# Robust Interpolation of Correspondences for Large Displacement Optical Flow

**Conference Paper** · July 2017

DOI: 10.1109/CVPR.2017.509

Yinlin Hu
Xidian University
**4** PUBLICATIONS **21** CITATIONS
SEE PROFILE

Rui Song
Xidian University
**18** PUBLICATIONS **59** CITATIONS
SEE PROFILE

# Robust Interpolation of Correspondences for Large Displacement Optical Flow

Yinlin Hu[1]
huyinlin@gmail.com

Yunsong Li[1]
ysli@mail.xidian.edu.cn

Rui Song[1,2,*]
rsong@xidian.edu.cn

## Abstract

*The interpolation of correspondences (EpicFlow) was widely used for optical flow estimation in most-recent works. It has the advantage of edge-preserving and efficiency. However, it is vulnerable to input matching noise, which is inevitable in modern matching techniques. In this paper, we present a Robust Interpolation method of Correspondences (called RicFlow) to overcome the weakness. First, the scene is over-segmented into superpixels to revitalize an early idea of piecewise flow model. Then, each model is estimated robustly from its support neighbors based on a graph constructed on superpixels. We propose a propagation mechanism among the pieces in the estimation of models. The propagation of models is significantly more efficient than the independent estimation of each model, yet retains the accuracy. Extensive experiments on three public datasets demonstrate that RicFlow is more robust than EpicFlow, and it outperforms state-of-the-art methods.*

## 1. Introduction

Optical flow estimation is one of the most fundamental problems in computer vision. The applications include motion segmentation [28], video saliency detection [32], action recognition [38], driver assistance [15, 25], *etc*. While there exists abundant literature of this topic, obtaining the reliable optical flow between frames is still an open problem, especially in the case of large displacements, which is common in real-world videos.

Since the optical flow problem is inherently ill-posed, typically energy optimization is involved in solving it. Many effective approaches [7, 6, 41, 35] have been proposed based on the pioneering work by Horn and Schunck [18] who cast the optical flow problem into an energy minimization framework. Although these approaches perform well in the case of small displacements, they often fail to estimate large displacements. The main reason is the failure of accurate initialization for the energy minimization to avoid local minima.

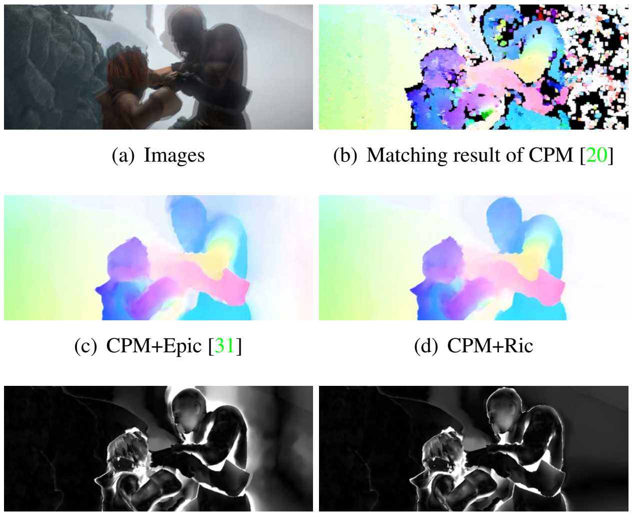

(a) Images (b) Matching result of CPM [20] (c) CPM+Epic [31] (d) CPM+Ric (e) Error map of CPM+Epic (f) Error map of CPM+Ric

Figure 1. Demonstration of the robustness of the proposed Ric method. Given the same noisy matches (CPM [20]), the flow interpolated from Ric is more tolerant of matching noise compared to that interpolated from Epic [31] (see the flat regions above the shoulder of the character). (e) and (f) are the error map of (c) and (d) respectively. Matching correspondences are shown as small colored patches as in [20], and the visualization of the error map follows [25] (the darker, the better).

Recently, a method called EpicFlow [31] that interpolates a sparse set of matches in a dense manner to initiate the optical flow estimation has shown great advantages over the traditional coarse-to-fine minimization scheme, especially in the case of occlusions, motion boundaries and large displacements. However, the flow result is vulnerable to input matching noise, which is inevitable, though the great advances have been made in the research of matching techniques [39, 20, 3, 21] (see Figure 1(b)).

In this paper, we propose a robust interpolation method of correspondences, which is more tolerant of matching noise. After over-segmenting the scenes into superpixels [1] and assuming the flow in each superpixel meets a piecewise affine model, we use RANSAC-like techniques (Random Sample Consensus) to estimate the model of each super-

*Corresponding author. [1]State Key Laboratory of Integrated Service Networks, Xidian University, China. [2]Key Laboratory of Infrared System Detection and Imaging Technology, Chinese Academy of Sciences.

pixel robustly. The affine model in each superpixel relies only on its local neighboring matches. This simple strategy is very effective and Figure 1 shows its robustness compared to EpicFlow which just uses an intuitive threshold to remove some noisy matches. Furthermore, considering the spatial continuity of flow, we propose a propagation mechanism between superpixels in the estimation of models. The propagation of models is significantly more efficient than the independent estimation of each model, yet retains the accuracy. Given the same matching correspondences as CPM-Flow [20], our approach, *RicFlow* (Robust Interpolation of Correspondences) outperforms the original CPM-Flow (interpolated using Epic) and performs best on the challenging MPI-Sintel dataset [8] and is competitive on KITTI-2012 [15] and KITTI-2015 [25].

We make the following contributions: **1)** We propose *RicFlow*, a robust interpolation method of correspondences based on the estimation of piecewise models. **2)** We propose a model propagation mechanism between superpixels in the estimation of piecewise models, which leads to a significant speed-up compared to the independent estimation of each model. Furthermore, model propagation can achieve more accurate results by utilizing the spatial relations between neighboring superpixels. **3)** We show the effectiveness of the proposed method by achieving state-of-the-art accuracy on the challenging datasets MPI-Sintel, KITTI-2012 and KITTI-2015.

## 2. Related work

Classic variational minimization framework for optical flow estimation has shown its success in the case of small displacements [18, 4]. Though using a coarse-to-fine scheme [6], the minimization still often gets stuck in local minima and leads to failure in the case of large displacements [35]. To handle this problem, some approaches [7, 41] introduce the descriptor matching to the minimization framework. However, their accuracy is still not satisfactory in real-world videos.

The interpolation scheme we use is similar to the sparse-to-dense interpolation framework [22, 31, 24]. Compared to [22] where the initial matching is obtained through the minimization of a matching energy, we use state-of-the-art matching methods as input directly. Closely related to our RicFlow, Revaud *et al.* [31] introduce an edge-preserving interpolation method (EpicFlow) based on geodesic distance. EpicFlow has shown great success in the case of occlusions, motion boundaries and large displacements, and many effective methods use it as the post refinement step [3, 20, 9, 26, 14]. However, the flow interpolated from EpicFlow is vulnerable to input matching noise, which is inevitable in modern matching techniques. To tackle this problem, we propose to use RANSAC-like techniques to estimate the piecewise models robustly. Li *et al.* [24] show the efficiency of an interpolation method based on the fast solver of weighted least squares. However, its accuracy is not satisfactory on MPI-Sintel.

Our method is related to piecewise parametric flow estimation. Xu *et al.* [40] fit affine models on segmented regions using total variation as its regularization. Ren [30] proposes to estimate piecewise flow based on grouped pixels using edge-based affinities. Nevertheless, these work often fails in the case of large displacements. Sun *et al.* [36, 34] use affine motion to regularize the flow in layered model estimation. Sevilla-Lara *et al.* [33] integrate the information of semantic segmentation into the model estimation of localized layers. Hornáček *et al.* [19] define per-pixel homography for flow estimation. Unlike this point-wise parametric model, Yang *et al.* [42] fit piecewise homography models on adaptive segments using an energy-based optimization. In contrast to the costly energy optimization, our method fits piecewise affine models for each superpixel [1] based on its neighboring matches.

The model estimation scheme we use is related to RANSAC [13] which is widely applied to robust estimation of a model from data due to its simple implementation and robustness. Choi *et al.* [10] make an insightful view of the numerous methods that have been derived from RANSAC. Raguram *et al.* [29] propose a uniform framework for RANSAC-based robust estimation. Recently, Ni *et al.* [27] introduce a binomial mixture model based on the assumption that there exists some grouping in the data, which leads to a better efficiency. In contrast, we achieve a significant efficiency gain by introducing a propagation scheme when estimating many spatially-related models simultaneously.

## 3. Robust interpolation of correspondences

In this section, we present our interpolation approach, *RicFlow*, and discuss its main features. Given two consecutive images $I_1, I_2$ and a set of the matches $\mathcal{M} = \{(p_1, p_1^{'})\}$ (sparse or dense) where the pixel $p_1 \in I_1$ and $p_1^{'} \in I_2$ define a correspondence, our goal is to estimate a dense correspondence field $\mathcal{F} : I_1 \to I_2$ between image $I_1$ and $I_2$.

### 3.1. Piecewise affine model

As in [37], we assume that most scenes of interest consist of regions with a constant motion pattern. We first over-segment the input image $I_1$ to $K$ non-overlapping superpixels $s_k$ using SLIC algorithm [1], where the average superpixel size is $\sigma$. That is a segmentation $\mathcal{S} = \{s_k | \bigcup_{k=1}^{K} s_k = I_1 \text{ and } \forall k \neq m, s_k \cap s_m = \varnothing\}$. For the flow in each superpixel $s_k$, we assume them meet a constant affine model $\mathcal{A}_k$, which is well-suited for most scenes of interest [34, 36, 37]. After the formulation of piecewise affine models on pre-segmented regions, the goal now is to estimate the model set $\{\mathcal{A}_k\}$ based on the input noisy matches.

## 3.2. Robust model estimation

**Graph construction.** We represent the over-segmented image as an undirected graph $G = (V, E)$, where $V$ is the vertices representing all superpixels, and $E$ is the edges connecting the superpixels with shared boundaries.

For the neighboring superpixels $s_a$ and $s_b$, that is $(s_a, s_b) \in E$, the edge length between them is defined as the cost connecting $s_a$ and $s_b$. In this paper, the cost is defined as the approximate geodesic distance between the center of neighboring superpixels, that is, the shortest distance with respect to a cost map. As in EpicFlow [31], we use the result of "structured edge detector" (SED) [12] as the cost map, which has shown prominent advantages over other choices. Based on the edge length, we define the distance function $\mathcal{D} : V \times V \rightarrow \mathbb{R}^+$ between any vertices as the shortest path connecting them on the graph $G$, which can be calculated efficiently using Dijkstra's algorithm.

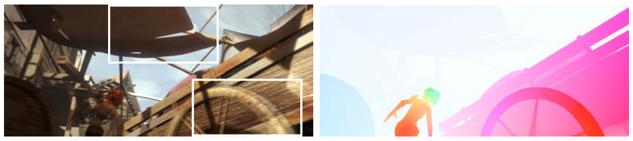
**Superpixel flow initialization.** Rather than using the raw matches directly, we generate superpixel flow from the input matches as the initialization, which can boost the efficiency of model estimation. Based on the defined superpixels $\{s_k\}$, we generate only one matching for each superpixel $s_k$, which we call it superpixel flow. We use the median of all the valid matches inside $s_k$ as the superpixel flow. For invalid superpixels (without any matches in it), the superpixel flow is set to the flow of the nearest valid superpixel. Figure 3(d) shows an example of the generated superpixel flow.

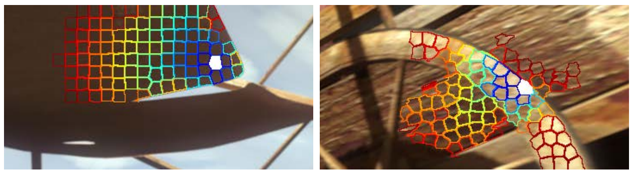
**Model estimation.** We use neighboring superpixel flow of each superpixel to estimate its model. $\mathcal{N}_n(k)$ is denoted as the set of $n$ nearest neighbors of superpixel $s_k$ on graph $G$, which we also call it support neighbors of $s_k$. Figure 2(b) shows two examples of support neighbors.

With the goal of estimating the model for each superpixel from its support neighbors, the problem is that the initial superpixel flow is noisy and the support neighbors are often "bleeding out" to the surface of different objects with different motion patterns, which affect the accuracy of model estimation significantly (see Figure 2). To handle this problem, we use RANSAC-like method to estimate the model for each superpixel.

As RANSAC family [13, 10], our model estimation runs in an iterative manner for each superpixel. For each iteration, we pick up a subset from its support neighbors randomly, and generate a model hypothesis from the superpixel flow of the support neighbors. Here, the size of the subset is fixed to 3 which is the minimum number of pairs to determine an affine transformation. After generating the model hypothesis, we use a criterion of hypothesis evaluation to update the current most probable model.

More precisely, the procedure can be formulated as an optimization problem:


(a) Images and the ground truth


(b) Two examples of support neighbors

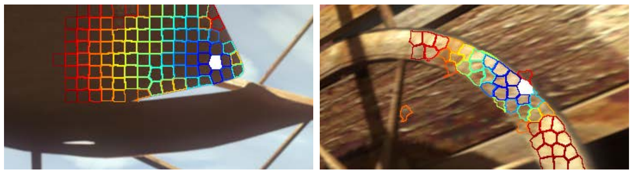

(c) Support neighbors after inlier extraction

Figure 2. Adaptive inlier extraction by robust model estimation. The support neighbors ($n = 100$) are represented by color (blue to red means closer to farther) according to the distance on graph $G$ from the superpixel seed (filled with white). The support neighbors on the right is "bleeding out" to different objects with different motion patterns. Our approach can handle this case effectively.

$$\hat{\mathcal{H}} = \arg\min_{\mathcal{H}} \{\mathcal{C}(\mathcal{H})\}, \tag{1}$$

where $\mathcal{C}(\mathcal{H})$ is the evaluation cost of the model hypothesis $\mathcal{H}$. In this paper, the evaluation cost is defined as:

$$\mathcal{C}(\mathcal{H}) = \sum_{s \in \mathcal{N}_n(k)} \mathcal{W}(s) \cdot \mathcal{L}\Big(\epsilon(s; \mathcal{H})\Big), \tag{2}$$

where $\mathcal{N}_n(k)$ is the support neighbors of superpixel $s_k$, $\mathcal{W}(s)$ is a weight function, $\mathcal{L}$ is a loss function, and $\epsilon(s; \mathcal{H})$ is the fitting error when applying model $\mathcal{H}$ for $s$, which is defined as the endpoint deviation of the fitting result from the original superpixel flow. The weight is defined as a function of the distance from $s_k$ to the neighbor $s$: $\mathcal{W}(s) = \exp(-\mathcal{D}(s, s_k)/\alpha)$. The loss function of the least square method can be represented as $\mathcal{L}(\epsilon) = \epsilon^2$, which is sensitive to noise. In contrast, we use the following loss function:

$$\mathcal{L}(\epsilon) = \begin{cases} |\epsilon| & |\epsilon| < \tau \\ \tau & \text{otherwise} \end{cases}, \tag{3}$$

where $\tau$ is a threshold (in our experiments, we fix $\tau = 5$), and the neighbors with fitting error $|\epsilon| < \tau$ are considered as inliers for the current superpixel.

After some iterations, the inliers can be extracted and Figure 2(c) shows some examples. Furthermore, the weighted least square (WLS) method is used to estimate the final affine model. Here, the WLS is only applied on the inliers and the weight is the same as $\mathcal{W}(s)$ in Equation 2.

## 3.3. Fast approximation

Estimating the piecewise models independently as described in Section 3.2 has high computational complexity. Considering the spatial continuity of flow, we propose a propagation scheme when estimating many spatially-related models simultaneously.

**Model propagation.** Inspired by PatchMatch [5], we improve the piecewise models in a spatially iterative manner. Superpixels are examined in *scan* order on odd iterations and in *reverse scan* order on even iterations. Each iteration undergoes *hypothesis propagation* followed by *random sampling* based on the spatial adjacency of superpixels.

Random initialization is often used for dense correspondences [5, 20]. Considering the superpixel flow is a strong prior information we can take advantage of, we use it as the initialization rather than the random initialization. That is, the model of each superpixel is initialized as a translation model derived from the superpixel flow, which is a degenerate case of the affine model.

Hypothesis propagation is the core scheme our fast approximation relies on. For a superpixel $s_k$ with model $h_k$ in an iteration, we denote the models of adjacent neighbors that is already examined in current iteration as set $\{\mathcal{H}_k\}$. Models are propagated from neighbors to itself:

$$\hat{h}_k = \arg\min_h(\mathcal{C}(h)), h \in \{h_k\} \cup \{\mathcal{H}_k\}, \tag{4}$$

where $\hat{h}_k$ denote the improved model, and $\mathcal{C}(h)$ is the evaluation cost of the hypothesis according to Equation 2.

A random sampling procedure is performed after the preceding propagation step. We pick up 3 matches randomly from the $n$ nearest neighbors and generate a model hypothesis by fitting the matches. The current best model will be improved by testing this new hypothesis.

**Algorithm 1:** Robust interpolation of correspondences

```
Input: a pair of images $I_1$, $I_2$, a set $\mathcal{M}$ of matches
Output: dense correspondence field $\mathcal{F} : I_1 \to I_2$
1 Over-segment $I_1$ to $K$ superpixels $\{s_k\}$
2 Superpixel flow initialization from $\mathcal{M}$
3 Construct superpixel graph $G = (V, E)$
4 for each superpixel $s_k$ do
5     Model initialization based on superpixel flow
6     Get the support neighbors $\mathcal{N}_n(k)$ of $s_k$
7     Inlier extraction based on model propagation
8     Weighted least square optimization on inliers
9 Variational refinement
```

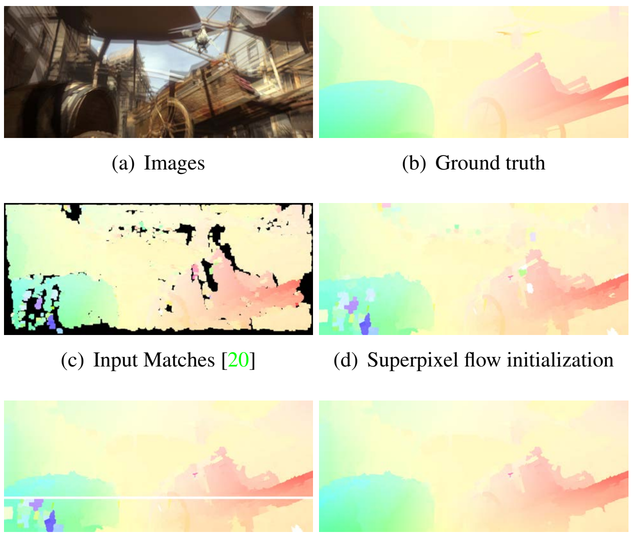

(a) Images (b) Ground truth (c) Input Matches [20] (d) Superpixel flow initialization (e) 3/4 iteration (f) 1 iteration

Figure 3. Illustration of model propagation. Notice how the majority of matching noises are removed after only 1 iteration.

**Termination criteria.** Although adaptive criteria of termination may be used as in RANSAC family [10, 27, 11], in practice, we have found it works well to iterate a fixed number of times. In this paper, the number of iteration $\gamma$ means the number of the entire examination of each superpixel in scan order (or reverse scan order). RicFlow converges rapidly only after a few iterations. Figure 3 shows an example illustrating the procedure of model propagation and demonstrates the speed of its convergence.

After obtaining the best model for each superpixel, the inlier extraction based on each model and WLS optimization are the same as Section 3.2.

## 3.4. Variational refinement

Variational minimization serves as our final refinement step by using the output of the interpolation as an initialization. As in [31, 21], the energy function is optimized only on the raw image scale without the coarse-to-fine scheme. The data term and smoothness term we used are the same as EpicFlow [31], which based on a classical variational framework [43]. The overall procedure of the proposed RicFlow is summarized in Algorithm 1.

## 4. Experiments

We evaluate RicFlow on three challenging datasets for optical flow estimation: • *MPI-Sintel* [8] is a synthetic benchmark based on an animated movie with many scenes containing large motions. Many rendering effects like motion, defocus blurs and atmospheric effects are contained in its "final" version, and we only use this challenging version to evaluate our method. • *KITTI-2012* [15] was created from a driving platform and contains realistic images of complex lighting conditions and large displacements. • *KITTI-2015* [25] also contains photos shot in city streets from a driving platform. Compared to KITTI-2012 datasets, it comprises dynamic scenes and is more challenging for optical flow estimation.

As in [31, 39], we optimize the parameters of our approach on 20% of the training set of MPI-Sintel. Then we use the *same* constant parameters to interpolate input matches for all evaluations: $\{\sigma, n, \alpha, \gamma\} = \{20, 150, 0.7, 4\}$, and report the results on the training set of MPI-Sintel, KITTI-2012 and KITTI-2015. We directly use the code of variational minimization from EpicFlow for flow refinement[1].

We implement RicFlow in C++ and make it run on an Intel i7 3.6GHz CPU. It requires only 5 seconds for an MPI-Sintel image pair ($1024 \times 436$). In detail, computing coarse-to-fine PatchMatch (CPM) takes 1.4s, constructing superpixel graph (include over-segmentation and extracting nearest neighbors) 1.1s, estimation of piecewise models 1.5s, and variational refinement 1s.

In Section 4.1 we analyze the interpolation performance of our approach and compare it to EpicFlow. Section 4.2 then studies the different parameters of RicFlow and the effect of model propagation. Finally, we show that RicFlow outperforms state-of-the-art methods on challenging datasets in Section 4.3

### 4.1. Comparison of interpolation.

**Input matches.** To evaluate the performance of our interpolation method, we generate input matches by using three state-of-the-art methods: KPM [17], DM [39] and CPM [20]. For KPM[2] and DM[3], we use the online code to extract the correspondences, and for CPM the binaries provided by their authors. Both DM and CPM include a reciprocal verification to remove the ambiguous matches in occluded areas. Similarly, we perform a forward-backward consistency check for KPM to remove incorrect matches. The second column in Figure 4 shows an example of the input matches.

**RicFlow versus EpicFlow.** We compare our RicFlow with the most widely used interpolation method EpicFlow [31]. Figure 4 shows an example of the comparison results using different input matches. Notice how the matching noise is resolved by RicFlow. Table 1 reports the comparison results on different datasets thoroughly. We can see that RicFlow outperforms EpicFlow in most cases. The only exception is that, RicFlow performs less well when using KPM as input in terms of average endpoint error (EPE), which is mainly caused by the low density of the input matches generated by KPM.

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

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

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

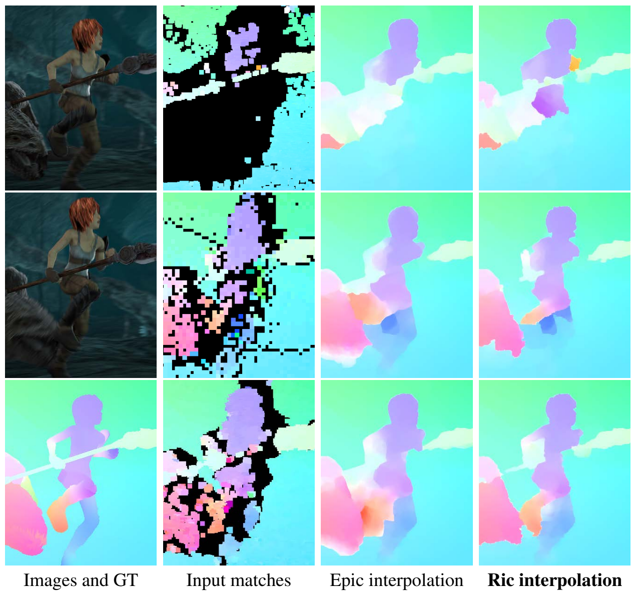


Figure 4. Example of the comparison between Epic [31] and the proposed Ric method for different matching inputs. The first column shows two images and the ground truth. The last three columns show the input matches (KPM [17], DM [39] and CPM [20] from top to bottom), and the results interpolating using Epic and Ric respectively. Notice how the matching noise is resolved by our Ric method near the leg of the character.

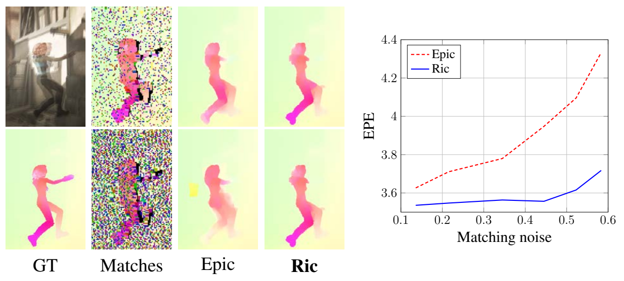


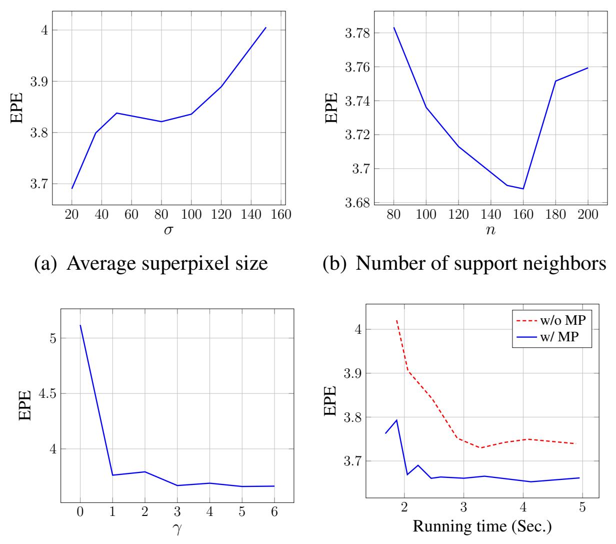
Figure 5. Comparison between Epic [31] and our Ric method with different levels of input matching noise. Notice how Ric is more tolerant of input matching noise compared with Epic.

As an interpolation method, RicFlow relies on the accuracy and density of input matches. We can observe that CPM consistently outperforms DM and KPM. We use CPM matches in the remainder of the experiments.

In order to demonstrate the robustness of RicFlow, we have evaluated its performance for different levels of input matching noise. We add synthetic noise to the matching results of CPM [20], and evaluate the performance of RicFlow and EpicFlow [31] on MPI-Sintel training subset using the same noisy matches as their input. Figure 5 shows the clear robustness of RicFlow over EpicFlow.

|  |  | EPE |  |  |  |  |  | Error percentage (%) |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | MPI-Sintel |  | KITTI-2012 |  | KITTI-2015 |  | MPI-Sintel |  | KITTI-2012 |  | KITTI-2015 |  |
|  |  | -Var. | +Var. | -Var. | +Var. | -Var. | +Var. | -Var. | +Var. | -Var. | +Var. | -Var. | +Var. |
| KPM | Epic | **7.16** | **6.74** | 17.58 | 17.04 | **26.35** | 26.01 | 19.52 | 16.64 | 57.84 | 51.65 | 56.98 | 53.53 |
|  | **Ric** | 7.56 | 6.93 | **16.52** | **15.85** | 27.88 | **25.35** | **18.26** | **15.86** | **52.94** | **46.30** | **53.03** | **49.45** |
| DM | Epic | 3.87 | 3.52 | 3.57 | 3.32 | 9.06 | 8.88 | 12.29 | 11.04 | 19.00 | 16.63 | 31.53 | 28.03 |
|  | **Ric** | **3.70** | **3.37** | **3.17** | **3.03** | **8.08** | **8.02** | **9.56** | **9.25** | **13.78** | **13.82** | **25.15** | **23.56** |
| CPM | Epic | 3.72 | 3.54 | 3.21 | 3.17 | 7.74 | 7.71 | 10.51 | 10.08 | 13.32 | 14.09 | 24.66 | 23.52 |
|  | **Ric** | **3.69** | **3.52** | **2.80** | **2.81** | **7.11** | **7.12** | **8.85** | **8.95** | **10.15** | **11.97** | **20.31** | **20.34** |

Table 1. Comparison of Epic [31] and the proposed Ric method for different matching inputs (KPM [17], DM [39] and CPM [20]) in terms of the average endpoint error (EPE) and the error percentage (EPE $\geq$ 3 pixels). The comparison is performed on three challenging datasets: MPI-Sintel [8], KITTI-2012 [15] and KITTI-2015 [25]. For a fair comparison, the results with variational refinement (**+Var.**) and without it (**-Var.**) are also reported. The proposed Ric interpolation method outperforms Epic in most cases.



(a) Average superpixel size (b) Number of support neighbors

(c) Iteration times (d) Effect of model propagation

Figure 6. Parameter analysis and the effect of model propagation. (a), (b) and (c) study the flow accuracy of different parameters. (d) shows the effect of model propagation (**w/ MP**) compared to independent estimation of models (**w/o MP**). The method with model propagation converges more rapidly, and can even achieve more accurate results after convergence. Matching time is not included.

## 4.2. Performance analysis

In order to get a better understanding of the performance of RicFlow, we evaluate the impact of different parameters and also the effect of model propagation. For better insights on the interpolation itself, the experiment in this section is performed without variational refinement, and all the results are reported on MPI-Sintel.

**Parameter analysis.** Figure 6(a) studies the effect of RicFlow with different superpixel size. We can see that with the increase of superpixel size $\sigma$, the result shows a clear trend of quality deterioration. However, large superpixel size often means low computational complexity. We find that $\sigma = 20$ gives a good balance. Figure 6(b) illustrates that too few support neighbors are not robust for model estimation, while too many support neighbors also leads to loss of accuracy. As Figure 6(c) shows, our method converges rapidly only after a few iterations. In our experiments, the iteration number $\gamma$ is fixed to 4, after which our method has almost converged.

**Effect of model propagation.** To demonstrate the effectiveness of model propagation, we compare it with the version without model propagation, which estimates the piecewise models independently. Independent model estimation often needs dozens of iterations to converge. However, considering that one iteration with model propagation requires more time, we use EPE over running times to compare them on the same computer for fair.

Figure 6(d) shows the comparison results. We can see that model propagation converges more rapidly. Furthermore, model propagation can even achieve more accurate results after convergence. The main reason of this phenomenon is that model propagation utilizes the spatial relations between neighboring superpixels.

## 4.3. Results on datasets

The results of RicFlow compared to other methods on MPI-Sintel can be seen in Table 2 which shows the top optical flow algorithms as published on the submission date. We clearly outperform CPM-Flow as well as all other methods. The original CPM-Flow use Epic [31] as its interpolation method. With the same matching inputs as CPM-Flow, we reduce the EPE by nearly 0.4 pixels compared to it. Note that RicFlow performs especially well in the occluded areas, which demonstrates its robustness. The approximate running time of the top methods (as declared by their authors) is also included in the Table. RicFlow is among the fastest methods, and is only slightly slower than the original CPM-Flow. Figure 7 shows some results of qualitative comparisons.

Table 3 and Table 4 show the top flow results on KITTI-

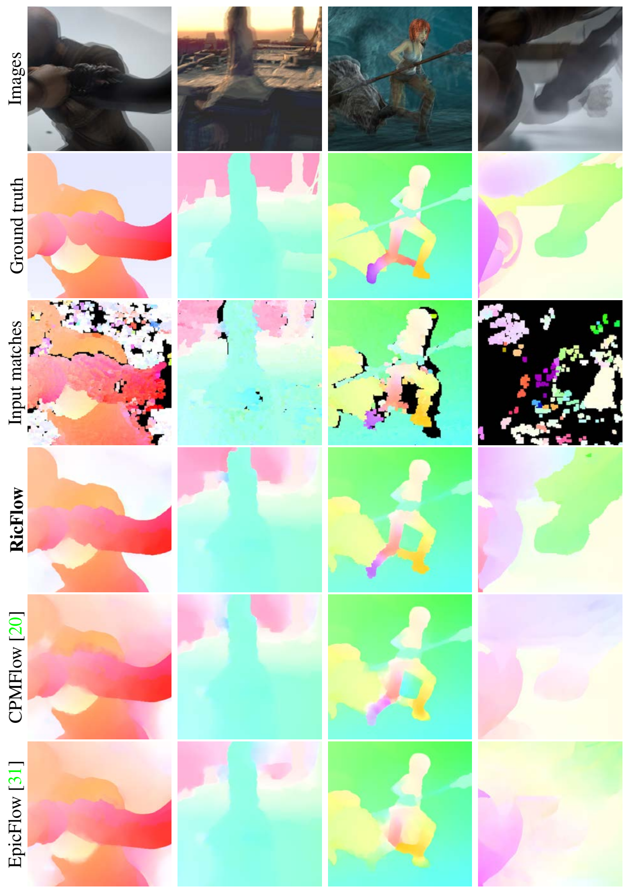


Figure 7. Example of the results on MPI-Sintel. Each column shows from top to bottom: mean of two consecutive images, ground truth, input matches (CPM), the proposed RicFlow and 2 state-of-the-art methods (CPMFlow [20] and EpicFlow [31]). RicFlow uses the same matching intput as CPMFlow. Notice how the flat regions with low saliency (see the background in the first column) and the motion discontinuities (see the leg of the character in the third column) are handled properly by our RicFlow.

2012 and KITTI-2015 test set for methods that do not use epipolar geometry or stereo vision, respectively. On KITTI-2012, RicFlow performs best in terms of EPE in non-occluded areas. However, many top methods achieve the same EPE. In terms of the percentage of erroneous pixels, which is more discriminative for the evaluation of large displacement optical flow, RicFlow achieves 4.96% in non-occluded areas, and it is the only method whose error percentage below 5% due to the submission date. On the more challenging dataset KITTI-2015, RicFlow also outperforms the original CPM-Flow, which reduce the error percentage by nearly 4%. Although RicFlow performs less well than SOF [33] which relies on semantic segmentations, it performs best in foreground regions. Furthermore, RicFlow is an order of magnitude faster than SOF. Figure 8 shows some results on KITTI.

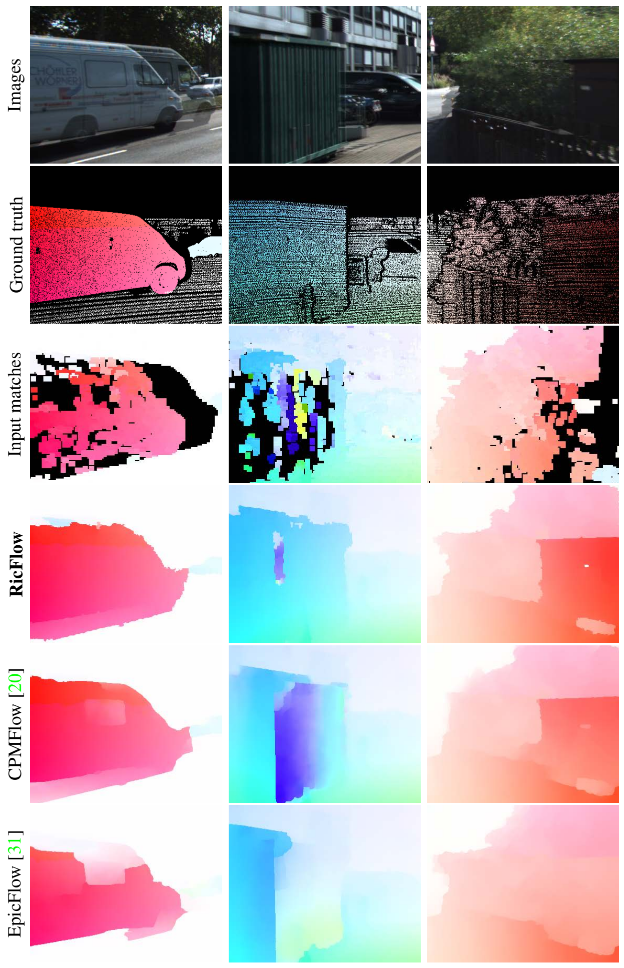


Figure 8. Example of the results on KITTI. Notice how the flow error in less-textured regions are reduced by RicFlow.

| Method | EPE All | EPE Noc | EPE Occ | Time ∼Sec. |
|---|---|---|---|---|
| **RicFlow** | **5.620** | 2.765 | **28.907** | 5 |
| DDF[16] | 5.728 | 2.623 | 31.042 | 60 |
| FlowFields[3] | 5.810 | **2.621** | 31.799 | 18 |
| SPM-BP2[23] | 5.812 | 2.754 | 30.743 | 42 |
| FullFlow[9] | 5.895 | 2.838 | 30.793 | 240 |
| CPM-Flow[20] | 5.960 | 2.990 | 30.177 | **4** |
| EpicFlow[31] | 6.285 | 3.060 | 32.564 | 17 |
| FGI[24] | 6.607 | 3.101 | 35.158 | 15 |
| PH-Flow[42] | 7.423 | 3.795 | 36.960 | 800 |
| MDPFlow2[41] | 8.445 | 4.150 | 43.430 | 700 |

Table 2. Top algorithms on MPI-Sintel test set (final). EPE-Noc (resp. EPE-Occ) is the EPE in non-occluded areas (resp. occluded areas).

**Limitations.** As an interpolation method, the accuracy of RicFlow relies on the input matches. Although RicFlow is more tolerant of matching noise than EpicFlow, it still

| Method | Out Noc3 | Out All3 | EPE Noc | EPE All | Time ∼Sec. |
|---|---|---|---|---|---|
| **RicFlow** | **4.96%** | 13.04% | **1.3** | 3.2 | 5 |
| PatchBatch[14] | 5.29% | 14.17% | **1.3** | 3.3 | 50 |
| DDF[16] | 5.73% | 14.18% | 1.4 | 3.4 | 60 |
| PH-Flow[42] | 5.76% | **10.57%** | **1.3** | **2.9** | 800 |
| FlowFields[3] | 5.77% | 14.01% | 1.4 | 3.5 | 23 |
| CPM-Flow[20] | 5.79% | 13.70% | **1.3** | 3.2 | **4** |
| DiscreteFlow[26] | 6.23% | 16.63% | **1.3** | 3.6 | 180 |
| EpicFlow[31] | 7.88% | 17.08% | 1.5 | 3.8 | 15 |

Table 3. Top algorithms on KITTI-2012 test set. Out-Noc3 (resp. Out-All3) is the percentage of pixels with flow error above 3 pixels in non-occluded areas (resp. all pixels).

| Method | Out All3 | Out Bg3 | Out Fg3 | Time ∼Sec. |
|---|---|---|---|---|
| SOF[33] | **16.81%** | **14.63%** | 27.73% | 360 |
| **RicFlow** | 19.52% | 18.73% | **23.49%** | 5 |
| PatchBatch[14] | 21.69% | 19.98% | 30.24% | 50 |
| DDF[16] | 21.92% | 20.36% | 29.69% | 60 |
| DiscreteFlow[26] | 22.38% | 21.53% | 26.68% | 180 |
| CPM-Flow[20] | 23.23% | 22.32% | 27.79% | **4** |
| FullFlow[9] | 24.26% | 23.09% | 30.11% | 240 |
| EpicFlow[31] | 27.10% | 25.81% | 33.56% | 15 |

Table 4. Top algorithms on KITTI-2015 test set. Out-Bg3 (resp. Out-Fg3) is the percentage of outliers averaged only over background regions (resp. foreground regions).

failed when using low-density matches as input (see KPM matching in Figure 4). On the other hand, the piecewise model estimation relies on the neighbor system of the graph constructed based on over-segmentations. The third column in Figure 7 shows an example of this failure case (near the feet of the character). It should be improved by using some information like semantic segmentations, which is also our nearly future work.

## 5. Conclusion

We have proposed an interpolation method of correspondences for optical flow estimation, which is tolerant of input matching noise. It uses superpixel techniques to oversegment the scenes under the assumption that most scenes of interest consist of regions with a consistent motion pattern. We focus on the estimation of the piecewise models using a RANSAC-like method for robust estimation. We introduce a novel model propagation to estimate many models simultaneously based on the insight that the models of neighboring superpixels are spatially-related. The extensive experiments on MPI-Sintel, KITTI-2012 and KITTI-2015 have shown its advantages over other competing methods. This work confirms the benefit of piecewise models for optical flow estimation. It may be possible to achieve better results using information of semantic segmentations [33, 2]. It is also an interesting research direction to implement our method on GPU for real-time optical flow estimation.

**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 by Xidian Graduate Innovation Project.

## References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2012. 1, 2

[2] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Exploiting semantic information and deep matching for optical flow. In *European Conference on Computer Vision (ECCV)*, 2016. 8

[3] 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. 1, 2, 7, 8

[4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV)*, 2011. 2

[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. 4

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, 2004. 1, 2

[7] 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. 1, 2

[8] 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, 4, 6

[9] 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. 2, 7, 8

[10] S. Choi, T. Kim, and W. Yu. Performance evaluation of RANSAC family. In *British Machine Vision Conference (BMVC)*, 2009. 2, 3, 4

[11] O. Chum and J. Matas. Matching with PROSAC progressive sample consensus. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 4

[12] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2015. 3

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 2, 3

[14] D. Gadot and L. Wolf. PatchBatch: a batch augmented loss for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 8

[15] 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. 1, 2, 4, 6

[16] F. Güney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision (ACCV)*, 2016. 7, 8

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

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

[19] M. Hornáček, F. Besse, J. Kautz, A. Fitzgibbon, and C. Rother. Highly overparameterized optical flow using patchmatch belief propagation. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[20] 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. 1, 2, 4, 5, 6, 7, 8

[21] T. Kroeger, R. Timofte, D. Dai, and L. V. Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 4

[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. 2

[23] 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. 7

[24] Y. Li, D. Min, M. N. Do, and J. Lu. Fast guided global interpolation for depth and motion. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 7

[25] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 5, 6

[26] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition (GCPR)*, 2015. 2, 8

[27] K. Ni, H. Jin, and F. Dellaert. GroupSAC: Efficient consensus in the presence of groupings. In *IEEE International Conference on Computer Vision (ICCV)*, 2009. 2, 4

[28] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2014. 1

[29] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. USAC: A universal framework for random sample consensus. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2013. 2

[30] X. Ren. Local grouping for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 2

[31] 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, 3, 4, 5, 6, 7, 8

[32] D. Rudoy, D. B. Goldman, E. Shechtman, and L. Zelnik-Manor. Learning video saliency from human gaze using candidate selection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1

[33] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 7, 8

[34] D. Sun, C. Liu, and H. Pfister. Local layering for joint motion estimation and occlusion detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[35] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision (IJCV)*, 2014. 1, 2

[36] D. Sun, E. B. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[37] C. Vogel, K. Schindler, and S. Roth. 3D scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision (IJCV)*, 2015. 2

[38] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 1

[39] 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. 1, 5, 6

[40] L. Xu, J. Chen, and J. Jia. A segmentation based variational model for accurate optical flow estimation. In *European Conference on Computer Vision (ECCV)*, 2008. 2

[41] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2012. 1, 2, 7

[42] 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, 7, 8

[43] H. Zimmer, A. Bruhn, and J. Weickert. Optic flow in harmony. *International Journal of Computer Vision (IJCV)*, 2011. 4