SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences

René Schuster[1]   Oliver Wasenmüller[1]   Georg Kuschk[2]   Christian Bailer[1]   Didier Stricker[1]
[1]DFKI - German Research Center for Artificial Intelligence   [2]BMW Group

`firstname.lastname@{dfki,bmw}.de`

Abstract

While most scene flow methods use either variational optimization or a strong rigid motion assumption, we show for the first time that scene flow can also be estimated by dense interpolation of sparse matches. To this end, we find sparse matches across two stereo image pairs that are detected without any prior regularization and perform dense interpolation preserving geometric and motion boundaries by using edge information. A few iterations of variational energy minimization are performed to refine our results, which are thoroughly evaluated on the KITTI benchmark and additionally compared to state-of-the-art on MPI Sintel. For application in an automotive context, we further show that an optional ego-motion model helps to boost performance and blends smoothly into our approach to produce a segmentation of the scene into static and dynamic parts.

1. Introduction

Scene flow describes the perceived 3D motion field with respect to the observer. It can thereby be considered as an extension to optical flow, which in comparison is the apparent motion field in 2D image space. Many applications such as robot navigation, high level vision tasks, *e.g.* moving object detection [25], and driver assistance systems rely on an accurate motion estimation of their surroundings. Especially the latter ones have great potential to make traffic more comfortable and much safer. While scene flow is not only a more detailed representation of the real world motion compared to optical flow, scene flow algorithms also reconstruct the 3D geometry of the environment. Due to the increased complexity when compared to depth or optical flow estimation, scene flow has only recently become of bigger interest. At the same time, simply estimating depth and optical flow separately to obtain scene flow (see [19, 28, 29, 31]) is not exploiting the full potential of the underlying data. The splitting produces incoherent results, limits the exploitation of inherent redundancies, and in general yields a non-dense scene flow field. Approaches that combine stereo depth estimation, *e.g.* [15], and 2D optical
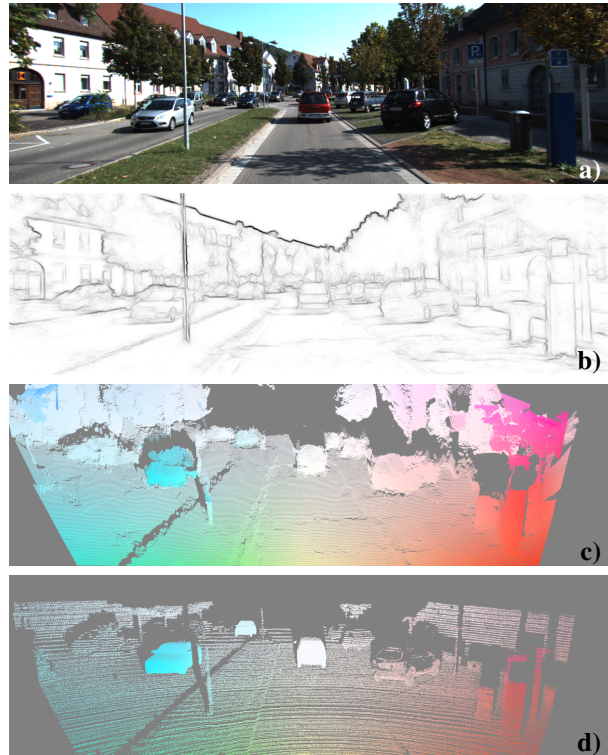

Figure 1. We present SceneFlowFields which uses stereo image pairs **(a)**, extracts scene flow boundaries **(b)**, and computes a dense scene flow field **(c)** that we compare to ground truth **(d)**. The color of the point clouds encodes the optical flow.

flow, *e.g.* [4, 30], have clearly been outperformed by a huge margin in the KITTI benchmark [10].

Most state-of-the-art approaches are either designed for indoor scenarios or describe outdoor scenes under mostly stationary or rigid motion assumptions. Contrary to that, our method is very versatile. In fact, we do not employ any a-priori regularization. Our matching process inherently encodes a first order local smoothness assumption and is itself solely data term based, while our interpolation scheme allows for very sharp discontinuities in the scene flow field. An optional ego-motion extension that incorporates additional assumptions improves accuracy for challenging traf-
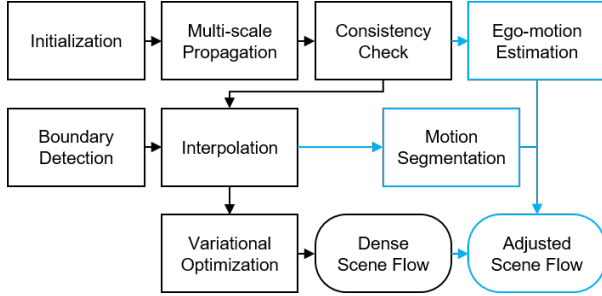
Figure 2. Overview of our SceneFlowFields. Blue color indicates the optional ego-motion extension.

fic data sets like KITTI, but is no mandatory part of our method. The basic version of the method and its optional extension are illustrated in Figure 2. The single stages of our approach are visualized in the supplementary video[1].

In more detail, we present a new scene flow approach – called SceneFlowFields – that densely interpolates sparse scene flow matches. The interpolation preserves boundaries of the 3D geometry and of moving objects by edge-preserving interpolation based on an improved edge detector to approximate scene flow boundaries. Matches are obtained by multi-scale propagation with random search to compute a dense scene flow field that in turn is filtered to remove outliers and leave sparse, robust correspondences across all images. The combination of multi-scale matching and edge-preserving interpolation sums to a novel method which is in strong contrast with any existing method that estimates scene flow. Matching can optionally be used to estimate ego-motion and obtain sparse motion indicators which can be interpolated to a dense motion segmentation. Optionally using ego-motion, we can reconstruct the scene flow for static parts of the scene directly. However, our method does not rely on ego-motion estimation. In particular, our contribution consists of:

- A novel method to find scene flow matches.
- A new interpolation method for scene flow that preserves boundaries of geometry and motion.
- An improved edge detector to approximate scene flow boundaries.
- An optional approach for straightforward integration of ego-motion.
- Thorough evaluation of our method on KITTI and MPI Sintel with comparison to state-of-the-art methods.

2. Related Work

Starting from Vedula *et al*. [32] who was among the first to compute 3D scene flow, many variational approaches followed. First using pure color images as input [2, 17] and later using RGB-D images [13, 18, 37]. While a varia-

tional formulation is typically complex, [18] achieved real-time performance with a primal-dual framework. Yet, all these approaches are sensitive to initialization and can not cope with large displacements [36], which is why they use a coarse-to-fine scheme. That in turn tends to miss finer details. Furthermore, the RGB-D approaches rely on depth sensors that either perform poorly in outdoor scenarios or are accordingly very expensive [39].

Since it is hard to capture ground truth scene flow information, there exist only very few data sets to evaluate scene flow algorithms on. Most of them use virtually rendered scenes to obtain the ground truth data [5, 9, 22]. To the best of our knowledge the only realistic data set that provides a benchmark for scene flow is the KITTI Vision Benchmark [10] that combines various tasks for automotive vision. Its introduction has played an important role in the development of stereo and optical flow algorithms, and the extension by [23] has also driven the progress in scene flow estimation.

Due to the advent of a piece-wise rigid plane model [33], scene flow has recently achieved a boost in performance. The majority of top performing methods at the KITTI Vision Benchmark employ this model to enforce strong regularization [21, 23, 34]. In [34] the authors encode this model by alternating assignment of each pixel to a plane segment and each segment to a rigid motion, based on a discrete set of planes and motions. In [23] the complexity of the model is further lowered by the assumption that a scene consists of only very few independently moving rigid objects. Thus each plane segment only needs to be assigned to one object. All segments assigned to the same object share the same motion. By propagation of objects over multiple frames, [24] achieves temporal consistency for [23]. The authors of [21] solve the pixel-to-plane assignment and the plane-to-motion assignment in a continuous domain.

Another promising strategy builds on the decomposition of a scene into static and moving parts [31]. While the motion of dynamic objects is estimated by solving a discrete labeling problem (as in [6]) using the Semi-Global-Matching (SGM) [15] algorithm, the perceived motion of all static parts is directly obtained from the 3D geometry of the scene and the ego-motion of the camera. This approach is especially convenient for scenes, where only a small proportion consists of moving objects, like it is typically the case in traffic scenarios. However, any a-priori assumption limits the versatility of a method. A rigid plane model performs poorly when applied to deformable objects, and ego-motion estimation for highly dynamic scenes is hard.

Our scene flow method differs from any of the mentioned approaches. We find sparse scene flow matches that are interpolated to a dense scene flow field, recovering the geometry of the scene and the 3D motion. Our method has to be distinguished from purely variational approaches. Although

---

[1]https://youtu.be/7fsXq5EA0Rw

we use variational optimization, it can be considered as a post-processing step for refinement. During interpolation, we assume that the geometry of a scene can be modeled by small planar segments, but we do not initially presume any segmentation. In fact, the size of our plane segments only depends on the density of the matches, which leads to smoothly curved shapes where matches are dense and to planar patches where matches are sparse. The same holds for our piece-wise affine motion model that is used to interpolate the 3D motion. These differences in the model and additionally the difference in the optimization method draw a clear boundary between our method and [21, 23, 34]. If we apply our optional ego-motion model there is a conceptual overlap to [31] which also uses ego-motion to estimate the 3D motion for the static parts of the scene. However, if we do not apply the ego-motion model, both methods have no noteworthy similarities. In any case, the way we estimate the ego-motion and the motion segmentation differs essentially from [31].

Finally, one has to differentiate between dual-frame [21, 23, 34] and multi-frame [24, 31, 34] approaches. Especially the images of KITTI have several characteristics that make matching between two frame pairs much more challenging than in a multi-view setting. First, considerably large stereo and flow displacements. Second, difficult lighting conditions and many reflective surfaces. Third, fast ego-motions combined with a low frame rate, which causes large regions to move out of image bounds. This has to be kept in mind when comparing results across these two categories.

3. SceneFlowFields

For scene flow computation we assume to have the typical stereo image information provided, *i.e.* two rectified stereo image pairs $(I_l^0, I_l^1, I_r^0, I_r^1)$ at times t_0 and t_1 along with the camera intrinsics. We further assume that the baseline B is known. For rectified images, the baseline describes the relative pose between the left and right cameras as translation parallel to the image plane. We represent scene flow as a 4D vector $\mathbf{u} = (u, v, d_0, d_1)^T$ consisting of two optical flow components u, v and the disparity values d_0, d_1 for both time steps. During matching, we jointly optimize all four components to obtain coherent scene flow. Given the mentioned information, we estimate a dense scene flow field as follows: For k subscales we initialize the coarsest by finding the best correspondences from kD-trees build with feature vectors using Walsh-Hadamard-Transform (WHT) [12]. For all $k + 1$ scales (the k subscales plus full resolution), we iteratively propagate scene flow vectors and adjust them by random search. Afterwards, the dense scene flow map on full resolution is filtered using an inverse scene flow field and a region filter. The filtered scene flow map is further thinned out by only taking the best match in each non-

overlapping 3×3 block. Scene flow boundaries are detected using a structured random forest. Geometry and 3D motion are separately interpolated based on a boundary-aware neighborhood. Finally, we refine the 3D motion by variational optimization. An overview is outlined in Figure 2.

3.1. Sparse Correspondences

Matching Cost. The matching cost in our algorithm solely depends on a data term. No additional smoothness assumptions are made like *e.g.* in [13, 17, 21, 23, 33, 34]. Given a scene flow vector, we define its matching cost by the sum of Euclidean distances between SIFTFlow features [20] over small patches for three image correspondences. These correspondences are the stereo image pair at time t_0, the temporal image pair for the left view point (standard optical flow correspondence) and a cross correspondence between the reference frame and the right frame at the next time step. This leads to the following cost C for a scene flow vector $\mathbf{u}$ at pixel p:

$$
\begin{aligned}
C = \sum_{\tilde{p} \in W(p)} & \left\| \phi\left(I_l^0\left(\tilde{p}\right)\right) - \phi\left(I_l^1\left(\tilde{p} + (u,v)^T\right)\right) \right\| + \\
& \left\| \phi\left(I_l^0\left(\tilde{p}\right)\right) - \phi\left(I_r^0\left(\tilde{p} + (-d_0, 0)^T\right)\right) \right\| + \\
& \left\| \phi\left(I_l^0\left(\tilde{p}\right)\right) - \phi\left(I_r^1\left(\tilde{p} + (u - d_1, v)^T\right)\right) \right\|
\end{aligned}
\tag{1}
$$

$W(p)$ is a 7×7 patch window centered at pixel p and $\phi(I(p))$ returning the first three principal components of a SIFT feature vector for image I and pixel p. The principal axes are computed for the combined SIFT features of all four images. At image boundaries we replicate the boundary pixel to pad images.

Initialization. Initialization is based on kD-trees similar to [11], but with three trees, using WHT features as in [1, 35]. For each frame other than the reference frame, we compute a feature vector per pixel and store them in a tree. To initialize a pixel of the reference image, we query the feature vector of the pixel to the pre-computed kD-trees. Scene flow matches are then obtained by comparing all combinations of the leafs for each queried node according to the matching data term introduced before (Equation 1). Since our stereo image pairs are rectified, for the images observed from the right camera view, we create kD-trees which regard the epipolar constraint, *i.e.* queries for such a tree will only return elements, which lie on the same image row as the query pixel. This way, we can efficiently lower the number of leaves per node for the epipolar trees, which speeds up the initialization process without loss of accuracy. For further acceleration, we use this initialization on the coarsest resolution only, and let the propagation fill the gaps when evolving to the next higher scale.

Multi-Scale Propagation. The initial matches get spread by propagation and steadily refined by random search. This

is done over multiple scales which helps to distribute rare correct initial matches over the whole image. For each scale, we run several iterations of propagation in one out of the four image quadrants so that each direction is used equally. During propagation, a scene flow vector is replaced if the propagated vector has a smaller matching cost. If this is not the case, the propagation along this path continues with the existing scene flow vector. After each iteration we perform a random search. That means that for all pixels we add a uniformly distributed random offset in the interval $]-1, 1[$ in pixel units of the current scale to each of the four scene flow components and check whether the matching cost decreases. Both propagation and random search help to obtain a smoothly varying vector field and to find correct matches even if the initialization is slightly flawed. For the different scale spaces, we simulate the scaling by smoothing the images and taking only every n-th pixel for a subsampling factor of $n = 2^k$ so that the patches consist of the same number of pixels for all scales. This way, we prevent (up)sampling errors because all operations are performed on exact pixel locations on the full image resolution. Smoothing is done by area-based downsampling followed by upsampling using Lanczos interpolation. Note that this matching method has already been used in [1], but while they use it for optical flow, we apply it to twice as many dimensions in search space.

Consistency Check. The matching procedure yields a dense map of scene flow correspondences across all images. However, many of the correspondences are wrong because of occlusions, out-of-bounds motion or simply because of mismatching due to challenging image conditions. To remove these outliers, we perform a two-step consistency check. First, we compute an inverse scene flow field for which the reference image is the right image at time t_1. Temporal order as well as points of view are swapped. Everything else remains as explained above. During consistency check, optical flow and both disparity maps for each pixel are compared to the corresponding values of the inverse scene flow field. If either difference exceeds a consistency threshold τ_c in image space, the scene flow vector gets removed. Secondly, we form small regions of the remaining pixels as in [1], where a pixel is added to a region if it has approximately the same scene flow vector. Afterwards, we check if we could add one of the already removed outliers in the neighborhood following the same rule. If this is possible and the region is smaller than s_c pixels, we remove the whole region. This way we obtain the filtered final scene flow correspondences of high accuracy and very few outliers (cf. Table 2). Because most of the times the joint filtering of the matches removes more disparity values than necessary, we fill up gaps with additional values. These values are the result of a separate consistency check for the disparity matches only. For the separate check we
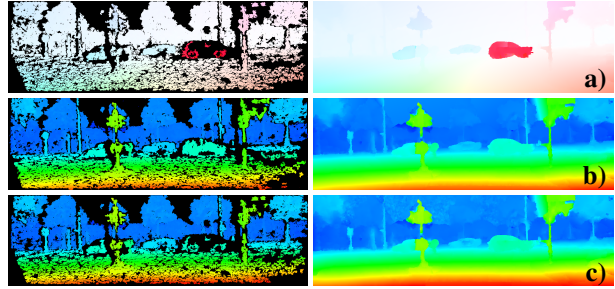

Figure 3. Sparse correspondences (left) and dense interpolation (right). Optical flow **(a)** and disparities at t_0 **(b)** and t_1 **(c)**.

compute a second disparity map with SGM [15] and use the same threshold τ_c as before. The additional disparity values that are retrieved this way are as accurate as the one from our standard consistency check but much denser which is shown in Figure 3 and Table 2.

3.2. Dense Interpolation

Sparsification. Before interpolating the filtered scene flow field to recover full density, an additional sparsification step is performed. This helps to extend the spatial support of the neighborhood during interpolation and speeds up the whole process [1]. For non-overlapping 3×3 blocks, we select the match with the lowest consistency error during filtering only. The remaining matches are called seeds with respect to the interpolation.

Interpolation Boundaries. A crucial part of the interpolation is the estimation of scene flow boundaries. While [1, 26] approximate motion boundaries for optical flow with a texture-agnostic edge detector [8], our edge detector is trained on semantic boundaries. We find that this models geometric boundaries as well as motion boundaries much better than image edges and is much more robust to lighting, shadows, and coarse textures. To do so, we have gathered about 400 images of the KITTI data set from [14, 27, 38] that have been labeled with semantic class information. Within these images we have merged semantic classes that in general neither align with geometric nor motion discontinuities, *e.g.* lane markings and road, or pole and panel. The boundaries between the remaining semantic labels are used as binary edge maps to train our edge detector. To this end, we utilize the framework of Structured Edge Detection (SED) [8] and train a random forest with the same parameters as in their paper, except for the number of training patches. We sample twice as many positive and negative patches during training because we use a bigger data set with images of higher resolution. The impact of the novel boundary detector will be evaluated in Section 5.

Interpolation Models. For the interpolation of geometry and motion, we use two different models. Both parts are interpolated separately which leads to a more accurate re-

Figure 4. Whereas SED [8] (**b**) detects all image boundaries, our new boundary detector (**c**) suppresses lane markings and shadows.

construction of the scene. This is due to the fact that the separate consistency check for disparity leaves more geometric matches where motion would leave image boundaries. Suppose a local, boundary-aware neighborhood of seeds is given for each unknown scene flow vector $\hat{\mathbf{u}}$ at pixel $\hat{p}$ for geometric and motion seeds respectively, $\mathcal{N}_{geo}$ and $\mathcal{N}_{motion}$. The depth of pixel $\hat{p}$ is reconstructed by fitting a plane $E(\hat{p}): a_1x + a_2y + a_3 = d_0$ through all seeds of the neighborhood $\mathcal{N}_{geo}$. This is done by solving a linear system of equations for all neighboring seed points p_g for which the disparity values are known, using weighted least squares. The weights for each seed are obtained from a Gaussian kernel $g(D) = \exp(-\alpha D)$ on the distance $D(\hat{p}, p_g)$ between target pixel and seed. The missing disparity value of $\hat{p}$ is obtained by plugging the coordinates of $\hat{p}$ into the estimated plane equation. In a similar fashion, but using a neighborhood of motion seeds $\mathcal{N}_{motion}$, the missing 3D motion is obtained by fitting an affine 3D transformation $\mathbf{x_1} = \mathbf{A}\mathbf{x_0} + \mathbf{t}$ using weighted least squares on all motion seeds p_m. Where $\mathbf{x_t} = (x_t, y_t, z_t)^T$ are the 3D world coordinates of motion seed p_m at time t_0 and t_1, and $[\mathbf{A}|\mathbf{t}] \in \mathbb{R}^{3\times 4}$ is the affine 3D transformation of twelve unknowns. The weights are computed by the same Gaussian kernel as for geometric interpolation, but using the distances $D(\hat{p}, p_m)$ between the target pixel and the motion seeds. To summarize, for the full reconstruction of scene flow $\hat{\mathbf{u}} = (u, v, d_0, d_1)^T$ at pixel $\hat{p}$, we compute d_0 using the plane model $E(\hat{p})$, reproject the point into 3D world space, transform it according to its associated affine transformation $[\mathbf{A}|\mathbf{t}]$, and project it back to image space to obtain u, v and d_1.

Edge-Aware Neighborhood. To find the local neighborhoods, we follow the idea of [26] using both their approximations. That is first, the n closest seeds to a pixel $\hat{p}$ are the $n-1$ closest seeds to the closest seed of $\hat{p}$, thus all pixels

with the same closest seed share the same local neighborhood. And secondly, the distance between $\hat{p}$ and its closest seed is a constant offset for all neighboring seeds which can be neglected. It is therefore sufficient to find a labeling that assigns each pixel to its closest seed and to find the local neighborhood for each seed. We use the graph-based method of [26] for this. Where the distances between seeds are geodesic distances that are directly based on the edge maps from our boundary detector.

3.3. Variational Optimization

To further refine the 3D motion after interpolation, we use variational energy minimization to optimize the objective

$$E(u, v, d') = E_{data}^{flow} + E_{data}^{cross} + \varphi \cdot E_{smooth} \quad (2)$$

Motion is represented in image space by optical flow and the change in disparity d'. The energy consists of three parts. Two data terms, one temporal correspondence and one cross correspondence, and an adaptively weighted smoothness term for regularization. The data terms use the gradient constancy assumption. Our experiments have shown, that a term for the color constancy assumption can be neglected.

$$E_{data}^*(I_1, I_2, \mathbf{x}, \mathbf{w}) =$$
$$\int_\Omega \beta(\mathbf{x}, \mathbf{w}) \cdot \Psi\left(\gamma \cdot |\nabla I_2(\mathbf{x} + \mathbf{w}) - \nabla I_1(\mathbf{x})|^2\right) dx \quad (3)$$

$$E_{data}^{flow} = E_{data}^*\left(I_l^0, I_l^1, \mathbf{x}, (u, v)^T\right) \quad (4)$$

$$E_{data}^{cross} = E_{data}^*\left(I_l^0, I_r^1, \mathbf{x}, (u - d_0 - d', v)^T\right) \quad (5)$$

The data terms do not contribute to the energy if the function

$$\beta(\mathbf{x}, \mathbf{w}) = \begin{cases} 1, & \text{if } (\mathbf{x} + \mathbf{w})^T \in \Omega \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

indicates that the scene flow is leaving the image domain. The smoothness term

$$E_{smooth} = \int_\Omega \Psi\left(|\nabla u|^2 + |\nabla v|^2 + \lambda \cdot |\nabla d'|^2\right) dx \quad (7)$$

penalizes changes in the motion field and is weighted by

$$\varphi(\mathbf{x}) = e^{-\kappa B(\mathbf{x})} \quad (8)$$

where $B(\mathbf{x})$ is the edge value of our boundary detector at pixel $\mathbf{x}$. All parts use the Charbonnier penalty $\Psi\left(x^2\right) = \sqrt{x^2 + \varepsilon^2}$ to achieve robustness. Since the smoothness term rather enforces constancy if β for both data terms is zero, we do not optimize the scene flow at pixels where the interpolated scene flow field leaves Ω. Our energy formulation is inspired by [3, 17, 37]. We use linear approximations of the Euler-Lagrange equations for the objective and apply the framework of Brox *et al.* [3] without the coarse-to-fine steps to find a solution by Successive Over-Relaxation (SOR).
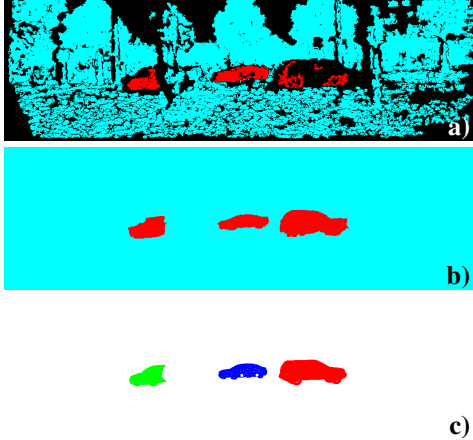
Figure 5. Example of our motion segmentation. Sparse motion indicators as obtained during ego-motion computation **(a)**, dense segmentation by interpolation **(b)** and moving ground truth objects as provided by KITTI [23] **(c)**.

4. Ego-Motion Model

In Section 5 we will show that our approach as described so far achieves results comparable to state-of-the-art. For the special challenges of the KITTI data set, we make an additional, optional assumption to further improve the performance of SceneFlowFields. Following [31], we argue that most parts of a scene are static and thus that the 3D motion for these areas is fully determined by the ego-motion of the observer. Given the ego-motion and a motion segmentation into static and dynamic areas, we apply the inverse ego-motion to all static points in the scene. Using our matching and interpolation scheme, both can be easily estimated with almost no additional effort.

Ego-Motion Estimation. The filtered scene flow field before interpolation provides very accurate matches across all images. We compute 3D-2D correspondences between the reference frame and the temporally subsequent frame by triangulation with the stereo matches. We limit the depth of these correspondences to 35 meters because disparity resolution for farther distances gets too inaccurate. This way, we obtain a Perspective-n-Point problem, which we solve iteratively using Levenberg-Marquardt and RANSAC to find the relative pose between the left cameras at time t_0 and t_1 by minimizing the re-projection error of all correspondences. For RANSAC, we consider a correspondence an outlier if the re-projection error is above 1 pixel. After first estimation, we recompute the set of inliers with a relaxed threshold of 3 pixels and re-estimate the pose $P = [R|t] \in \mathbb{R}^{3 \times 4}$. The two stage process helps to avoid local optima and to find a trade-off between diverse and robust correspondences.

Motion Segmentation. An initial sparse motion segmentation can directly be obtained as side product of the ego-motion estimation. Outliers in the correspondences are considered in motion, while points in conformity with the estimated ego-motion are marked as static. We use our boundary-aware interpolation to compute a dense segmentation (cf. Figure 5). Pixels labeled as moving are spread up to the object boundaries within they are detected. Because the segmentation is only a binary labeling, no complex interpolation model is needed. An unknown pixel gets assigned with the weighted mean of its local neighborhood. The weights are again based on the geodesic distances between matches. This interpolation method is similar to the Nadaraya-Watson estimator in [26]. The interpolated motion field is then thresholded to obtain a dense, binary motion segmentation. The quality of this segmentation is evaluated in Section 5. Finally, the inverse estimated ego-motion is applied to all points that are labeled as static.

5. Experiments and Results

We use the explicit values of the previous sections and the following parameters for all our experiments, even across different data sets. For $k = 3$ subscales and full resolution we run 12 iterations of propagation and random search. We use a consistency threshold of $\tau_c = 1$ and a minimal region size of $s_c = 150$ for the region filter. During interpolation we use geometry and motion neighborhoods of 160 and 80 seeds respectively and $\alpha = 2.2$ for the Gaussian kernel to weight the geodesic distances. For the variational energy minimization we set $\kappa = 5$, $\gamma = 0.77$, $\lambda = 10$ and $\varepsilon = 0.001$. We run two outer and one inner iteration in our optimization framework with 30 iterations for the SOR solver using a relaxation factor of $\omega = 1.9$. We threshold the interpolated motion field with $\tau_S = 0.4$ to obtain a binary segmentation when applying the ego-motion model.

5.1. Boundary Detection

To test the impact of our motion boundary detector, we evaluate different variants of our method twice. Once, using standard edge detection as in [8], and a second time using our structured random forest trained on semantic edges. The results are compared in Table 2. The major improvements are visualized in Figure 4. High image gradients at lane markings or shadows (especially shadows of vehicles) are effectively suppressed when using our boundary detector, while at the same time it accurately detects all kinds of objects. This helps greatly to smoothly recover the street surface during interpolation, sharpen discontinuities in depth and motion in general, and allows for accurate boundaries when interpolating the motion segmentation.

5.2. KITTI Scene Flow Benchmark

Our main experiments are taken out on the KITTI Scene Flow Benchmark [23]. The results of our public submission are presented in Table 1 where we compare to state-of-the-art methods. At the time of writing, our method

Rank	Method	dual	D1-bg	D1-fg	D1-all	D2-bg	D2-fg	D2-all	Fl-bg	Fl-fg	Fl-all	SF-bg	SF-fg	SF-all	Runtime
1	PRSM [34]		**3.02**	10.52	**4.27**	**5.13**	**15.11**	**6.79**	**5.33**	13.40	**6.68**	**6.61**	20.79	**8.97**	300 s
2	OSF+TC [24]		4.11	**9.64**	5.03	5.18	15.12	6.84	5.76	**13.31**	7.02	7.08	**20.03**	9.23	3000 s
3	OSF [23]	x	4.54	12.03	5.79	5.45	19.41	7.77	5.62	18.92	7.83	7.01	26.34	10.23	3000 s
4	FSF+MS [31]		5.72	11.84	6.74	7.57	21.28	9.85	8.48	25.43	11.30	11.17	33.91	14.96	**2.7 s**
5	CSF [21]	x	4.57	13.04	5.98	7.92	20.76	10.06	10.40	25.78	12.96	12.21	33.21	15.71	80 s
6	**SceneFlowFields**	x	5.12	13.83	6.57	8.47	21.83	10.69	10.58	24.41	12.88	12.48	32.28	15.78	65 s
7	PRSF [33]	x	4.74	13.74	6.24	11.14	20.47	12.69	11.73	24.33	13.83	13.49	31.22	16.44	150 s
8	SGM+SF [15, 16]	x	5.15	15.29	6.84	14.10	23.13	15.60	20.91	25.50	21.67	23.09	34.46	24.98	2700 s
9	PCOF-LDOF [7]	x	6.31	19.24	8.46	19.09	30.54	20.99	14.34	38.32	18.33	25.26	49.39	29.27	50 s
10	PCOF+ACTF [7]	x	6.31	19.24	8.46	19.15	36.27	22.00	14.89	60.15	22.43	25.77	67.75	32.76	(0.08 s)

Table 1. Results on the KITTI Scene Flow Benchmark [23]. The column *dual* indicates whether only two frame pairs are used by this method. Run times in parentheses are using a GPU. We achieve the third best result among all dual-frame methods. Our SceneFlowFields yields especially good results at foreground regions (*SF-fg*).

Variant	D1-bg	D1-fg	D1-all	D2-bg	D2-fg	D2-all	Fl-bg	Fl-fg	Fl-all	SF-bg	SF-fg	SF-all	Density	Edges
full+ego	**5.36**	**10.85**	**6.20**	7.94	18.23	**9.51**	**10.36**	22.85	**12.28**	**12.04**	28.31	**14.53**	100.00 %	Ours
full	**5.36**	**10.85**	**6.20**	15.91	**18.03**	16.23	22.33	**21.69**	22.23	24.78	**27.37**	25.18	100.00 %	
no var	**5.36**	**10.85**	**6.20**	15.77	18.81	16.24	23.75	23.72	23.75	25.60	28.99	26.12	100.00 %	
full+ego	5.48	11.99	6.47	9.07	19.98	10.74	11.63	25.47	13.75	13.39	30.92	16.07	100.00 %	SED [8]
full	5.48	11.99	6.47	16.90	20.80	17.50	23.57	25.22	23.82	26.14	30.71	26.84	100.00 %	
no var	5.48	11.99	6.47	16.73	21.38	17.44	25.00	27.04	25.32	26.93	32.16	27.73	100.00 %	
matches	1.91	**3.74**	2.18	2.48	4.08	2.71	2.10	2.78	2.20	3.87	6.24	4.21	38.82 %	–
disparity	**1.42**	4.06	**1.82**	–	–	–	–	–	–	–	–	–	57.81 %	–

Table 2. Evaluation of the different parts of our method on KITTI [23] training data. Our new edge detector outperforms SED [8]. The ego-motion model helps greatly to improve overall results. The bottom two rows show the amount of outliers for our sparse correspondences before interpolation. The density is computed with respect to available KITTI ground truth.

was ranked 6th and achieved the 3rd best result out of all dual-frame methods while at the same time being considerably faster than the top three performing methods (cf. Table 1). That our method generalizes better to other data sets is shown in Section 5.3 where we often outperform the best dual-frame method [23]. In Figure 6, we give an visual example of our results and compare it to the two top performing methods in dual-frame [23] and multi-frame [34] categories. It can be seen that our interpolation produces very sharp edges. This in combination with our matching method helps to obtain accurate scene flow, especially for (moving) objects. Methods with comparable overall performance on KITTI ([21, 31]), perform worse on moving foreground objects than our SceneFlowFields.

Apart from the official evaluation, we test the different components of our method in Table 2 to evaluate the effect of each part. We use all training images of the KITTI data set, and evaluate our basic method without the variational optimization (no var), the full basic approach (full) and our method with the optional ego-motion extension (full+ego). Additionally we compute the accuracy and densities with respect to the KITTI ground truth of our sparse scene flow matches (matches) and the separately filtered sparse stereo correspondences (disparity). The variational optimization is primarily useful for optical flow and foreground. All variants using our improved edge detector outperform their according variant using basic image edges.

Finally, we use the provided object maps of KITTI, to test the performance of our motion segmentation (cf. Figure 5). To this end, we compute the precision and recall for our binary segmentation. Precision is defined as the percentage of estimated pixels that are correctly labeled as in motion. The recall is the relative amount of ground truth pixels that are labeled as moving and covered by our estimation. Over all frames, we achieve a precision of about 28 % and a recall of about 83 %. Most of the missed ground truth foreground pixels belong to objects which are far away and moving parallel to the direction of viewing. This way, the re-projection error of the 3D-2D correspondences during ego-motion estimation is below the threshold. Two remarks have to be considered regarding the precision. First, KITTI only annotates cars that are mostly visible, *i.e.* pedestrians, cyclists, other vehicles, or partly occluded cars are not included in the ground truth, but will be marked as moving if they are in motion. Secondly, since areas that are wrongly classified as dynamic will be filled with our basic scene flow estimation which is still of high quality, we did tune in favor of a high recall.

5.3. MPI Sintel

We claim that our proposed method is very versatile and not at all restricted to any setup. Therefore, we have additionally evaluated SceneFlowFields on MPI Sintel [5] without changing any parameters. The only difference to our
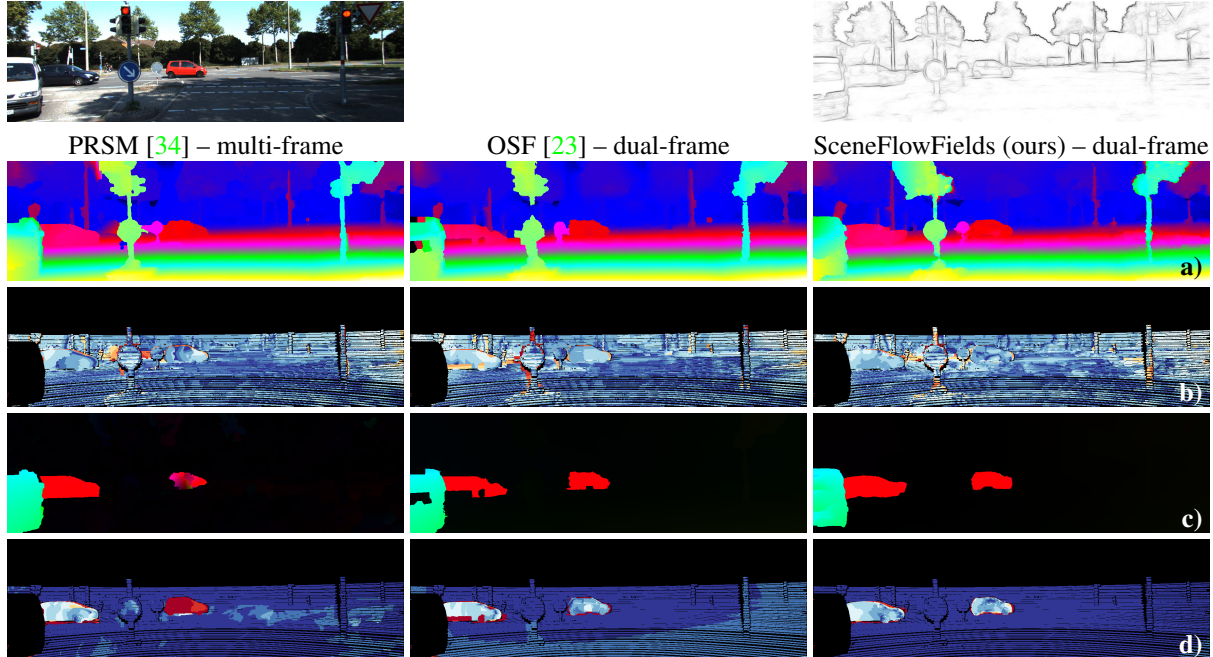
PRSM [34] – multi-frame   OSF [23] – dual-frame   SceneFlowFields (ours) – dual-frame

Figure 6. Exemplary visual comparison on KITTI Scene Flow Benchmark [23]. We show disparity **(a)** and optical flow **(c)** results along with the corresponding error maps **(b)** and **(d)** for PRSM [34], OSF [23] and our SceneFlowFields. We accurately detect moving objects and reconstruct sharp boundaries. More examples are visualized in the supplementary video and on the public homepage of KITTI.

evaluation on KITTI is, that we do not use our semantic edge detector which was trained on KITTI imagery, but instead SED to obtain edge maps. We test both, our basic approach and the ego-motion extension (+ego) for our method. All training frames over all but two sequences for which a subsequent frame exists are processed. The *final* rendering passes for all images are used. We measure the percentage of outliers according to the KITTI metric for disparity and optical flow. The sequences *cave_2* and *sleeping_1* are left out because they have not been evaluated in [31] – to which we want to compare – due to varying camera parameters. The relative amounts of outliers over all evaluated sequences are given in Table 3 and are compared to [23, 31, 34] using the results published by [31]. Our results can keep up with state-of-the-art scene flow methods, although we have not tuned our method on MPI Sintel. For sequences with close-up, non-rigid motion, *e.g. ambush_7* or *bandage_1*, our depth estimation even beats the multi-frame scene flow method that is ranked first on KITTI.

6. Conclusion

Our novel approach to interpolate sparse matches to a dense scene flow achieves state-of-the-art performance on different data sets. At time of submission SceneFlowFields is ranked third on KITTI and achieves state-of-the-art performance on MPI Sintel. We have shown that a stochastic matching approach works for higher dimensional search

Sequence	Disparity				Optical Flow				
	PRSM	OSF	FSF	Ours	PRSM	OSF	FSF	Ours	+ego
Average	15.99	19.84	**15.35**	18.15	**13.70**	28.16	18.32	29.24	22.20
alley_1	7.43	**5.28**	5.92	8.81	**1.58**	7.33	2.11	5.94	3.95
alley_2	**0.79**	1.31	2.08	1.73	1.08	1.44	1.20	2.85	**0.87**
ambush_2	41.77	55.13	**36.93**	51.72	**51.33**	87.37	72.68	90.92	83.84
ambush_4	24.09	24.05	**23.30**	37.78	**41.99**	49.16	45.23	60.03	42.65
ambush_5	**17.72**	19.54	18.54	25.52	25.23	44.70	**24.82**	46.92	29.86
ambush_6	29.41	**26.18**	30.33	37.13	**41.98**	54.75	44.05	57.06	47.65
ambush_7	35.07	71.58	23.47	**16.34**	3.35	22.47	27.87	13.66	7.35
bamboo_1	**7.34**	9.71	9.67	14.53	**2.41**	4.04	4.11	6.11	4.15
bamboo_2	**17.06**	18.08	19.27	19.89	**3.58**	4.86	3.65	5.84	3.97
bandage_1	21.22	19.37	20.93	**16.42**	**3.30**	18.40	4.00	3.82	4.03
bandage_2	22.44	23.53	22.69	**21.77**	**4.06**	13.12	4.76	10.72	9.06
cave_4	**4.27**	5.86	6.22	6.20	16.32	33.94	14.62	15.63	**12.95**
market_2	**5.27**	6.61	6.81	6.71	**4.77**	10.08	5.17	7.11	6.09
market_5	15.38	13.67	**13.25**	26.66	28.38	29.58	**26.31**	40.77	28.87
market_6	**8.99**	10.29	10.63	14.53	**10.72**	16.39	13.13	28.92	16.69
mountain_1	0.42	0.78	0.23	**0.15**	3.71	88.60	17.05	90.60	89.57
shaman_2	25.49	28.27	24.77	**21.13**	**0.46**	1.67	0.56	8.85	4.31
shaman_3	33.92	52.22	**27.09**	35.37	1.75	11.45	**1.31**	15.91	8.51
sleeping_2	**1.74**	2.97	3.52	3.07	**0.00**	0.01	0.02	0.61	0.03
temple_2	**4.92**	5.54	5.96	6.98	9.51	10.52	9.66	29.58	12.57
temple_3	11.04	16.62	10.65	**8.61**	32.10	81.39	62.34	72.28	49.18

Table 3. Results on MPI Sintel [5]. Average outliers show that SceneFlowFields can keep up with state-of-the-art.

spaces and that the applied consistency filters produce very robust correspondences. Boundary-aware interpolation has turned out to be a powerful tool to fill the gaps in the scene flow field due to filtering. To cope with missing correspondences across the images, we have applied an optional ego-motion model that helps to overcome this issue. For future work, we want to improve robustness of the ego-motion or extend SceneFlowFields to use multiple frame pairs.

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *International Conference on Computer Vision (ICCV)*, 2015. 3, 4

[2] T. Basha, Y. Moses, and N. Kiryati. Multi-view scene flow estimation: A view centered variational approach. *International Journal of Computer Vision (IJCV)*, 2013. 2

[3] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *European Conference on Computer Vision (ECCV)*, 2004. 5

[4] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011. 1

[5] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, 2012. 2, 7, 8

[6] Q. Chen and V. Koltun. Full Flow: Optical flow estimation by global optimization over regular grids. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[7] M. Derome, A. Plyer, M. Sanfourche, and G. Le Besnerais. A prediction-correction approach for real-time optical flow computation using stereo. In *German Conference on Pattern Recognition (GCPR)*, 2016. 7

[8] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *International Conference on Computer Vision (ICCV)*, 2013. 4, 5, 6, 7

[9] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 2

[11] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kD-trees. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 3

[12] Y. Hel-Or and H. Hel-Or. Real-time pattern matching using projection kernels. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2005. 3

[13] E. Herbst, X. Ren, and D. Fox. RGB-D Flow: Dense 3-D motion estimation using color and depth. In *International Conference on Robotics and Automation (ICRA)*, 2013. 2, 3

[14] A. Hermans and G. Floros. https://omnomnom.vision.rwth-aachen.de/data/rwth_kitti_semantics_dataset.zip. 4

[15] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2008. 1, 2, 4, 7

[16] M. Hornacek, A. Fitzgibbon, and C. Rother. SphereFlow: 6 DoF scene flow from RGB-D pairs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7

[17] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *International Conference on Computer Vision (ICCV)*, 2007. 2, 3, 5

[18] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers. A primal-dual framework for real-time dense RGB-D scene flow. In *International Conference on Robotics and Automation (ICRA)*, 2015. 2

[19] P. Lenz, J. Ziegler, A. Geiger, and M. Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Intelligent Vehicles Symposium (IV)*, 2011. 1

[20] C. Liu, J. Yuen, and A. Torralba. SIFT Flow: Dense correspondence across scenes and its applications. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011. 3

[21] Z. Lv, C. Beall, P. F. Alcantarilla, F. Li, Z. Kira, and F. Dellaert. A continuous optimization approach for efficient and accurate scene flow. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 3, 7

[22] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[23] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 3, 6, 7, 8

[24] M. Neoral and J. Šochman. Object scene flow with temporal consistency. In *Computer Vision Winter Workshop (CVWW)*, 2017. 2, 3, 7

[25] C. Rabe, U. Franke, and S. Gehrig. Fast detection of moving objects in complex scenarios. In *Intelligent Vehicles Symposium (IV)*, 2007. 1

[26] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 5, 6

[27] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, and A. Lopez. Vision-based offline-online perception paradigm for autonomous driving. In *Winter Conference on Applications of Computer Vision (WACV)*, 2015. 4

[28] R. Schuster, C. Bailer, O. Wasenmüller, and D. Stricker. Combining stereo disparity and optical flow for basic scene flow. In *Commercial Vehicle Technology Symposium (CVTS)*, 2018. 1

[29] R. Schuster, O. Wasenmüller, G. Kuschk, C. Bailer, and D. Stricker. Towards flow estimation in automotive scenarios. In *ACM Computer Science in Cars Symposium (CSCS)*, 2017. 1

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

[31] T. Taniai, S. N. Sinha, and Y. Sato. Fast multi-frame stereo scene flow with motion segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 6, 7, 8

[32] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *International Conference on Computer Vision (ICCV)*, 1999. 2

[33] C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In *International Conference on Computer Vision (ICCV)*, 2013. 2, 3, 7

[34] C. Vogel, K. Schindler, and S. Roth. 3D scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision (IJCV)*, 2015. 2, 3, 7, 8

[35] A. S. Wannenwetsch, M. Keuper, and S. Roth. Probflow: Joint optical flow and uncertainty estimation. In *International Conference on Computer Vision (ICCV)*, 2017. 3

[36] O. Wasenmüller, B. Schenkenberger, and D. Stricker. Towards non-rigid reconstruction - How to adapt rigid RGB-D reconstruction to non-rigid movements? In *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2017. 2

[37] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. Efficient dense scene flow from sparse or dense stereo data. In *European Conference on Computer Vision (ECCV)*, 2008. 2, 5

[38] P. Xu, F. Davoine, J.-B. Bordes, H. Zhao, and T. Denœux. Multimodal information fusion for urban scene understanding. *Machine Vision and Applications (MVA)*, 2016. 4

[39] T. Yoshida, O. Wasenmüller, and D. Stricker. Time-of-flight sensor depth enhancement for automotive exhaust gas. In *International Conference on Image Processing (ICIP)*, 2017. 2