Coarse-to-fine PatchMatch for Dense Correspondence

Article *in* IEEE Transactions on Circuits and Systems for Video Technology · June 2017

DOI: 10.1109/TCSVT.2017.2720175

Coarse-to-fine PatchMatch for Dense Correspondence

Yunsong Li, Yinlin Hu, Rui Song*, Peng Rao and Yangli Wang

***Abstract*—While matching technique has been studied in various areas of computer vision for decades, efficient dense correspondence remains an open problem. In this paper we present a simple but powerful matching method that works in a coarse-to-fine scheme for optical flow and also stereo matching. 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 efficient dense correspondence. Unlike existing NNF techniques, which is efficient but the result is often too noisy 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 dense correspondence than NNF techniques. Furthermore, our approach can also capture the tiny structures with large motions which is a problem for traditional coarse-to-fine methods. Interpolated by an edge-preserving interpolation method (EpicFlow), our method outperforms state-of-the-art optical flow methods on MPI-Sintel and KITTI, and runs much faster than the competing methods.**

***Index Terms*—coarse-to-fine, stereo matching, dense correspondence, optical flow.**

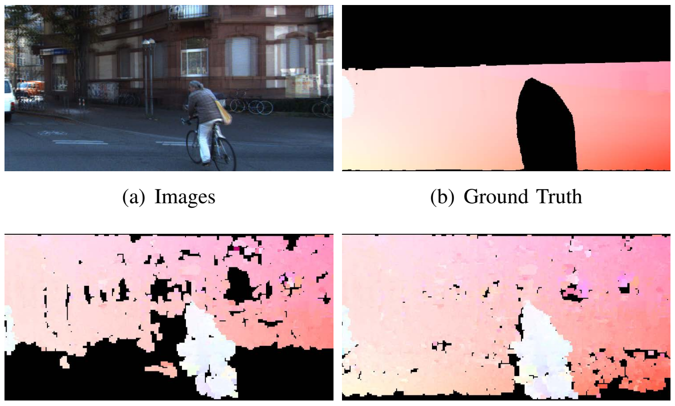

(a) Images (b) Ground Truth (c) PatchMatch [5] (d) Coarse-to-fine PatchMatch

Fig. 1. The comparison between PatchMatch and coarse-to-fine PatchMatch (after outlier removal). (a) and (b) are the means of two consecutive images and the ground truth flow respectively. (c) shows the matching result of single-scale PatchMatch [5], and (d) shows the result of the proposed coarse-to-fine PatchMatch. Notice how the weak matches in texture-less regions (roads) are recovered by (d). Matching correspondences are shown as small colored patches (color follows [6], see Figure 7(a)).

I. Introduction

OPTICAL flow, a formulation of dense correspondence between two consecutive frames in a video, 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.

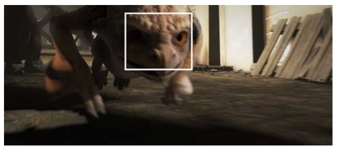
Since the pioneering work by Horn and Schunck [1] who formulated the optical flow estimation as a problem of energy minimization, many effective approaches have emerged for improving the performance with complex motions [2], [3], [4]. Though combined with a coarse-to-fine scheme, such approaches still 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.

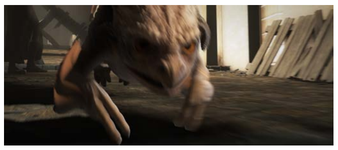
The visual similarity between two image regions is the most important clue for large optical flow estimation. Recently, optical flow interpolated from sparse descriptor matching correspondences directly has shown great success for large displacements, especially with significant occlusions [7]. While 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 [8], the nearest neighbor fields (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 [5] and the improved methods [9], [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 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. Figure 1(c) shows an example of PatchMatch after outlier removal.

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

* Corresponding author. 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.


(a) First Image


(b) Second Image

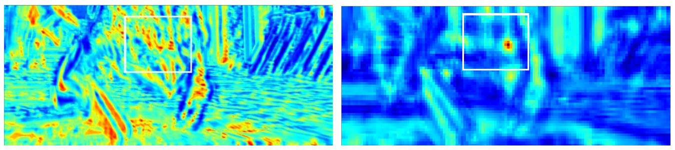

(c) Two examples of response map with different patch size

Fig. 2. Example of response map on the target image with different reference patch size. **Top row** consists of the two consecutive images. **Bottom row** are the response maps of two patches with different size (larger on the right) located at the white rectangle. We can see that the response maps with larger patch size are more discriminative.

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 [11], 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 [5] on each level, our approach, *CPM* (Coarse-to-fine PatchMatch), interpolated using EpicFlow [7] outperforms the original EpicFlow and also most state of the art.

This paper extends its earlier version [12] with the following major differences: **1)** We introduce a strategy of adaptive search radius which can determine the search radius automatically, and experiments show that the adaptive strategy outperforms the fixed one considerably. **2)** We explain our method more illustratively to make it more clear in Section III. **3)** We evaluate the performance of our method with different descriptors to demonstrate that our choice is not ad-hoc, and compare our matching method with other matching methods more extensively. Furthermore, we add more convincing experimental results on stereo matching in Section IV.

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 pixel 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 [13] and KITTI [14], [15].

II. Related Work

We do not review the entire literature on optical flow and only discuss the related literature. We refer to publications like [3], [6] for a detailed overview of optical flow methods based on the classic variational minimization framework first proposed in [1].

Our method is inspired by NNF estimation. The efficiency of computing NNF has advanced remarkably through the work [5], [9], [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.* [16] used the computed NNF as a hint for motion segmentation before variational optimization. Bao *et al.* [17] 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. Our method is closely related to FlowFields [18] which proposed a similar pipeline to ours to handle the noisiness of NNF. Unlike FlowFields which relies on a dedicated hierarchical structure and many complicated techniques, our method just use some simple natural heuristics for ambiguity handling.

Also inspired by PatchMatch [5], Li *et al.* [19] proposed a PatchMatch belief propagation to estimate the optical flow which was formulated as a labeling problem. Yang *et al.* [20] 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 [2] introduced a descriptor matching term to the classic variational minimization framework of optical flow. Furthermore, Xu *et al.* [4] 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. On the other hand, Hu *et al.* [21] proposed a filtering framework for superpixel matching. To handle the sparsity of descriptor matching, Leordeanu *et al.* [22] extended sparse matching to dense matching with a locally affine constraint. For efficiency, Wulff *et al.* [23] 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.* [7] introduced an edge-preserving interpolation framework for optical flow based on a matching algorithm termed DeepMatching [11]. However, the interpolation results is mainly constrained by the efficiency of DeepMatching. Closely related to our method, Kim *et al.* [24] and Hur *et al.* [25] also investigated hierarchical matching, but their methods requires inexact inference using loopy belief propagation.

III. 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 derivate PatchMatch in Section III-A, and then detail the matching procedure on one level of the pyramid in Section III-B, and finally describe the hierarchical structures

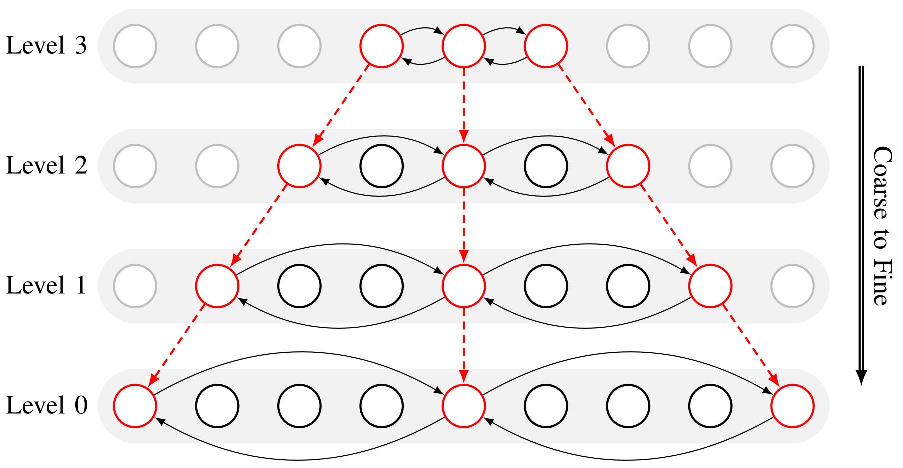


Fig. 3. Overview of the proposed CPM algorithm. After the construction of the pyramid, we choose seed points on each level (simple grid). The red circles represent the seeds, and the *neighbor propagation* is performed between adjacent seeds (black arrows) to propagate good matches to adjacent seeds. For the coarse-to-fine scheme, the seeds on lower levels are initialized by high-level seeds (red arrows).

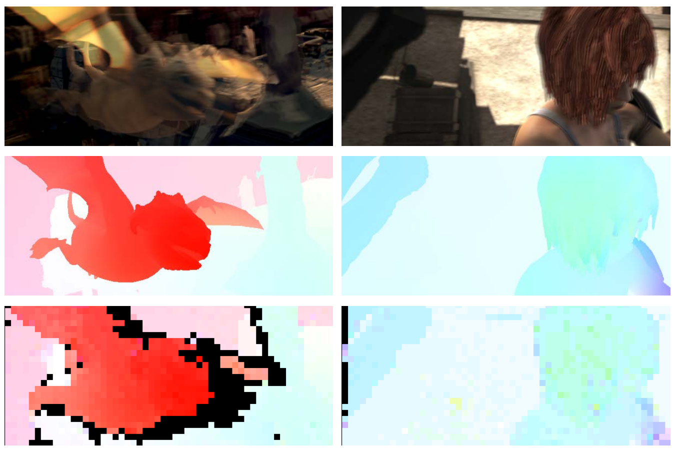


Fig. 4. Example of PatchMatch on grid structure. Each column shows from top to bottom: mean of two consecutive images, ground truth and the matching results of CPM with $d = 11$. Note that, even with so large grid spacing, the result can capture most of the motion details.

of our approach as well as the propagation step between the levels in Section III-C.

A. PatchMatch

The objective of nearest neighbor fields (NNF) algorithms is the find of all patches in one image for the most similar patch in another image under some patch distance metric. Connelly *et al.* proposed a randomized algorithm, called PatchMatch [5], to efficiently compute approximate NNF.

PatchMatch has three main components. First, every pixel in the image is initialized with either random matching offsets or some prior information. Next, an iterative refinement process is applied over every pixel in an interleaved zig-zag and reverse zig-zag manner, in which good matches are propagated to adjacent pixels, followed by random search near the best matching offset so far.

neighborhood propagation and *random search* are the core idea of PatchMatch. Propagation is under the assumption that the patch offsets are likely to be the same. For example, if there is a good matching near a pixel, the matching of this pixel will be updated by the matching of the neighbor if the patch distance (in some patch distance metric) of the neighbor is smaller. After propagation, a sequence of randomly selected candidate offsets will be tested to improve the current matching. The propagation and random search are performed iteratively until convergence or some fixed iteration numbers. We refer readers to [5] for more detailed information.

PatchMatch has shown prominent advantages in scene correspondence [8] and structural image editing [5]. However, the computed NNF is often very noise due to the lack of global regularization. We introduce a coarse-to-fine structure into the PatchMatch framework, which can reduce the noise significantly (see Figure 1(d)).

B. PatchMatch on Grid Structure

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 (see Figure 4). 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 III-C).

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

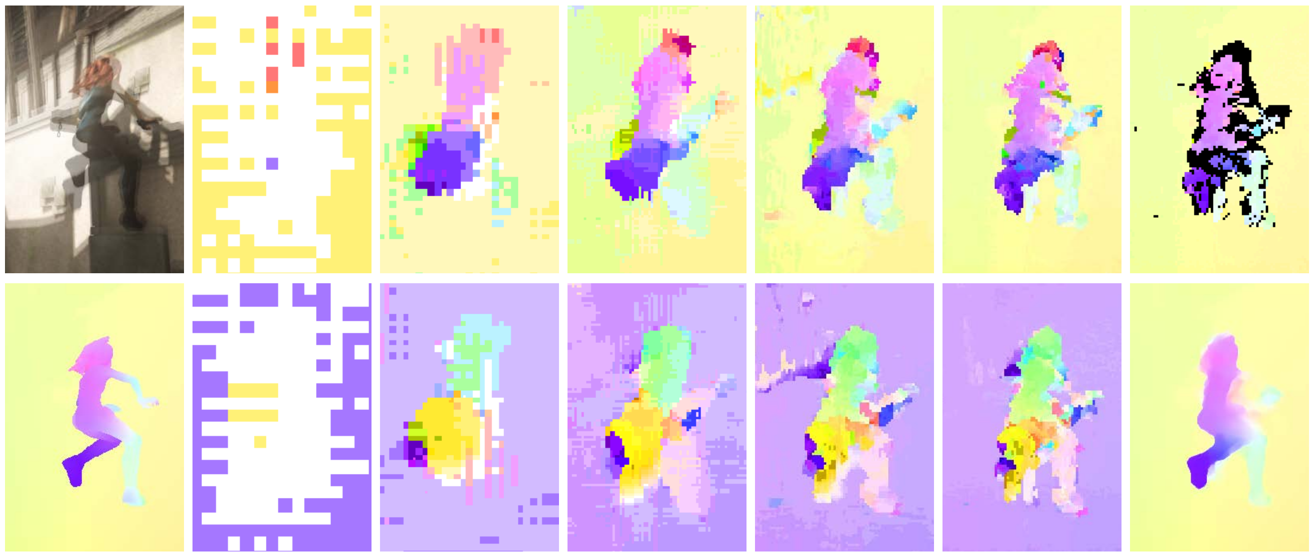


Fig. 5. Example of coarse-to-fine PatchMatch. The first column shows the mean of two consecutive images and the ground truth. The following columns show the matching results on different levels from coarse to fine (from left to right, the top row is the forward matching and the bottom row is the backward matching). The last column shows the matching results after outlier handling and the flow results after interpolation with Epic [7]. Note how the motion details are recovered gradually as the matching gets finer.

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

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

where $C(f(\cdot))$ denote the matching 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 matching cost in Section IV.

After the preceding propagation step, a random search as in PatchMatch [5] 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 [5], 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.

C. Coarse-to-fine Scheme

Our basic matching is similar to PatchMatch [5] 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 results.

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 seed in I_1^0 over 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 (see Figure 3):

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

For 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

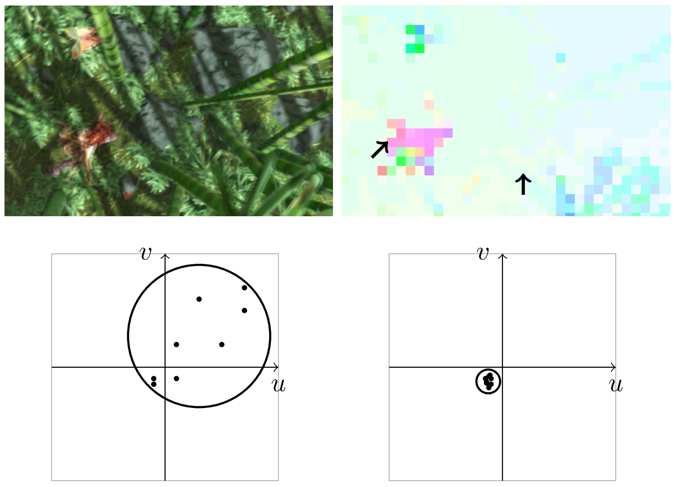


Fig. 6. Demonstration of the adaptive search radius. The top row is the mean of two consecutive images and the matching results on some pyramid level after initialization. The search radius of the two marked seeds are shown in the bottom row, which is the radius of the minimum circle containing all the neighboring matches.

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. Figure 5 shows an examples of coarse-to-fine matching.

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.

The determination of r is not trivial. Unlike the fixed r for all seeds as in our conference paper [12], we propose a strategy of adaptive search radius for each seed. We set the search radius of each seed as the radius of the minimum circle containing all the initial neighboring matches of the seed (see Figure 6). This adaptive search radius is closely related to the matching consistency near the seed, which can determine the search radius of each seed effectively. The seeds near motion discontinuities usually have large search radius, which can help them to find more reasonable matches. On the other hand, the seeds with consistent neighboring matches will have compact search radius, which is beneficial to computational efficiency and also less ambiguities. Section IV demonstrates the advantages of this strategy.

As in [17], [26], [27], a simple forward-backward consistency check is performed to detect the occlusions and remove the outliers, and the matches larger than 400 pixels are also removed. The overall procedure of CPM is summarized in Algorithm 1.

For stereo matching, it is similar to optical flow with the except that the result of stereo matching have no vertical movement (after stereo calibration). Our matching framework can be used in stereo matching directly after some simple adaption. Instead of random initialization, we initialize the seed matching on the top level with no vertical movement. Furthermore, we pick matching candidates in the procedure of *random search* along the horizontal line through the seed itself. Figure 7 shows the effect of this adaption.

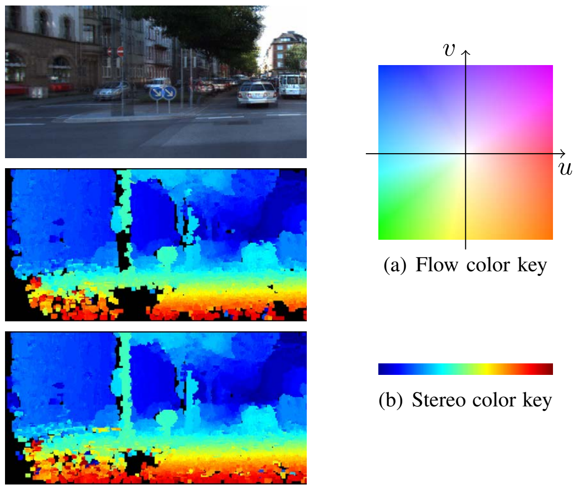


Fig. 7. Comparison of CPM in stereo case. In left column, the top row shows the means of two consecutive images and the following two rows show the results of raw CPM and CPM with stereo in consideration. The right column shows the color key of optical flow and stereo matching.

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 adaptive radius
Outlier handling
```

IV. Experiments

In this section, we evaluate our matching method on two datasets: MPI-Sintel [13] and KITTI [14], [15].

MPI-Sintel dataset 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.

KITTI dataset was created from a driving platform and contains images of city streets. It contains complex lighting conditions and large displacements. Note that the KITTI dataset consists of two versions: KITTI2012 [14] and KITTI2015 [15]. We test our method on KITTI for both flow evaluation and stereo evaluation.

TABLE I

CPM WITH DIFFERENT DESCRIPTORS ON MPI-SINTEL. THE "#" COLUMN REFERS TO THE AVERAGE NUMBER OF MATCHES PER IMAGE. THE AEE (AVERAGE ENDPOINT ERROR) AFTER INTERPOLATION (EPICFLOW) IS ALSO REPORTED. ERROR RATIO IS THE PERCENTAGE OF PIXELS WITH FLOW ERROR ABOVE 3 PIXELS.

Features	#	Density	Precision	AEE	Error Ratio
SIFT-F [8]	**39.4k**	**0.900**	0.953	**3.437**	9.97%
MLDP [28]	38.2k	0.882	**0.961**	3.451	**9.74%**
CRT [29]	39.0k	0.894	0.957	3.508	9.87%
CT [30]	38.7k	0.891	0.956	3.532	9.99%
RGB	37.6k	0.893	0.884	5.356	16.83%

TABLE II

CPM WITH DIFFERENT DESCRIPTORS ON KITTI FLOW DATASET.

(a) KITTI-2012

	Non-occluded		Occluded	
Features	AEE	Error Ratio	AEE	Error Ratio
SIFT-F [8]	**1.231**	**5.37%**	**2.850**	**12.38%**
MLDP [28]	1.276	5.77%	3.023	13.11%
CRT [29]	1.389	6.21%	3.299	13.63%
CT [30]	1.517	6.74%	3.507	14.32%
RGB	4.191	15.97%	7.351	23.33%

(b) KITTI-2015

	Non-occluded		Occluded	
Features	AEE	Error Ratio	AEE	Error Ratio
SIFT-F [8]	**3.117**	**13.11%**	**7.167**	**21.25%**
MLDP [28]	3.712	14.54%	8.374	22.94%
CRT [29]	3.801	15.00%	8.686	23.35%
CT [30]	4.063	15.79%	9.042	24.08%
RGB	7.505	26.14%	14.397	33.57%

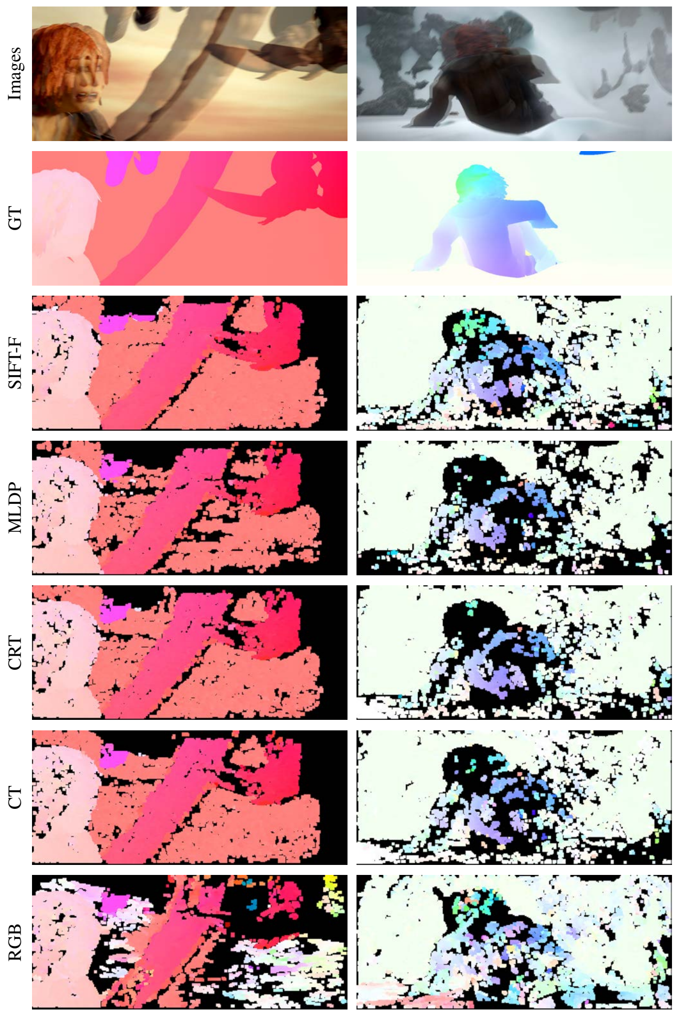


Fig. 8. Example of CPM results with different descriptors. Each column shows from top to bottom: mean of two consecutive images, ground truth, the matching results of CPM with descriptor: SIFT-F [8], MLDP [28], CRT [29], CT [30] and RGB. Note the difference of *density* and *precision* of the matching results.

To fill the gaps created by outlier handling we use EpicFlow [7] 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: $\{d, k, n\} = \{3, 5, 6\}$. This demonstrates the robustness of our method. We use the online code with default parameters used in EpicFlow [7] for interpolation[1].

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 4.5 seconds for one color image pair (1024×436) from the MPI-Sintel training set. In detail, descriptor computation takes 0.7s, coarse-to-fine matching (including forward-backward consistency check) 0.8s, and interpolation (EpicFlow) 3s. We can observe that 70% of the time is spent on interpolation. The source code of CPM is available at https://github.com/yinlinhu/cpm.

A. *Evaluation of different descriptors*

First, we perform experiments to select a reasonable descriptor for our matching method. Considering different matching methods often produce matches at different locations, we use a comparison method similar to [11] for fair matching comparison. After assigning 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 5 pixels.

We compare the performance of our CPM matching method with five different descriptors: SIFT flow (SIFT-F) [8] [2], MLDP [28], Complete Rank Transform (CRT) [29], Census Transform (CT) [30] and RGB. The simple RGB is only used for baseline comparison. We choose the patch-based matching cost to be the sum of absolute differences over all the dimensions of the descriptor at the matching points.

Table I and II show the performance of each descriptor on MPI-Sintel and KITTI flow dataset, and Figure 8 shows some examples of the matching results. We can see that, SIFT-F is the best performer, which has also shown prominent performance demonstrated in [8], [18], [31]. In the following

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

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

TABLE III
COMPARISON OF DIFFERENT MATCHING METHODS ON MPI-SINTEL.

Method	#	Density	Precision	AEE	Time
CPM	39.4k	0.900	**0.953**	**3.437**	1.5s
DeepMatching [11]	5.9k	0.876	0.911	3.774	15s
KPM [10]	**446.5k**	**1.000**	0.537	6.961	**0.4s**
SIFT-NN [32]	1.2k	0.156	0.564	29.835	0.5s

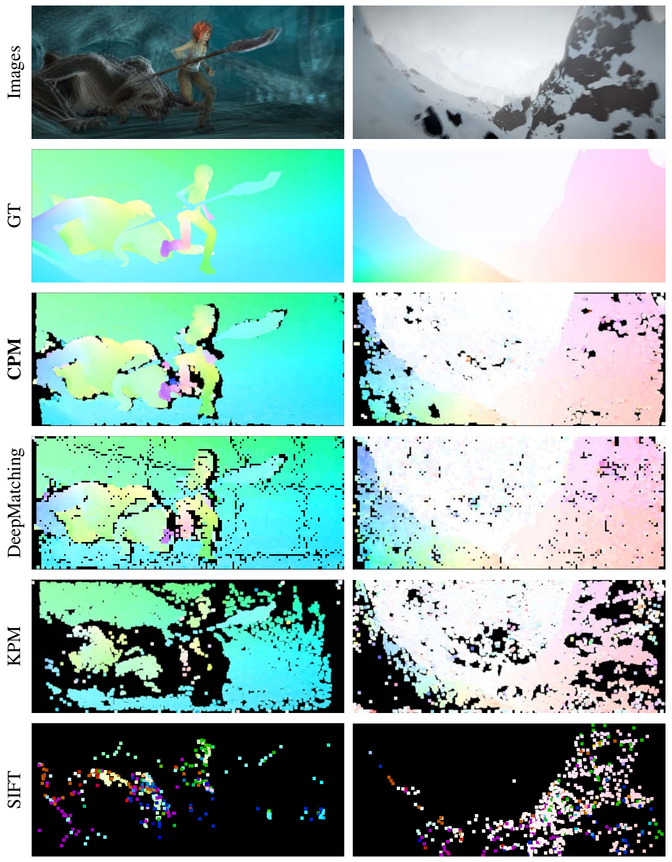


Fig. 9. Comparison of different matching methods. Each column shows from top to bottom: mean of two consecutive images, ground truth, the matching results of CPM, DeepMatching [11], KPM [10] and SIFT [32]. Unlike SIFT and KPM, out CPM can produce highly dense matching correspondences. Similar to DeepMatching, CPM can produce reasonable matching even in flat regions (last column). Furthermore, as Table. III shows, the matching correspondences produced by CPM are more accurate than DeepMatching.

experiments, we will use SIFT-F as our default descriptor except state explicitly.

B. Comparison of different matching

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 [7]).

We first compare our matching method with the following matching algorithms: sparse SIFT keypoints [32] matched with FLANN [33] (referred to as SIFT-NN), Kd-tree PatchMatch [10] (KPM)[3] and DeepMatching [11] (DM)[4].

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

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

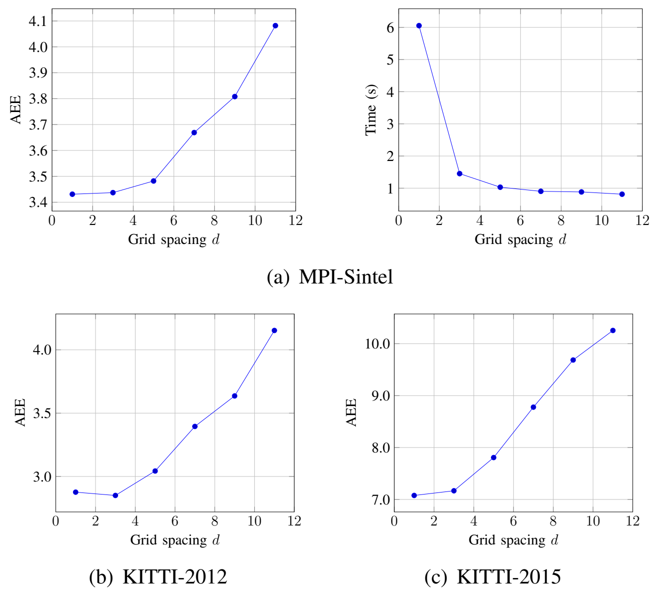


Fig. 10. Effect of different grid spacing on different flow dataset. Note that $d = 3$ hardly impairs the quality but leads to a significant speed-up.

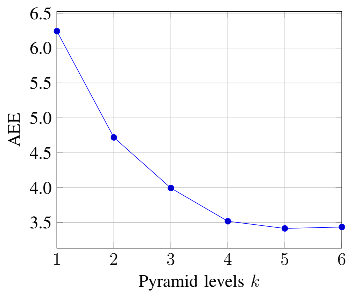
Quantitative results are listed in Table III, and qualitative results in Figure 9. As we can see that, the matching results produced by SIFT is too sparse to obtain a reasonable dense flow after interpolation. To handle the noisiness of KPM which is a dense NNF technique, we perform a forward-backward consistency check before the interpolation. Note that, our matching method can produce comparable matching results with DeepMatching [11] but runs much faster. Furthermore, as Table III 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.

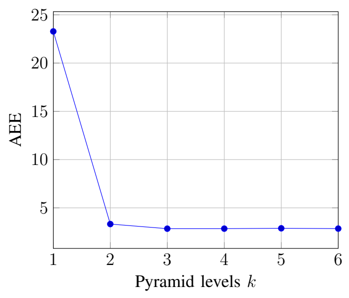
C. Parameter sensitivity analysis

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 IV 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 of MPI-Sintel and also KITTI.

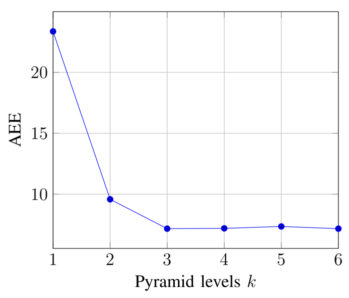
Figure 10 report the performance of CPM with different grid spacing d, the average running time on MPI-Sintel is also



(a) MPI-Sintel



(b) KITTI-2012



(c) KITTI-2015

Fig. 11. Effect of different pyramid levels on different flow dataset. More pyramid levels shows consistent performance gain on the three dataset. While, less pyramid levels is enough for good performance on KITTI.

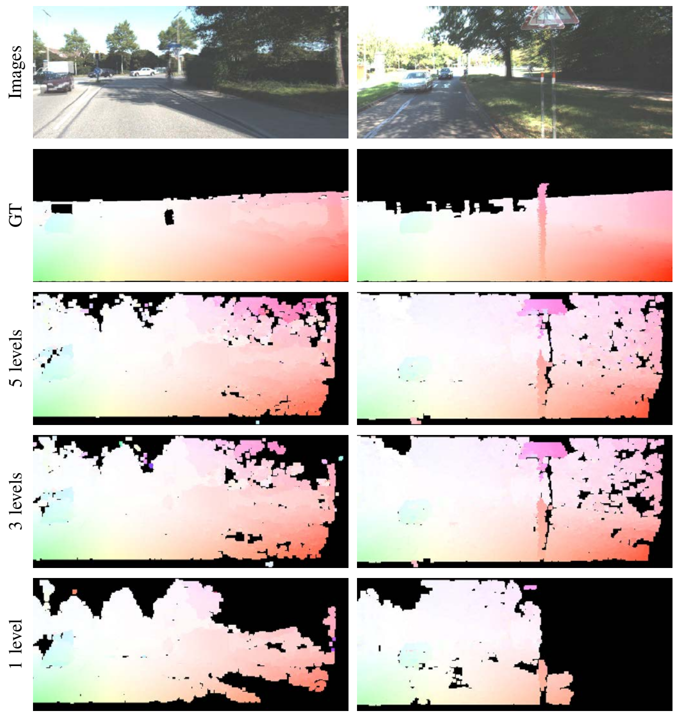


Fig. 12. Effect of different pyramid levels. Each column shows from top to bottom: mean of two consecutive images, ground truth, the matching result of CPM with $k = 5$, $k = 3$ and $k = 1$. We can see that, the single-scale PatchMatch ($k = 1$) have much less matches due to matching ambiguities. On the other hand, as the pyramid levels k gets larger, the effective *density* of the matching result is significantly increased. The sparse ground truth from KITTI dataset are also showed as small colored patches as Figure 1.

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

Table IV studies the effect of our method with different search radius r and the advantages of adaptive strategy. 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.

TABLE IV
COMPARISON OF THE FIXED AND ADAPTIVE SEARCH RADIUS. THE ADAPTIVE STRATEGY CLEARLY OUTPERFORMS THE FIXED ONE.

	Radius	MPI-Sintel	KITTI-2012	KITTI-2015
Fixed	1	3.612	3.198	8.530
	2	3.574	3.017	7.866
	4	<u>3.517</u>	2.982	7.548
	8	3.597	<u>2.909</u>	7.420
	16	3.571	3.014	<u>7.333</u>
	32	3.656	3.029	7.551
	64	3.739	3.187	7.832
	128	3.722	3.238	7.953
	Adaptive	**3.437**	**2.850**	**7.167**

The problem is that the best search radius is not easy to determine across different datasets. Compared to this fixed strategy, the results with adaptive search radius outperforms that with fixed search radius on all three datasets.

Next, we evaluate the effect of our hierarchical method with different number of hierarchical levels k . As we can see in Figure 11 and 12, 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.

D. *Results on datasets*

1) Flow results on MPI-Sintel: Interpolated from CPM, CPM-Flow is currently among the best methods measured in the average endpoint error (AEE) [6] on the MPI-Sintel test set. Table V summarizes the main results. Our method outperforms all published methods on the clean version, and the CPM with adaptive search radius (termed *CPM2*) clearly

TABLE V

RESULTS ON MPI-SINTEL TEST SET. AEE-NOC (RESP. AEE-OCC) IS THE AEE ON NON-OCCLUDED AREAS (RESP. OCCLUDED AREAS).

	Method	AEE All	AEE Noc	AEE Occ	Time
Clean Set	**CPM2**	**3.253**	**0.980**	**21.812**	4.5s
	SPM-BP2[19]	3.515	1.020	23.865	42s
	DCFlow[34]	3.537	1.103	23.394	9s
	RicFlow[35]	3.550	1.264	22.220	5s
	CPM-Flow[12]	3.557	1.189	22.889	**4.3s**
	DiscreteFlow[36]	3.567	1.108	23.626	180s
	FullFlow[37]	3.601	1.296	22.424	240s
	FlowFields[18]	3.748	1.056	25.700	18s
	CNN-HPM[38]	3.778	0.996	26.469	23s
	DDF[39]	3.863	1.296	24.820	60s
	EpicFlow[7]	4.115	1.360	26.595	16.4s
Final Set	DCFlow[34]	**5.119**	**2.283**	**28.228**	9s
	CNN-HPM[38]	5.363	2.303	30.313	23s
	RicFlow[35]	5.620	2.765	28.907	5s
	DDF[39]	5.728	2.623	31.042	60s
	FlowFields[18]	5.810	2.621	31.799	18s
	SPM-BP2[19]	5.812	2.754	30.743	42s
	FullFlow[37]	5.895	2.838	30.793	240s
	CPM-Flow[12]	5.960	2.990	30.177	**4.3s**
	DiscreteFlow[36]	6.077	2.937	31.685	180s
	CPM2	6.180	3.012	32.008	4.5s
	EpicFlow[7]	6.285	3.060	32.564	16.4s

TABLE VI

RESULTS ON KITTI FLOW 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). FL-BG (RESP. FL-FG) IS THE PERCENTAGE OF OUTLIERS AVERAGED ONLY OVER BACKGROUND REGIONS (RESP. FOREGROUND REGIONS).

(a) KITTI-2012

Method	Out Noc3	Out All3	AEE Noc	AEE All	Time
CNN-HPM [38]	**4.89%**	13.01%	**1.2**	3.0	23s
RicFlow [35]	4.96%	13.04%	1.3	3.2	5s
PatchBatch [40]	5.29%	14.17%	1.3	3.3	50s
CPM2	5.60%	13.52%	1.3	3.3	4.5s
DDF [39]	5.73%	14.18%	1.4	3.4	60s
PH-Flow [20]	5.76%	**10.57%**	1.3	**2.9**	800s
FlowFields [18]	5.77%	14.01%	1.4	3.5	23s
CPM-Flow [12]	5.79%	13.70%	1.3	3.2	**4.2s**
NLTGV-SC [41]	5.93%	11.96%	1.6	3.8	16s
DiscreteFlow [36]	6.23%	16.63%	1.3	3.6	180s
DeepFlow [11]	7.22%	17.79%	1.5	5.8	17s
EpicFlow [7]	7.88%	17.08%	1.5	3.8	15s

(b) KITTI-2015

Method	Fl-all	Fl-bg	Fl-fg	Time
DCFlow [34]	**14.86%**	**13.10%**	23.70%	8.6s
CNN-HPM [38]	18.68%	18.33%	20.42%	23s
RicFlow [35]	18.79%	18.73%	**19.09%**	5s
CPM2	19.91%	19.22%	23.37%	4.5s
PatchBatch [40]	21.07%	19.98%	26.50%	50s
DDF [39]	21.17%	20.36%	25.19%	60s
DiscreteFlow [36]	22.38%	21.53%	26.68%	180s
CPM-Flow [12]	23.23%	22.32%	27.79%	**4.2s**
FullFlow [37]	24.26%	23.09%	30.11%	240s
EpicFlow [7]	27.10%	25.81%	33.56%	15s
DeepFlow [11]	29.18%	27.96%	35.28%	17s

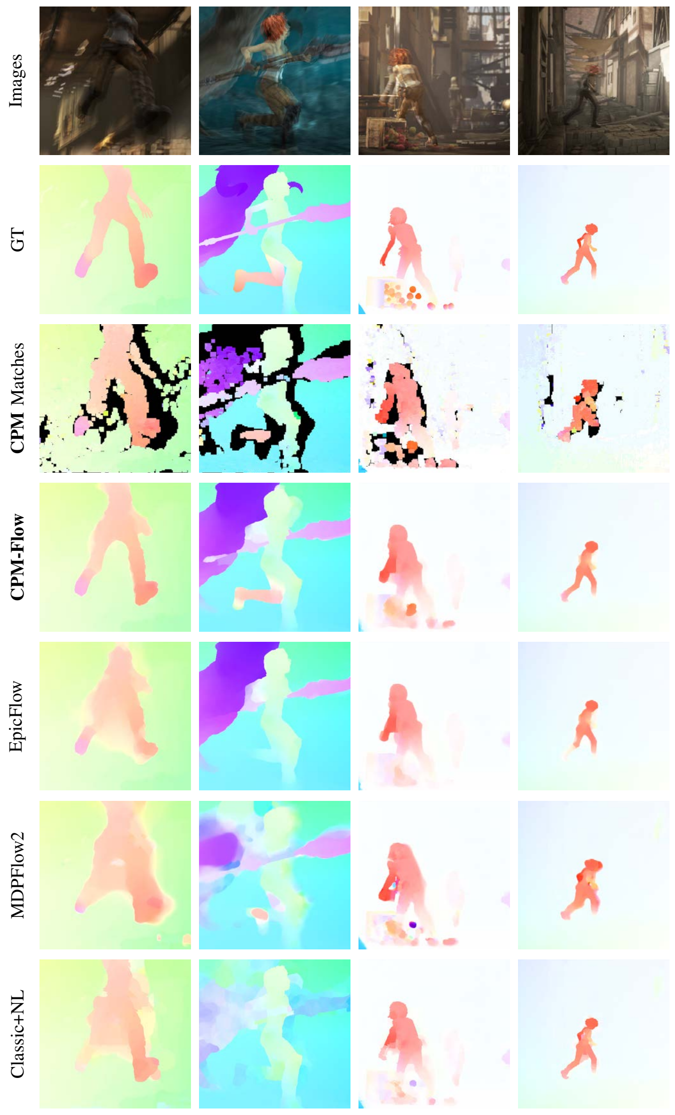


Fig. 13. Example of our results on MPI-Sintel. Each column shows from top to bottom: mean of two consecutive images, ground truth, CPM matches, CPM-Flow and 3 state-of-the-art methods (EpicFlow [7], MDPFlow2 [4] and Classic+NL [3]). In addition to the sharp motion boundaries which is similar to EpicFlow, CPM-Flow can survive in the case of tiny structures with large motions (like the limbs of the character) which is usually a problem.

outperforms the conference version CPM-Flow. On the final version, CPM2 performs slightly worse than CPM-Flow.

Figure 13 shows a comparison to two state-of-the-art methods. One using an extended coarse-to-fine scheme (MDPFlow2 [4]) and the other is a modern method interpolated from matching correspondences (EpicFlow [7]). Similar to EpicFlow [7], 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.

2) *Flow results on KITTI:* Table VI compares our method to state-of-the-art methods that do not use epipolar geometry or stereo vison on the KITTI flow dataset. Note that we use the *same* matching parameters for both KITTI and MPI-Sintel.

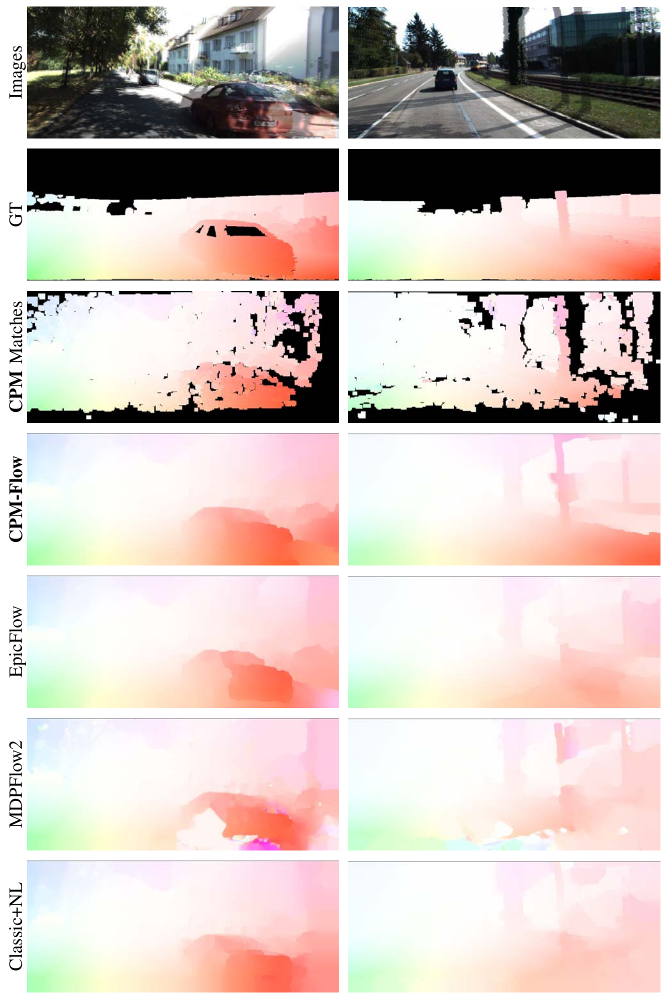


Fig. 14. Some qualitative results on KITTI flow dataset. Each column shows from top to bottom: mean of two consecutive images, ground truth, CPM Matches, CPM-Flow and 3 state-of-the-art methods (EpicFlow [7], MDPFlow2 [4] and Classic+NL [3]). Note how dense correspondences in flat regions (like the wall and the road) are captured by CPM.

Our method clearly outperforms the original EpicFlow [7] as well as most other methods. As can be seen, the CPM2 outperforms the original CPM-Flow, especially on KITTI-2015. Figure 14 shows some qualitative results on KITTI.

3) Stereo results on KITTI: As an integer matching method, CPM alone is not suited for competing with other stereo methods. However, CPM can capture most of the motion details quickly and can also be interpolated to obtain the final stereo results like in optical flow estimation. Compared with Epic which is not intended for stereo matching, SPS [42] can often produce more reasonable stereo matching results. Figure 16 shows an example. Although the result of Epic is more eye-pleasing, its performance is sacrificed by smooth motion boundaries.

We use SPS as our stereo interpolator. Rather than constructing the whole cost volume in raw SPS, we can construct only a tiny fraction of the whole cost volume guided by the CPM initialization. In our experiments, we only compute the cost of disparities near the corresponding CPM Matching (± 3 pixels). Figure 15 shows some examples of the CPM stereo matching. As Table VII shows, our CPM have similar performance as the raw SPS method, which demonstrates the effectiveness of the CPM initialization. Using more powerful tools for cost computation like MC-CNN [43] may increase the performance, while we do not investigate it further.

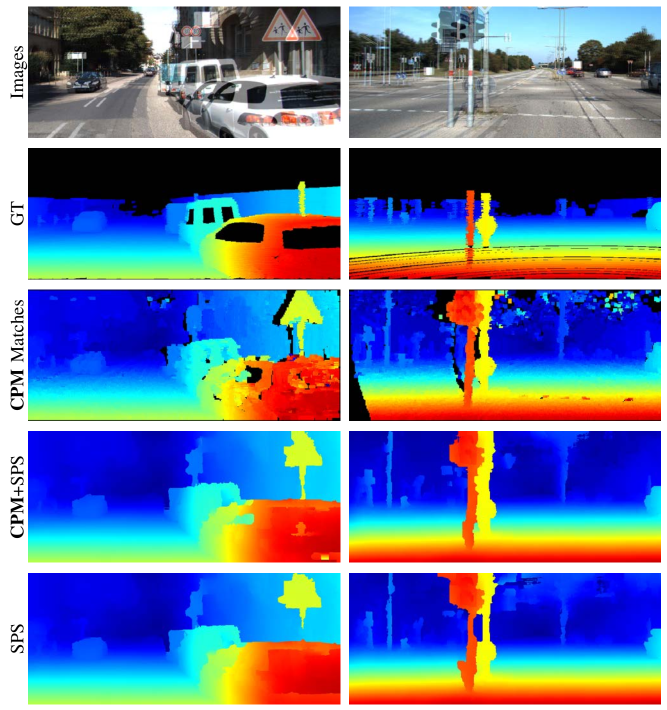


Fig. 15. Results of CPM stereo matching. Each column shows from top to bottom: mean of two consecutive images, ground truth, CPM results, SPS [42] using CPM as initialization and the raw SPS method. Using CPM alone can capture most of the disparities.

E. Failure cases

Although CPM works well in most cases, it still fails in some challenging situations. Figure 17 shows some examples where CPM fails in the regions with glass reflections or large motion deformations. For glass reflection on KITTI dataset, introduction of some restriction like epipolar information may be help. For large motion deformation, we need more discriminative descriptors (like MC-CNN [43] *etc.*), which is also our nearly future work.

V. Conclusion

We present an efficient coarse-to-fine matching framework for dense correspondence. Observe that the matching results is more discriminative on higher pyramidal levels and the dense correspondence is smooth spatially, we combine an efficient random search with the coarse-to-fine scheme on a sparse image grid structure for the computation of dense correspondence. The evaluation on modern optical flow and stereo datasets shows that our approach is capable of providing highly accurate results after interpolation, 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.

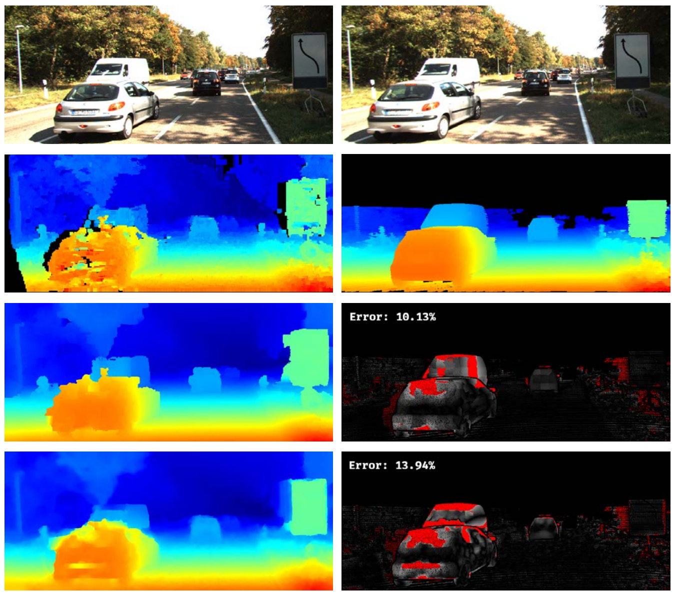


Fig. 16. Example of different stereo interpolator. The first row shows two consecutive images, and the second row shows the matching result of CPM and the ground truth. The third row shows the stereo result and error map of SPS [42] using CPM as its initialization, and the last row shows the result from Epic [7]. Although the result of Epic is more eye-pleasing, its performance is sacrificed by smooth motion boundaries.

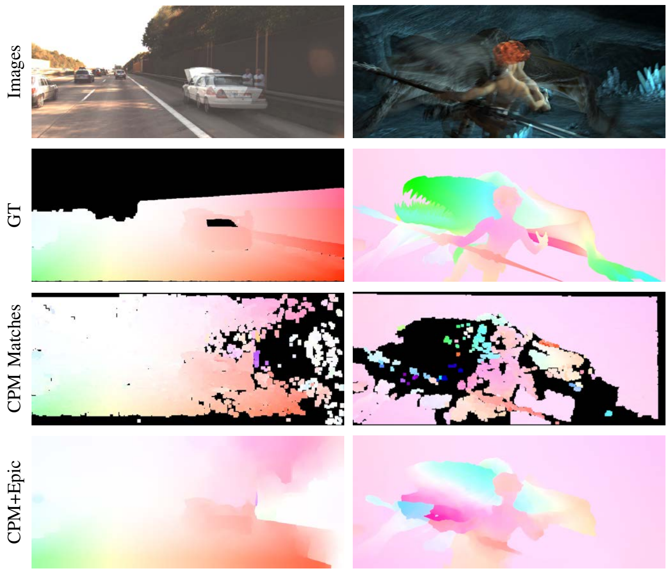


Fig. 17. Example of failure cases. Each column shows from top to bottom: mean of two consecutive images, ground truth, CPM matching results and the flow results interpolated by Epic. The failure is mainly caused by glass reflections and large motion deformations.

TABLE VII
RESULTS ON KITTI STEREO 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). FL-BG (RESP. FL-FG) IS THE PERCENTAGE OF OUTLIERS AVERAGED ONLY OVER BACKGROUND REGIONS (RESP. FOREGROUND REGIONS).

(a) KITTI-2012

Method	Out Noc3	Out All3	AEE Noc	AEE All	Time
MC-CNN [43]	**2.61%**	3.84%	0.8	1.0	100s
PRSM [44]	2.78%	**3.00%**	**0.7**	**0.7**	300s
Content-CNN [45]	3.07%	4.29%	0.8	1.0	**0.7s**
SPS-St [42]	3.39%	4.41%	0.9	1.0	2s
CPM2	3.58%	4.41%	0.9	1.1	1.8s
PCBP [46]	4.04%	5.37%	0.9	1.1	300s
PR-Sceneflow [47]	4.36%	5.22%	0.9	1.1	150s

(b) KITTI-2015

Method	Fl-all	Fl-bg	Fl-fg	Time
PRSM [44]	**4.27%**	**3.02%**	10.52%	300s
Content-CNN [45]	4.54%	3.73%	**8.58%**	**1s**
SPS-St [42]	5.31%	3.84%	12.67%	2s
CPM2	5.44%	4.13%	12.03%	1.8s
OSF [15]	5.79%	4.54%	12.03%	3000s
PR-Sceneflow [47]	6.24%	4.74%	13.74%	150s

REFERENCES

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

[2] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.

[3] 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.

[4] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

[5] 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.

[6] 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.

[7] 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.

[8] C. Liu, J. Yuen, and A. Torralba, "SIFT Flow: Dense correspondence across scenes and its applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.

[9] S. Korman and S. Avidan, "Coherency sensitive hashing," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[10] K. He and J. Sun, "Computing nearest-neighbor fields via propagation assisted kd-trees," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] 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.

[12] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine PatchMatch for large displacement optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] 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.

[14] 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.

[15] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

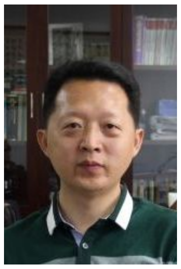
[16] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[17] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving PatchMatch for large displacement optical flow," *IEEE Trans. on Image Processing (TIP)*, 2014.

[18] 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.

[19] 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.
[20] 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.
[21] Y. Hu, R. Song, Y. Li, P. Rao, and Y. Wang, "Highly accurate optical flow estimation on superpixel tree," *Image and Vision Computing (IVC)*, 2016.
[22] 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.
[23] 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.
[24] 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.
[25] J. Hur, H. Lim, C. Park, and S. C. Ahn, "Generalized deformable spatial pyramid: Geometry-preserving dense correspondence estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[26] 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.
[27] 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.
[28] M. A. Mohamed, H. A. Rashwan, B. Mertsching, M. A. García, and D. Puig, "Illumination-robust optical flow using a local directional pattern," *IEEE Trans. on Circuits and Systems for Video Technology (CSVT)*, 2014.
[29] O. Demetz, D. Hafner, and J. Weickert, "The complete rank transform: A tool for accurate and morphologically invariant matching of structures," in *British Machine Vision Conference (BMVC)*, 2013.
[30] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *European Conference on Computer Vision (ECCV)*, 1994.
[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.
[32] L. DG, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.
[33] M. M and L. DG, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Applications (VISSAPP)*, 2009.
[34] J. Xu, R. Ranftl, and V. Koltun, "Accurate optical flow via direct cost volume processing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[35] Y. Hu, Y. Li, and R. Song, "Robust interpolation of correspondences for large displacement optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[36] M. Menze, C. Heipke, and A. Geiger, "Discrete optimization for optical flow," in *German Conference on Pattern Recognition (GCPR)*, 2015.
[37] Q. Chen and V. Koltun, "Full Flow: Optical flow estimation by global optimization over regular grids," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[38] C. Bailer, K. Varanasi, and D. Stricker, "CNN-based patch matching for optical flow with thresholded hinge embedding loss," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[39] F. Güney and A. Geiger, "Deep discrete flow," in *Asian Conference on Computer Vision (ACCV)*, 2016.
[40] D. Gadot and L. Wolf, "PatchBatch: a batch augmented loss for optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[41] R. Ranftl, K. Bredies, and T. Pock, "Non-local total generalized variation for optical flow estimation," in *European Conference on Computer Vision (ECCV)*, 2014.
[42] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *European Conference on Computer Vision (ECCV)*, 2014.
[43] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[44] C. Vogel, K. Schindler, and S. Roth, "3d scene flow estimation with a piecewise rigid scene model," *IJCV*, 2015.
[45] W. Luo, A. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[46] K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun, "Continuous markov random fields for robust stereo estimation," in *European Conference on Computer Vision (ECCV)*, 2012.
[47] C. Vogel, K. Schindler, and S. Roth, "Piecewise rigid scene flow," in *IEEE International Conference on Computer Vision (ICCV)*, 2013.



Yunsong Li received his M.S. and Ph.D. degree in Signal and Information Processing from Xidian University, China, in 1999 and 2002 respectively. He is currently an Professor in the State Key Laboratory of Integrate Service Network, School of Telecommunications at Xidian University. His research interests include spectral image coding and image analysis.

Yinlin Hu received his M.S. degree in Xidian University, China, in 2011. He was a senior software engineer and technical leader in Zienon, LLC. He is currently pursuing the Ph.D. degree under the direction of Prof. Yunsong Li in the State Key Laboratory of Integrate Service Network at Xidian University. His research interests include optical flow, video segmentation and motion analysis.

Rui Song received his M.S. and Ph.D. degree in Signal and Information Processing from Xidian University, China, in 2006 and 2009 respectively. He is currently an Associate Professor in the State Key Laboratory of Integrate Service Network, School of Tele-communications at Xidian University. His research interests include image and video quality assessment, video coding algorithms and VLSI architecture design, and 3D reconstruction.

Peng Rao received the B.S. degree from Huazhong University of Science and Technology in 2000, M.S. degree and Ph.D. degree from Chinese Academy of Sciences. He is currently a research fellow of Shanghai Institute of Technical Physics, Chinese Academy of Sciences. His research interests are in the fields of Optical imaging and processing.

Yangli Wang received the B.S. degree from Peking University in 1995, M.S. degree from microelectronics School of Xidian University in 1998, and Ph. D. degree from telecommunication school of Xidian University in 2002. He is currently a Professor at the State Key Laboratory of Integrated Service Networks of Xidian University. His research interests are in the fields of image and video compression and communication, distribute video coding, and image processing.