015 dataset. As Table 4 shows, our method outperforms EpicFlow and is 40+ times faster than the 1st.

5. Conclusion

We present an efficient coarse-to-fine matching framework for optical flow estimation. On the basis of the observation that the matching results is more discriminative on higher pyramidal levels and the optical flow is smooth spatially, we combine an efficient random search with the coarse-to-fine scheme on a sparse image grid structure for optical flow. The evaluation on modern datasets shows that our approach is capable of providing highly accurate optical flow for large displacements and is more efficient than state-of-the-art methods with similar quality. For our future work, we are interested in the parallelization of our matching framework in the GPU for real-time processing. Exploring the proposed matching framework to facilitate other tasks is also an interesting research direction.

Acknowledgements. This work was supported by NSFC Grant (No. 61401337 and No. 61222101), the 111 Project (B08038 and B08028), Program of Innovative Research Team in Shaanxi Province, Fundamental Research Funds for the Central Universities, and Key Laboratory of Infrared System Detection and Imaging Technology, Shanghai Institute of Technical Physics, Chinese Academy of Sciences. This work was also supported partially by Xidian Graduate Innovation Project.

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 4, 5, 8

[2] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 2, 4, 5, 7

[3] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 5

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. In *Proc. of ACM SIGGRAGH*, 2009. 1, 2, 3

[5] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Trans. PAMI*, 2011. 1, 2

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, 2012. 2, 5

[7] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[8] L. DG. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 5, 6

[9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 5, 7, 8

[10] K. He and J. Sun. Computing nearest-neighbor fields via propagationassisted kd-trees. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2, 5, 6

[11] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981. 1, 2

[12] J. Hur, H. Lim, C. Park, and S. C. Ahn. Generalized deformable spatial pyramid: Geometry-preserving dense cor-

respondence estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[13] R. Kennedy and C. J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2015. 5, 8

[14] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[15] S. Korman and S. Avidan. Coherency sensitive hashing. In *IEEE International Conference on Computer Vision (ICCV)*, 2011. 1, 2

[16] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[17] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu. SPM-BP: Sped-up patchmatch belief propagation for continuous MRFs. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[18] C. Liu, J. Yuen, and A. Torralba. SIFT Flow: Dense correspondence across scenes and its applications. *IEEE Trans. PAMI*, 2011. 1, 4

[19] J. Lu, H. Yang, D. Min, and M. N. Do. PatchMatch Filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 5

[20] M. M and L. DG. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISSAPP)*, 2009. 5, 6

[21] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 5, 7, 8

[22] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition (GCPR)*, 2015. 5, 8

[23] R. Ranftl, K. Bredies, and T. Pock. Non-local total generalized variation for optical flow estimation. In *European Conference on Computer Vision (ECCV)*, 2014. 8

[24] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 5, 6, 7, 8

[25] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 5

[26] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 2014. 1, 2, 5

[27] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2, 5, 6, 8

[28] J. Wulff and M. J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 5

[29] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Trans. PAMI*, 2012. 1, 2, 5, 7

[30] J. Yang and H. Li. Dense, accurate optical flow estimation with piecewise parametric model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 5, 8

[31] J. Yang, B. Price, S. Cohen, Z. Lin, and M.-H. Yang. PatchCut: Data-driven object segmentation via local shape transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4