Large Displacement 3D Scene Flow with Occlusion Reasoning

Andrei Zanfir[1] and Cristian Sminchisescu[2,1]
[1]Institute of Mathematics of the Romanian Academy
[2]Department of Mathematics, Faculty of Engineering, Lund University
andrei.zanfir@imar.ro, cristian.sminchisescu@math.lth.se

Abstract

The emergence of modern, affordable and accurate RGB-D sensors increases the need for single view approaches to estimate 3-dimensional motion, also known as scene flow. *In this paper we propose a coarse-to-fine, dense, correspondence-based* scene flow *formulation that relies on explicit geometric reasoning to account for the effects of large displacements and to model occlusion. Our methodology enforces local motion rigidity at the level of the 3d point cloud without explicitly smoothing the parameters of adjacent neighborhoods. By integrating all geometric and photometric components in a single, consistent, occlusion-aware energy model, defined over overlapping, image-adaptive neighborhoods, our method can process fast motions and large occlusions areas, as present in challenging datasets like the MPI Sintel Flow Dataset, recently augmented with depth information. By explicitly modeling large displacements and occlusion, we can handle difficult sequences which cannot be currently processed by state of the art scene flow methods. We also show that by integrating depth information into the model, we can obtain correspondence fields with improved spatial support and sharper boundaries compared to the state of the art, large-displacement optical flow methods.*

1. Introduction

The *scene flow* is a 3-dimensional translation field of an observed scene, and can be predicted from a variety of inputs which encode depth and color information: RGB and depth from stereo or multiple cameras, depth from sensors (structured light, time-of-flight) or even synthetic depth. Computing a 3d motion field is useful for several computer vision applications such as 3d segmentation, navigation, scene understanding and interaction[16, 1, 7, 32, 21].

The idea of estimating a 3d motion field is relatively new in contrast with considerably more work focused on estimating an optical flow field[10, 3, 30, 15, 23, 31, 13, 29]. An open question raised by the emergence of sensors capturing appearance augmented with depth is how to efficiently exploit this additional source of information.

In this paper, we show that by reasoning based on a 3d point cloud representation, we can produce physically plausible explanations of the scene motion across successive RGB-D frames separated by large displacements and under significant occlusion. Instead of linearizing infinitesimal motion explicitly, which may be difficult or invalid, we initially estimate large-displacement correspondences that act as anchors. We then rely on local rigidity by constraining 3d point cloud neighborhoods to individually follow rigid transformations. In order to explain the observed data, the anchors and the geometric terms based on local rigidity are augmented with constraints on the appearance of the point cloud. Appearance is used such that points in correspondence match, whereas depth alignment constraints are used to prevent the estimated solution from drifting away from the target point cloud. Traditionally, the entire 3d motion field has been regularized to obtain smooth solutions, but we show that by just using rigid, geometric constraints on overlapping, large and image-adaptive neighborhoods, reliable results can be obtained. The links between neighbors are based on depth and occlusion estimates in order to preserve structure and correctly pass or block information.

The main contribution of this work is the design of a scene flow solver that can handle large displacements, uses explicit 3d occlusion reasoning, and integrates large-displacement and geometric terms. The model is distinct with respect to classical state of the art approaches in both variational optical flow and scene flow optimization. By establishing correspondences constrained by local rigid transformation, and without explicitly enforcing the smoothness of the motion field, we can obtain reliable scene flows with good consistency with the RGB-D input data.

2. Related Work

The scene flow was introduced in the seminal work of Vedula [25] as the 3d motion field of the scene. Since then, several computational approaches have been developed. Whenever a stereo or a multi-view camera system is

available, the scene flow can be computed by imposing consistency with the observed optical flow [25], by decoupled [28] or joint [11, 28] estimation of structure and motion, or by relying on local scene rigidity [26], [27]. In this work we assume that a depth sensor is available, therefore estimating the 3d structure is not needed.

Spies et al. [22] combine intensity and depth information by extending the optical flow formulation of Horn and Schunck [10] to take advantage of depth information, integrated as an additional channel in the variational framework. The depth flow is estimated based on the observed data with a smoothness prior. However, the camera is assumed to be orthographic and there is no explicit connection between the optical and depth (range) flow. Consistency issues are addressed in [14], where the scene flow is directly solved based on the range data. This constrains the 3d motion in image space, although the method still does not use range constraints.

Quiroga et al. [18] define a 2d warping function to couple apparent and 3d motion, through a joint local scene flow constraint on both intensity and depth data. Although the method is able to handle large displacements, it fails on untextured regions and more complex motions, such as rotations. In order to solve for dense scene flow, a regularization procedure is required. Usually, the 3d motion field is assumed to be piecewise smooth and total variation (TV) is used as regularizer, resulting in a better handling of discontinuities.

A rigid over-parametrization of the scene flow is further introduced in [17], but the used neighborhoods are small, thus offering limited information during drastic appearance changes which are frequent with fast motions. We differ from their approach in the handling of large displacements, in relying on both anchors and (re-estimated) geometric correspondences and in constructing an energy model that is occlusion-aware. Our model operates over large, image-adaptive neighborhoods, under local rigidity assumptions but without smoothness priors. The potential drift is compensated by aligning and matching, followed by snapping using the point cloud grid, and not by modeling the residual motion of the camera as performed in [17]. To deal with large displacements, [14] use SIFT matching to initialize their optimizer, in spirit similar to [3], by fixing anchors in the flow and setting a corresponding energy in their formulation. While we also rely on initial anchors and continuously estimating correspondences, we additionally provide a complete occlusion-aware energy formulation which enforces both geometric and appearance consistency.

Herbst [9] relies on a scene flow adaption of [4] with robust penalty terms and gradient constancy assumptions. To avoid smoothing across motion boundaries, they use a neighborhood structure that weights connected points according to color, normal and depth similarities, but do not infer occlusion. In [19], a novel variational extension of [18] is presented. A weighted TV is applied on each component of the 3d motion field in order to preserve motion discontinuities across depth edges.

In an alternative approach, Hadfield and Bowden [8] estimate the scene flow using particle filtering and a 3d colored point cloud representation of the scene. In their method, a large set of motion hypotheses are generated and tested, offering a beneficial degree of robustness, as multiple solutions are propagated consistently over time, at additional computational cost.

3. Model Formulation

We work with RGB-D images represented using their RGB and depth components, I and Z, respectively. Specifically, $I : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}^3$ and $Z : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}$. For two-frame modeling, we use images I_t, I_{t+1} with corresponding depth maps Z_t, Z_{t+1}, but time indexes will be dropped whenever unambiguous. Given calibrated cameras and a projection function $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, our goal is to align the two point-clouds describing the scene at two close moments in time in a way that is consistent with the given RGB-D input and physically plausible at multiple scales l, in the sense of local rigidity. The function Π projects a 3d point onto the image domain Ω:

$$\Pi(\mathbf{x} = (x, y, z)) = \left(f_x \frac{x}{z} + c_x, \quad f_y \frac{y}{z} + c_y\right)^\top \qquad (1)$$

where f_x and f_y are the focal lengths of the camera and (c_x, c_y) its principal point. We also consider the function $z(\mathbf{x}) = \mathbf{z}^\top \mathbf{x} = z$, where the vector $\mathbf{z} = [0, 0, 1]^\top$ isolates the depth component of the argument. Notice that for a virtual 3d point $\mathbf{y}$, placed (or predicted) within the point cloud measured by the sensor, it can happen that $z(\mathbf{y}) \neq Z(\Pi(\mathbf{y}))$.

The inverse projection function $\Pi_Z^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ back-projects a point from the image plane to the corresponding 3d location, given a map of depths:

$$\Pi_Z^{-1}(x, y) = \left(Z(x, y)\frac{x - c_x}{f_x}, \quad Z(x, y)\frac{y - c_y}{f_y}, \quad Z(x, y)\right)^\top \qquad (2)$$

We define the point-cloud associated with I_t, Z_t, as X, the matrix containing the 3d locations of N points in the first frame. Scene flow estimation addresses the problem of finding the corresponding 3d locations in the second frame Y, or more precisely, the displacement field ΔX:

$$Y = X + \Delta X \qquad (3)$$

We formulate the solution as a dense correspondence problem, solving for Y while making assumptions about the local rigidity of the displacement field, by ensuring appearance consistency and by geometrically reasoning about occlusion.

3.1. Local Rigidity Assumption

We enforce neighboring points on the same object surface to obey, *as truthfully as possible*, a *rigid 3d* geometric transformation, by relying on techniques of point-cloud registration. A rigid transformation $\mathbf{D}$ applied to a 3d point $\mathbf{x} \in \mathbb{R}^3$, consists of a rotation matrix $\mathbf{R}(\mathbf{v}) \in SO(3)$, the 3d rotation group, and a translation vector $\mathbf{t} = [t_x, t_y, t_z]^\top$, such that:

$$\mathbf{D}(\mathbf{x}; \mathbf{v}, \mathbf{t}) = \mathbf{R}(\mathbf{v})\mathbf{x} + \mathbf{t} \tag{4}$$

Parametrization of rotations: We rely on standard rigid representations, *e.g*. [6, 17], in order to define the parameter $\mathbf{v}$ as a vector in $\mathbb{R}^3$ describing the axis of rotation $\widehat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$, and $\phi = \|\mathbf{v}\|$ the angle of rotation around that axis. To build $\mathbf{R}(\mathbf{v})$ we use an *exponential map* from $\mathbb{R}^3$ to the 3-sphere S^3, $\mathbf{q} \in S^3$, and a quaternion-to-matrix transformation for conversion to $SO(3)$, *i.e*. $\mathbf{R}(\mathbf{q}(\mathbf{v}))$. The exponential map is defined as:

$$\begin{aligned} &e^{[0,0,0]^\top} = [0,0,0,1]^\top \\ &e^{\mathbf{v}} = \sum_{i=0}^{\infty} \left(\frac{1}{2}\widetilde{\mathbf{v}}\right)^i = \left[\sin(\frac{1}{2}\phi)\widehat{\mathbf{v}}, \cos(\frac{1}{2}\phi)\right]^\top = \mathbf{q}, \mathbf{v} \neq \mathbf{0} \end{aligned} \tag{5}$$

where $\widetilde{\mathbf{v}}$ is a quaternion created by extending the vector $\mathbf{v}$ with a scalar component 0, $\mathbf{q}$ is the resulting quaternion, and $[0,0,0,1]^\top$ is the quaternion corresponding to the identity rotation. The exponentiation $\widetilde{\mathbf{v}}^i$ is performed by quaternion multiplication. For numerical stability, (5) is rewritten as:

$$e^{\mathbf{v}} = \left[\frac{\sin(\frac{1}{2}\phi)}{\phi}\mathbf{v}, \cos(\frac{1}{2}\phi)\right]^\top \tag{6}$$

3.2. Neighborhood Link Structure

We encode the neighborhood by means of matrices $\mathbf{W}^g$ and $\mathbf{W}^a$. These measure the likelihood of connected neighbors, capturing how likely $\mathbf{x}_i$ and each one of its neighbors are to obey the same local rigid motion (in case of $\mathbf{W}^g$), as well as, additionally, if they also remain visible in the second image (in case of $\mathbf{W}^a$). The distinction is made by integrating occlusion estimates: the data term becomes invalid for points that lose visibility, unlike the geometric term, that continuously constrains the motion.

$$\mathbf{W}^g_{ij} = \begin{cases} \exp(-\tau_g \|\mathbf{x}_i - \mathbf{x}_j\|_2^2), & j \in \mathcal{N}(i). \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$\mathbf{W}^a_{ij} = \mathbf{W}^g_{ij} \exp(-\tau_o \cdot \text{Occ}(\Pi(\mathbf{x}_j)) \tag{8}$$

where τ_g, τ_o are parameters with values found by validation. The occlusion map Occ is a function of the motion field and can be queried at a specific position, as detailed next.

3.3. Occlusion Reasoning

The availability of depth information allows better reasoning about image locations where occlusions may occur, based on the current model estimates. Using the correspondences Y (inferred at processing level l, dropped here for simplicity), we present two approaches to integrate 3d geometric cues for occlusion estimation. These are presented in Algorithms 1 and 2. If we denote the two occlusion maps O_1 and O_2 produced, respectively, by each algorithm, we define the occlusion term Occ:

$$\text{Occ} \leftarrow \max(O_1, O_2) \tag{9}$$

where $\max$ is applied element-wise on the values in O_1, O_2 to produce the final Occ visibility map. Algorithm 1 aims to identify those points that are moving behind or in front of a previously occupied location. Our inference is based on the depth at frame *t*. If, initially, the end points in correspondence have different depths (*i.e*. we can view them in layers of constant depth that are nearer or further away from the camera), the depth disparity is used as a cue for occlusion: either the first point shifts 'under' an occluding layer or the second point is occluded by the movement of the initial layer.

Data: X, Y, Z_t
Result: Occlusion states O_1 for all image points
$O_1 \leftarrow \mathbf{0}$;
for *each point* $\mathbf{x}_i$ *in* X **do**
  if $\Pi(\mathbf{y}_i) \notin \Omega_t$ **then**
    $O_1(\Pi(\mathbf{x}_i)) = 1$;
  end
  $\Delta Z = Z_t(\Pi(\mathbf{y}_i)) - z(\mathbf{x}_i)$;
  if $\Delta Z < 0$ **then**
    $O_1(\Pi(\mathbf{x}_i)) = \min(1, O_1(\Pi(\mathbf{x}_i)) + \varepsilon_Z|\Delta Z|)$;
  else
    $O_1(\Pi(\mathbf{y}_i)) = \min(1, O_1(\Pi(\mathbf{y}_i)) + \varepsilon_Z|\Delta Z|)$;
  end
end

Algorithm 1: Geometric occlusion reasoning algorithm based on Z_t and correspondences estimates X in frame t and Y in frame $t+1$. The parameter ε_Z is set by validation.

Algorithm 2 integrates both data and estimates, by considering the initial and measured depths at the corresponding target location. The differences from the first algorithm are: it does not take into consideration the initial depth ordering of end-points and it only predicts the occlusion state for the initial location. If a point ends up with a depth value higher than it had initially, we obtain an occlusion cue, assuming points are at equal depth. Notice that differently from advanced 2d optical flow models[24], we do

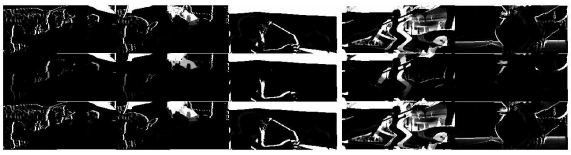

Figure 1. Occlusion maps generated by our Alg. 1 (first row) and Alg. 2 (second row) as well as their max operator, *c.f.* §3.3. In practice we found that combining both estimates provides robustness and complementarity: the first estimate (Alg. 1) tends to be effective in the case of foreground motions against quasi-static backgrounds, whereas the latter (Alg. 2) is adequate in cases when the entire scene, or the camera, is moving.

not work with object support hypotheses or explicit planar layers: instead we reason based on the real depth values and the hypothesized 3d inter-frame point displacements. While the use of relative depth cues and geometric reasoning has proven effective in our case (§5), in practice, training an occlusion classifier based on such features, and other appearance-based ones, can also be effective and we are exploring such options in ongoing work.

Data: X, Y, Z_t, Z_{t+1}
Result: Occlusion states O_2 for all image points
$O_2 \leftarrow \mathbf{0}$;
for *each point* $\mathbf{x}_i$ *in* X **do**
  if $\Pi(\mathbf{y}_i) \notin \Omega_{t+1}$ **then**
    $O_2(\Pi(\mathbf{x}_i)) = 1$;
  end
  $\Delta Z = z(\mathbf{x}_i) - Z_{t+1}(\Pi(\mathbf{y}_i))$;
  $O_2(\Pi(\mathbf{x}_i)) =$
  $\min(1, \max[0, O_2(\Pi(\mathbf{x}_i)) + \varepsilon_Z \Delta Z])$;
end

Algorithm 2: Geometric occlusion reasoning algorithm based on Z_t, Z_{t+1} and correspondences estimates X in frame t and Y in frame $t+1$.

3.4. Energy Components

We design a dense energy model that can handle large displacement and occlusion. The model is expressed in terms of several energy sub-components. It relies on geometric large-displacement correspondence anchors, local rigidity assumptions defined over large neighborhoods constructed using the RGB-D point cloud, on appearance consistency, and depth constancy terms. Geometric occlusion estimates are used in order to control the strength of the neighborhood connections and automatically mask regions for which correspondences cannot be established. In this section, we review each component of the energy. The complete scene-flow algorithm appears in §4. Detail on our coarse-to-fine second-order optimization scheme is given in the Appendix.

Rigid Parameter Fields. Our geometric model links $\mathbf{Y}$ to $\mathbf{X}$ by assuming an underlying field of rigid parameters $\mathbf{\Theta} = [\boldsymbol{\theta}_1^\top, \boldsymbol{\theta}_2^\top, ..., \boldsymbol{\theta}_N^\top]^\top$, where $\boldsymbol{\theta}_i = [\mathbf{v}_i^\top, \mathbf{t}_i^\top]^\top$ are rigid parameters that constrain each neighborhood $\mathcal{N}(i)$ of a point $\mathbf{x}_i \in X$. A neighborhood represents the closest *K* points to $\mathbf{x}_i$ in 3d, where *K* is chosen appropriately, according to the pyramid level. The neighborhoods are large in both 3d and 2d: in practice we use 150-250 neighbors, which can extend to 60-80 cm in 3d and, depending on viewpoint, between 20-200 pixels radius, or up to $1/4$ of the image diagonal. This creates interesting long range connections in the image plane, as *e.g.* an isolated part of the background will still link with its neighbours on other parts of the background even though a foreground object might interpose.

Large-Displacement Anchors. If initial correspondences $\mathbf{Y}^0$ from a large-displacement matching process are available, we define an energy term that constrains (biases) the solution around $\mathbf{Y}^0$:

$$E_M(\mathbf{\Theta}) = \sum_i \sum_{j \in \mathcal{N}(i)} W_{ij}^a \Psi(\Delta \mathbf{D}_{ij}^0) \qquad (10)$$

where for brevity, we use $\Delta \mathbf{D}_{ij} = \mathbf{D}_{ji} - \mathbf{y}_j$, $\mathbf{D}_{ji} = \mathbf{D}(\mathbf{x}_j; \boldsymbol{\theta}_i)$, and parameters $\boldsymbol{\theta}_i \in \mathbf{\Theta}$. The robust error penalty Ψ is defined as $\Psi(\mathbf{x}) = \sqrt{\mathbf{x}^\top \mathbf{x} + \epsilon^2}$.[1]

Locally Rigid Geometric Constraint. We define an energy component that penalizes the deviation of each neighborhood from a rigid transformation:

$$E_G(\mathbf{Y}, \mathbf{\Theta}) = \sum_i \sum_{j \in \mathcal{N}(i)} W_{ij}^g \Psi(\Delta \mathbf{D}_{ij}) \qquad (11)$$

Notice that we do not enforce smoothness across the rigid parameters of neighboring points [17]. Instead, our neighborhoods are large and overlapping, so a degree of smoothness, as well as robustness to large displacements, are achieved implicitly. However, we will also need additional appearance and depth constancy terms in order to obtain solutions that match all the observed data, 2d and 3d, as described next.

Appearance Term. Rather than directly enforcing similar appearance between points in correspondence, *i.e.* $\mathbf{x}_i$ and $\mathbf{y}_i$, we place a constraint on the underlying *matched 3d regions*. The projections Π of neighborhoods in the two images, and their disparity are integrated into the following term:

$$E_A(\mathbf{\Theta}) = \sum_i \sum_{j \in \mathcal{N}(i)} \mathbf{W}_{ij}^a \sum_{c=1}^{3} \Psi(\Delta I_{ij}^c) \qquad (12)$$

[1] A specific noise sensor model can be estimated from data and seamlessly included with the robust error penalty in our formulation.

where we abbreviate $\Delta I^c_{ij} = I^c_{t+1}(\Pi \circ \mathbf{D}_{ji}) - I^c_t(\Pi(\mathbf{x}_j))$, '$\circ$' denotes function composition, and c indexes the color channel in RGB data.

This energy constraint derives from our assumption that regions are locally rigid. If that assumption is correct, warping the local appearance through the rigid displacement $\mathbf{D}(\mathbf{x};\boldsymbol{\theta})$ is valid with respect to all measurements and not only consistent with a subset of the matches. Therefore, we impose that the projection of a region in the first image (plane) must remain coherent, in terms of color and/or gradient information, with the projection of the rigid transformation of the same region, in the second image.

Depth Constancy Term. The point cloud associated to the scene in one frame, moving under the estimated local rotations and translations needs to match the point cloud in the other frame, as estimates can otherwise drift if left unconstrained. The appearance penalty is not sufficient, as it only partially constrains the projection. We therefore introduce an energy component that integrates the disparity between the expected (model predicted) depth and the measured depth, obtained from the sensor:

$$E_Z(\boldsymbol{\Theta}) = \sum_i \sum_{j \in \mathcal{N}(i)} \mathbf{W}^a_{ij} \Psi(\Delta Z_{ij}) \qquad (13)$$

where we abbreviate $\Delta Z_{ij} = Z_{t+1}(\Pi \circ \mathbf{D}_{ji}) - \mathbf{z}^\top \mathbf{D}_{ji}$, with $\mathbf{z} = [0,0,1]^\top$ isolating the depth component of a point displaced by the rigid transformation. The point $\mathbf{x}_j \in \mathbf{X}$ in frame t is transformed by the appropriate local rigid transformation, gets projected onto the image plane in frame $t+1$ and we compare it to the value read from the depth map Z_{t+1} at that location. This guarantees that $\mathbf{x}_j$ remains close to the point cloud, after the transformation.

4. Scene Flow Optimization

Our complete energy model (§4.1) measures the consistency of the solution with respect to the observed data and locally penalizes non-rigid displacements. After each energy minimization iteration, we update an occlusion estimate that is used by the energy model (*c.f.* §3.3). Similarly in spirit to ICP methods [2], we 'snap' our predicted *visible* points to their nearest neighbour in the cloud $\Pi^{-1}_{Z_{t+1}}(\Omega)$ in the target frame. The algorithm is presented in Alg. 3.

4.1. Dense Energy

The full energy model to optimize can be written as:

$$E(\mathbf{Y}, \boldsymbol{\Theta}) = E_A + \alpha E_Z + \beta E_M + \lambda E_G \qquad (14)$$

where α, β and λ are weights estimated by validation.

Minimization. We solve (14) over the parameter fields $\mathbf{Y}, \boldsymbol{\Theta}$ by alternating variable optimization:

$$\boldsymbol{\Theta}^{k+1} \leftarrow \arg\min_{\boldsymbol{\Theta}} E(\mathbf{Y}^k, \boldsymbol{\Theta}^k) \qquad (15)$$

$$\mathbf{Y}^{k+1} \leftarrow \arg\min_{\mathbf{Y}} E(\mathbf{Y}^k, \boldsymbol{\Theta}^{k+1}) \qquad (16)$$

where k is the iteration index. We run the optimization to convergence. See Appendix A for derivations.

4.2. Complete Scene Flow Algorithm

Ingredients are now in place to describe the complete scene flow algorithm, Alg. 3. We rely on a coarse to fine scheme based on 3 processing levels, where the image domain Ω is sampled every n_l pixels, with $l \in \{0,1,2\}$, on both the x and y dimensions. Let Ω_l be the regular grid at processing level l. We define the 3d locations $\mathbf{X}_l$ and the parameter fields $\mathbf{Y}_l, \boldsymbol{\Theta}_l$ associated with Ω_l.

Our computational pipeline starts by estimating large-displacement correspondences.[2] Due to its excellent performance, we use DeepMatching[29] in order to obtain an initial set of correspondences, which we interpolate (using bilinear functions on the displacements induced by the four closest matches) in order to obtain a dense field (§4) for the initial discrete grid of points Ω_0. However, in principle, any other large-displacement 2d or 3d, sparse or dense matching method can be used. We generically refer to the large-displacement routine as `LDMatchingInterp`). We transform 2d correspondences to 3d, by inverse projection in order to obtain $\mathbf{Y}_0$. Whenever 2d locations fall outside image boundaries, we set the depth from the closest matched neighbour, and we inverse project using Π^{-1}. The rigid parameter field $\boldsymbol{\Theta}_0$ is initialized by setting all the exponential parameters to identity rotations, while translations are set to $\mathbf{Y}_0 - \mathbf{X}_0$.

At each level l, we optimize the energy defined by (14) using (15) and (16), starting from estimates $\mathbf{Y}_l$ and $\boldsymbol{\Theta}_l$, to obtain solutions $\mathbf{Y}^*_l$ and $\boldsymbol{\Theta}^*_l$. We 'snap' non-occluded points in $\mathbf{Y}^*_l$ to their nearest neighbours in the point cloud associated with the target frame $t+1$ by means of function `ICPsnap`. To pass the parameter fields $\mathbf{Y}^*_l$ as initialization to the next processing level $l+1$, we need to interpolate between *translations* rather than correspondences: if $\mathbf{y}_i \in \mathbf{Y}_{l+1}, \mathbf{x_i} \in \mathbf{X}_{l+1}$ and $\mathbf{y}_i = \mathbf{x}_i + \Delta\mathbf{x}_i$, we need to find the closest (4) neighbours of $\mathbf{x}_i$ in $\mathbf{X}_l$, and interpolate between the corresponding translations to compute the motion $\Delta\mathbf{x}_i$. We refer to this interpolation routine as `Interp3d`.

To initialize $\boldsymbol{\Theta}_{l+1}$ with the rigid parameter field at the previous level, $\boldsymbol{\Theta}^*_l$, we select neighbours as we did in the initialization of $\mathbf{Y}_{l+1}$. For the exponential map components

[2] Our method can operate without this energy cost E_M, with large-displacements still handled through our large neighborhood matching term E_G. Results provided by such a model are shown in fig. 2.

$\mathbf{v}$, we pass to a quaternion representation $\mathbf{q}$ via the exponential map, use spherical linear interpolation (*i.e*. function `SLERP`), and then map back to the original axis-rotation representation. For the translational component of the rigid transformations we also use `Interp3d`.

Data: I_t, I_{t+1}, Z_t, Z_{t+1}
// initialize large-displacement correspondences
$(\{\mathbf{p}_i, \mathbf{q}_i)\}_i \leftarrow$ `LDMatchingInterp`(I_t, I_{t+1})
$X \leftarrow [\ldots, \Pi_{Z_t}^{-1}(\mathbf{p}_i), \ldots]$
$Y^0 \leftarrow [\ldots, \Pi_{Z_{t+1}}^{-1}(\mathbf{q}_i), \ldots]$
//initialize local rigid parameters $\boldsymbol{\theta}_i = (\mathbf{v}_i, \mathbf{t}_i)$
$\mathbf{v}_i \leftarrow [0,0,0]^\top$, $\mathbf{t}_i \leftarrow \mathbf{y}_i - \mathbf{x}_i, \forall \mathbf{x}_i \in X, \mathbf{y}_i \in Y_0$
for *each pyramid level* $l \in \{0, 1, 2\}$ **do**
  while *not converged* **do**
    // estimate occlusion
    `Update` Occ,$\mathbf{W}^a$, $\mathbf{W}^g$ from (9), (7) ;
    $\boldsymbol{\Theta}_l^* \leftarrow$ `Optimize` (15);
    $\mathbf{Y}_l^* \leftarrow$ `Optimize` (16);
    $\mathbf{Y}_l^* \leftarrow$ `ICPsnap`$(\mathbf{Y}_l^*)$;
  end
  // Pass output to the next level
  $\mathbf{Y}_{l+1} \leftarrow$ `Interp3d`$(\mathbf{Y}_l^*)$;
  $\mathbf{v}_i \leftarrow$ `SLERP`$(\mathbf{v}_i^*), \forall \mathbf{v}_i \in \boldsymbol{\Theta}_{l+1}, \forall \mathbf{v}_i^* \in \boldsymbol{\Theta}_l^*$;
  $\mathbf{t}_i \leftarrow$ `Interp3d`$(\mathbf{t}_i), \forall \mathbf{t}_i \in \boldsymbol{\Theta}_{l+1}, \forall \mathbf{t}_i^* \in \boldsymbol{\Theta}_l^*$;
end

Algorithm 3: Computational steps of RGB-D scene flow.

5. Experiments

In all our experiments, the image resolution was 640×480, with a 3-level pyramid where each layer samples the original data at $5, 3,$ and 1 pixel strides, respectively. The iterative optimization procedure is run to convergence.

RGB-D Datasets. For training and testing purposes, we use RGB-D sequences taken from the CAD 120 dataset [12], with noisy depth captured by a Kinect device, as well as RGB sequences, newly augmented with accurate depth, from the MPI Sintel dataset[5]. Parameters are validated separately, on withheld data, for each dataset as these have significantly different depth, noise, motion and appearance statistics.

CAD 120 consists of video frames with no camera motion that are 40 to 50 ms apart, so we sample once every 4 to 5 frames to ensure large displacements. We selected several challenging sequences for illustration, provided in fig. 2.

Sintel. We use the new MPI Sintel dataset and benchmark[5], consisting of a test set of 552 image pairs (12 folders) and 1040 training image pairs (23 folders). The ground truth used in the evaluation is the actual motion field in both visible and occluded (unmatched) areas. The dataset contains computer generated video sequences from the animated movie Sintel, with extremely difficult cases of very large displacements, occlusion, motion and defocus blur, specular reflections and strong atmospheric effects. As depth data for the test set was unavailable, we used 2 subsets of the training data: Sintel-Train23 with 23 pairs, with each single pair selected from the middle of its corresponding training folder and Sintel-Test92 with 92 pairs, and each 4 pairs selected similarly, in an uniform fashion. We used Sintel-Train23 to validate our parameters, and Sintel-Test92 for testing.

Running times. Our method takes roughly 4 minutes to run for a pair of images, with 2 minutes of preprocessing (initial matches, image filtering, *etc*.), on a 3.2Ghz 16-core machine. The heaviest computational burden is in the energy minimization, which is linearly dependent on the number of iterations, the neighborhood size, and image resolution. But it is also highly parallelizable.

Results. The first set of experiments uses pairs of frames from CAD120. We analyze the robustness of several methods w.r.t. articulated human motion observed from a fixed viewpoint. fig. 2 shows side-by-side results of a large displacement optical flow method [3] and ours, indicating that the use of 3d information indeed offers gains.

The second set of experiments focuses on Sintel, with qualitative results for four different state of the art 2d and 3d methods shown in fig. 3 and quantitative results shown in table 1. Our scene flow and the corresponding occlusion estimates are accurate in challenging situations where large-displacements and a variety of other effects occur. While we expect 3d methods to perform better, this is still non-trivial as we do not have access to ground truth boundaries. Although the results are visually compelling, with comparatively superior spatial support and sharper boundaries for 3d models over 2d ones, the error on Sintel can be dominated by the displacement of structures like hair, which is difficult to infer, but easy to render accurately based on a synthesis model, as used to generate the Sintel dataset. Whether a detailed flow and occlusion representation can, or should be, extracted for such structures remains unclear both in principle, and in terms of utility.

Algorithms	LDOF[3]	EpicFlow[20]	Ours
Error(in pixels)	7.44	4.82	4.6

Table 1. Quantitative evaluation of state of the art 2d and 3d methods on the challenging Sintel dataset (*Sintel-Test92*). Our method uses 3d information and offers improved accuracy as well as accurate estimates of occlusion.

6. Conclusions

We have presented a new model to compute dense scene flows from RGB-D images by adopting a matching approach in conjunction with a consistent coarse to fine for-

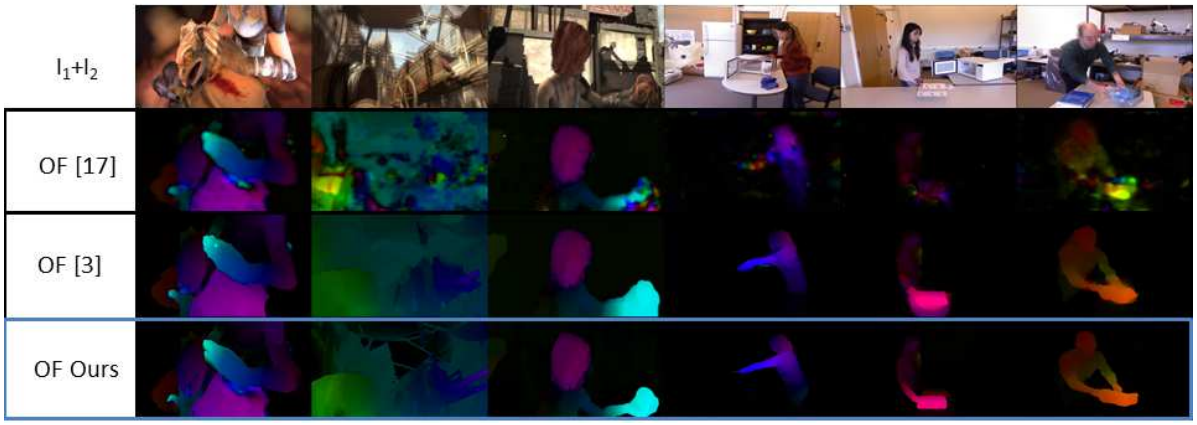

Figure 2. Sample scene flow estimation with results in the x-y plane for image pairs in the Sintel (first three columns) and CAD120 (last three columns). Displacements are *moderate*. We show results from 3 different methods: **1st row:** 3d scene flow[17]; **2nd row:** large-displacement 2d optical flow[3]; **3rd row:** our proposed 3d scene flow method without relying on large-displacement anchors (no E_M component). Displacements for both us and [17] are initialized to 0. Notice the improved results of 3d methods and the good handling of detail of our model.

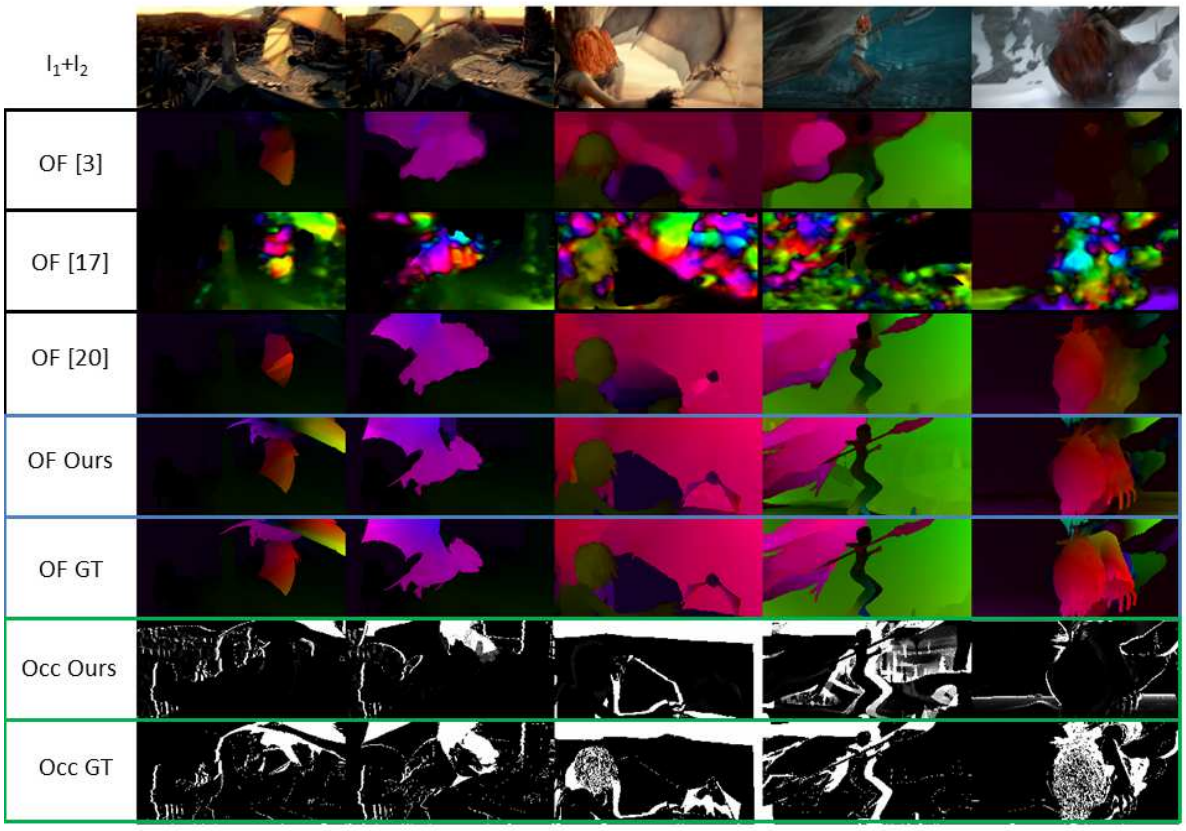

Figure 3. Five pairs of images and four methods illustrated on the Sintel dataset: **1st row:** input images overlaid, **2nd row:** [3], **3rd row:** [17] (N.B. this state of the art 3d scene flow method is not designed to handle large displacements, but produces excellent results for small to moderate ones), see also fig. 2, **4th row:** EpicFlow[20], **5th row:** our proposed approach shows sharper boundaries and improved spatial support estimates, **6th row:** ground truth optical flow. On the last two rows we show the ground truth occlusion states and our soft estimates.

mulation. Our geometric, locally rigid formulation can efficiently requlalarize the solution, whereas the optimization scheme, operating on a unique energy formulation, has been demonstrated to handle large displacements. Explicit 3d geometric occlusion reasoning, integrated in the energy formulation, offers increased robustness in areas where correspondences cannot be established. By explicitly modeling large displacements and occlusion, we can successfully work with difficult sequences which cannot be processed by current state of the art 3d scene flow methods. We have also shown that by using depth information, whenever available, we can, as expected, obtain superior correspondence fields compared to those of state of the art large-displacement 2d optical flow methods.

Appendix A: Optimization Details

A. 1. Optimizing the Rigid Motion Parameters Θ

Expanding (14) yields:

$$E(\mathbf{Y},\mathbf{\Theta}) = \sum_i \sum_{j\in\mathcal{N}(i)} \left[\mathbf{W}^a_{ij} \sum_{c=1}^{3} \Psi(\Delta I^c_{ij}) + \alpha \mathbf{W}^a_{ij} \Psi(\Delta Z_{ij}) + \beta \mathbf{W}^a_{ij} \Psi(\mathbf{D}^0_{ij}) + \lambda \mathbf{W}^g_{ij} \Psi(\mathbf{D}_{ij}) \right] \quad (17)$$

Linearization. The non-linearity of our objective function is handled by means of a 2^{nd} order Taylor expansion around a current estimate $\mathbf{\Theta}^k$, at iteration $k+1$:

$$\begin{aligned} E(\mathbf{Y},\mathbf{\Theta}^k+\Delta\mathbf{\Theta}) = \sum_i \sum_{j\in\mathcal{N}(i)} \Bigg[& \mathbf{W}^a_{ij} \sum_{c=1}^{3} \Psi(\Delta I^c_{ij}) + \\ & + \alpha \boldsymbol{W}^a_{ij} \Psi(\Delta Z_{ij} + (\mathbf{J}_{Z_{t+1}} - \mathbf{z}^\top \mathbf{J}_T)\Delta\boldsymbol{\theta}_i) + \\ & + \beta \boldsymbol{W}^g_{ij} \Psi(\Delta \mathbf{D}^0_{ij} + \mathbf{J}_D \Delta\theta_i) + \\ & + \lambda \boldsymbol{W}^g_{ij} \Psi(\Delta \mathbf{D}_{ij} + \mathbf{J}_D \Delta\theta_i) \Bigg] \end{aligned} \quad (18)$$

where

$$\mathbf{J}_{I^c_{t+1}} = \nabla I^c_{t+1} \mathbf{J}_\pi \mathbf{J}_\mathbf{D} \quad (19)$$

$$\mathbf{J}_{Z_{t+1}} = \nabla Z_{t+1} \mathbf{J}_\pi \mathbf{J}_\mathbf{D} \quad (20)$$

$$\mathbf{J}_\pi = \begin{bmatrix} \frac{f_x}{z} & 0 & -f_x \frac{x}{z^2} \\ 0 & \frac{f_y}{z} & -f_y \frac{y}{z^2} \end{bmatrix} \quad (21)$$

$$\mathbf{J}_\mathbf{D} = \begin{bmatrix} (\partial\mathbf{R}/\partial\mathbf{v})\mathbf{x} \\ \mathbf{I}_{3\times 3} \end{bmatrix}^\top \quad (22)$$

are the Jacobian matrices evaluated at $\mathbf{x}_j$, $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix, and ∇ is used to compute the gradient of the color channels I^c_{t+1} and depth map Z_{t+1} of the second frame.

To evaluate the derivatives of $\mathbf{R}$ w.r.t. the exponential map parameters $\mathbf{v}$, we use the chain rule:

$$\frac{\partial\mathbf{R}}{\partial\mathbf{v}} = \frac{\partial\mathbf{R}}{\partial\mathbf{q}} \frac{\partial\mathbf{q}}{\partial\mathbf{v}} \quad (23)$$

where efficient formulas exist for the computation of both $\partial\mathbf{R}/\partial\mathbf{q}$ and $\partial\mathbf{q}/\partial\mathbf{v}$, using (5) and (6).

Update. Because the vectors $\boldsymbol{\theta}_i$ are independent, we can optimize them in parallel. Taking the partial derivatives of (18) and setting them to zero yields:

$$\Delta\boldsymbol{\theta}^*_i = -\mathbf{H}^{-1}_i \mathbf{g}_i \quad (24)$$

where:

$$\begin{aligned} \mathbf{H}_i = \sum_{j\in\mathcal{N}(i)} \Bigg[& \mathbf{W}^a_{ij} \sum_{c=1}^{3} \Psi'(\Delta I^c_{ij}) \mathbf{J}^\top_{I^c_{t+1}} \mathbf{J}_{I^c_{t+1}} + \\ & + \alpha \mathbf{W}^a_{ij} \Psi'(\Delta Z_{ij}) (\mathbf{J}_{Z_{t+1}} - \mathbf{z}^\top \mathbf{J}_\mathbf{D})^\top (\mathbf{J}_{Z_{t+1}} - \mathbf{z}^\top \mathbf{J}_\mathbf{D}) + \\ & + \beta \mathbf{W}^g_{ij} \Psi'(\Delta \mathbf{D}^0_{ij}) \mathbf{J}^\top_\mathbf{D} \mathbf{J}_\mathbf{D} + \\ & + \lambda \mathbf{W}^g_{ij} \Psi'(\Delta \mathbf{D}_{ij}) \mathbf{J}^\top_\mathbf{D} \mathbf{J}_\mathbf{D} \Bigg] \end{aligned} \quad (25)$$

is the 6×6 Hessian matrix and

$$\begin{aligned} \mathbf{g}_i = \sum_{j\in\mathcal{N}(i)} \Bigg[& \mathbf{W}^a_{ij} \sum_{c=1}^{3} \Psi'(\Delta I^c_{ij}) \mathbf{J}^\top_{I^c_{t+1}} \Delta I^c_{ij} \\ & + \alpha \mathbf{W}^a_{ij} \Psi'(\Delta Z_{ij}) (\mathbf{J}_{Z_{t+1}} - \mathbf{z}^\top \mathbf{J}_\mathbf{D})^\top \Delta Z_{ij} \\ & + \beta \mathbf{W}^g_{ij} \Psi'(\Delta \mathbf{D}^0_{ij}) \mathbf{J}^\top_\mathbf{D} \Delta \mathbf{D}^0_{ij} \\ & + \lambda \mathbf{W}^g_{ij} \Psi'(\Delta \mathbf{D}_{ij}) \mathbf{J}^\top_\mathbf{D} \Delta \mathbf{D}_{ij} \Bigg] \end{aligned} \quad (26)$$

is the 6×1 gradient.
We then update:

$$\begin{aligned} \Delta\mathbf{\Theta} &= [\Delta\boldsymbol{\theta}^\top_1, \Delta\boldsymbol{\theta}^\top_2, \ldots, \Delta\boldsymbol{\theta}^\top_N]^\top \\ \mathbf{\Theta}^{k+1} &= \mathbf{\Theta}^k + \Delta\mathbf{\Theta} \end{aligned} \quad (27)$$

A. 2. Optimizing the Correspondence Field $\mathbf{Y}$

This component is straightforward, as we must solve:

$$\mathbf{Y}^{k+1} = \arg\min_\mathbf{Y} E_G(\mathbf{Y}^\mathbf{k}, \mathbf{\Theta}^{k+1}) \quad (28)$$

In practice, this optimization can be performed in parallel for all sites $\mathbf{y}_i \in \mathbf{Y}$. We use a second order, damped Newton trust-region method.

Acknowledgements. This work was supported in part by CNCS-UEFISCDI under CT-ERC-2012-1, PCE-2011-3-0438, and JRP-RO-FR-2014-16.

References

[1] D. Banica and C. Sminchisescu. Second-order constrained parametric proposals and sequential search-based structured prediction for semantic segmentation in RGB-D images. In *Proceedings of Computer Vision and Pattern Recognition*. IEEE, 2015.

[2] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[3] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Proceedings of Computer Vision and Pattern Recognition*, 2009.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*. Springer, 2004.

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

[6] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3):29–48, 1998.

[7] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571. IEEE, 2013.

[8] S. Hadfield and R. Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2290–2295. IEEE, 2011.

[9] E. Herbst, X. Ren, and D. Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2276–2282. IEEE, 2013.

[10] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3), 1981.

[11] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE, 2007.

[12] H. S. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8), 2013.

[13] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1721–1728. IEEE, 2013.

[14] A. Letouzey, B. Petit, E. Boyer, et al. Surface flow from depth and color images. In *British Machine Vision Conference*, 2011.

[15] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[16] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of European Conference on Computer Vision*, 2012.

[17] J. Quiroga, T. Brox, F. Devernay, and J. Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *Computer Vision–ECCV 2014*, pages 567–582. Springer, 2014.

[18] J. Quiroga, F. Devernay, and J. Crowley. Scene flow by tracking in intensity and depth data. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 50–57. IEEE, 2012.

[19] J. Quiroga, F. Devernay, and J. L. Crowley. Local/global scene flow estimation. In *ICIP-IEEE International Conference on Image Processing*, 2013.

[20] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[21] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015.

[22] H. Spies, B. Jahne, and J. L. Barron. Dense range flow from depth and intensity data. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 131–134. IEEE, 2000.

[23] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[24] D. Sun, E. B. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1768–1775. IEEE, 2012.

[25] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 722–729. IEEE, 1999.

[26] C. Vogel, K. Schindler, and S. Roth. 3d scene flow estimation with a rigid motion prior. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011.

[27] C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1377–1384. IEEE, 2013.

[28] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. *Efficient dense scene flow from sparse or dense stereo data*. Springer, 2008.

[29] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1385–1392. IEEE, 2013.

[30] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic huber-l1 optical flow. In *British Machine Vision Conference*, 2009.

[31] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(9), 2012.

[32] S. Xu, D. Honegger, M. Pollefeys, and L. Heng. Real-time 3d navigation for autonomous vision-guided MAVs. In *IROS 2015 (IEEE/RSJ International Conference on Intelligent Robots and Systems)*.