Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation

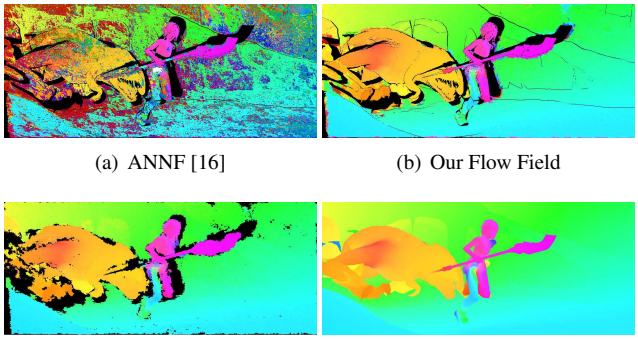
Christian Bailer[1] Bertram Taetz[1,2] Didier Stricker[1,2]
Christian.Bailer@dfki.de Bertram.Taetz@dfki.de Didier.Stricker@dfki.de
[1]German Research Center for Artificial Intelligence (DFKI), [2]University of Kaiserslautern

Abstract

Modern large displacement optical flow algorithms usually use an initialization by either sparse descriptor matching techniques or dense approximate nearest neighbor fields. While the latter have the advantage of being dense, they have the major disadvantage of being very outlier prone as they are not designed to find the optical flow, but the visually most similar correspondence. In this paper we present a dense correspondence field approach that is much less outlier prone and thus much better suited for optical flow estimation than approximate nearest neighbor fields. Our approach is conceptually novel as it does not require explicit regularization, smoothing (like median filtering) or a new data term, but solely our novel purely data based search strategy that finds most inliers (even for small objects), while it effectively avoids finding outliers. Moreover, we present novel enhancements for outlier filtering. We show that our approach is better suited for large displacement optical flow estimation than state-of-the-art descriptor matching techniques. We do so by initializing EpicFlow (so far the best method on MPI-Sintel) with our Flow Fields instead of their originally used state-of-the-art descriptor matching technique. We significantly outperform the original EpicFlow on MPI-Sintel, KITTI and Middlebury.

1. Introduction

Finding the correct dense optical flow between images or video frames is a challenging problem. While the visual similarity between two image regions is the most important clue for finding the optical flow, it is often unreliable due to illumination changes, deformations, repetitive patterns, low texture, occlusions or blur. Hence, basically all dense optical flow methods add prior knowledge about the properties of the flow, like local smoothness assumptions [18], structure and motion adaptive assumptions [30], the assumption that motion discontinuities are more likely at image edges [26], or the assumption that the optical flow can be approximated by a few motion patterns [9]. The most popular of these assumptions is the local smoothness assumption. It is usually incorporated into a joint energy based regularization that rates data consistency together with the smoothness in a variational setting of the flow [18]. One major drawback of this setting is that fast minimization techniques usually rely on local linearization of the data term and thus can adapt the motion field only very locally. Hence, these methods have to use image pyramids to deal with fast motions (large displacements) [6]. In practice, this fails in cases where the determined motion on a lower scale is not very close to the correct motion of a higher scale.



(a) ANNF [16] (b) Our Flow Field

(c) Our outlier filtered Flow Field (d) Ground truth

Figure 1. Comparison of state-of-the-art approximate nearest neighbor fields (a) and Flow Fields (b) with the same data term. a) and b) are shown with ground truth occlusion map (black pixels). c) is after outlier filtering, occluded regions are successfully filtered. It can be used as initialization for an optical flow method.

In contrast, for purely data based techniques like approximate nearest neighbor fields [16] (ANNF) and sparse descriptor matches [32] there are fast approaches that can efficiently perform a global search for the best match on the full image resolution. However, as there is no regularization, (approximate) nearest neighbor fields (NNF) usually contain many outliers that are difficult to identify. Furthermore, even if outliers can be identified they leave gaps in the motion field that must be filled. Sparse descriptor matches

usually contain fewer outliers as matches are only determined for carefully selected points with high confidence. However, due to their sparsity the gaps between matches are usually even larger than in outlier filtered ANNF. Gaps can be problematic, since a motion for which no match is found cannot be considered. Despite these difficulties, ANNF and sparse descriptor matches gained a lot of popularity in the last years as initial step of large displacement optical flow algorithms. Nowadays, nearly all top-performing methods on challenging datasets like MPI-Sintel [8] rely on such techniques. However, while there are descriptor matching approaches like Deep Matching [32] that are tailored for optical flow, dense initialization is usually simply based on ANNF – which is suboptimal. The intention behind ANNF is to find the visually closest match (NNF), which is often not identical with the optical flow. An important difference is that NNF are known to be very noisy regarding the offset of neighboring pixels, while optical flow is usually locally smooth and occasionally abrupt (see Figure 1).

In this paper we show that it is possible to utilize this fact, to create dense correspondence fields that contain significantly fewer outliers than ANNF regarding optical flow estimation – not because of explicit regularization, smoothing (like median filtering) or a different data term, but solely because of our novel search strategy that finds most inliers while it effectively avoids finding outliers. We call them *Flow Fields* as they are tailored for optical flow estimation, while they are at the same time dense and purely data term based like ANNF. Flow Fields are conceptually novel as we avoid building on the popular, but for optical flow estimation inappropriate, (A)NNF concept. Our contributions are:

- A novel hierarchical correspondence field search strategy that features powerful non-locality in the image space (see Figure 5 a)), but locality in the flow space (for smoothness) and can utilize hierarchy levels (scales) as effective outlier sieves. It allows to obtain better results with hierarchies/scales than without, even for tiny objects and other details.
- We extend the common forward backward consistency check by a novel two way consistency check as well as region and density based outlier filtering.
- We show the effectiveness of our approach by clearly outperforming ANNF and by obtaining the best result on MPI-Sintel [8] and the second best on KITTI [13].

2. Related Work

Dense optical flow research started more than 30 years ago with the work of Horn and Schunck [18]. We refer to publications like [2, 27, 29] for a detailed overview of optical flow methods and the general principles behind it.

One of the first works that integrated sparse descriptor matching for improved large displacement performance was Brox and Malik [7]. Since then, several works followed the idea of using sparse descriptors [34, 32, 20, 28, 26]. Only few works used dense ANNF instead [19, 9]. Chen et al. [9] showed that remarkable results can be achieved on the Middlebury evaluation portal by extracting the dominant motion patterns from ANNF. Revaud et al. [26] compared ANNF to Deep Matching [32] for the initialization of their approach. They found that Deep Matching clearly outperforms ANNF. We will use their approach for optical flow estimation and show that this is not the case for our Flow Fields.

An important milestone regarding fast ANNF estimation was Patchmatch [4]. Nowadays, there are even faster ANNF approaches [21, 16]. There are also approaches that try to obtain correspondence fields tailored to optical flow. Lu et al. [23] used superpixels to gain edge aware correspondence fields. Bao et al. [3] used an edge aware bilateral data term instead. While the edge aware data term helps them to obtain good results – especially at motion boundaries, their approach is still based on the ANNF strategy to determine correspondences, although it is unfavorable for optical flow. HaCohen et al. [15] presented a hierarchical correspondence field approach for image enhancement. While it does well in removing outliers, it also removes inliers that are not supported by a big neighborhood (in each scale). Such inliers are especially important for optical flow as they cannot be determined by the classical coarse to fine strategy. Our approach cannot only preserve such isolated inliers, but can also spread them if needed (Figure 5 a)).

A technique that shares the idea of preferring locality (to avoid outliers) with our approach is region growing in 3D reconstruction [14, 12]. It is usually computationally expensive. A faster GPU parallelizable alternative based on PatchMatch [4] was presented in our previous work [1]. It shares some ideas with our basic approach in Section 3.1, but was not designed for optical flow estimation and lacks many important aspects of our approach in this paper.

3. Our Approach

In this section we detail our Flow Field approach, our extended outlier filter and the data terms used in the tests of our paper. Flow Fields are described in two steps. First we describe a basic (non-hierarchical) Flow Field approach. Afterwards, we build our full (hierarchical) Flow Field approach on top of it. Given two images $I_1, I_2 \subset \mathbb{R}^2$ we use the following notation: $P_r(p_i)$ is an image patch with patch radius r centered at a pixel position $p_i = (x, y)_i \in I_i$ $i = 1, 2$. The total size of our rectangular patch is $(2r + 1) \times (2r + 1)$ pixels. Our goal is to determine the optical flow field of I_1 with respect to I_2 i.e. the displacement field for all pixels $p_1 \in I_1$, denoted by $F(p_1) = M(p_1) - p_1 \in \mathbb{R}^2$ for each pixel p_1. $M(p_1)$ is the corresponding matching position $p_2 \in I_2$ for a position $p_1 \in I_1$. All parameters mentioned below are assigned in Section 4.

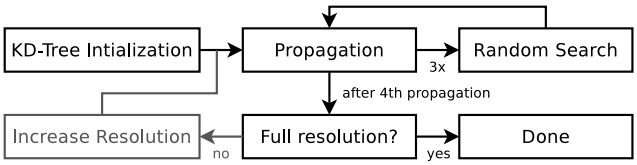


Figure 2. The pipeline of our Flow Field approach. For the basic approach we only consider the full resolution.

3.1. Basic Flow Fields

The first step of our basic approach is similar to the kd-tree based initialization step of the ANNF approach of He and Sun [16]. We do not use any other step of [16] as we have found them to be harmful for optical flow estimation, since they introduce *resistant outliers* whose matching errors are below those of the ground truth. Once introduced, a purely data based approach without regularization cannot remove them anymore. The secret is to avoid finding them.[1]

Our approach, outlined in Figure 2, works as follows: First we calculate the Walsh-Hadamard Transform (WHT) [17] for all patches $P_r(p_2)$ centered at all pixel positions p_2 in image I_2 similar to [16].[2] In contrast to them we use the first 9 bases for all three color channels in the CIELab color space. The resulting 27 dimensional vectors for each pixel are then sorted into a kd-tree with leaf size l. We also split the tree in the dimension of the maximal spread by the median value. After building the kd-tree we create WHT vectors for all patches $P_r(p_1)$ at all pixel positions in image I_1 as well and search the corresponding leaf within the kd-tree (where it would belong to if we would add it to the tree). All l entries L in the leaf found by the vector of the patch $P_r(p_1)$ are considered as candidates for the initial Flow Field $F(p_1)$. To determine which of them is the best we calculate their matching errors E_d with a robust data term d and only keep the candidate with the lowest matching error in the initial Flow Field, i.e.

$$F(p_1) = arg\,min_{p_2 \in L}(E_d(P_r(p_1), P_r(p_2))) - p_1 \quad (1)$$

This is similar to *reranking* in [16]. We call points in the initial Flow Field arising directly from the kd-tree *seeds*. Larger l increase the probability that both correct seeds and resistant outliers are found. However, if both are found at a position the resistant outlier prevails. Thus, it is advisable to keep l small and to utilize the local smoothness of optical flow to propagate rare correct seeds in the initial Flow Field into many surrounding pixels – outliers usually fail in this regard as their surrounding does not form a smooth surface. The propagation of our initial flow values works similar to the propagation step in the PatchMatch approach [4] i.e. flow values are propagated from position $(x, y-1)_1$ and $(x-1, y)_1$ to position $p_1 = (x, y)_1$ as follows:

$$F(p_1) = arg\,min_{p_2 \in G_1}(E_d(P_r(p_1), P_r(p_2))) - p_1$$
$$G_1 = \{F(p_1), F\big((x, y-1)_1\big), F\big((x-1, y)_1\big)\} + p_1 \quad (2)$$

G_1 are the considered flows for our first propagation step. It is important to process positions $(x, y-1)_1$ and $(x-1, y)_1$ with Equation 2 before position $(x, y)_1$ is processed. This allows the propagation approach to propagate into arbitrary directions within a 90 degree angle (see Figure 3 a)). As optical flow varies between neighboring pixels, but propagation can only propagate existing flow values our next step is a random search step. Here, we modify the flow of each pixel p_1 by a random uniformly distributed offset O_{rnd} of at most R pixels. If the matching error E decreases we replace the flow F by the new flow $F + O_{rnd}$. O_{rnd} is a subpixel accurate offset which leads to subpixel accurate positions $M(p_1)$. The pixel colors of $M(p_1)$ and $P_r(M(p_1))$ are determined by bilinear interpolation. Early subpixel accuracy not only improves accuracy, but also helps to avoid outliers as subpixel accurate matches have a smaller matching error.

In total we perform alternately 4 propagation and 3 random search steps (all with the same R) as shown in Figure 2. While the first propagation step is performed to the right and bottom, the subsequent three propagation steps are performed into the directions shown in Figure 3 c). Many approaches that perform propagation (e.g. [16]) do not consider different propagation directions. Even the original PatchMatch approach only considers the first two directions. While these already include all 4 main directions, we have to consider that propagation actually can propagate into all directions within a quadrant (see Figure 3 a)) and that there are 4 quadrants in the full 360 degree range.

Extensive propagation with random search is important to distribute rare correct seeds into the whole Flow Field. The locality of single propagation steps and random search (with small R) effectively prevents the Flow Field from introducing new outliers not existing in the initial Flow Field.

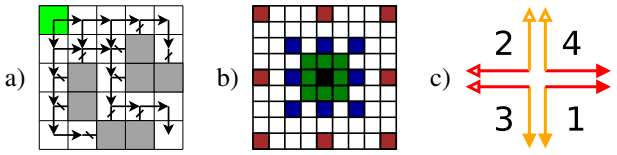


Figure 3. a) Example for the ability of propagation to propagate into different directions within a 90 degree angle. Gray pixels reject the flow of the green seed pixel. In practice each pixel is a seed. b) Pixel positions of P_1 (green), P_1^2 (blue) and P_1^4 (red). The central pixel is in black. c) Our propagation directions.

3.2. Flow Fields

Our basic Flow Fields still contain many resistant outliers arising from kd-tree initialization. We can further re-

[1] ANNF try to reproduce the NNF that contains all resistant outliers.

[2] For WHTs patches must be split in the middle. We found that quality does not suffer from spiting uneven patches $(2r+1)$ into sizes r and $r+1$.

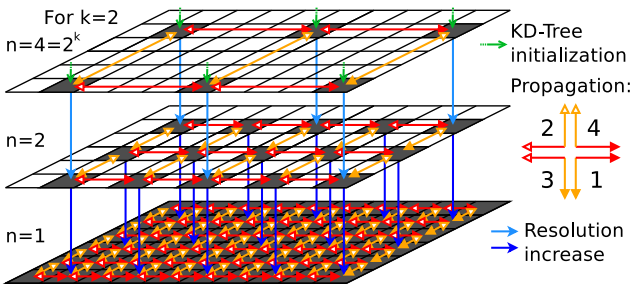


Figure 4. Illustration of our hierarchical Flow Field approach. Flow offsets saved in pixels are propagated in all arrow directions.

duce their amount (and the amount of inliers) by not determining an initial flow value for each pixel. This helps as inliers usually propagate much further than outliers (optical flow is smooth, outliers are usually not). However, to cover the larger flow variations between fewer inliers (that are further apart from each other) the random search distance R must be increased, which raises the danger of adding close by resistant outliers. A way to avoid this is to increase r, as well. This helps e.g. in the presence of repetitive patterns or poorly textured regions, but also significantly increases computation time and creates new failure cases e.g. close to motion discontinuities and for small objects. Furthermore, a larger r (and R) leads to less accurate matches.

We found a powerful solution (outlined in Figure 4) that avoids most of the disadvantages of large patches while being even more robust: First we define that $P_r^n(p_i)$ is a subsampled patch at pixel position p_i with patch radius $r * n$ that consists of only each nth pixel within its radius including the center pixel i.e. (see Figure 3 b) for an illustration):

$$(x^*, y^*) \in P_r^n((x,y)) \Rightarrow \begin{cases} |(x^* - x)| \bmod n = 0 \\ |(y^* - y)| \bmod n = 0 \end{cases} \quad (3)$$

The pixel colors for $P_r^n(p_i)$ are not determined from image I_i, but from a smoothed version of I_i that we call I_i^n. This is similar to using image pyramids and using P_r on a higher pyramid level. The difference is that I_i^n has the full image resolution and that p_i is an actual pixel position on the full resolution, which effectively prevents upsampling errors.

As I_i^n only has to be calculated once we can afford to use an expensive low pass filter without noticeable difference in overall processing speed. In practice, we downsample I_i by a factor of n with area based downsampling, before upsampling it again with Lanczos interpolation [11] to obtain I_i^n. We always start with $n = 2^k$. Our full Flow Field approach first initializes only each nth pixels $p_1^n = (x_n, y_n)_1$ with $x_n \mod n = 0$ and $y_n \mod n = 0$ (see Figure 4). Initialization is performed similar to the basic approach:

$$F(p_1^n) = arg\,min_{p_2 \in L}\Big(E_d\big(P_r^n(p_1^n), P_r^n(p_2)\big)\Big) - p_1^n \quad (4)$$

Note that the kd-tree samples L are identical to those of the basic approach. We still use non-subsampled patches $P_r(p_i)$ for the WHT vectors for an accurate initialization.

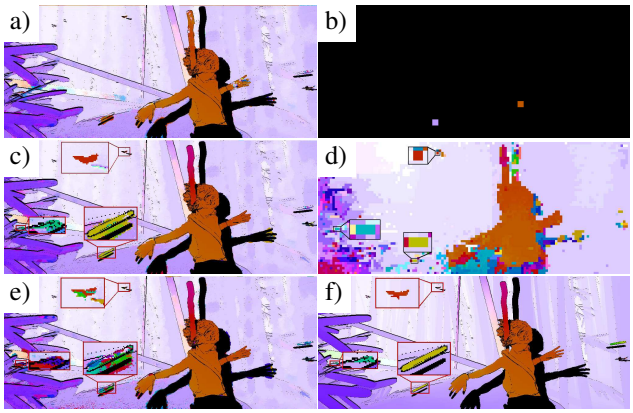


Figure 5. **a)** Flow Field obtained with $k = 3$ with b) as only initialization (black pixels in b) are set to infinity). It shows the powerfulness of our hierarchical propagation. **c)** Like a) but with kd-tree initialization. The 3 marked details are preserved due to their availability in the coarse level d). **e)** like c) but without hierarchies (basic approach). Details are not preserved. **f)** ground truth. Note: As correspondence estimation is impossible in occluded areas and as orientation we blacked such areas out.

After initialization we perform propagation and random search similar to the basic approach. Except that we only propagate between points p_1^n i.e. $(x_n - n, y_n)_1, (x_n, y_n - n)_1 \rightarrow (x_n, y_n)_1$ etc. (see Figure 4) and that we use $R_n = R * n$ as maximum random search distance. After determining $F(p_1^n)$ using patches P^n, we determine $F(p_1^m), m = 2^{k-1}$ in the same way using patches P^m. Hereby, the samples $F(p_1^n)$ are used as seeds instead of kd-tree samples. Positions p_1^m that are not part of p_1^n receive an initial flow value in the first propagation step of the hierarchy level $k - 1$. This approach is repeated up to the full resolution $F(p_1^1) = F(p_1)$ (see Figure 2 and 4).

Propagation and random search (with small enough R) are usually too local in flow space to introduce new outliers, while propagations of lower hierarchy levels are likely to remove most outliers persisting in higher levels, since resistant outliers are often not resistant on all levels. Thus, hierarchy levels serve as effective outlier sieves (see videos in supplementary material). Also, matching patches P_r^n is mostly significantly more robust than matching patches P_{r*n} if r is sufficiently large. Deformations affect smoothed patches e.g. less, as smoothing allows more matching inaccuracy for a good match. Still, we obtain accurate flow values as we are iteratively increasing the resolution.

In contrast to ordinary multi scale approaches, our hierarchical approach is non-local in the image space. Figure 5 a) demonstrates how powerful this non-locality is. The Flow Field is only initialized by two flow values with a flow offset of 52 pixels to each other (Figure 5 b)). This is more

than the random search step of all hierarchy levels together can traverse. Thus, the orange flow is a propagation barrier for the violet flow (Like gray pixels in Figure 3 a)). Anyhow, our approach manages to distribute the violet flow and similar flows determined by random search throughout the whole image. We originally performed the experiment to prove that the flow can be propagated into the arms starting from the body, but our approach even can obtain the flow for nearly the whole image with such poor initialization.

Figure 5 c) shows that we can even find tiny objects with our hierarchical approach: The 3 marked objects are well persevered in c) due to their availability in the coarse level d). Remarkably, these objects are only preserved when using hierarchical matching. Our basic approach without hierarchies only preserves parts of the upper object (a butterfly) riddled with outliers, although its seeds are a superset of the seed of the hierarchical approach – but it fails in avoiding resistant outliers. Our hierarchical approach preserves tiny objects due to unscaled WHTs (initialization) and since the image gradients around tiny objects create local minima in E_d, even for huge patches P_r^n. This is sufficient as lower minima (resistant outliers) are successfully avoided by our search strategy. Our visual tests showed that our approach with $k = 3$ in general preserves tiny objects and other details better than our basic approach. With too large k (> 3) tiny objects are due to lack of seeds not that well preserved.

3.3. Data Terms

In our paper we consider two data terms:

1. Census transform [36]. It is computationally cheap, illumination resistant and to some extend edge aware. We use the sum of census transform errors over all color channels in the CIELab color space for E_d.
2. Patch based SIFT flow [22]. A SIFT flow pixel usually has $S = 3$ channels. The colors are determined by first calculating the 128 dimensional SIFT vector for each pixel and then reducing it by PCA to S dimensions. The error between Sift Flow colors is determined by the L_2 distance. For the images I_i^n we found it advantageous to smooth the Sift Flow images as described in Section 3.2 and to not use larger SIFT features instead. WHTs are still calculated in the CIELab color space.

3.4. Outlier Filtering

A common approach of outlier filtering is to perform a forward backward consistency check. We found that the robustness can be improved by a consistency check between two Flow Fields with different patch radii, as outliers often diverge into different directions. Practically, we calculate a backward flow for two patch radii r and r_2 and delete a pixel if it is not consistent to both backward flows i.e. if

$$|F(p_1) + F_j^b(p_1 + F(p_1))| < \epsilon, j \in 1, 2 \qquad (5)$$

is not fulfilled for one of the two backward flows F_j^b. For a 3 way check an additional forward flow could be added, but for a 2 way check an extra backward flow performs better (see supplementary material for an explanation).

After the consistency check many of the remaining outliers form small regions that were originally connected to removed outliers. Thus, we remove these regions as follows: First, we segment the partly outlier filtered Flow Field into regions. Neighboring pixels belong to the same region if the difference between their flow is below 3 pixels.[3] Then, we test for regions with less than s pixels if it is possible for that region to add at least one outlier that was removed by the consistency check with the same rule. If this is possible, we found a small region that was originally connected to an outlier and we remove all points in that region.

3.5. Sparsification and Dense Optical Flow

To fill the gaps created by outlier filtering we use the edge preserving interpolation approach proposed by Revaud et al. [26] (EpicFlow). We found that EpicFlow does not work very well with too dense samples. Thus, we select one sample in each 3x3 region in the outlier filtered Flow Field if the region still contains at least e samples. This is our last consistency check. We found that even after region based filtering most remaining outliers are in sparse regions where most flow values were removed. The sample that is selected is the sample for which the sum of both forward backward consistency check errors is the smallest.

4. Evaluation

We evaluate our approach on 3 optical flow datasets:

- MPI-Sintel [8]: It is based on an animated movie and contains many large motions up to 400 pixels per frame. The test set consists of two versions: *clean* and *final*. *Clean* contains realistic illuminations and reflections. *Final* additionally adds rendering effects like motion, defocus blurs and atmospheric effects.
- Middlebury [2]: It was created for accurate optical flow estimation with relatively small displacements. Most approaches can obtain an endpoint error (EPE) in the subpixel range.
- KITTI [13]: It was created from a platform on a driving car and contains images of city streets. The motions can become large when the car is driving.

In Section 4.1 we perform experiments to analyze our approach and compare it to ANNF. In Section 4.2 we present our results in the public evaluation portals of the introduced datasets. For simplicity, we use $k = 3$ and $R = 1$ which we

[3] Only the flow differences between neighboring pixels count. The flow values of a region can vary by an arbitrary offset.

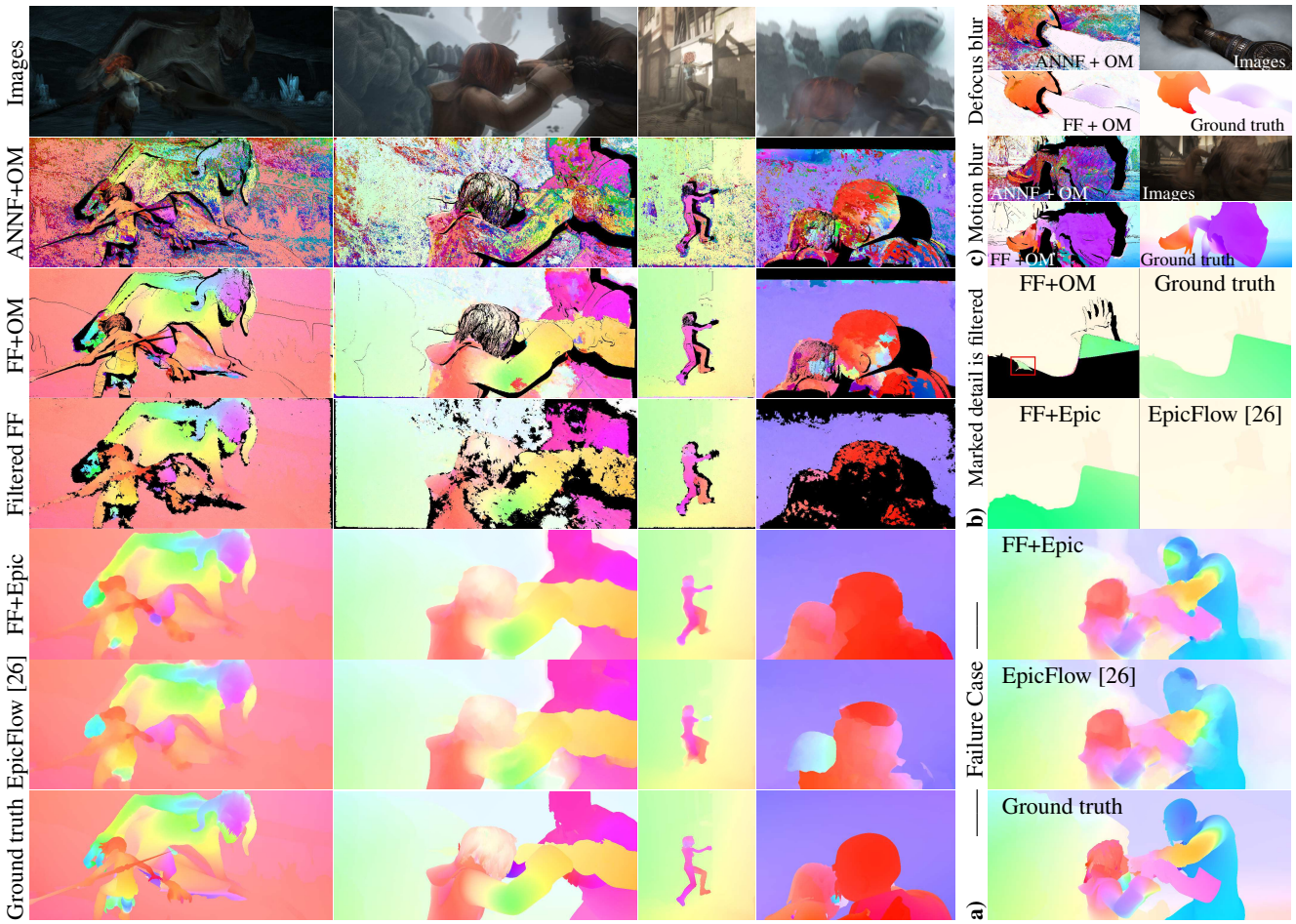


Figure 6. The left 4 columns show example results. *Images* is the average of both input images. For ANNF we use [16] in a fair way (see text). *FF* means Flow Fields. *OM* means that the ground truth occlusion map is added (black pixels, it is incomplete at image boundaries). *Filtered FF* is after outlier filtering (deleted pixels in black). *FF+Epic* is EpicFlow applied on our Flow Fields. *EpicFlow* is the original EpicFlow. Right column: a) Our approach fails in the face of the right person (outlier) and at its back (blue samples too far right). Still our EPE is smaller due to more preserved details. b) The marked bright green flow is not considered due to too strong outlier filtering. This makes a huge difference here. c) We show that our Flow Fields (bottom left) perform much better in presence of blur than ANNF (top left).

have found to perform well based on a few incoherent tests (and Table 1 and 2 for k), $l = 8$ equivalent to [16] and $r = 8$ and $r_2 = 6$ as runtime tradeoffs for the census transform. Only ϵ (± 1), e (± 1), s (± 50) and $r = r_2 + 1$ for SIFT flow were tuned coherently on all training frames. ϵ, e and s are set to 5, 4 and 50 for MPI-Sintel, to 1, 7 and 50 for Middleburry and to 1, 9 and 150 for KITTI, respectively. If not mentioned differently we use the census transform as data term. For EpicFlow applied on Flow Fields we use their standard parameters which are tuned for their Deep Matching features [32]. For a fair comparison we use the same parameters (tuning ϵ, e, s for ANNF does not affect our results), data term and WHTs in CIELab space for the ANNF approach [16] (the original approach performs even worse). This includes ANNF results in Section 4.1 and in Figure 1 and 6. More details regarding parameter selection and more experiments can be found in our supplementary material.

Visual results are shown in Figure 6. EpicFlow can preserve considerably more details with our Flow Fields than with the original Deep Matching features. Even in failure cases like in Figure 6 a) (right column), our approach often still achieves a smaller EPE thanks to more preserved details. Note that the shown failure cases also happen to the original EpicFlow. Despite more details our approach in general does not incorporate more outliers. The occasional removal of important details like the one marked in Figure 6 b) remains an issue – even for our improved outlier filtering approach. The marked detail is important as the flow of the very fast moving object is different on the left (brighter green). Still, we can in general preserve more details than the original EpicFlow. Figure 6 c) shows that our approach also performs well in presence of motion and defocus blur.

4.1. Experiments

In the introduction we claimed that our Flow Fields are better suited for optical flow estimation than ANNF and contain significantly fewer outliers. To prove our statement quantitatively we compare our Flow Fields with different number of hierarchy levels k to the state-of-the-art ANNF approach presented in [16]. We also compare to the real NNF calculated in several days on the GPU. The comparison is performed in Table 1 with 4 different measures:

- The percentage of flows with an EPE below 3 pixels.
- The EPE bounded to a maximum of 10 pixels for each flow value (EPE10). Outliers in correspondence fields can have arbitrary offsets, but the difficulty to remove them does not scale with their EPE. Local outliers can even be more harmful since they are more likely to pass the consistency check. The EPE10 considers this.
- The real endpoint error (EPE) of the raw correspondence fields. It has to be taken with care (see EPE10).
- The EPE after outlier filtering (like in Section 3.4) and utilizing EpicFlow to fill the gaps (Epic).

All 4 measures are determined in non-occluded areas only, as it is impossible to determine data based correspondences in occluded areas. As can be seen, we can determine nearly 90% of the pixels on the challenging MPI-Sintel training dataset with an EPE below 3 pixels, relying on a purely data based search strategy which considers each position in the image as possible correspondence. With weighted median filtering (weighted by matching error) this number can even be improved further, but the distribution is unfavorable for EpicFlow (it probably removes important details similar to some regularization methods). In contrast, more hierarchy levels up to the tested $k = 3$ have a positive effect on the EPE as they successfully can provide the required details.

Bao et al. [3] also used hierarchical matching in their approach to speed it up. However, despite joined bilateral upsampling combined with local patch matching in a 3x3 window they found that the quality on Middlebury drops clearly due to hierarchical matching. As can be seen in Table 2 this is not the case for our approach. As expected from the experiment in Figure 5 the quality even rises. Note that the Epic result does not rise much as EpicFlow is not designed for datasets like Middlebury with EPEs in the subpixel area. Even with the ground truth it does not perform much better than with our approach. Our upsampling strategy requires 11 patch comparisons while [3] requires 9 comparisons and joined bilateral upsampling. However, in contrast to their upsampling strategy ours is non-local which means that we can easily correct inaccuracies and errors from a coarser level (the non-locality is demonstrated in Figure 5 a)).

Method	≤ 3 pixel	EPE10	EPE	Epic
$k = 3$+median	92.17%	0.91	4.41	2.13
$k = 3$	89.20%	1.30	6.04	2.04
$k = 2$	88.79%	1.36	8.84	2.08
$k = 1$	86.88%	1.57	14.65	2.27
$k = 0$	79.13%	2.29	32.51	2.81
ANNF [16]	68.05 %	3.38	59.11	3.41
NNF	60.20 %	4.18	110.30	- [4]
Original EpicFlow	-			2.48

Table 1. Comparison of different correspondence fields on a representative subset (2x every 10th frame) on non-occluded regions of the MPI-Sintel training set (*clean* and *final*). See text for details.

Method	≤ 1 pixel	EPE3	EPE	Epic
Ground truth	100%	0.0	0.0	0.214
$k = 3$	87.08 %	0.499	1.16	0.239
$k = 2$	86.81%	0.508	2.32	0.240
$k = 0$	81.93%	0.670	12.33	0.240
Original EpicFlow	-			0.380

Table 2. Comparison of our approach with different hierarchy levels on the Middlebury training dataset to demonstrate that the quality does not suffer from hierarchical matching like in [3]. Note that the Epic result is biased to the value in the first row.

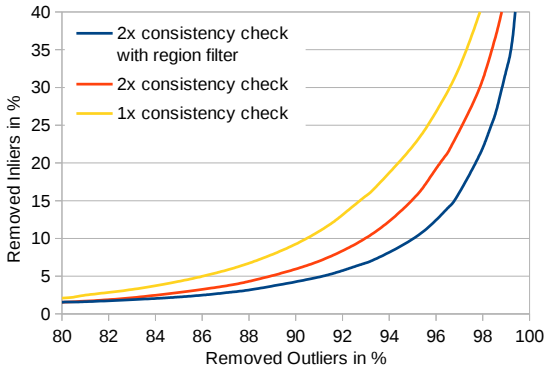


Figure 7. Percentage of removed outliers versus percentage of removed inliers, for an outlier threshold of 5 pixels (We vary ϵ).

Outlier Filtering Figure 7 shows the percentage of outliers that are removed versus the percentage of inliers that are removed by different consistency checks on the MPI-Sintel training set. Both the 2x consistency check as well as the region filter increase the amount of removed outliers for a fixed inlier ratio. We also considered using the matching error E_d for outlier filtering, but there is no big gain to achieve (see supplementary material).

4.2. Results

MPI-Sintel Our results compared to other approaches on MPI-Sintel can be seen in Table 3. We clearly outperform the original EpicFlow as well as all other approaches. We can reduce the EPE on *final* by nearly 0.5 pixels and nearly 0.4 pixels on *clean*. Most of this advance is obtained in the non-occluded area but EpicFlow also rewards our bet-

[4] No backward flow calculated

ter input in the occluded areas. On *clean* we can reduce the EPE in non-occluded areas to only 1.056 pixels, which is far from the performance of most other approaches. On *final* we can drastically reduce the error of fast motions of more than 40 pixels (s40+). Our approach also performs well close to occlusion boundaries (d0-10).

Middlebury On Middlebury we obtain an average rank of 38.0 (EpicFlow: 52.2) and an average EPE of 0.33 (EpicFlow: 0.39). Our rank is either exactly the same as EpicFlow (e.g. 69 on Army) or better (e.g. 4 instead of 53 on Urban). As already discussed in Section 4.1 the EPE rank that can be obtained with EpicFlow on Middlebury is limited, as EpicFlow is not designed for such datasets. Nevertheless, we can improve the result on some datasets.

KITTI On KITTI patch based approaches seem to either perform poorly [9], use scale robust features [25] or special techniques like plane fitting [3]. We think this is because image patches of walls and the street are undergoing strong scale changes and deformations (due to high view angle). With the census transform our results are good for an unmodified patch based approach but not state-of-the-art (see supplementary material). However, as our approach allows to exchange data terms as easily as parameters we use the more deformation and scale robust SIFT flow data term to obtain the results on KITTI presented in Table 4.[5] We use small patches with $r = 3$ and $r_2 = 2$ as the benefit of SIFT to be scale and deformation robust is otherwise destroyed. Due to the small patch sizes we use $S = 12$ and $S_2 = 18$ for the 2. consistency check as runtime tradeoffs. As can be seen, we just missed the best approach by 0.01% in *> 3 pixel nocc*. Our approach only fails slightly in *>3 pixel all*. However, note that interpolation into the occluded areas is performed by EpicFlow. There might be better interpolation methods for the specific application of planar street scenes. Compared to the original EpicFlow we are much better. Indeed, our approach is currently the only one with top performance on Sintel clean and final, as well as KITTI.

Interesting is that although we have to use very small patches on KITTI, our hierarchical approach (with enlarged but blurred patches) still works very well. This demonstrates that the concept of hierarchical matching works even in challenging cases when matching large patches fails.

Runtime Our approach including EpicFlow requires 18s for a frame in MPI-Sintel running on the CPU.[6] By using patches with $r = 6$ and no second consistency check we can reduce the total time to 10s with an EPE increase of only 0.13 on *final* (training set) and even a decrease of 0.02 on *clean* as smaller patches perform better here. On KITTI our approach with SIFT flow needs 23 seconds per image (13 seconds without PCA). The best approach PPR-Flow needs 800s and the third best NLTGV-SC 16s, but on the GPU.

[5] Our approach with SIFT flow also outperforms EpicFlow on the MPI-Sintel and Middlebury training sets (but less). See supplementary material.

[6] In detail: $3 \times 0.4s$ for kd-tree initialization, $2 \times 5s + 1 \times 3s$ for the three Flow Fields, $0.1s$ for outlier filtering and $3.5s$ for EpicFlow.

Method (Final)	EPE all	EPE nocc.	EPE occ.	d0-10	s40+
Flow Fields	**5.810**	**2.621**	**31.799**	**4.851**	**33.890**
EpicFlow [26]	6.285	3.060	32.564	5.205	38.021
TF+OFM [20]	6.727	3.388	33.929	5.544	39.761
SparseFlowFused[28]	7.189	3.286	38.977	5.567	44.319
DeepFlow [32]	7.212	3.336	38.781	5.650	44.118
NFF-Local [9]	7.249	2.973	42.088	4.896	44.866
Method (Clean)	EPE all	EPE nocc.	EPE occ.	d0-10	s40+
Flow Fields	**3.748**	**1.056**	**25.700**	2.784	**23.602**
EpicFlow [26]	4.115	1.360	26.595	3.660	25.859
PH-Flow [35]	4.388	1.714	26.202	3.612	27.997
NNF-Local[9]	5.386	1.397	37.896	**2.722**	36.342

Table 3. Results on MPI-Sintel. Bold results are the best, underlined the 2. best. (n)occ = (non) occluded. d0-10 = 0 - 10 pixels from occlusion boundary. s40+ = motions of more than 40 pixels.

Method	Rank	>3 pixel nocc.	>3 pixel all	EPE nocc.	EPE all
PH-Flow [35]	1	**5.76 %**	**10.57 %**	**1.3 px**	**2.9 px**
Flow Fields	2	5.77 %	14.01 %	1.4 px	3.5 px
NLTGV-SC [25]	3	5.93 %	11.96 %	1.6 px	3.8 px
DDS-DF [31]	4	6.03 %	13.08 %	1.6 px	4.2 px
TGV2ADCSIFT [5]	5	6.20 %	15.15 %	1.5 px	4.5 px
EpicFlow [26]	13	7.88 %	17.08 %	1.5 px	3.8 px

Table 4. Results on KITTI test set. The table rank is the original rank excluding non optical flow methods. nocc. = Non-occluded.

5. Conclusion and Future Work

In this paper we presented a novel correspondence field approach for optical flow estimation. We showed that our Flow Fields are clearly superior to ANNF and better suited than state-of-the-art descriptor matching approaches, regarding optical flow estimation. We also presented advanced outlier filtering and demonstrated that we can obtain promising optical flow results, utilizing a state-of-the-art optical flow algorithm like EpicFlow. With our results, we hope to inspire the research of dense correspondence field estimation for optical flow. So far, sparse descriptor matching techniques are much more popular as too little effort was spent in improving dense techniques.

In future work, more advanced data terms can be tested. Thanks to intensive research mainly in stereo estimation there are nowadays e.g. many improvements for the census transform [10, 25, 24, 33]. These can probably be used to further improve our approach. Promising is also to estimate patch deformations by random search [15]. It is known that this works well for patch normals in 3D reconstruction [1].

Acknowledgements

This work was partially funded by the BMBF project DYNAMICS (01IW15003) and the EU 7th Framework Programme project AlterEgo (600610).

References

[1] C. Bailer, M. Finckh, and H. P. Lensch. Scale robust multi view stereo. In *ECCV*, pages 398–411. Springer, 2012.

[2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011.

[3] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *CVPR*, pages 3534–3541. IEEE, 2014.

[4] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *ECCV*, pages 29–43. Springer, 2010.

[5] J. Braux-Zin, R. Dupont, and A. Bartoli. A general dense image matching framework combining direct and feature-based costs. In *ICCV*. IEEE, 2013.

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36. Springer, 2004.

[7] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 33(3):500–513, 2011.

[8] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625. Springer, 2012.

[9] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow with nearest neighbor fields. In *CVPR*, pages 2443–2450. IEEE, 2013.

[10] O. Demetz, D. Hafner, and J. Weickert. The complete rank transform: A tool for accurate and morphologically invariant matching of structures. In *BMVC*, 2013.

[11] C. E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, 1979.

[12] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8):1362–1376, 2010.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 2013.

[14] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, pages 1–8. IEEE, 2007.

[15] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)*, 30(4):70, 2011.

[16] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, pages 111–118. IEEE, 2012.

[17] Y. Hel-Or and H. Hel-Or. Real-time pattern matching using projection kernels. *PAMI*, 27(9):1430–1445, 2005.

[18] B. K. Horn and B. G. Schunck. Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

[19] O. Jith, S. A. Ramakanth, and R. V. Babu. Optical flow estimation using approximate nearest neighbor field fusion. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 673–674. IEEE, 2014.

[20] R. Kennedy and C. J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 364–377. Springer, 2015.

[21] S. Korman and S. Avidan. Coherency sensitive hashing. In *ICCV*, pages 1607–1614. IEEE, 2011.

[22] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *PAMI*, 33(5):978–994, 2011.

[23] J. Lu, H. Yang, D. Min, and M. N. Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *CVPR*, pages 1854–1861. IEEE, 2013.

[24] X. Luan, F. Yu, H. Zhou, X. Li, D. Song, and B. Wu. Illumination-robust area-based stereo matching with improved census transform. In *Measurement, Information and Control (MIC)*, volume 1, pages 194–197. IEEE, 2012.

[25] R. Ranftl, K. Bredies, and T. Pock. Non-local total generalized variation for optical flow estimation. In *ICCV*, 2014.

[26] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR*, 2015.

[27] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, pages 2432–2439. IEEE, 2010.

[28] R. Timofte and L. Van Gool. Sparse flow: Sparse matching for small to large displacement optical flow. In *Applications of Computer Vision (WACV)*, pages 1100–1106. IEEE, 2015.

[29] C. Vogel, S. Roth, and K. Schindler. An evaluation of data costs for optical flow. In *Pattern Recognition (GCPR)*, pages 343–353. Springer, 2013.

[30] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *ICCV*, pages 1663–1668. IEEE, 2009.

[31] D. Wei, C. Liu, and W. Freeman. A data-driven regularization model for stereo and flow. In *3DTV-Conference*. IEEE, 2014.

[32] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

[33] G. Xiong, X. Li, J. Gong, H. Chen, and D.-J. Lee. Color rank and census transforms using perceptual color contrast. In *Control Automation Robotics & Vision (ICARCV)*, pages 1225–1230. IEEE, 2010.

[34] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *PAMI*, 34(9):1744–1757, 2012.

[35] J. Yang and H. Li. Dense, accurate optical flow estimation with piecewise parametric model. In *CVPR*, pages 1019–1027, 2015.

[36] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, pages 151–158. Springer, 1994.