2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

RGB-D Flow: Dense 3-D Motion Estimation Using Color and Depth

Evan Herbst Xiaofeng Ren Dieter Fox

Abstract— **3-D motion estimation is a fundamental problem that has far-reaching implications in robotics. A scene flow formulation is attractive as it makes no assumptions about scene complexity, object rigidity, or camera motion. RGB-D cameras provide new information useful for computing dense 3-D flow in challenging scenes. In this work we show how to generalize two-frame variational 2-D flow algorithms to 3-D. We show that scene flow can be reliably computed using RGB-D data, overcoming depth noise and outperforming previous results on a variety of scenes. We apply dense 3-D flow to rigid motion segmentation.**

I. Introduction

The ability to detect and estimate fine-grained motion is a fundamental capability for many robotic applications, including person and object tracking, gesture recognition, and object manipulation. The difficulty of motion estimation strongly depends on the assumptions made on scenes and object movements. For instance, if objects of interest can be segmented from the scene easily at all times, tracking can be performed by associating segments in one frame with segments in the next. This has led to robust techniques for problems such as person tracking [1] and manipulator tracking [2]. Assuming that the number of rigid objects moving through the scene is small and known, moving objects can be segmented and tracked via scene differencing and matching, as has been demonstrated for 2-D laser scans [3].

For more complex motion scenarios, robotics researchers often resort to feature-based approaches, where either distinctive markers are placed on the moving objects or it is assumed that the moving objects have color features that can be tracked over time. Recently, such approaches have yielded impressive results on problems such as learning kinematic models of articulated objects [4], [5]. However, their dependence on point features limits the applicability of these techniques to textured objects only.

In this paper we investigate motion estimation in its most general form: we make no assumption about the number of moving objects, whether they are rigid, whether they have visual texture, or whether the camera moves. We introduce RGB-D Flow, a scene flow algorithm that combines dense depth and color information to estimate the 3-D motion of each point in an RGB-D frame. Many existing RGB-D flow algorithms make simplifying assumptions, e.g. using point features [6], [7] or linearized flow constraints [8]). RGB-D Flow builds on successful optical flow algorithms that use robust norms, nonlinearized data constraints and variational solutions [9]. We extend recent optical flow energy formulations to produce dense 3-D flow.

We show motion estimation results on a variety of scenes containing human hands and bodies, robot arms and multiple moving objects. Our algorithm computes dense motion accurately and robustly, handling both small and large motions and overcoming sensor noise. We can compute fine-grained motion on small structures such as fingers noticeably better than previous work. We also apply dense flow to rigid motion segmentation, where we use flow to segment multiple objects in an active vision scenario in which a robot pushes surfaces to discover objects.

This paper is organized as follows. In sec. III we introduce optical flow, scene flow and our objective function. We discuss using scene flow to perform motion segmentation in sec. IV. Sec. V shows results for flow estimation and segmentation, and we conclude in sec. VI.

II. Related Work

The most common types of dense motion estimation in computer vision are optical flow and scene flow. *Optical flow* is apparent motion (translation) in the image plane; *scene flow* is translation in 3-D. Both are usually estimated as vector fields on the pixel grid. Optical flow is an extensively studied problem that goes back to Horn and Schunck [10] (global solution) and Lucas and Kanade [11] (local solution). One development of note is that of Brox et al. [9], which introduces a variational optimization in which approximations commonly made in flow estimation are pushed to an inner loop to improve accuracy. Another is that of the "combined local and global" approach of Bruhn et al. [12]. Brox and Malik [13] incorporate local descriptor matching to improve differential optical flow for large motion. We will extend variational optical flow to the RGB-D case, using both color and depth reasoning.

The term "scene flow" was introduced by Vedula et al. [14], who estimate 3-D flow from color and depth data by first estimating optical flow, then using a first-order approximation of the depth map. Spies et al. [15] introduce a depth analog to brightness constancy that they called the range flow constraint. Other notable scene flow techniques are that of Huguet and Devernay [16], who add explicit occlusion reasoning for pixels, and that of Vogel et al. [17], who add a local rigidity term. These papers computed scene flow from stereo pairs, from which only sparse depth measurements can be extracted with any accuracy. There are few examples of dense depth flow before the arrival of the Kinect, one example being Lukins and Fisher [18], who add color to depth flow in the Lucas-Kanade framework.

E. Herbst and D. Fox are with the University of Washington Department of Computer Science & Engineering, Seattle, WA 98195. X. Ren is with ISTC Pervasive Computing, Intel Labs, Seattle, WA 98195.

With the introduction of dense depth sensors, there has been growing interest in computing flow combining color and depth. Letouzey et al. [6] regularize flow computation using the Laplacian of the graph of points in 3-D rather than the image Laplacian. Their method is designed for large motions, not for objects close in space. Quiroga et al. [7] track regions in color and depth, essentially extending the KLT feature tracker [19] to 3-D. This has the same shortcomings that the Lucas-Kanade optical flow approach does. Hadfield and Bowden [20] use particle filtering in 3-D to estimate the surface in each frame. Sophisticated resampling is necessary to assure that the surface is densely represented at all times. Gottfried et al. [8] use a variational approach with 2-D regularization. Major differences are that we use robust convex norms, avoid early linearization, and carry out more evaluation. The work of Wedel et al. [21] is the most similar to ours, differing mainly in the regularization and in the dependence of various weights on depth values.

Motion segmentation in point clouds is harder than in color images alone due to irregular sampling and the high noise levels of depth sensors. One recent approach is that of Van de Ven et al. [3], who run an expensive belief propagation algorithm on a conditional random field to perform data association, motion estimation and segmentation. Object segmentation is a key to many problems such as object discovery [22] and manipulation [5].

III. Dense Motion Estimation

We consider estimating the 3-D motion of points in the scene using two RGB-D frames. We want to be able to estimate motion for untextured surfaces. Optical flow and scene flow are the most common formulations. *Optical flow* estimates translation in the image plane: each pixel location $\vec{x} = (x, y)$ in the domain of the image I is assigned a motion vector $\vec{u} = (u, v)$ in the plane. u and v are measured in pixels. *Scene flow* estimates translation in 3-D at each pixel: the scene flow maps $\vec{x} \in I$ to $\vec{u}(\vec{x}) = (u, v, w) \in \mathbb{R}^3$. u and v, parallel to the image plane, are measured in pixels; w, orthogonal to the image plane, is measured in meters.

The scene flow formulation is attractive in many ways: (1) it does not make any assumption about the scene motion, making it capable of handling moving cameras, multiple moving objects as well as non-rigid deformations; (2) it does not rely on the availability of identifiable sparse features such as SIFT; (3) it generates motion at all the points in the scene, which is useful for motion segmentation and dense tracking tasks. While differential formulations of scene/optical flow require small motion, reasonably large motions can be recovered through the use of a scale-space pyramid, computing flow vectors from coarse to fine, and warping images across levels.

Our flow formulation is based largely on that of Brox et al. [9]. This is a *variational* method, which optimizes a global energy functional E of the flow field:

$$\vec{u}^* = \operatorname{argmin} E(\vec{u}) = \operatorname*{argmin}_{\vec{u}} \int_{\vec{x} \in I} F(\vec{x}, \vec{u}(\vec{x})) d\vec{x}. \quad (1)$$

Usually F is composed of *data terms*, which match image data between two or more images I_i, and *smoothness terms*, which regularize the problem. We include data terms for color and depth as well as smoothness terms.

A. Color Consistency

All optical flow methods to date try to enforce *brightness constancy*, the idea that the color of an object is constant over time. For a pair of images I_1 and I_2, this can be written as

$$\Delta_t I(\vec{x}) = I_2(\vec{x} + \vec{u}(\vec{x})) - I_1(\vec{x}) = 0 \quad (2)$$

Usually brightness constancy is optimized in its linearized form by approximating with a first-order Taylor expansion to make the constraint linear in $\vec{u}$:

$$\Delta_t I \approx I_2 + \nabla I_2 \cdot \vec{u} - I_1. \quad (3)$$

Eqn. 3 is also called the *optical flow constraint equation* (OFCE). The experience of the optical flow community has been that nonlinearized brightness constancy, as used by [9], leads to more accurate flow than a linearized version. Therefore we use a nonlinearized color consistency term.

The classical formulation of optical flow would use a quadratic penalty (L_2 norm) on the equation above, as adopted in a recent adaptation of optical flow to RGB-D data [8]. In our approach, we make use of the recent experience of optical flow researchers to design our data term. First, instead of the L_2 norm, we use a robust data term that uses the Charbonnier penalty

$$\psi(a^2) = \sqrt{a^2 + \epsilon^2}, \quad (4)$$

a convex approximation of the L_1 norm. This deals with outliers much better than does the quadratic penalty. Second, we use the constancy of both brightness and gradients, as gradients are often more robust than brightness values under real-world illumination. Our color data term is then

$$E_C(\vec{u}) = \psi((\Delta_t I)^2). \quad (5)$$

B. Depth Consistency

The data term for depth looks slightly different from that for color because the measured depth of a physical point is not constant over time. The constraint we want to enforce, therefore, is that change in depth is consistent with observed depths in both frames. Spies et al. [15] propose an analog of the OFCE that they call the *range flow constraint equation* (RFCE), given two depth maps Z_1, Z_2 defined over the same domains as I_1, I_2:

$$Z_2(\vec{x}) + \nabla Z_2^T \vec{u}(\vec{x}) - Z_1(\vec{x}) = 0. \quad (6)$$

We instead use a nonlinearized depth consistency term similar to our color consistency energy:

$$\Delta_t Z = Z_2(\vec{u}(\vec{x})) - Z_1(\vec{x}), \quad (7)$$

$$E_Z(\vec{u}) = \mu(\vec{x}) \psi((\Delta_t Z - w(\vec{x}))^2). \quad (8)$$

The difference between eqn. 5 and eqn. 8 is clear: by definition, when a point $\vec{x}$ moves in 3-D, the depth motion

w is the change of its measured depth. As in the color case, we use the robust Charbonnier penalty.

$\mu(\vec{x})$ weights the depth data term against the color data term. We use a measure of uncertainty in measured depth, which grows with distance from the camera. Specifically, $\mu(\vec{x}) = \frac{\sigma_Z(1)}{\sigma_Z(d)}$, where $\sigma_Z(d)$ is the depth resolution of the stereo sensor around depth d. This is approximately quadratic in d.

C. Regularization

Regularization is necessary for variational flow computation. The flow problem is underconstrained locally; this is commonly known as the *aperture problem*. The optical flow constraint in eqn. 5 has no information on how a point moves perpendicular to the ∇I direction. With regularization, flow algorithms are often capable of producing accurate motion estimates even in regions of uniform appearance where no local signal is present. The regularization term is traditionally quadratic, but a quadratic penalty on the smoothness term leads to many small changes in flow at motion boundaries rather than few large changes, so some recent methods replace this also with a robust penalty.

The most commonly used regularization is an isotropic regularizer that penalizes a high gradient in the flow [9]. Again using the Charbonnier penalty, we can define an isotropic regularization term for RGB-D data as

$$E_S(\vec{u}) = \psi(|\nabla u|^2 + |\nabla v|^2 + \beta(\vec{x})|\nabla w|^2), \qquad (9)$$

where we adapt the usual optical flow smoothness term to the scene flow case by scaling the depth term to trade off meters against pixels: $\beta(\vec{x}) = f^2$, f the focal length of the color camera, into whose coordinate frame we assume the depth maps have been put.

When optimizing eqn. 9, we will use the gradient of the flow field only via the Laplacian L of the flow field. This is traditionally calculated using an image Laplacian that weights all pixels equally. While an isotropic regularizer can work well for the optical flow case, we find that it's beneficial to use an anisotropic smoothing that takes into account the local gradient signals: image gradient, depth difference, and surface orientation difference. For the case of color-only data, one commonly used anisotropic regularizer is that of Nagel and Enkelmann [23], which reduces smoothing across edges. The RGB-D flow method of Letouzey et al. [6] substitutes for L a *graph* Laplacian over the point cloud of the scene, with weights decreasing with increase in depth difference:

$$l(\vec{x}, \vec{y}) = e^{-c(z(\vec{x})-z(\vec{y}))^2}, \ \vec{y} \in N(\vec{x}), \qquad (10)$$

where $N(\vec{x})$ is the 3-D neighbor points of $\vec{x}$. (See [6] for how to build L from l, which is there called w.) Both these methods optimize an objective including the quadratic smoothness term $E_S(\vec{u}) = \vec{u}^T L \vec{u}$, where $\vec{u}$ stands for the concatenation of the flows $\vec{u}(\vec{x})$ for all pixels $\vec{x}$. While our objective takes a different form (we use the Laplacian in the style of [12] during optimization), we also use the flow gradient only via the Laplacian. One regularizer with which we experiment takes a form similar to Letouzey's, making use also of color and surface normal. The Laplacian matrix for this regularizer is defined using

$$l(\vec{x}, \vec{y}) = 1 - \max \begin{pmatrix} e^{-c_z|z(\vec{x})-z(\vec{y})|^2} \\ e^{-c_c|rgb(\vec{x})-rgb(\vec{y})|^2} \\ e^{-c_n|1-n(\vec{x})\cdot n(\vec{y})|^2} \end{pmatrix} \qquad (11)$$

where z is the depth, rgb the RGB color value, and n the surface normal. This term, inspired by Pb [24], computes an approximate probability of an object boundary between a pair of adjacent points to avoid smoothing across boundaries. Other sources of information for such a regularizer could be an improved Pb, perhaps trained on environment-specific data, or motion segmentation results from past video frames.

Due to the max operator, this regularizer likes to smooth between points unless there is overwhelming evidence that they represent different objects. A more correct formulation would use depth-dependent color and surface normal terms in the style of [22]; we have used the simpler formulation here for expository purposes.

Regularization that downweights the connection between flow at different points can leave some points that have no local signal with no sources of information if they also have weak connections to their neighbors. Therefore we include a small zero-motion bias in our objective. We use a quadratic penalty in order to penalize large flows while penalizing small ones as little as possible, because empirically we find that when we use robust data and smoothness penalties, only large flow values indicate such underconstrained points. The full objective is then

$$E(\vec{u}) = \int_{\vec{x} \in I} (E_C(\vec{u}) + E_Z(\vec{u}) + \alpha E_S(\vec{u}) + \gamma E_B(\vec{u}))\, d\vec{x}, \qquad (12)$$

where $E_B(\vec{u}) = |\langle 1, 1, \eta(\vec{x})\rangle \cdot \vec{u}|^2$ is the flow magnitude penalty, with $\eta(\vec{x}) = \frac{f}{z}$ balancing pixels against meters. We have abused notation to save space: inside the integral $\vec{u}$ is actually $\vec{u}(\vec{x})$.

D. Variational Solution of RGB-D Flow

Our solution of RGB-D FLOW is largely motivated by and adapted from proven variational optical flow techniques such as those of Brox et al. [9]. We have three Euler-Lagrange equations, one for each component of the flow $\vec{u} = (u, v, w)$. The Euler-Lagrange equation for u is:

$$\begin{aligned}
0 =& \frac{\partial F}{\partial u} - \frac{\partial}{\partial x}\frac{\partial F}{\partial \frac{\partial u}{\partial x}} - \frac{\partial}{\partial y}\frac{\partial F}{\partial \frac{\partial u}{\partial y}} \\
=& \psi'((\Delta_t I)^2) 2(\Delta_t I)\frac{\partial I_2}{\partial x}(\vec{x}+\vec{u}) \\
&+ \mu(\vec{x})\psi'((\Delta_t Z - w(\vec{x}))^2) 2(\Delta_t Z - w(\vec{x}))\frac{\partial Z_2}{\partial x}(\vec{x}+\vec{u}) \\
&- \frac{\partial}{\partial x}(\alpha\psi'(|\nabla\vec{u}|^2)2\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(\alpha\psi'(|\nabla\vec{u}|^2)2\frac{\partial u}{\partial y}) + 2\gamma u \\
&\text{[rewrite } u \to u + du\text{]} \\
\approx& \psi'_D \frac{\partial I_1}{\partial x}(\vec{x})(\Delta_t I + \frac{\partial I_1}{\partial x}(\vec{x})du + \frac{\partial I_1}{\partial y}(\vec{x})dv) \\
&+ \mu(\vec{x})\psi'_Z \frac{\partial Z_1}{\partial x}(\vec{x})(\Delta_t Z + \frac{\partial Z_1}{\partial x}(\vec{x})du + \frac{\partial Z_1}{\partial y}(\vec{x})dv - dw) \\
&- \alpha\psi'_S \nabla^2(u + du) + \gamma(u + du)
\end{aligned}$$

where we define

$$\psi'_D = \psi'((\Delta_t I)^2),$$
$$\psi'_Z = \psi'((\Delta_t Z - w(\vec{x}))^2),$$
$$\psi'_S = \psi'(|\nabla \vec{u}|^2).$$

Then the SOR update step, with ω the SOR parameter, is

$$du \leftarrow (1-\omega)du + \omega \frac{num}{denom}$$
$$\begin{aligned} \text{with } num = &-\psi'_D \frac{\partial I_1}{\partial x}(\vec{x})(\Delta_t I + \frac{\partial I_1}{\partial y}(\vec{x})dv) \\ &- \mu(\vec{x})\psi'_Z \frac{\partial Z_1}{\partial x}(\vec{x})(\Delta_t Z + \frac{\partial Z_1}{\partial y}(\vec{x})dv - dw) \\ &+ \alpha\nabla^2(u) + \alpha\nabla_N^2(du) - \gamma u \\ denom = &\psi'_D \frac{\partial I_1}{\partial x}(\vec{x})^2 + \mu(\vec{x})\psi'_Z \frac{\partial Z_1}{\partial x}(\vec{x})^2 - \alpha\nabla_P^2(du) + \gamma \end{aligned}$$

where we approximate the image Laplacian of du $\nabla^2(u)$ (weighted by ψ'_S) using Bruhn et al.'s [12] discretization and let ∇_P^2 and ∇_N^2 stand for the Laplacian terms that do and do not contain du, respectively. The derivation for v is analogous and omitted.

The Euler-Lagrange equation for w is

$$\begin{aligned} 0 = &\frac{\partial F}{\partial w} - \frac{\partial}{\partial x}\frac{\partial F}{\partial \frac{\partial w}{\partial x}} - \frac{\partial}{\partial y}\frac{\partial w}{\partial y} \\ &\text{[making the approximation that } \frac{\partial \beta}{\partial \vec{x}} \approx 0 \text{]} \\ = &\mu(\vec{x})\psi'((\Delta_t Z - w(\vec{x}))^2)2(\Delta_t Z - w(\vec{x}))(-1) + 2\gamma\eta(\vec{x})w \\ &- \frac{\partial}{\partial x}(\alpha\psi'(|\nabla\vec{u}|^2)2\beta(\vec{x})\frac{\partial w}{\partial x}) - \frac{\partial}{\partial y}(\alpha\psi'(|\nabla\vec{u}|^2)2\beta(\vec{x})\frac{\partial w}{\partial y}) \\ &\text{[rewrite } w \rightarrow w + dw\text{]} \\ &\approx \mu(\vec{x})\psi'_Z(\Delta_t Z + \frac{\partial Z_1}{\partial x}(\vec{x})du + \frac{\partial Z_1}{\partial y}(\vec{x})dv - dw)(-1) \\ &- \alpha\psi'_S\nabla^2(w + dw) + 2\gamma\eta(\vec{x})(w + dw). \end{aligned}$$

The SOR update step for w is

$$dw \leftarrow (1-\omega)dw + \omega\frac{num}{denom}$$
$$\begin{aligned} \text{with } num = &\mu(\vec{x})\psi'_Z(\Delta_t Z + \frac{\partial Z_1}{\partial x}(\vec{x})du + \frac{\partial Z_1}{\partial y}(\vec{x})dv) \\ &+ \alpha\nabla^2(w) + \alpha\beta(\vec{x})\nabla_N^2(dw) - \gamma\eta(\vec{x})w \\ denom = &\mu(\vec{x})\psi'_Z - \alpha\beta(\vec{x})\nabla_P^2(dw) + \gamma\eta(\vec{x}). \end{aligned}$$

E. Practical Issues

We optimize over a scale-space pyramid. At each scale we use the fixed-point optimization suggested by [9] to iteratively linearize this system of equations as a function of a small change $d\vec{u}$ in the flow field, and use a parallel sparse linear solver. The smoothing used to downsample the image and upsample the flow field is done in 2-D, making it not ideal for scene flow computation since neighboring pixels are not always close in 3-D. However, if we assume small total motion this is still a reasonable approximation.

The most variable factor in the speed of flow estimation is how long the optimizer takes to converge. Convergence is affected by the number of variables being estimated, by the amount of smoothing (smoothing slows convergence), and by the strength of the flow magnitude penalty (we find that a large penalty slows convergence).

The biggest factor in the numerical stability of optimization is the magnitude of derivatives of the depth map. In general the largest depth-map derivatives are much larger than the largest image derivatives; image edges tend to be blurred over several pixels, whereas depth discontinuities reported by the Kinect do not. (Measured depth edges are noisy in different ways.) Large derivatives break the smoothness assumptions made when the depth data term is linearized, which happens eventually even in methods that use "nonlinearized" data terms (in an inner loop). At present we fight numerical instability by limiting the magnitude of the depth derivative used in optimization. This slows convergence slightly but avoids having the flow bounce back and forth across the true value each time image gradients are recalculated.

IV. Object Segmentation from Motion

One motivation for this work is autonomous exploration of indoor scenes. The robot, given an environment full of objects, should be able to actively determine where all objects are. Some of this can be accomplished with object instance recognition, but in general some objects will be previously unseen and some will be heavily occluded, necessitating active vision. Once the robot has manipulated part of the scene, for example by pushing a surface, it needs to perform motion segmentation to estimate object boundaries. We previously demonstrated a system for rigid object segmentation from motion that works by comparing views of the scene before and after the manipulation [25]. The method works by sampling a very large number of sets of possible point correspondences, fitting a rigid motion to each and very quickly roughly evaluating these motions, discarding those that do not fit many scene points. It then scores in more detail the best of all the candidate motions, ultimately choosing the best few to use as labels in a Markov random field (MRF) over scene points. We run inference in this MRF to add spatial smoothness to our segmentation. That algorithm is applicable to any two scenes, regardless of motion magnitude, but is slow due to the need for a very large number of samples, and its method for selecting a small number of candidate motions from a long list is error-prone.

We find that the candidate selection step is the hardest part of the motion segmentation process, partly due to the large number of similar motions that are proposed by almost any sampling-based approach. If seed points for motion fitting are densely sampled in space, many tuples of correspondences can be obtained by shifting each point of another tuple slightly in a given direction, and so will suggest very similar motions. In order to avoid missing the true motion, our previous work [25] allows each point in one scene to correspond to any of hundreds in the other scene. Since then, to make motion candidate selection faster and more robust, we have experimented with limiting the set of possible point correspondences for each pixel to one. Two ways to produce a single correspondence per point are rigid point cloud alignment—using, e.g., the iterated closest points algorithm (ICP) [26]—and nonrigid motion estimation such as that

provided by optical flow or scene flow. A major limitation of ICP is that it must be run separately for each region that moves differently. That is, it requires that segmentation be known. Therefore in this work we use scene flow to choose a single candidate point correspondence per pixel. Optical or scene flow, while accurate to less than a pixel for some well-behaved datasets, is known to have problems with untextured regions, where flow is determined mostly by smoothness constraints. However, for small motions, such as between consecutive video frames, scene flow is a fast way to find a short list of reasonable-quality point correspondences.

To perform motion segmentation, we run scene flow on two frames and assign each pixel its scene flow as a proposed motion. We run the algorithm of [25] to iteratively fit a rigid motion using RANSAC and remove that motion's inliers from the seed set. Efficiency improvements in RANSAC are possible given the knowledge that each point has at most one correspondence, so this step is faster than it is when used in the general case. We follow the fitting procedure with MAP inference in an MRF to select a single motion per point while enforcing spatial smoothness of labels.

V. Experiments

We demonstrate our scene flow method on a variety of real data relevant to domestic robotics. Fig. 1 shows results for four scene flow algorithms on videos we collected, and fig. 2 shows our results on a frame pair from [8], for comparison to their results. The **Liu+DM** algorithm estimates scene flow by first estimating optical flow from color information only using the implementation of Liu [27] (on which we base our scene flow implementation), followed by sampling the depth map Z_2 at the flowed-to location in the image plane. This is a good baseline and is even slightly more sophisticated than the scene flow method of [14], which only *approximated* the depth map in the depth sampling step. The **RGB-D nonrobust** algorithm is our method modified to use quadratic penalties on the data and smoothness terms. This method tends to oversmooth both the optical and z-flow estimates, demonstrating the reason for using robust penalty functions in flow estimation. We then show results from running RGB-D Flow with isotropic smoothing and with the anisotropic smoothing of eqn. 11. Results of our algorithm with isotropic smoothing should be similar to results of running [21]. The parameters of each method were adjusted manually; in most cases the values are the same. The first row of each column shows the two color images. Each column shows estimated flow in xy and in z separately. The xy-flows are shown in the Middlebury color coding [28]. The z-flows are shown using the x axis of the same coding. Within each column all images are coded using the same maximum flow. For ease of interpretation we include blowups of the xy-flow fields visualized with arrows rather than Middlebury colors.

One of the frame pairs in fig. 1 shows human full-body motion. This is intended as a comparison to a method similar to that of Letouzey et al. [6] on data similar to those shown in their paper. Here RGB-D Flow with anisotropic smoothing is similar to Letouzey's method but uses a more sophisticated regularization (and robust penalty functions).

In fig. 3 we show segmentations obtained using the results of RGB-D Flow with anisotropic regularization on the two of these frame pairs that involve rigid motion. In both we obtain the correct number of motions and good segmentations. The shampoo bottle in the second scene doesn't move and so is identified as background.

Fig. 2 demonstrates our method on the data used by Gottfried et al. [8], who published their data. They didn't register the color and depth images in the published data, so we approximately registered them by shifting the depth images in 2-D by a manually chosen number of pixels so that the alignment looks reasonable. Also, these data give depth in unspecified units, so our results may be worsened slightly due to our not adjusting the depth uncertainty model to use a unit other than millimeters.

Our method results in similar or better xy- and z-flow to the optical-flow-based method on all videos shown. It is clear also that using quadratic penalties worsens flow estimation. Anisotropic smoothing appears helpful on most scenes. On the first frame pair of fig. 1, the outline of the box is noticeably cleaner with anisotropic than with isotropic smoothing, and on the second frame pair, the outline of the robot arm is also very clean. On the third frame pair, anisotropy makes the outlines of the bottom of the ball and the bottom of the person clearer, although it's not clear whether this improves flow accuracy on the person's left leg. On the frame pair of fig. 2, the anisotropic regularizer makes the hand outline clearer particularly around the pointer finger. We judge that anisotropy is usually an improvement on the close- and medium-range scenes in which we are interested.

The runtime of RGB-D Flow varies between 8 and 30 seconds per frame pair at 320x240 resolution; the variance is due mostly to the varying amounts of smoothing necessary in different scenes, since we optimize to convergence rather than to a fixed number of iterations. The implementation is on the CPU except for the sparse linear solver [29]. We experimented with speeding up the algorithm by instead optimizing with SOR and limiting its iterations, as was done by [21], and did not find this to provide enough smoothing on our data.

Videos showing scene flow for the whole of the second sequence shown in fig. 1 and others are available from `https://www.cs.washington.edu/node/8501/`.

VI. Conclusions

We have shown that using dense depth information improves estimation of scene flow, the incremental 3-D translation of scene points. Further, scene flow can be used for motion segmentation, even for objects without texture.We have developed a state-of-the-art scene flow estimation algorithm that uses RGB-D data.

The largest problem with the current algorithm is its lack of occlusion handling, which forces pixels that disappear to take on large estimated movement in order to satisfy the

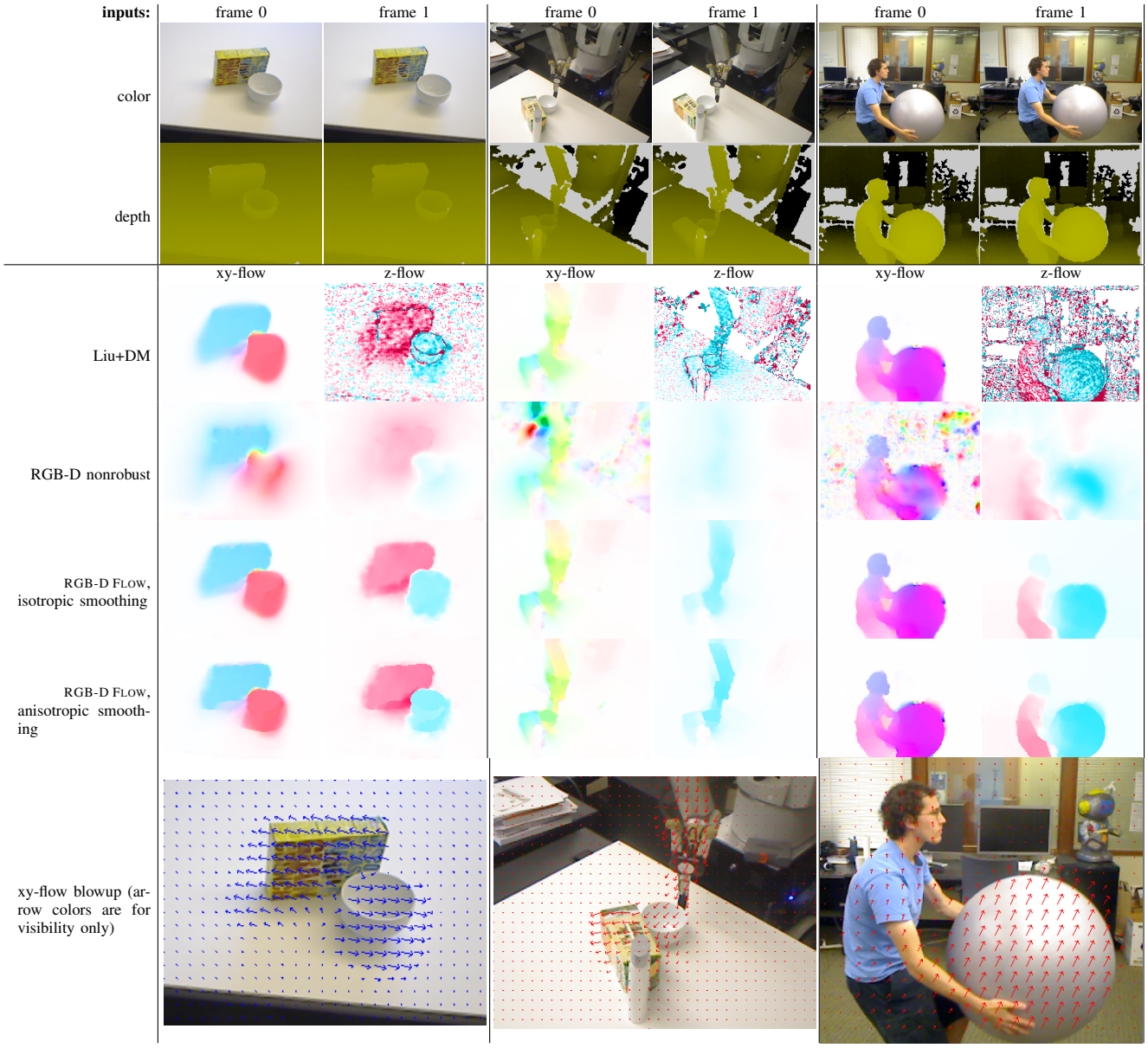


Fig. 1: scene flow results on three frame pairs: two rigid objects moving; two rigid objects and a robot arm moving; full-body human motion. Gray pixels in depth images show invalid readings. First column for each frame pair is xy-flow; second column is z-flow. Shown in the Middlebury optical flow color coding (for z-flow we use colors along the x axis) with maximum flow (10, 5, and 10) pixels in xy respectively, and all with maximum .01 m in z.

data constraints. This is visible in the results for the human-motion frame pair of fig. 1, where pixels around the edges of the ball are assigned large flows. We are working with a formulation that adds occlusion handling to the scene flow optimization in a way that makes use of our RGB-D data, but this objective introduces interactions between variables that aren't present in the formulation given in this paper.

Representing the uncertainty of estimated flow would speed the sampling step of rigid motion fitting by allowing us to preferentially select seed points with low flow uncertainty. Work such as [30], [31], [32] has suggested multiple ways to measure the uncertainty of optical flow; these should be extensible to scene flow.

We plan to use scene flow for interactive object segmentation and manipulation. When objects are being moved continuously, differential motion estimation between consecutive frames should provide an improvement over the motion estimation of our previous work [25], which required two static scenes.

VII. Acknowledgments

This work was funded in part by the Intel Science and Technology Center for Pervasive Computing and by ONR MURI grant N00014-09-1-1052.

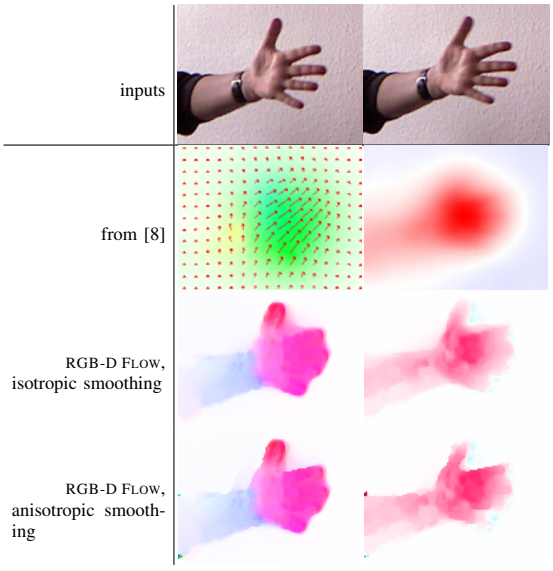
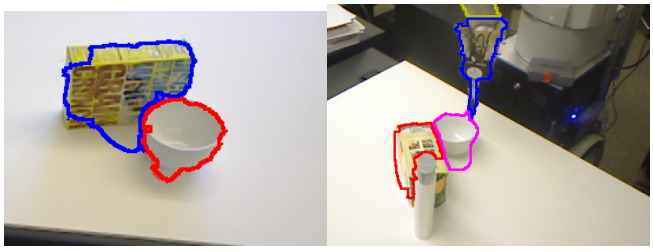


Fig. 2: scene flow results on a frame pair from [8]. First column: xy-flow; second column: z-flow. Shown in the Middlebury optical flow color coding (for z-flow we use colors along the x axis) with maximum flow 4 pixels in xy, .005 m in z. For comparison we include part of the first row of fig. 3 of that paper (the best-looking of their results). The color scheme, of course, differs from ours.



(a) (b)

Fig. 3: rigid object segmentation results on two frame pairs from fig. 1, with different motions shown in different colors.

REFERENCES

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2011, p. 3.

[2] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in hand 3d object modeling," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 9, 2011.

[3] J. van de Ven, F. Ramos, and G. Tipaldi, "An integrated probabilistic model for scan matching, moving object detection and motion estimation," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2010.

[4] J. Sturm, V. Pradeep, C. Stachniss, C. Plagemann, K. Konolige, and W. Burgard, "Learning kinematic models for articulated objects," in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[5] D. Katz and O. Brock, "Interactive segmentation of articulated objects in 3-d," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2011.

[6] A. Letouzey, B. Petit, and E. Boyer, "Scene flow from depth and color images," in *British Machine Vision Conference (BMVC)*, 2011.

[7] J. Quiroga, F. Devernay, and J. Crowley, "Scene flow by tracking in intensity and depth data," in *CVPR Workshops*, 2012.

[8] J. Gottfried, J. Fehr, and C. Garbe, "Computing range flow from multimodal kinect data," in *International Symposium on Visual Computing (ISVC)*, 2011.

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

[10] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, 1981.

[11] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Image Understanding Workshop*, 1981.

[12] A. Bruhn, J. Weickert, and C. Schnoerr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International Journal of Computer Vision*, 2005.

[13] T. Brox and J. Malik, "Large-displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

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

[15] H. Spies, B. Jaehne, and J. Barron, "Range flow estimation," 2002.

[16] F. Huguet and F. Devernay, "A variational method for scene flow estimation from stereo sequences," in *International Conference on Computer Vision (ICCV)*, 2007.

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

[18] T. Lukins and R. Fisher, "Colour-constrained 4-d flow," in *British Machine Vision Conference (BMVC)*, 2005.

[19] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University Technical Report CMU-CS-91-132, Tech. Rep., 1991.

[20] S. Hadfield and R. Bowden, "Kinecting the dots: Particle-based scene flow from depth sensors," in *International Conference on Computer Vision (ICCV)*, 2011.

[21] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers, "Stereoscopic scene flow computation for 3d motion understanding," *International Journal of Computer Vision*, 2011.

[22] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward object discovery and modeling via 3-d scene comparison," in *IEEE International Conference on Robotics & Automation (ICRA)*, 2011.

[23] H. Nagel and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1986.

[24] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color and texture cues," 2004.

[25] E. Herbst, X. Ren, and D. Fox, "Object segmentation from motion with dense feature matching," in *ICRA Workshop on Semantic Perception, Mapping and Exploration*, 2012.

[26] P. Besl and N. McKay, "A method for registration of 3-d shapes," 1992.

[27] C. Liu, W. Freeman, E. Adelson, and Y. Weiss, "Human-assisted motion annotation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[28] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[29] NVIDIA Corporation. Cusp. https://developer.nvidia.com/cusp/.

[30] O. Nestares, D. Fleet, and D. Heeger, "Likelihood functions and confidence bounds for total-least-squares problems," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[31] A. Bruhn and J. Weickert, "A confidence measure for variational optical flow methods," Saarland University Mathematics Department technical report 106, Tech. Rep., 2004.

[32] C. Kondermann, R. Mester, and C. Garbe, "A statistical confidence measure for optical flows," in *European Conference on Computer Vision (ECCV)*, 2008.