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

Christian Bailer, Bertram Taetz and Didier Stricker

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 article 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 does not require explicit regularization, smoothing (like median filtering) or a new data term. Instead we solely rely on patch matching techniques and a novel multi-scale matching strategy. We also present enhancements for outlier filtering. We show that our approach is better suited for large displacement optical flow estimation than modern descriptor matching techniques. We do so by initializing EpicFlow with our approach instead of their originally used state-of-the-art descriptor matching technique. We significantly outperform the original EpicFlow on MPI-Sintel, KITTI 2012, KITTI 2015 and Middlebury. In this extended article of our former conference publication we further improve our approach in matching accuracy as well as runtime and present more experiments and insights.

Index Terms—optical flow, dense matching, correspondence fields.

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 [1], structure and motion adaptive assumptions [2], the assumption that motion discontinuities are more likely at image edges [3], or the assumption that the optical flow can be approximated by a few motion patterns [4]. 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 [1]. 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) [5]. In practice, this fails in cases where the determined motion on a coarser scale is not very close to the correct motion of a finer scale.

In contrast, for purely data based techniques like approximate nearest neighbor fields [6] (ANNF) and sparse descriptor matches [7] 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 are 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 as initial step of large displacement optical flow algorithms.

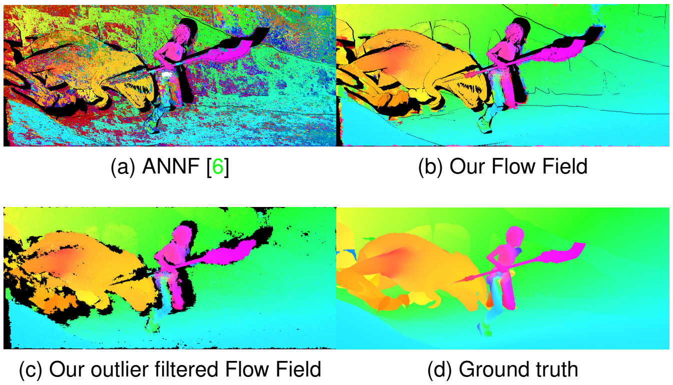

(a) ANNF [6]
(b) Our Flow Field
(c) Our outlier filtered Flow Field
(d) Ground truth

Fig. 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.

- *All authors are with the Department of Augmented Vision, German Research Center of Artificial Intelligence, 67663 Kaiserslautern, Germany E-mail: see http://av.dfki.de/members/*
- *Bertram Taetz and Didier Stricker are also with the University of Kaiserslautern.*

Manuscript received XXXX XX, 2017; revised XXXX XX, 2017.

Nowadays, most top-performing methods on challenging datasets like MPI-Sintel [8] rely on such techniques.

However, although most pixel-dense approaches use powerful patch matching [9] techniques like propagation and random search, conventional patch matching approaches are tailored to find the ANNF. This is suboptimal for optical flow estimation. The intention behind ANNF is to find the visually closest match (NNF) for each pixel, which is often not identical to 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 article we show that it is possible 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 by using carefully designed multi-scale patch matching. In contrast to common patch matching we are locally restricting the random search step to very few pixels (in flow space) and are using multi-scale matching to compensate for the small random search distance. While, there are some similarities to common pyramid approaches used in optical flow estimation, our approach does not require explicit regularization. Instead, it inherently avoids outliers, due to effects like the outlier sieve effect (see Figure 7). We call our correspondence fields *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. Our main contributions are:

- A novel multi-scale correspondence field matching strategy that features powerful non-locality in the image space (matches can, if flow is consistent, propagate an arbitrary number of pixels in just one iteration, see Figure 8 a), but locality in the flow space (the movement speed by iteration is restricted, for smoothness) and can utilize scales as effective outlier sieves. It allows to obtain better results with 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 competitive results on MPI-Sintel [8], KITTI 2012 [10] and 2015 [11].
- Several experiments to analyze our approach.

In this extended article we also present improved versions of our conference approach [12], that are much more accurate (*Flow Fields+*) or more accurate and at the same time much faster (*Flow Fields+ Fast*) than our conference version. We also present additional experiments and insights.

2 Related Work

Dense optical flow research started more than 30 years ago with the work of Horn and Schunck [1]. We refer to publications like [13], [14], [15] 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 [16]. Since then, several works followed the idea of using (sparse) descriptors [3], [17], [18], [19], [20], while few works used dense ANNF instead [4], [21]. Chen et al. [4] showed that remarkable results can be achieved on the Middlebury evaluation portal by extracting the dominant motion patterns from ANNF. Revaud et al. [3] compared ANNF to Deep Matching [18] for the initialization of their approach, called EpicFlow. 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 approach. Deep Matching is a semi-dense descriptor matching technique tailored for optical flow that does not use patch matching techniques like our approach.

An important milestone regarding fast ANNF estimation was PatchMatch [9]. They showed that an efficient way of computing an ANNF is to first initialize the ANNF with random seeds, then propagate these seeds into neighboring pixels if the matching error decreases – with a powerful propagation method that can propagate many pixels in one iteration. Finally, they perform several iterations of random movements and propagations for every pixel (if error decreases), while the maximum random movement in each iteration decreases quadratically starting from imageSize/2. In our approach we perform random movements only very locally to avoid finding difficult outliers that are part of the NNF but not optical flow.

Nowadays, there are even faster ANNF approaches [6], [22]. Korman et al. [22] used hashing to speedup the process of finding a good ANNF, while He et al. [6] used kd-trees for that. Seeds obtained in that way are then improved with patch matching techniques. 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. [24] 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. [25] presented a multi-scale correspondence field approach for image enhancement. While it does well in removing outliers, it also removes inliers that are not supported by a large 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 8 a).

A technique that shares the idea of preferring locality (to avoid outliers) with our approach is region growing in 3D reconstruction [26], [27]. It is usually computationally expensive. A faster GPU parallelizable alternative for region growing based on PatchMatch [9] was presented in our prior work [28]. It shares some ideas with our basic approach in Section 3.1, but was not designed for optical flow estimation and lacks important aspects of our approach in this article.

Recently, Hu et al. [29] improved the runtime of our multi-scale matching strategy [12] by not performing bilinear interpolation and by not considering every pixel in propagation. This improves runtime speed at the cost of accuracy. Furthermore, we recently created a CNN based data term [30] for our Flow Fields approach.

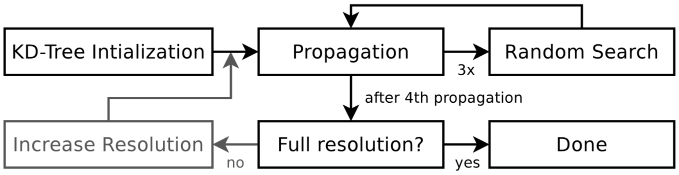


Fig. 2. The pipeline of our Flow Fields approach. For the basic approach we only consider the full resolution.

3 Our Approach

In this section we detail our Flow Fields approach, our extended outlier filter and the data terms used in the tests of this article. The idea of our approach is described in two steps. First we introduce a basic (single-scale) Flow Fields approach in Section 3.1. Then we build our full multi-scale Flow Fields approach on top of it in Section 3.2. This approach we also call *conference approach*, as it was already presented in the conference version of this article [12]. In addition, we present in this extended article improved versions of our approach called *Flow Fields+* in Section 3.3 and a faster version of *Flow Fields+* called *Flow Fields+ Fast* in Section 3.4.

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.

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 [6]. We do not use any other step of [6] 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. Hence, the secret is to avoid finding them. ANNF approaches try to reproduce the NNF that contains all resistant outliers, but due to their approximate nature they fail at doing so – which is beneficial for optical flow estimation. In our (basic) approach we want to reinforce this property even more to find even less resistant outliers, while still keeping track of inliers.

Our approach, outlined in Figure 2, works as follows: First we calculate the Walsh-Hadamard Transform (WHT) [31] for all patches $P_r(p_2)$ centered at all pixel positions p_2 in image I_2 similar to [6].[1] 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 (see Section 3.5), 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 [6]. We call points in the initial flow field arising directly from the kd-tree *seeds*. Larger l increase the probability that both correct seeds (inliers) and resistant outliers are found as they both have a lower matching error than the third possible state: non resistant outliers (due to $arg\,min$ the matching error decreases with larger l). 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 [9] i.e. flow values are propagated to position $p_1 = (x,y)_1$ from position $(x, y-1)_1$ and $(x-1, y)_1$ as follows:

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

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 4 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 $\mathcal{R}$ 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 overall accuracy, but also helps to avoid resistant outliers in our tests. An important reason for this is probably that inliers are faster in gradient descent due to propagation support by neighboring pixels (see Figure

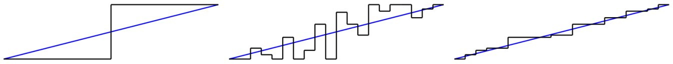


Fig. 3. Sketch shows propagation support in gradient descent in 1D space (x-axis: 1D image space, y-axis: 1D flow space). Blue lines: ground truth. Black lines: current flow. Left: after propagation of initial seeds. Middle: Random search (very slow gradient descent). Right: propagation of noisy random search samples leads to much faster gradient decent. In 2D space this effect is even more powerful, as propagation is more powerful here (see Figure 4 a).

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

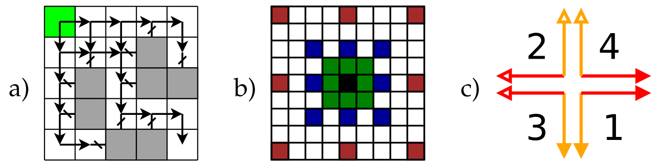


Fig. 4. a) Example for the ability of propagation to propagate into different directions within a 90 degree angle. Gray pixels reject the flow of the green seed pixel. In practice each pixel is a seed. b) Pixel positions of P_1 (green), P_1^2 (blue) and P_1^4 (red). The central pixel is in black. c)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

3). Outliers are usually not smooth. Here, the randomized gradient descent performs poorly due to lack of proper propagation support (which is good). As a result, early accurate inliers can sometime wipe out outliers before they get too accurate (and thus resistant).

In total we perform alternately 4 propagation and 3 random search steps (all with the same $\mathcal{R}$) 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 4 c). Many approaches that perform propagation (e.g. [6]) 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 4 a) and that there are 4 quadrants in the full 360 degree range.

Extensive propagation with random search (which we call *spreading*) is important to distribute rare correct seeds into the whole Flow Field. The locality of spreading (with small $\mathcal{R}$) prevents the flow field from introducing new outliers not existing in the initial flow field (see Figure 5).

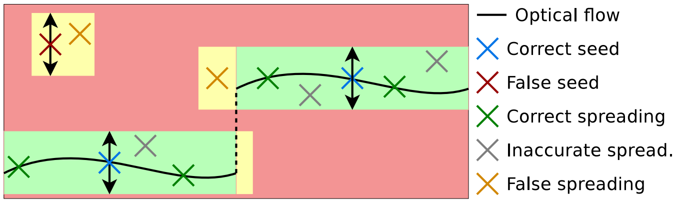


Fig. 5. Illustration of spreading of seeds, based on intuitions underlying the proposed method. X axis is image position, y axis optical flow displacement. From a seed, spreading (propagation + random search) can distribute the flow far in image direction X (propagation) but only in a narrow range in flow direction X (due to small random search distance $\mathcal{R}$). This allows inaccurate matches but no real resistant outliers with large EPE (=y axis error) if started from a correct seed. An exception are motion discontinuities (yellow ends of green areas) or false seeds. Here, outliers with large EPE are possible. However, outliers in yellow regions should not propagate well, which keeps these regions small. Propagation requires smoothness and in contrast to inliers, outlier regions are usually not smooth. If this would not be the case the smoothness assumption of optical flow [1] would not work.

3.2 Flow Fields

Our basic Flow Fields still contain many resistant outliers arising from kd-tree initialization. We can further reduce their number (and the number of initial inliers) by not determining an initial flow value for each pixel. This helps as inliers usually propagate much further than outliers.[2] However, to cover the larger flow variations between fewer inliers (that are further apart from each other) the random search distance $\mathcal{R}$ must be increased, which raises the danger of adding close by resistant outliers. A way to avoid this is to increase the patch influence area as well, either by raising r or by determining the optical flow on a downsampled image. This helps for instance in the presence of repetitive patterns or poorly textured regions, but creates new failure cases e.g. close to motion discontinuities and for small objects. Furthermore, a larger influence area and larger $\mathcal{R}$ leads to less accurate matches.

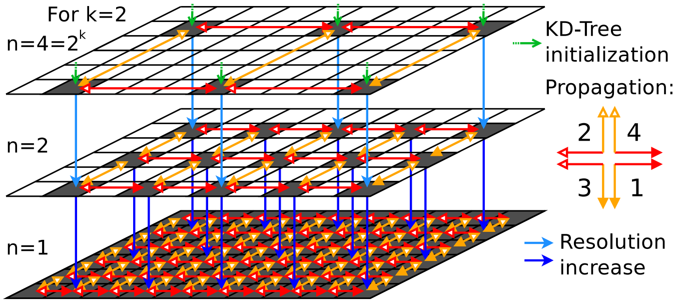


Fig. 6. Illustration of our multi-scale Flow Fields approach. Flow offsets saved in pixels are propagated in all arrow directions.

Our solution (outlined in Figure 6) avoids most of the disadvantages of large influence areas 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 rn that consists of only each nth pixel within its radius including the center pixel, i.e. (see Figure 4 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 low-pass filtered version of I_i that we call I_i^n, i.e. we use scale-spaces [32]. While scale-spaces are similar to using image pyramids and using P_r on a n times downsampled image, scale-spaces have the advantage that we can perform high-quality interpolation at very low computational cost up to pixel accuracy in the full image resolution. This is as scale-space interpolations only have to be computed once for every pixel and can be sped up e.g. with Fourier transform. In contrast, interpolation on demand has to be computed at each propagation or random search iteration. Interpolation on demand is still required for subpixel interpolation (we use fast bilinear interpolation), but in contrast to image pyramids we can use accurate pixel interpolations as starting point.

Furthermore, p_i is an actual pixel position on the full resolution, which prevents upsampling errors. Our low-pass filtering approach to obtain I_i^n is described in Section 3.6.

We always start with $n = 2^k$. Our full Flow Fields approach first initializes only each nth pixels $p_1^n = (x_n, y_n)_1$

[2] Propagation with local random search only works in case of smoothness. Due to intensive studies of the smoothness assumption [1] of optical flow we can in general assume: optical flow is smooth and outliers are usually not. Thus, they cannot propagate well.

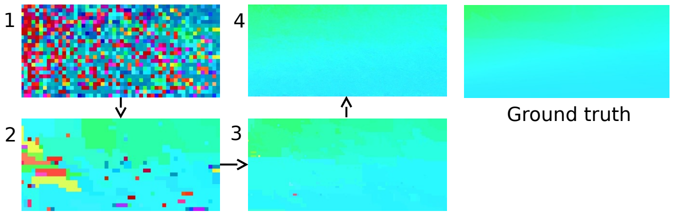


Fig. 7. Outlier sieve effect. Outliers disappear through propagations on different scales. For visualization purposes the valid gray pixels of the scales in Figure 6 are enlarged to fill the whole pixel space. Scales for the numbers are: 1: n=8 after KD-tree initialization, 2:n=8 after propagation, 3:n=4 after propagation, 4:n=1 after propagation (we skipped n=2). The full images can be found in our supplementary material.

with $x_n \mod n = 0$ and $y_n \mod n = 0$ (see Figure 6). Initialization is performed similar to the basic approach:

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

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. This is better if small objects should be preserved and also leads to slightly lower overall endpoint errors in our tests.

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 6) and that we use $\mathcal{R}_n = \mathcal{R}n$ 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 scale $k-1$. This approach is repeated up to the full resolution $F(p_1^1) = F(p_1)$ (see Figure 2 and 6).

As demonstrated in Figure 5 our spreading (propagation + random search) is usually too local to introduce new (resistant) outliers. On the other hand, spreading of finer scales has a good chance of removing outliers persisting in coarser scales, since resistant outliers are often not resistant on all scales. This is due to the fact that matching error minima are different on different scales. Formally: If $G_n = \arg\min_{p_2} E_d(P_r^n(p_1), P_r^n(p_2))$ is the global minimum match at scale n then we cannot imply that it is the minimum for a different scale as well i.e. $G_{n_1} = p_2 \nRightarrow G_{n_2} = p_2$. As a result, scales serve as a kind of outlier sieve. The outlier sieve effect is described in more detail in Section 4.3. Furthermore, Figure 7 demonstrates how spreading of different scales gradually sieves outliers scale by scale.

In contrast to ordinary multi-scale approaches, our approach is non-local in image space i.e. matches can propagate arbitrary far into image directions. Figure 8 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 8 b)). This is more than the random search step of all scales together can traverse. Thus, the orange flow is a propagation barrier for the violet flow (like gray pixels in Figure 4 a). Anyhow, our approach manages to spread 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 spread into the arms starting from the body, but our approach even can obtain the flow for nearly the whole image with such poor initialization.

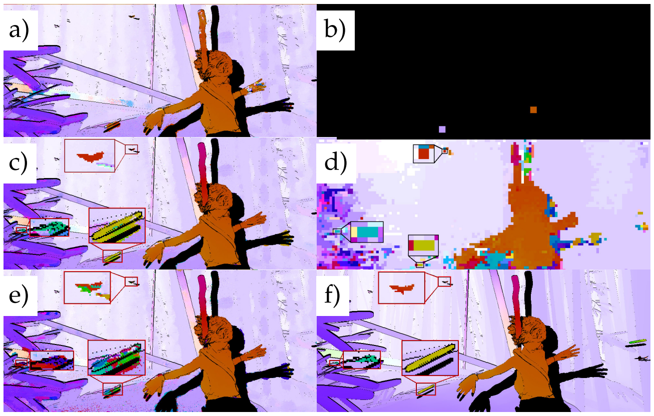


Fig. 8. **a)** Flow field obtained with $k = 3$ with b) as only initialization (black pixels in b) are set to infinity). It shows the powerfulness of our multi-scale propagation. **c)** Like a) but with kd-tree initialization. The 3 marked details are preserved due to their presence in the coarsest scale d). **e)** like c) but without scales (basic approach). Details are not preserved. **f)** ground truth. As correspondence estimation is impossible in occluded areas and as orientation we blacked such areas out.

Figure 8 c) shows that we can even find tiny objects with our multi-scale approach: The 3 marked objects are well persevered in c) due to their presence in the coarse scale d). Remarkably, these objects are only preserved when using multi-scale matching. Our basic approach without scale-spaces only preserves parts of the upper object (a butterfly) riddled with outliers, although its seeds are a superset of the seed of the multi-scale approach – but it fails in avoiding resistant outliers. Our multi-scale 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 Flow Fields+

Our original approach uses 4 propagation iterations containing 3 random search iterations with a fixed random search distance R. In our improved approach first presented in this article, we instead use two different random search distances. First we perform 4 propagation iterations (containing 3 random search iterations) with $\mathcal{R}^+ = 2\mathcal{R}$ and then 8 propagation iterations (containing 7 random search iterations) with $\mathcal{R}$. For the different scales this means that we use $\mathcal{R}_n^+ = \mathcal{R}^+ n$ and $\mathcal{R}_n = \mathcal{R}n$. Our four search directions are hereby repeated every 4 propagation iterations. The larger $\mathcal{R}^+$ helps to further distribute sparse matches in difficult situations like large flow variations with only few correct seeds, while the smaller $\mathcal{R}$ is required for accurate convergence. Large random search distances increase the risk of finding resistant outliers, but we found that the positive effect prevails if $\mathcal{R}$ and $\mathcal{R}^+$ are chosen reasonably (see Section 4.1).

n	8	4	4	2	2	1
n^*	8	6	4	3	2	1

TABLE 1
Scales and *sub-scales* used for our improved approach Flow Fields+.

3.3.1 *Sub-Scales*

Besides different random search distances our improved approach also uses *sub-scales*. While our ordinary scales are limited to scaling factors of $n \in \{2^k, k \in \mathbb{N}\}$, sub-scales $n^* \in \mathbb{N}$ can additionally contain values that are not a multiple of two. In our improved approach we use sub-scales for the patch size $P_r^{n^*}(p_i)$, image blur $I_i^{n^*}$ and random search distances $\mathcal{R}_{n^*}^+ = \mathcal{R}^+ n^*$ and $\mathcal{R}_{n^*} = \mathcal{R}n^*$, but not for propagation and random search positions of the scales (i.e. everything shown in Figure 2). Here we only use valid n. Table 1 shows the n and n^* used for the different scales in our tests of our improved approach with *sub-scales*.

3.4 Flow Fields+ Fast

The *Flow Fields+ Fast* approach aims to be much faster and still more accurate than the original *Flow Fields* approach. Compared to *Flow Fields+* we omit sub-scales. Furthermore, we use only 4 propagations with $\mathcal{R}$, similar to the original *Flow Fields* approach for the finest and thus computationally most expensive scale. Coarser scales are still executed with $4 \times \mathcal{R}^+$ and $8 \times \mathcal{R}$ like in the *Flow Fields+* approach.

3.4.1 *Flow Fields+ Fast x2*

Flow Fields+ Fast x2 is an even faster version that does not execute the finest scale at all and only uses 4 propagations with $\mathcal{R}$ on the 2nd finest scale. Starting from the 3rd finest scale this approach also uses $4 \times \mathcal{R}^+$ and $8 \times \mathcal{R}$. Furthermore, we only add one pixel in a 2x2 region to the KD-Tree as KD-Tree creation would otherwise be a significant time factor. As the approach does not process the finest scale, it creates only one match in each 2x2 region. This is not an issue since we sparsify matches before computing the final optical flow (See Section 3.8).

3.5 Data Terms

In this article we consider the following data terms:

1) Census transform [33]. It is computationally cheap, illumination robust 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 [34] (for experiments with our original conference approach) and Pixel-wise SIFT features [7] (for experiments with our improved approaches). Reasoning for the decision to switch to SIFT is provided in Section 3.6.

    a) Pixel-wise SIFT features: the error between SIFT features is determined with the L_2 distance. Due to the large feature vector of $S = 128$ dimensions only $r = 0$ is affordable in our approach ($r = 1$ has already 9 times more operations).

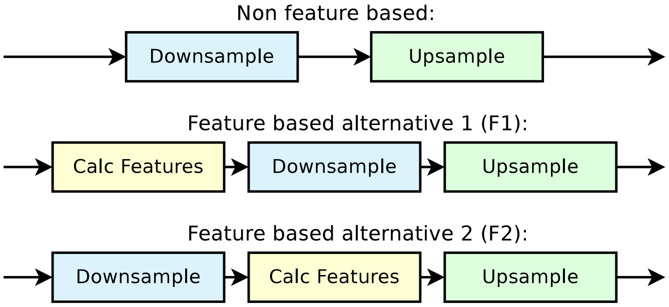


Fig. 9. Besides direct Fourier based low-pass filling an image based low-pass can be calculated by successive down and upsampling. If feature extraction is used, there are two options: features can either be calculated before or after downsampling. After feature extraction up/downsampling is performed not on the image but on the feature map.

    b) Patch based SIFT flow: the colors are determined by first calculating the 128 dimensional SIFT vector for each pixel and then reducing it by PCA to $S << 128$ dimensions. The error between Sift Flow vectors is also determined by the L_2 distance.

3) Our Convolutional Neural Network based data term presented in our very recent paper [30]. Results of the data term were recently reported in our CNN paper and are not further detailed in this article, but for completeness we also report these results here.

3.6 Features, Low-Pass Filtering and Border Condition

To perform low-pass filtering we consider the approaches shown in Figure 9. Downsampling is performed by downsampling I_i by a factor of n with area based downsampling. For upsampling we use Lanczos interpolation [35]. While ordinary low-pass filtering (e.g. for the Census Transform data term) just requires up and downsampling, feature based data terms additionally require to calculate the features. This can either be performed before (F1) or after downsampling (F2). Which setup performs better seems to depend on the data term. While in our tests SIFT flow performs better with F1, SIFT performs better with F2. As SIFT with F2 outperforms SIFT flow with F1, F2 allows us to drop the slower and more complicated SIFT flow data term in favor of SIFT features.

As feature creation for every pixel and upsampling of large feature vectors is time consuming, it is unsuitable for our fast approaches. Thus, we also introduce a fast version of F2 which we call F2F. Here we use bilinear interpolation for upsampling instead of Lanczos. Furthermore, we only calculate one feature vector for each 2x2 pixel region on the finest scale. Pixels for which no feature vector is calculated are linearly interpolated from neighboring pixels. For all other scales we still calculate feature vectors for all pixels.

3.6.1 *Border Condition*

Patch matching for border pixels requires matching pixels outside the image area. To do so we use a replicative boundary condition. This means that pixels outside the image area obtain the pixel color of the visible pixel closest to them.

3.7 Outlier Filtering

A common approach of outlier filtering is to perform a forward backward consistency check. We found that the robustness of the consistency check can be further improved by calculating the backward flow two instead of only one time. This helps as our approach is randomized. Hence, two backward flows with different seeds for the pseudo-random number generator are not identical which is why outliers often diverge into different directions. This property can be further reinforced by using different patch radii r and r_2 for both backward flows. We delete a pixel if it is not consistent to both backward flows i.e.

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

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 in our tests.

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.

3.8 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. [3] (EpicFlow). We found that EpicFlow does not work very well with too dense samples. Thus, we select only one sample in each $q \times q$ region in the outlier filtered flow field if the region still contains at least e samples. q is set to 3, except for *Flow Fields+ Fast x2* where $q = 3$ does not fit (a sampling size of 2x2 cannot be assigned to 3x3 patches). Here, we use $q = 4$ (and we also test $q = 8$ as a faster alternative). 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 4 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 [13]: 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 2012 [10]: 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.
- KITTI 2015 [11]: An improved version of KITTI 2012, where other cars actually drive (in KITTI 2012 other cars are just standing in the street).

The remainder of this section is structured as follows: In Section 4.1 we detail parameter selection. In Section 4.2 – 4.5 we analyze our approach with various kinds of experiments. In Section 4.6 we evaluate our different approaches on the MPI-Sintel and KITTI 2015 training set, while we evaluate our best approach on the test sets of all four major evaluation portals in Section 4.7. Finally, in Section 4.8 we present visual results of our approach. Further evaluations can be found in our supplementary material.

4.1 Parameter Selection

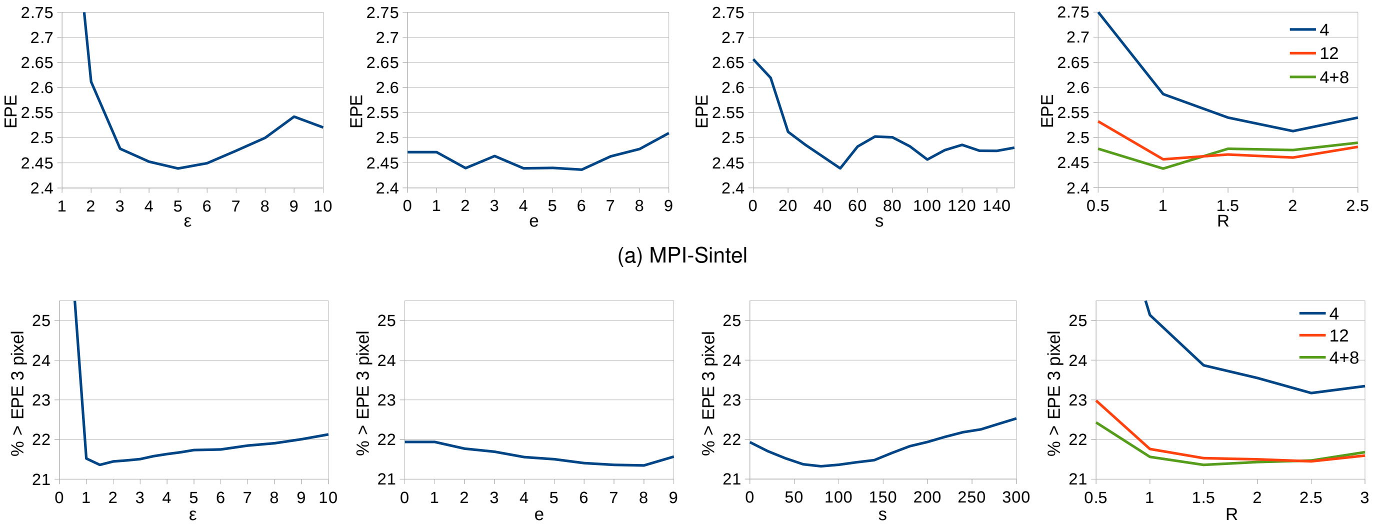
Here we detail parameter selection for our approach. In our experiments we use a kd-tree leaf size of $l = 8$ equivalent to [6] and use $k = 3$ scales as it showed to perform best for the tested optical flow benchmarks. In general, visual tests with large images showed that $k_{good} \approx \log_4(NumImagePixels/6000)$ seems to be a reasonable approximation for a good k. Note that this is only based on few visual observations and might vary depending on other parameters and the dataset.

We set random search distance $\mathcal{R}$ to $\mathcal{R} = 1$ for experiments on MPI-Sintel and $\mathcal{R} = 1.5$ for experiments on KITTI. These values are based on the experiments in Figure 10 right. The results of our conference approach *Flow Fields* are created with a fixed $R = 1$. The parameters ϵ (outlier filter threshold), e and s are tuned coherently for our results in Section 4.7 on the corresponding training set with stepsizes $\epsilon \pm 0.5$ (i.e. ϵ = 0.5,1,1.5,2...) $e \pm 1$ and $s \pm 50$. Determined parameters for our public results can be found in our supplementary material.

In our experiments we use the census transform data term for the MPI-Sintel and Middlebury datasets with a patch radius of $r = 8$ and $r_2 = 6$ for our conference approach *Flow Fields* and $r = 4$ and $r_2 = 3$ for our improved approaches *Flow Fields+ (Fast)*. While the values of the conference approach are based on a few incoherent tests with the *Flow Fields* approach the values of our improved approaches are based on tests of different r on the whole MPI-Sintel training set.

For our experiments on KITTI 2012 and 2015 we use data terms based on the deformation and scale robust SIFT features instead (improved approach: SIFT, conference approach: SIFT flow with $r = 3$, $S = 12$ PCA dimensions, $r_2 = 2$, $S_2 = 12$ PCA dimensions for 2. backward flow). We use SIFT here as the KITTI dataset contains image patches of walls and the streets that are undergoing extreme scale changes and deformations (due to large viewing angles). Thus, patch based approaches perform poorly here [4]. By using a different more appropriate data term for KITTI we also demonstrate that in our approach the data term can easily be adapted to the problem.

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



(a) MPI-Sintel

(b) KITTI 2015

Fig. 10. The influence of different parameters of our approach. We plot the main measures for each dataset.

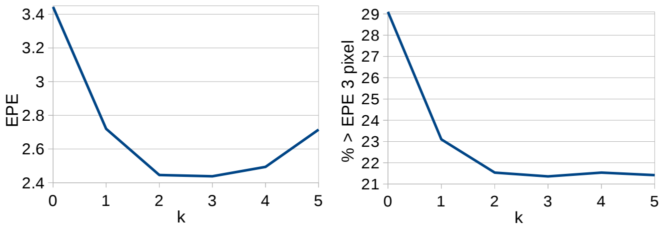


Fig. 11. The influence of the parameter k on our approach. We plot the main measures for each dataset.

For EpicFlow [3] applied on our approach we use their standard parameters which are tuned for Deep Matching features [18]. As there are no standard parameters for KITTI 2015 we use slightingly modified KITTI 2012 parameters. 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 our tests with the ANNF approach [6] (the original approach performs even worse). This includes ANNF results in Section 4.2 and in Figure 1 and 16.

4.1.1 *Influence of parameters*

The influence of our parameters can be seen in Figure 10 and 11. The optimal value of ϵ depends strongly on the dataset and data term, but is in our tests always monotonically decreasing in a large range around the minimum. Too small values are more harmful than too large ones. e and s are noisy on MPI-Sintel but also contain a clear minimum with monotonically decreasing range on KITTI 2015. We think the noise on MPI-Sintel is caused by the fact that it contains different sub-datasets with different challenges. These sub-datasets have different optimums for e and s. The rightmost plots show the parameter $\mathcal{R}$. “4” means 4 propagation iterations, “12” means 12 iterations. “4+8” means 4 iterations with $\mathcal{R}^+$ and 8 iterations with $\mathcal{R}$ as described in Section 3.3. As can be seen, our approach of using 4+8 iterations performs the best if $\mathcal{R}$ is chosen reasonably. For too large $\mathcal{R}$ the error increases faster than with 12 fixed $\mathcal{R}$ iterations, as the 4+8 approach also uses $\mathcal{R}^+ = 2\mathcal{R}$. While the difference between 4 and 12 iterations is larger than between 12 and 8+4, 8+4 has the benefit that it has the same runtime as 12. In our conference approach we simply used 4 iterations with $\mathcal{R} = 1$, which is suboptimal.

Figure 11 shows the influence of k. While $k = 3$ is optimal for both MPI-Sintel as well as KITTI 2015 the error only increases slightly for larger k on KITTI but significantly on MPI-Sintel. This is likely caused by the fact that MPI-Sintel contains many more small independently moving objects than KITTI. These cannot be determined anymore if k is too large.

4.2 Comparison to ANNF

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 scales k to the state-of-the-art ANNF approach presented in [6]. We also compare to the real NNF calculated in several days on a GPU. The comparison (to our *Flow Fields* approach) is performed in Table 2 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 3.7) 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

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 [6]	68.05 %	3.38	59.11	3.41
NNF	60.20 %	4.18	110.30	- [4]
Original EpicFlow		-		2.48

TABLE 2
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*). Results are based on our conference approach *Flow Fields*. 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 3
Comparison of our conference approach *Flow Fields* with different scales on the Middlebury training dataset to demonstrate that the quality does not suffer from multi-scale matching like in [24]. Note that the Epic result is biased to the value in the first row.

determine nearly 90% of the pixels on the challenging MPI-Sintel training set with an EPE below 3 pixels, relying on a purely data based search strategy which considers each position in the image as a 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 scales up to the tested $k = 3$ have a positive effect on the EPE as they successfully can provide the required details. The ANNF approach of He et al. [6] underperforms our approach clearly, but in contrast to the ground truth NNF approach it fails in finding all resistant outliers. Thus, the ground truth NNF approach performs even worse.

4.2.1 *Differences to scaled matching of Bao et al. [24]*

Bao et al. [24] also used multi-scale 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 accuracy on Middlebury drops clearly due to multi-scale matching. As can be seen in Table 3, this is not the case for our approach. As expected from the experiment in Figure 8 the accuracy 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 (of our *Flow Fields* approach) requires 11 patch comparisons while [24] 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 scale (the non-locality is demonstrated in Figure 8 a).

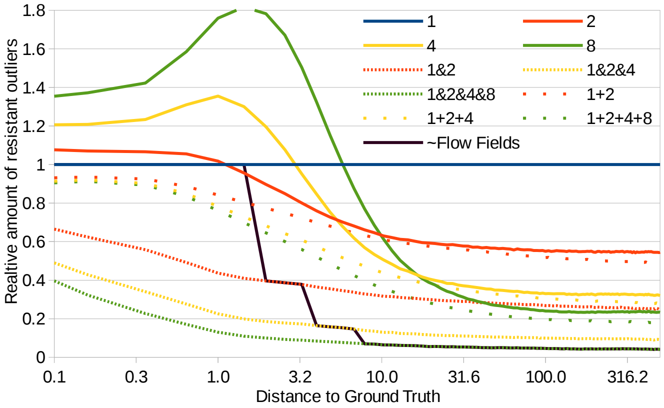


Fig. 12. We determined the probably that a point is a resistant outlier depending on the distance of the point to the ground truth match. Probabilities are plotted relative to the blue plot "1". They were determined on the MPI-Sintel dataset with many million random points.

4.3 Analysis of Outlier Sieve Effect

In this subsection we analyze the outlier sieve effect of our approach (Figure 7) on a pixel level. Particularly, we want to examine what happens if inliers and outliers are confronted with each other in pixel positions (e.g. the inlier propagates into the outlier or vice versa). Obviously (considering that the inlier is accurate) the probability that an outlier succeeds over the inlier is the probability that it is resistant. We want to determine this probability of resistance $\mathcal{P}_{\mathcal{S}}(d_f)$ for different scales $\mathcal{S}$ and distances $d_f = \|p_2 - p_2^*\|_2$ to the ground truth match (inlier) p_2^*. Besides matching a single scale $\mathcal{S} = x$ we also want to consider approaching an pixel position several times on different scales $\mathcal{S} = x\&y\ldots$. This means that the outlier only prevails if it prevails on all approached scales. Furthermore, as comparison we also consider matching several patches of several scales at once $\mathcal{S} = x + y\ldots$ (this is not part of our approach). With the matching error abbreviation

$$E_x^\circ(p_2) = E_d(P_r^x(p_1), P_r^x(p_2)), \tag{6}$$

we can define the following configurations for $\mathcal{S}$:

$$\begin{aligned} C_{\mathcal{S}=x} &= E_x^\circ(p_2) < E_x^\circ(p_2^*) & (7)\\ C_{\mathcal{S}=x\&y\ldots} &= C_x \wedge C_y \ldots & (8)\\ C_{\mathcal{S}=x+y\ldots} &= E_x^\circ(p_2) + E_y^\circ(p_2)\ldots < E_x^\circ(p_2^*) + E_y^\circ(p_2^*)\ldots & (9) \end{aligned}$$

Then, the probability $\mathcal{P}_{\mathcal{S}}(d_f)$ can be written as:

$$\mathcal{P}_{\mathcal{S}}(d_f) = \mathcal{P}(C_{\mathcal{S}} \mid d_f = \|p_2 - p_2^*\|_2). \tag{10}$$

Since the raw probabilities $\mathcal{P}_{\mathcal{S}}(d_f)$ are difficult to read in a plot and as we are mainly interested in the relation between probabilities we plot the relation of probabilities $\mathcal{P}_{\mathcal{S}}^{rel}(d_f) = \frac{\mathcal{P}_{\mathcal{S}}(d_f)}{\mathcal{P}_1(d_f)}$ in Figure 12, instead. $\mathcal{P}_1(d_f)$ is the resistant outlier probability for patches on the finest scale. The values in in the plot are determined from million random pixel positions of the MPI-Sintel dataset, with uniform distribution (for the plot this means uniformly distributed on a circle of radius d_f around the ground truth for distance d_f).

[4] No backward flow calculated

As can be seen in the Figure 12, outliers that are far from the the ground truth match p_2^* (i.e. d_f is large) are less likely to be resistant on a coarser scale (like 4,8), while outliers (inaccurate matches) that are close to the ground truth match p_2^* are less likely to be resistant on a finer scale (like 1,2).

Assuming that outliers, once removed, cannot be reintroduced anymore this allows our multi-scale approach to benefit from the strengths of all scales regarding outlier removal. In fact, not only the strengths. For instance both scale 1 and 2 are clearly inferior to scale 8 for large distances but combined (1&2) they can compete with scale 8. This shows that there is a certain degree of stochastic independence and that even for large distances scale 1 is sometimes superior to scale 2 (although not on average). Without the possibility of outliers being reintroduced the outlier sieving effect would approximate to $S = 1\&2\&4\&8$ (4 way sieve).

However, *random search* can reintroduce resistant outliers (which can just be inaccurate minimums if d_f is small) in its search range. If for instance on scale $S = 2$ the exact inlier position p_2^* is found, this is worth nothing if on scale $S = 1$ there is a resistant outlier within the random search range $Range$ i.e. $d_f < Range \approx \mathcal{R}$, around the inlier position p_2^*, as random search on scale $S = 1$ can vary the position provided by scale 2 in this range.

The black curve in Figure 12 considers this effect by ignoring scales in the & operation where an inlier found by a rougher scale can be undone when there is a resistant outlier on a finer scale within the random search range of the finer scale $Range_n \approx \mathcal{R}_n$ around the inlier. The curve can be seen as a rough approximation of what to expect from Flow Fields.[5] It shows that the sieving effect is very effective for outliers with large endpoint error (d_f), while it can contribute nothing for subpixel optimization. At a distance of $d_f = 200$ pixels the joint resistance outlier rate is only 4.3% of scale 1 and 23.2% of scale 8. As mentioned above the Flow Fields curve is a rough approximation. In the supplementary material we discuss the main approximations made.

We also tested the resistance of matching patches of several scales at once (e.g. 1+2+4+8). While it also decreases the probability of resistant outliers it is computationally expensive and by far not as effective for large distances as our approach. A small benefit is the higher robustness for small distances. This is likely as it can, due to the larger patches of rougher scales, also match in textureless areas.

4.4 Texture effect tests

Due to the small random search distance $\mathcal{R}$ we can expect from our approach that it can flawlessly match repetitive patterns as long as the influence area of the coarsest scale is larger than the ambiguous repetitive pattern. That this is actually the case for our approach is demonstrated in the experiment in Figure 13. While we get a perfect match only with $k >= 3$ the matching error strongly decreases already for fewer scales. We think that this is among other things due to the outlier sieve effect: as corner pixels can be matched they can overwrite matches of non-corner pixels, but not the other way around. In fact this effect is also required for $k = 3$ as the images are 95x95 pixels in size. However, with the used $r = 4$ the matching patch size is only $(2r + 1)2^k = 72$ pixels. The figure also shows that we can expect a similar effect for texture-less objects.

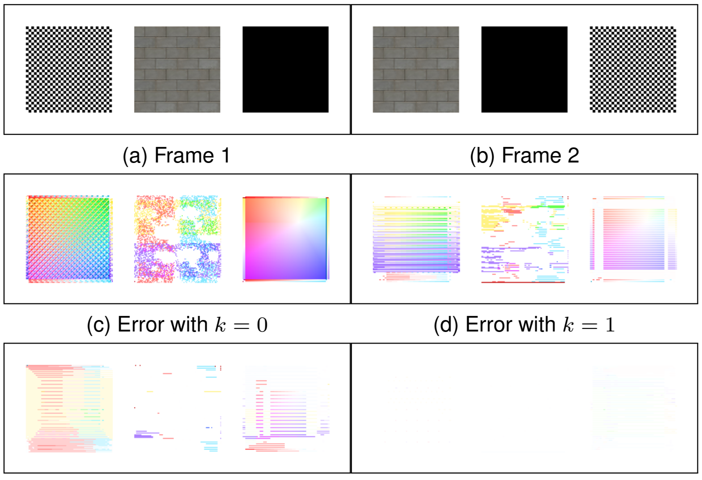

(a) Frame 1
(b) Frame 2
(c) Error with $k = 0$
(d) Error with $k = 1$
(e) Error with $k = 2$
(f) Error with $k = 3$

Fig. 13. Experiments with repetitive patterns and texturelessness. For a moire free illustration one might have to zoom in. Colors show the flow error i.e. white is perfect. Matching gets better with more scales. The horizontal structures in the error maps occur as our approach prefers horizontal propagation in case of identical matching errors.

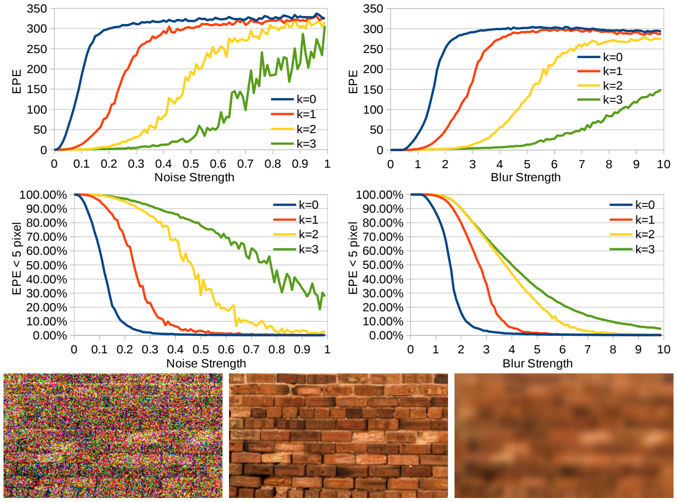


Fig. 14. Our multi-scale approach is noise and blur resistant. The left image in the lower row shows a part of the tested texture with noise factor $\sigma = 0.5$, the unmodified texture and the texture with blur factor $\sigma = 5$. In the tests we match the noisy/blurred texture to the unmodified one. Our approach with $k = 3$ still matches the shown examples well.

In natural images repetitive and texture-less objects are usually not completely ambiguous. Thus, our approach should be able to match them even if the repetitive structure exceeds the influence area of the coarsest scale. Figure 14 shows that our multi-scale approach is also noise and blur resistant. Blur resistance is also confirmed by Figure 16 c).

4.5 Outlier Filtering

Figure 15 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.

[5] In the supplementary material we discuss some of the inaccuracies of the curve. For the point we want to make these are not important.

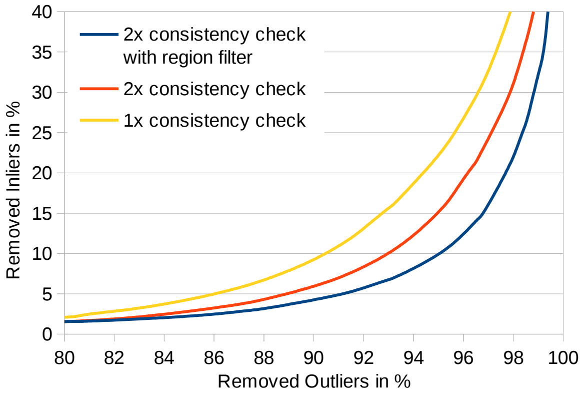


Fig. 15. Percentage of removed outliers versus percentage of removed inliers, for an outlier threshold of 5 pixels (we vary ϵ).

Method	parameter	EPE	time*	time Epic
Flow Fields+	c×2, q=3	2.410	14.0s	3.1s
Flow Fields+ no sub-scales	c×2, q=3	2.438	11.4s	3.1s
Flow Fields+ Fast	c×2, q=3	2.448	6.7s	3.1s
Flow Fields+ Fast	c×1, q=3	2.461	4.5s	3.1s
Flow Fields+ Fast x2	c×2, q=4	2.526	1.8s	1.8s
Flow Fields+ Fast x2	c×1, q=4	2.535	1.2s	1.8s
Flow Fields+ Fast x2	c×1, q=8	2.693	1.2s	1.1s
Original Flow Fields (tuned r)	c×2, q=3	2.574	6.1s	3.1s
Original Flow Fields	c×2, q=3	2.587	14.2s	3.2s

TABLE 4
Accuracy and runtime of our approaches on the MPI-Sintel training set. c×1 or 2 in filter column means 1x or 2x consistency check. *Runtime without EpicFlow.

Method	parameter	>3px	time*	time Epic
Flow Fields+	c×2, F2	21.22%	25.8s	1.8 s
Flow Fields+ no sub-scales	c×2, F2	21.36%	21.1s	1.8 s
Flow Fields+ Fast	c×2, F2	21.82%	10.8s	1.9 s
Flow Fields+ Fast	c×1, F2	21.98%	8.4s	1.9 s
Flow Fields+ Fast	c×1, F2F	21.96%	6.5s	1.9 s
Flow Fields+ Fast x2	c×2, F2F	23.34%	4.0s	1.3s
Flow Fields+ Fast x2	c×1, F2F	23.54%	3.1s	1.2s
Original Flow Fields	c×2, F1	24.74%	39.4s[6]	1.8s

TABLE 5
$>$ 3px EPE failure rate and runtime of our approaches on the KITTI 2015 training set. c×1 or 2 in filter column means 1x or 2x consistency check. *Single core runtime without EpicFlow.

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).

4.6 Evaluation of our approaches

Here we compare the performance and runtime of our different approaches on the MPI-Sintel and KITTI 2015 training sets (on the test sets only the best version of an approach shall be submitted). As can be seen in the first two results in Table 4 and 5, sub-scales improve the matching accuracy, with a reasonable increase in runtime. If speed matters the *Flow Fields+ Fast* approach can provide results much faster, with relatively small accuracy trade-off. *Flow Fields+ Fast x2* is again much faster, while even this approach still outperforms our original conference approach in accuracy (if we do not use q=8). The tables also show that while the 2nd consistency check improves the result, it also requires extra runtime which is why it is not recommendable for our fast approaches. On MPI-Sintel the runtime of EpicFlow exceeds the runtime of our *Flow Fields+ Fast x2* approach. We can decrease it by increasing q. However, this has a clear impact on the accuracy. Tuning r also improves our original approach slightly (we also found $r = 4$ to be the best. Due to two local minima $r = 8$ stays the 2nd best).

Our results on KITTI show that our fast feature approach F2F is comparable to the F2 approach regarding matching accuracy. In the shown test it even performs slightly better which we consider as noise as in another test it performed slightly worse.

4.7 Public Results

In this subsection we present the public results of our approach on different public evaluation portals. We consider our conference approach *Flow Fields* [12], our improved approach *Flow Fields+* and for completeness also our very recent CNN based publication [30] that uses CNN based features as *Flow Fields+CNN*. *Flow Fields+ Fast* is not considered here as the evaluation portals request to submit only the best approach of a publication and to test variations of an approach on the training set. As results in the evaluation portals change regularly we only compare to similar approaches. For a full overview of approaches we refer to the corresponding evaluation portals [8], [10], [11], [13] (links in reference section).

4.7.1 MPI-Sintel

Our results on MPI-Sintel are shown Table 6. Our conference approach *Flow Fields* already clearly outperforms the original EpicFlow that is based on Deep Matching features [18]. Most of this advance is obtained in the non-occluded area but EpicFlow also rewards our better input in the occluded areas. Our improved approach *Flow Fields+* again performs clearly better than our conference approach – especially on the clean set. Here it is at the moment of writing this article the best submission on the non-occluded area with an EPE of only 0.820, while the 2. best recent submission (MR-Flow, yet unpublished) has an EPE of 0.983. Still, our approach is only the 2. best for the overall error (EPE all) as MR-Flow seems to have a better interpolation into the occluded area (for which we still use EpicFlow). With better interpolation on top of our approach it might perform better here, as well. Our approach with CNN-based features [30] performs best on the final set. We think that learned features benefit from the motion blur that is only in the final set while on the clean set there is not such a big improvement possible.

4.7.2 Middlebury

On Middlebury *Flow Fields* obtains an average EPE of 0.33, *Flow Fields+* an average EPE of 0.32 and EpicFlow an average EPE of 0.39. *Flow Fields+* is is as good as or better than

[6] Single core runtime, in conference paper we reported multicore

Method (Final set)	EPE all	EPE nocc.	EPE occ.	d0-10	s40+
Flow Fields+CNN [30]	5.363	2.303	30.313	4.718	32.422
Flow Fields+	5.707	2.684	30.356	4.691	34.167
Flow Fields	5.810	2.621	31.799	4.851	33.890
CPM-Flow [29]	5.960	2.990	30.177	5.038	35.136
EpicFlow [3]	6.285	3.060	32.564	5.205	38.021

Method (Clean set)	EPE all	EPE nocc.	EPE occ.	d0-10	s40+
Flow Fields+	3.102	0.820	21.718	2.340	18.549
CPM-Flow [29]	3.557	1.189	22.889	3.032	21.900
Flow Fields	3.748	1.056	25.700	2.784	23.602
Flow Fields+CNN [30]	3.778	0.996	26.469	2.604	23.582
EpicFlow [3]	4.115	1.360	26.595	3.660	25.859

TABLE 6
Results on MPI-Sintel. (n)occ = (non-)occluded. d0-10 = 0 - 10 pixels from occlusion boundary. s40+ = motions of more than 40 pixels.

Method	>3 pixel nocc.	>3 pixel all	EPE nocc.	EPE all	runtime
Flow Fields+CNN [30]	4.89%	13.01%	1.2 px	3.0 px	23s
Flow Fields+	5.06%	13.14%	1.2 px	3.0 px	28s
Flow Fields	5.77%	14.01%	1.4 px	3.5 px	23s
CPM-Flow [29]	5.79%	13.70%	1.3 px	3.2 px	4.2s
EpicFlow [3]	7.88 %	17.08%	1.5 px	3.8 px	15s

TABLE 7
Results on KITTI 2012 test set. nocc. = Non-occluded.

Method	Fl-bg	Fl-fg	Fl-all	Fl-bg nocc.	Fl-fg nocc.	Fl-all nocc.
Flow Fields+CNN [30]	18.33%	24.96%	19.44%	8.91%	20.78%	11.06%
Flow Fields+	19.51%	25.37%	20.48%	9.69%	21.06%	11.75%
CPM-Flow [29]	22.32%	27.79%	23.23%	12.77%	23.84%	14.78%
EpicFlow [3]	25.81%	33.56%	27.10%	15.00%	29.39%	17.61%

TABLE 8
Results on KITTI 2015 test set. nocc. = Non-occluded. "fg" means only foreground pixels, "bg" only background pixels. Results are <3 pixel.

Flow Fields for all sequences (mostly better). *Flow Fields* again is as good as or better than EpicFlow (also mostly better). As already discussed in Section 4.2 the EPE that can be obtained with EpicFlow on Middlebury is limited, as EpicFlow is not designed for such datasets. Nevertheless, we can strongly improve the result on some datasets. Most improvement is obtained on the urban dataset where *Flow Fields+* obtains the 4th best result while EpicFlow obtains the 63th best.

4.7.3 KITTI 2012 and 2015

Our results on KITTI 2012 and 2015 can be seen in Table 7 and 8, respectively. As can be seen, our conference approach *Flow Fields* already clearly outperforms the original EpicFlow with Deep Matching features on KITTI 2012. Our improved approach *Flow Fields+* performs even better. To the best of our knowledge our *Flow Fields+* approach is so far the best approach both on KITTI 2012 and 2015 that does not use CNNs like [30], [36] or object segmentation and rigidity assumptions for the segmented objects like [37], [38], [39]. Thus, in contrast to all better performing approaches ours also works for non-rigid scenes or scenes where object segmentation fails and does not require to train a neural network, for which proper training data is required. Our CNN-based approach [30] performs even better, but does require proper training data.

4.8 Visual Results

Visual results of our approach are shown in Figure 16. EpicFlow can preserve considerably more details with our Flow Fields approach than with the original Deep Matching features. With Flow Fields+ even more details are preserved that are not or worse preserved with the original Flow Fields (e.g. bottom of elbow in image 2 and neck in image 4). Even in failure cases like in Figure 16 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 16 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 16 c) shows that our approach also performs well in the presence of motion and defocus blur.

5 Conclusion

In this article 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 techniques, regarding optical flow estimation. We also presented extended outlier filtering and demonstrated that we can obtain promising optical flow results, utilizing a modern optical flow algorithm like EpicFlow. Compared to the conference version we further improved our approach both in accuracy and runtime efficiency. We also gave a deeper insight into our approach. With our results, we hope to inspire the research of dense correspondence field estimation for optical flow.

Acknowledgments

This work was partially funded by the BMBF projects DYNAMICS and VIDETE.

References

[1] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical Symposium East*. International Society for Optics and Photonics, 1981, pp. 319–331. 1, 2, 4

[2] A. Wedel, D. Cremers, T. Pock, and H. Bischof, "Structure-and motion-adaptive regularization for high accuracy optic flow," in *ICCV*. IEEE, 2009, pp. 1663–1668. 1

[3] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," in *CVPR*, 2015. 1, 2, 7, 8, 12, 13

[4] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow with nearest neighbor fields," in *CVPR*. IEEE, 2013, pp. 2443–2450. 1, 2, 7

[5] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *ECCV*. Springer, 2004, pp. 25–36. 1

[6] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," in *CVPR*. IEEE, 2012, pp. 111–118. 1, 2, 3, 4, 7, 8, 9, 13

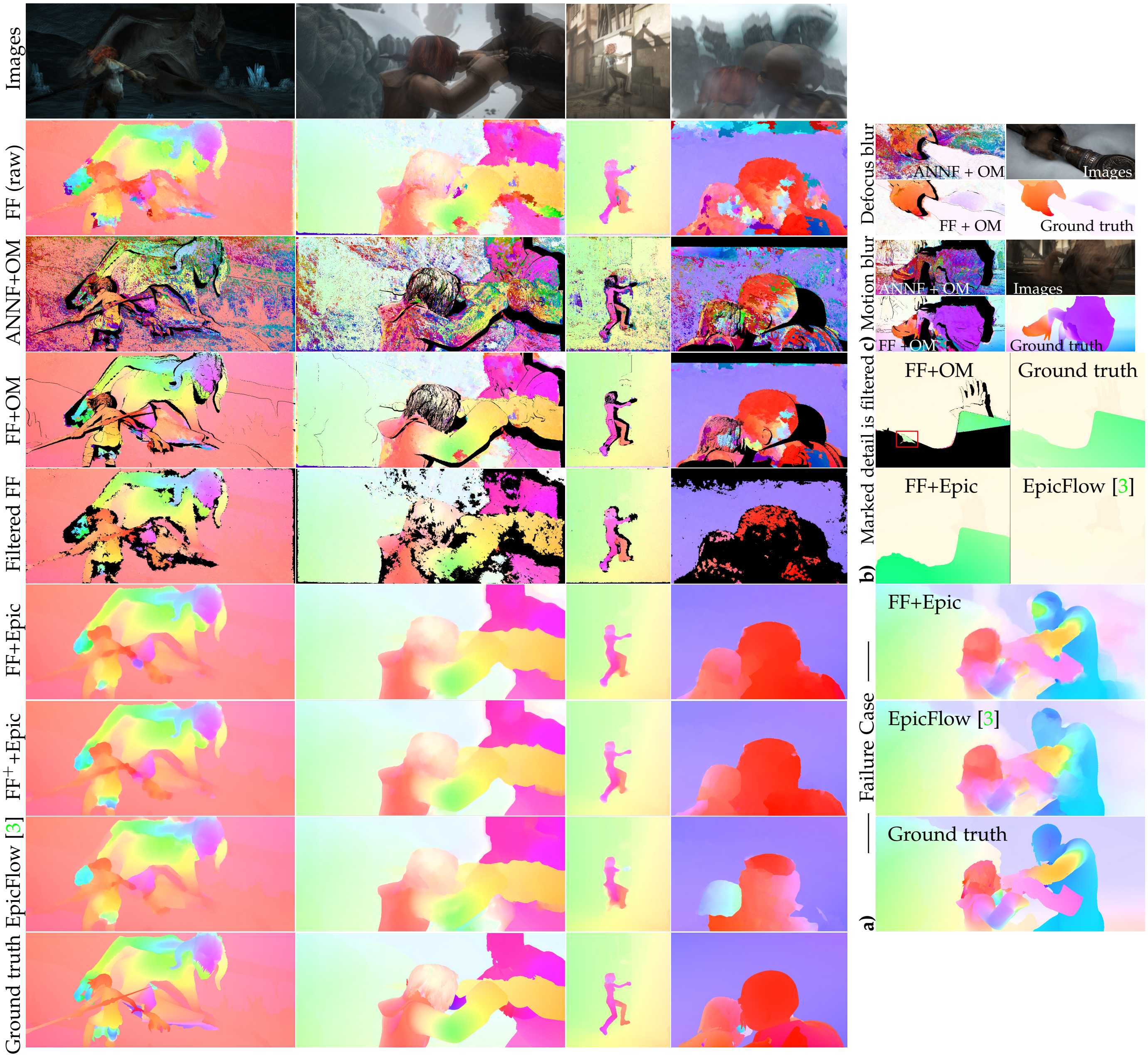


Fig. 16. Results on the MPI-Sintel Final set. The left 4 columns show example results. *Images* is the average of both input images. For ANNF we use [6] in a fair way (see text). *FF* means Flow Fields, *FF*$^+$ means Flow Fields+ . *OM* means that the ground truth occlusion map is added (black pixels, it is incomplete at image boundaries). This is done as data based matching is only possible in non occluded areas. *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 the presence of blur than ANNF (top left).

[7] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004. 1, 6

[8] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, “A naturalistic open source movie for optical flow evaluation,” in *ECCV*. Springer, 2012, pp. 611–625. [Online]. Available: http://sintel.is.tue.mpg.de/results 2, 7, 11

[9] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, “The generalized patchmatch correspondence algorithm,” in *ECCV*. Springer, 2010, pp. 29–43. 2, 3

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, 2013. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=flow 2, 7, 11

[11] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=flow 2, 7, 11

[12] C. Bailer, B. Taetz, and D. Stricker, “Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation,” in *ICCV*. IEEE, 2015. 2, 3, 11

[13] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, “A database and evaluation methodology for optical flow,” *IJCV*, vol. 92, no. 1, pp. 1–31, 2011. [Online]. Available: http://vision.middlebury.edu/flow/eval/results/results-e1.php 2, 7, 11

[14] D. Sun, S. Roth, and M. J. Black, “Secrets of optical flow estimation and their principles,” in *CVPR*. IEEE, 2010, pp. 2432–2439. 2

[15] C. Vogel, S. Roth, and K. Schindler, “An evaluation of data costs for optical flow,” in *Pattern Recognition (GCPR)*. Springer, 2013,

pp. 343–353. 2

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

[17] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *PAMI*, vol. 34, no. 9, pp. 1744–1757, 2012. 2

[18] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *ICCV*, 2013. [Online]. Available: http://hal.inria.fr/hal-00873592 2, 8, 11

[19] 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*. Springer, 2015, pp. 364–377. 2

[20] R. Timofte and L. Van Gool, "Sparse flow: Sparse matching for small to large displacement optical flow," in *Applications of Computer Vision (WACV)*. IEEE, 2015, pp. 1100–1106. 2

[21] 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)*. IEEE, 2014, pp. 673–674. 2

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

[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*. IEEE, 2013, pp. 1854–1861. 2

[24] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving patchmatch for large displacement optical flow," in *CVPR*. IEEE, 2014, pp. 3534–3541. 2, 9

[25] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM transactions on graphics (TOG)*, vol. 30, no. 4, p. 70, 2011. 2

[26] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *ICCV*. IEEE, 2007, pp. 1–8. 2

[27] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *PAMI*, vol. 32, no. 8, pp. 1362–1376, 2010. 2

[28] C. Bailer, M. Finckh, and H. P. Lensch, "Scale robust multi view stereo," in *ECCV*. Springer, 2012, pp. 398–411. 2

[29] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine patchmatch for large displacement optical flow," in *CVPR*. IEEE, 2016. 2, 12

[30] C. Bailer, K. Varanasi, and D. Stricker, "Cnn-based patch matching for optical flow with thresholded hinge loss," *arXiv preprint arXiv:1607.08064*, 2016. 2, 6, 11, 12

[31] Y. Hel-Or and H. Hel-Or, "Real-time pattern matching using projection kernels," *PAMI*, vol. 27, no. 9, pp. 1430–1445, 2005. 3

[32] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994. 4

[33] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *ECCV*. Springer, 1994, pp. 151–158. 6

[34] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *PAMI*, vol. 33, no. 5, pp. 978–994, 2011. 6

[35] C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of Applied Meteorology*, vol. 18, no. 8, pp. 1016–1022, 1979. 6

[36] D. Gadot and L. Wolf, "Patchbatch: a batch augmented loss for optical flow," in *Computer Vision and Pattern Recognition (CVPR)*, 2016. 12

[37] 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. 12

[38] J. Hur and S. Roth, "Joint optical flow and temporally consistent semantic segmentation," in *European Conference on Computer Vision (ECCV)*, 2016. 12

[39] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black, "Optical flow with semantic segmentation and localized layers," in *Computer Vision and Pattern Recognition (CVPR)*, 2016. 12

Christian Bailer received his diploma (equivalent to M.Sc.) in computer science from Ulm University, in 2012. Currently he is pursuing a PhD at the German Research Center of Artificial Intelligence in Kaiserslautern, Germany in the area of motion estimation. His current research interests are motion estimation and machine learning for computer vision.

Bertram Taetz received his B.Sc. and M.Sc. in applied mathematics, minor subject physics, from Ruhr University Bochum, Germany, in 2009. During his PhD, at the Ruhr University, he focused on numerical methods for dynamical systems. He finished his PhD in 2012. Since 2013 he works as senior researcher at the German Research Center for Artificial Intelligence and also joined the department of computer science at the University of Kaiserslautern in 2015. His research interests are numerical and statistical methods for motion estimation and dynamical systems.

Didier Stricker is professor at the university of Kaiserslautern and scientific director at the "German Research Center for Artificial Intelligence" (DFKI) in Kaiserslautern where he leads the research department Augmented Vision. From 2002 to June 2008 Didier Stricker lead the department "Virtual and Augmented Reality" at the Fraunhofer Institute for Computer Graphics (Fraunhofer IGD) in Darmstadt, Germany. In this function, he initiated and participated to many national and international projects in the areas of computer vision and virtual and augmented reality. In 2006, he received the Innovation Prize of the German Society of Computer Science. Didier Stricker serves as reviewer for different European or national research organizations, and is a regular reviewer for the most important journals and conferences in the areas of VR/AR and computer vision.