Fast Odometry and Scene Flow from RGB-D Cameras based on Geometric Clustering

Mariano Jaimez[1,2], Christian Kerl[2], Javier Gonzalez-Jimenez[1] and Daniel Cremers[2]

Abstract— **In this paper we propose an efficient solution to jointly estimate the camera motion and a piecewise-rigid scene flow from an RGB-D sequence. The key idea is to perform a two-fold segmentation of the scene, dividing it into geometric clusters that are, in turn, classified as static or moving elements. Representing the dynamic scene as a set of rigid clusters drastically accelerates the motion estimation, while segmenting it into static and dynamic parts allows us to separate the camera motion (odometry) from the rest of motions observed in the scene. The resulting method robustly and accurately determines the motion of an RGB-D camera in dynamic environments with an average runtime of 80 milliseconds on a multi-core CPU. The code is available for public use/test.**

I. INTRODUCTION

The joint estimation of the motion of a camera and the motion of the objects it observes is a problem of great interest with numerous applications in robotics, computer vision and beyond: tracking and mapping in dynamic scenarios, manipulation of fast-moving objects, or autonomous navigation are a few prominent examples. However, it is also a complex and computationally demanding problem that has not been properly solved yet. On the one hand, great progress have been made in visual odometry under the assumption of static or quasi-static environments [1]–[3], but the performance of these methods deteriorates when the number of pixels observing non-static parts becomes significant. On the other hand, scene flow (motion of the scene objects) is often estimated as the non-rigid velocity field of the observed points with respect to the camera relative position. This approach alone does not yield the camera motion because all points in the scene are treated equally and, therefore, static and non-static regions are indistinguishable when the camera moves. Moreover, the scene flow estimation tends to be computationally expensive, and most existing approaches require between several seconds and few minutes to align just a pair of images, which prevents them from being used in practice.

In this paper we present a new method to estimate both the motion of an RGB-D camera and the scene flow. Our approach relies on a two-fold segmentation of the scene. First, the scene is divided into geometric clusters by running K-Means on the 3D coordinates of the observed points. These clusters are treated as rigid bodies and are mostly exploited for the scene flow estimation, which greatly reduces its computational cost without sacrificing much accuracy. Second, the scene is also segmented into static and moving parts. The static regions (background) are used to derive the camera motion while the scene flow is estimated for the moving parts. To increase robustness, we propagate the background segmentation through time since static and moving parts of the scene are likely to be consistent along the image sequence.

We perform an extensive evaluation of our approach, comparing it with several state-of-the-art methods in visual odometry and scene flow estimation. Results show that our approach estimates the camera motion more accurately, in particular when the scene is highly dynamic, and ranks second in the scene flow evaluation. Above all, its main advantage is its significantly lower runtime, of about 80 milliseconds running on multiple CPU cores at QVGA resolution (several orders of magnitude faster than most scene flow algorithms). For this reason, it can be applied online, a feature that existing approaches lack.

The code, together with the demonstration video, can be found here:

`http://mapir.isa.uma.es/work/Joint-VO-SF`

II. RELATED WORK

Thus far, visual odometry and scene flow estimation are two highly related problems that are usually addressed separately because of their intrinsic complexity. Though some joint solutions have been reported, they typically lack precision and efficiency. Next, we review some of the latest proposals for these problems.

Traditionally, visual odometry approaches have exploited sparse feature correspondences to estimate the camera motion [4]. While they are resilient to large numbers of outliers, they usually require optimization over multiple frames to achieve accurate camera localization. In general, these methods cannot provide a dense scene flow, but there exist extensions to estimate the motion of multiple rigid objects [5]. However, they require enough feature points to sufficiently constrain the motion, which is not guaranteed for objects projected as small regions on the image. With the advent of consumer RGB-D cameras, which provide dense depth maps at comparably high resolution, dense direct methods gained popularity. Typical cost functions penalize the intensity error [1], [6], inverse or direct depth error [3], [7], point-to-plane error [8], or an alternative error in the feature space

This work has been funded by the Spanish Government under project DPI2014-55826-R, the grant program FPI-MICINN 2012 and the ERC Consolidator Grant *3DReloaded*.

[1] Department of Systems Engineering and Automation, University of Málaga, Spain (emails: {marianojt,javiergonzalez}@uma.es).

[2] Department of Computer Science, Technical University of Munich, Germany (emails: {christian.kerl,cremers}@tum.de).

[9]. The main difference to sparse, feature-based methods is that they do not require explicit correspondences, but rather compute and update them implicitly during the optimization. To achieve robustness against unmodelled effects, these approaches combine multiple cost functions and use robust penalties like Huber, Tukey, or Cauchy [1], [10]. This strategy works well if most of the scene is static and only little portions of the input images observe moving parts, but fails when the moving parts become more significant.

Scene flow has traditionally been estimated with stereo systems [11], but this trend also changed in 2010 with the appearance of affordable RGB-D cameras. Several variational approaches have been proposed [12]–[14] to compute scene flow from RGB-D image pairs, using different norms (weighted L_2 / L_1) for the data and the regularization terms. Jaimez *et al.* [15] presented a real-time implementation using a Primal-Dual solver, which provides good estimates only for small motions. The semi-rigid scene flow proposed in [16] uses 6-DoF representation for the flow and also includes the camera motion into its formulation, which makes it the best candidate for comparisons. However, only the accuracy of the scene flow was evaluated in their paper. Sun et al. [17] presented a probabilistic approach which relies on a depth-based segmentation of the scene. They regularize the estimation process by retrieving a mean rigid-body motion for each layer, and allow for small deviations of the motion field from the layer's mean motion. A big downside is its high runtime: it requires several minutes to align just a pair of images. The smooth piecewise-rigid flow proposed in [18] achieves very accurate flow estimates by jointly segmenting the scene into its rigidly moving parts and computing their underlying motions. However, optimizing for the segments makes it also computationally very expensive. Other approaches focus on the estimation of large motions. SphereFlow [19] also parametrizes the motion field with 6 DoF, and proposes to match corresponding points within a 3D spherical search range instead of using traditional planar patch comparisons. GraphFlow [20] outperforms SphereFlow by looking for and registering sparse correspondences between points in geometric edges, and densify the flow at a later optimization stage. None of these two methods [19] [20] provide information about their runtime.

Recent works on non-rigid 3D reconstruction also estimate the camera motion and a deformation flow for the particular objects they reconstruct. DynamicFusion [21] estimates the camera motion using KinectFusion [8], which is not prepared to handle moving objects. Afterwards, they estimate a set of sparse volumetric rigid transformations to align the moving object to the model, and interpolate between these transformations to obtain the dense warp-field. VolumeDeform [22] follows a similar sequence, estimating the camera motion by aligning sparse color correspondences and computing the dense deformation field associated to the deformable object at the finest resolution. They both provide impressive results for moderate camera motions and deformations, but unfortunately their code is not available. Fusion4D [23] addresses the same problem using multiple static cameras, and therefore only a nonrigid motion field is estimated (odometry is not necessary).

III. Overview of the method

The proposed method to jointly estimate the camera and the scene motions from RGB-D sequences comprises several sequential blocks, as illustrated in Fig. 1. As inputs, a pair of I-D frames (I_1, Z_1) and (I_2, Z_2) is given, where $I_{(.)} : \Omega \to \mathbb{R}$ and $Z_{(.)} : \Omega \to \mathbb{R}$ stand for the intensity and depth images defined on the image domain $\Omega \subset \mathbb{R}^2$. First, the frame (I_1, Z_1) is segmented into N geometric clusters $\boldsymbol{C} = \{C_i, i = 1, ..., N\}$ by applying K-Means to the 3D coordinates of the scene points. Each cluster is considered to behave as a rigid body, which greatly simplifies the scene flow estimation because the motion is estimated cluster-wise instead of pixel-wise (reducing the number of unknowns by 3-4 orders of magnitude). The velocity associated to the each cluster is represented by the 3D twist $\boldsymbol{\xi}_i \in \mathfrak{se}(3)$. Second, an initial guess for the odometry $\boldsymbol{\xi}_R \in \mathfrak{se}(3)$ is computed by minimizing the photometric and geometric residuals within a robust M-estimator. Thus, we obtain the predominant rigid motion of the scene which, save in cases of very dynamic environments, corresponds to the camera motion itself. Subsequently, this estimate is used to segment the scene into static parts (or background) and moving objects. To this end, the I-D images are warped according to $\boldsymbol{\xi}_R$ and the average residuals per cluster are computed. Only the clusters moving according to $\boldsymbol{\xi}_R$ will have low residuals, and therefore will be segmented as background. Clusters with high residuals after the warping are those whose motion does not coincide with $\boldsymbol{\xi}_R$, and hence are tagged as moving objects. Instead of using a binary segmentation, which would require to set a sharp threshold on the residuals, we use a continuous representation and define $b_i \in [0, 1]$ as the probability of any cluster i to be a moving object. For simplicity, we will use $\boldsymbol{b}$ to refer to the segmentations of all clusters. More details about the background segmentation are given in Section VI.

Once the segmentation is known, it can be used to break the motion estimation process into two separate steps. First, all the clusters that have been tagged as background are used to obtain a more precise odometry $\boldsymbol{\xi}_O$, now excluding the non-static parts. Second, a piecewise rigid scene flow is estimated for the rest of the scene, assuming that each cluster behaves as a rigid body. Last, the background segmentation is recomputed with the new refined odometry and warped to the next frame, leading to $B^T : \Omega \to [0, 1]$. Since moving objects are likely to be moving for more than one frame, and the same applies to still parts, we make use of B^T in the next iteration to obtain segmentations that are consistent through time.

IV. Geometric Clustering

Our proposal to reduce the complexity of a per-pixel estimation of the motion field is to cluster the scene points in sets that will be treated as rigid bodies. Other existing algorithms have employed this strategy as well to obtain a faster (and often more precise) scene flow either by using

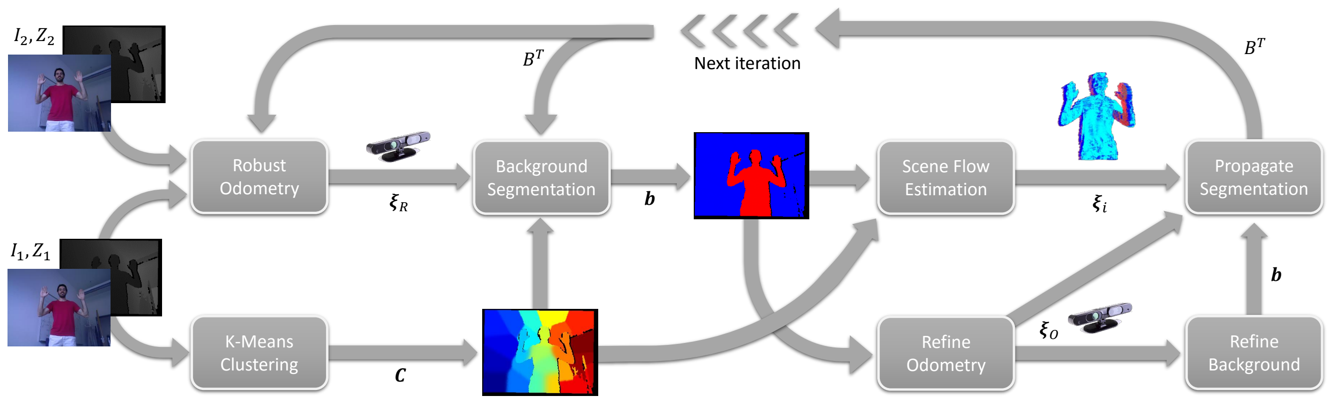

Fig. 1. Schematic representation of the main components of our algorithm. The estimation process starts (left) by reading a pair of RGB-D images. After the process finishes, the background segmentation $\boldsymbol{b}$ is propagated to the next frame according to $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_O$ to be used in the next iteration.

superpixels [24] or K-Means [17] [18]. We follow the idea presented in [18] and compute K-Means clusters based on the 3D coordinates of the observed points. This strategy is advantageous because:

- It has a physical ground, in the sense that close points in space are very likely to belong to the same rigid body.
- It is a very convenient representation when working with image pyramids (coarse-to-fine). Once the clusters are computed for a given image, they can be easily computed for the rest of the pyramid by using the spatial coordinates of the k-means that have already been obtained. This is efficient and also provides a consistent clustering throughout all the image levels.
- It is suitable to propagate information from one frame to the next because the cluster centers can be mapped efficiently from (Z_1, I_1) to (Z_2, I_2) with $\boldsymbol{\xi}_O$ and $\boldsymbol{\xi}_i$.

Two additional steps are performed after obtaining the clusters. First, we build a connectivity graph $G = \{G_{ij},\ i = 1, ..., N,\ j = 1, ..., N\}$ which indicates which clusters are contiguous in space ($G_{ij} = 1$) and which ones are separated ($G_{ij} = 0$). This graph is exploited later on for the background segmentation by fostering contiguous clusters to be segmented similarly (spatial regularization). Moreover, contiguous clusters are smoothed to avoid sharp motion transitions along their boundaries, but this smoothing mostly affects the scene flow estimation and does not play any role in the other modules of our algorithm.

Another important aspect is the number of clusters to consider. Too few leads to very large scene regions which will likely include parts of the scene with different motions. On the other hand, if many clusters are extracted, then they will not contain enough points for their motion to be robustly estimated. We have empirically chosen to use 24 clusters per image, which leads to middle-size clusters that can be homogeneously initialized on the image domain (6×4). More elaborate strategies could be adopted in the future, like using an adaptive number of cluster to fuse redundant regions or to create new ones in areas with high residuals.

The main limitation of our approach is the assumption that each cluster behaves as a rigid body. Since the only criterion to form the clusters is the spatial proximity of points, there can always be clusters which actually contain points from different rigid bodies. For instance, if a person stands close to a wall, there might be some clusters containing points from both the person and the wall. In this case, the estimated motion for those clusters will be the predominant motion between the two (person or wall). This limitation could be alleviated or even solved by increasing the number of cluster and regularizing the motion between them, at the expense of a significantly higher computational cost.

V. Robust Odometry

The odometry is computed by minimizing the photometric and geometric residuals between consecutive RGB-D pairs. The geometric residuals r_Z and the photometric residuals r_I are defined as

$$r_Z^k(\boldsymbol{\xi}) = Z_2(\mathcal{W}(x^k, \boldsymbol{\xi})) - \left| T(\boldsymbol{\xi})\, \pi^{-1}(x^k, Z_1(x^k)) \right|_z \ , \quad (1)$$

$$r_I^k(\boldsymbol{\xi}) = I_2(\mathcal{W}(x^k, \boldsymbol{\xi})) - I_1(x^k) \ , \quad (2)$$

where x^k represents a given pixel of the image (the superscript k is omitted from here on) and $| \bullet |_z$ denotes the z-coordinate of a 3D point. The function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ projects 3D points onto the image plane according to the pinhole model, and $T(\boldsymbol{\xi}) \in \mathrm{SE}(3)$ is the homogeneous transformation associated to the twist $\boldsymbol{\xi}$. The warping function is given by:

$$\mathcal{W}(x, \boldsymbol{\xi}) = \pi(T(\boldsymbol{\xi})\, \pi^{-1}(x, Z_1(x))) \ . \quad (3)$$

We formulate a dense optimization problem to obtain the camera motion, and compute the Cauchy M-estimator of the residuals:

$$\boldsymbol{\xi}_R = \arg\min_{\boldsymbol{\xi}} \left\{ \sum_{k=1}^{M} \left[F(w_R^k r_Z^k(\boldsymbol{\xi})) + F(\alpha_I w_R^k r_I^k(\boldsymbol{\xi})) \right] \right\} \ , \quad (4)$$

$$F(r) = \frac{c^2}{2} \log\left(1 + \left(\frac{r}{c}\right)^2\right) \ , \quad (5)$$

where M is the number of pixels in Z_1 with non-null depth. The Cauchy M-estimator represents a good compromise between robustness and basin of convergence, since it is much more robust than L_2 / L_1 norms but never gets flat like the Tukey's biweight function. The parameter α_I weights the photometric and the geometric terms. The parameter c marks the inflection point of F, and can be tuned to make the estimation more or less robust against high residuals. Furthermore, a pre-weighting (w_R) is applied to help the solver converge to the true camera motion:

$$w_R(x) = Z_1(x)(1 - B^T(x)) \ . \tag{6}$$

The pre-weighting has a two-fold function. First, it down-weights pixels of the clusters which were previously segmented as moving objects (remember that $B^T(x)$ encodes the probability of pixel x to have been moving in the previous frame). Second, it gives more significance to distant points which are more likely to observe static parts of the scene.

Since (4) is highly non-linear, the motion is solved within a coarse-to-fine scheme, linearizing the residuals at every level of the image pyramid, as done in [1], [3]. At each level, the solution of (4) is obtained via Iteratively Reweighted Least Squares.

VI. Background segmentation

In order to estimate both odometry and scene flow, we need to separate the static parts of the scene from the moving ones. This would be straightforward if the camera was still, but when the camera moves every region of the scene is in apparent motion and, hence, static and non-static objects become hard to distinguish. To identify them, we propose to use the robust odometry $\boldsymbol{\xi}_R$ (previously computed) to check which regions/clusters follow this pattern of motion and which do not. This evaluation is not performed pixel-wise but cluster-wise, since we assume that all pixels in a cluster have the same rigid body motion.

Initially, the RGB-D frames are warped according to $\boldsymbol{\xi}_R$. After the warping, clusters belonging to the background will have low photometric and geometric residuals, whereas the residuals associated to moving objects will still be high. In theory, this criterion should suffice to segment the scene into static and non-static parts, but in practice the process is much more complicated because residuals are not always a good metric to evaluate precise image alignment:

- Intensity and depth images are never registered perfectly. This means that some pixels of the background (close to object boundaries) tend to get the color of the foreground and vice versa. Thus, some clusters could be perfectly aligned and still bear high residuals due to this misregistration of color and depth.
- Occluded pixels will always exhibit high residuals even if the images are perfectly aligned.
- Since the depth measurement error grows quadratically with depth, the geometric residuals of distant clusters tend to be much higher than those of clusters close to the camera.

To cope with these issues, the background segmentation is divided into two steps. First, we compute a robust metric of the residuals per cluster ($\boldsymbol{\delta}$). Second, we formulate a minimization problem to obtain the segmentation of the clusters $\boldsymbol{b}$ based on their average residuals, their geometry and their previous segmentations $\boldsymbol{b}^T$ ($\boldsymbol{b}^T$ is computed by averaging $B^T(x)$ per cluster).

The robust average residuals are computed as:

$$\delta_i = \frac{\sum_{k=1}^{S_i - O_i} \alpha_I r_I^k + r_Z^k / \bar{Z}_i}{S_i - O_i} \ , \tag{7}$$

where S_i is the size of the i cluster, O_i is the number of occluded pixels in the cluster (which are excluded from the computation of δ_i) and $\bar{Z}_i$ is the average depth of the cluster. The occluded pixels are considered to be those with geometric residuals below a certain threshold, that is, a pixel x is occluded if

$$r_Z(\boldsymbol{\xi}, x) < -\Delta Z_{occ} \ . \tag{8}$$

Next, we formulate a minimization problem to obtain the background segmentation. The energy to be minimized is composed of four terms:

$$E(\boldsymbol{b}) = E_D(\boldsymbol{b}, \boldsymbol{\delta}) + E_R(\boldsymbol{b}) + E_T(\boldsymbol{b}, \boldsymbol{b}^T) + E_Z(\boldsymbol{b}, \bar{\boldsymbol{Z}}) \ , \tag{9}$$

where $\boldsymbol{b}$ is the only unknown (dependencies with $\boldsymbol{\delta}$, $\boldsymbol{b}^T$ and $\bar{\boldsymbol{Z}}$ are shown for clarity).

The dataterm E_D pushes the clusters to be segmented as background when their residuals are low, and vice versa. To this end, we need to define a mapping between $\boldsymbol{\delta}$ and $\boldsymbol{b}$, and specify thresholds for low and high residuals (δ_L and δ_H respectively). For the sake of simplicity, we employ the following piece-wise linear function:

$$g(\delta_i) = \begin{cases} 0 & \delta_i < \delta_L \\ (\delta_i - \delta_L)/(\delta_H - \delta_L) & \delta_L \leq \delta_i \leq \delta_H \\ 1 & \delta_i > \delta_H \end{cases} , \tag{10}$$

and define the dataterm E_D as

$$E_D(\boldsymbol{b}, \boldsymbol{\delta}) = \sum_{i=1}^{N} w_D(\delta_i) \left(b_i - g(\delta_i)\right)^2 \ , \tag{11}$$

with

$$w_D(\delta_i) = \sqrt{\left(\frac{2\delta_i - (\delta_H + \delta_L)}{\delta_H - \delta_L}\right)^2 + 1} \ . \tag{12}$$

The function $w_D(\delta_i)$ increases the weight of the dataterm when the residuals are far from the area of uncertainty ($\delta_L < \delta_i < \delta_H$), giving more strength to clusters which are clearly either part of the background or a moving object.

The regularization term E_R tries to force neighbouring clusters to get a similar segmentation, and is defined as

$$E_R(\boldsymbol{b}) = \lambda_R \sum_{i=1}^{N} \sum_{j=i+1}^{N} G_{ij} \left(b_i - b_j\right)^2 \ . \tag{13}$$

We have chosen to minimize a quadratic term in (13) because it helps to smooth occasional wrong labellings of clusters with misleading residuals. We have also tried to minimize the absolute value of the differences (total variation), which allows for sharp discontinuities between connected clusters, but it did not provide better results.

Temporal regularization (E_T) is also imposed, since both static and dynamic parts of the scene are very likely to remain still and moving (respectively) through time:

$$E_T(\boldsymbol{b}, \boldsymbol{b}^T) = \lambda_T \sum_{i=1}^{N} (b_i - b_i^T)^2 \ . \tag{14}$$

Last, we include an extra term that introduces a bias towards the background ($b_i \to 0$) for all the clusters which are far from the camera. This models the fact that, in indoor scenarios, moving objects tend to be at the foreground while distant observations are likely to capture the fix elements of the environment (walls, ceiling, floor, furniture, etc.).

$$E_Z(\boldsymbol{b}, \bar{\boldsymbol{Z}}) = \lambda_Z \sum_{i=1}^{N} \max\left(0, e^{\bar{Z}_i} - e^{Z_{Min}}\right) b_i^2 \ . \tag{15}$$

Since all the terms in $E(\boldsymbol{b})$ are squares with respect to $\boldsymbol{b}$, the optimization problem (9) is convex and its solution can be obtained in closed-form. Detailed information about the parameters introduced here is given in section VIII.

VII. scene Flow Estimation and Odometry Refinement

Once the scene is segmented, we divide the motion estimation into two separate processes. All the clusters segmented as background will be considered as a single rigid block and used to re-estimate the odometry. On the other hand, the rigid body motions of the moving clusters $\boldsymbol{\xi}_i$ will be computed independently. Knowing $\boldsymbol{\xi}_i$, the scene flow $\boldsymbol{s}(x)$ associated to each point $\boldsymbol{p}(x)$ of the cluster i is calculated as

$$\boldsymbol{s}(x) = (T(\boldsymbol{\xi}_i) - I)\, \boldsymbol{p}(x) \ . \tag{16}$$

Since $\boldsymbol{b}$ are continuous in the range $[0, 1]$, we need to create a partition of that interval to separate static and moving objects. Instead of imposing a simple binary threshold at 0.5, we consider the following three regions:

- If $b_i < 1/3$, the cluster i is assumed to be static and is only utilized for the odometry estimation.
- If $b_i > 2/3$, the cluster i is assumed to be moving and is only utilized for scene flow estimation.
- If $1/3 < b_i < 2/3$, the state of the cluster i is uncertain and therefore it is utilized for both the odometry and the scene flow estimation.

In this way, clusters that are not clearly segmented contribute the odometry estimation (because they could be background), but we also compute their own independent motion.

The rigid motions of the moving clusters are obtained by minimizing the following energy:

$$\boldsymbol{\xi}_i = \arg\min_{\boldsymbol{\xi}} \left\{ \sum_{k=1}^{S_i} \left[F(w_Z^k r_Z^k(\boldsymbol{\xi})) + F(\alpha_I w_I^k r_I^k(\boldsymbol{\xi})) \right] \right\} . \tag{17}$$

The final odometry $\boldsymbol{\xi}_O$ is computed similarly by minimizing (17) for the background pixels. These optimization problems are almost the same as the one described in Section V; the only difference is the pre-weighting strategy. Once the scene is segmented, we use pre-weights that penalize occlusions and discontinuities, favouring smooth regions in the optimization process to find a precise solution:

$$w_Z = \frac{1}{K_Z + (\nabla_x Z_1)^2 + (Z_1 - Z_2)^2} \ , \tag{18}$$

$$w_I = \frac{1}{K_I + (\nabla_x I_1)^2 + (I_1 - I_2)^2} \ . \tag{19}$$

The weights w_Z and w_I penalize pixels where either the spatial or the temporal gradients are high. Although we do not provide an explicit comparison in the paper, this pre-weighting strategy provides better results than pure robust minimization without pre-weights, and also helps the IRLS solver to converge in fewer iterations.

VIII. Implementation details

In this section we describe important details of our algorithm which are not part of its theoretical core but have an impact on its performance, and set the values of the parameters introduced throughout the paper.

The Cauchy M-estimator includes the parameter c that controls how robustly the residuals are minimized. In all cases, this parameter is set proportional to the average photometric and geometric residuals ($\bar{r}$), which are evaluated after each iteration of the IRLS solver:

$$c = \begin{cases} 0.2\,\bar{r} & \text{Robust odometry (section V)} \\ \bar{r} & \text{Scene flow (section VII)} \\ 2\,\bar{r} & \text{Odometry refinement (section VII)} \end{cases} \ . \tag{20}$$

Another important aspect is the selection of the parameters δ_L and δ_H in the segmentation stage. To obtain them, we compute the median of the robust residuals associated to the clusters ($\hat{\delta}$). Since the clusters of moving objects will typically have residuals considerably higher than $\hat{\delta}$, we set this value as a threshold. We have also observed that average residuals grow with the motion of the camera and, therefore, we also make δ_L and δ_H to be dependent on the norm of the camera motion:

$$\hat{\delta}_t = \min(t_M, \max(t_B, \hat{\delta})) \ , \tag{21}$$

$$\delta_L = \hat{\delta}_t + 10\, \|\boldsymbol{\xi}_R\|_2 \, , \quad \delta_H = 2\hat{\delta}_t + 10\, \|\boldsymbol{\xi}_R\|_2 \ . \tag{22}$$

The median residual $\hat{\delta}$ has to be truncated because clusters with residuals below a certain low threshold (t_B) are always assumed to belong to the background and those above a high threshold ($\sim 2\,t_M$) are assumed to be moving objects.

The rest of the parameters introduced in sections V, VI and VII are set as follows:

- α_I is set to 0.15 so that photometric and geometric residuals contribute equally.
- The occlusion threshold (ΔZ_{occ}) is set to 0.2 meters, and the Z_{Min} of the geometric prior in the background segmentation is set to the mean depth of the scene divided by four.
- The weights of the different terms in the segmentation stage are set to $\lambda_R = 0.5$, $\lambda_T = 1.5$ and $\lambda_Z = 0.15$.

- The constants of the pre-weights in (18) and (19) are set to $K_I = 0.1$ and $K_Z = 10^{-4}$.

These values are the ones used for all the experiments presented in the paper and, although they have been obtained empirically, they provide good results for the indoor scenarios typically found when using RGB-D cameras. However, these values are not optimal in general and would need to be retuned if this method is applied with different sensors and/or configurations (e.g. stereo system and outdoor).

IX. Experiments

This section is divided into two main parts: evaluation of the odometry and evaluation of the scene flow estimation. In all the experiments, we used registered RGB-D images at QVGA resolution (240×320). The experiments has been run on a laptop with an Intel Core i7-4712 HQ CPU at 2.3 GHz and Ubuntu 14.04. Besides analyzing the results presented herein, we encourage the reader to watch the demonstration video of our method (link above).

A. Odometry Evaluation

We test the accuracy of our algorithm with several sequences of the TUM dataset [25]. Some of the selected sequences do not contain moving objects (*Freiburg1*), while others (*Freiburg3*) are very challenging and present, at least for some time intervals, scenes mostly composed of moving objects or where the percentage of pixels with null depth goes beyond 50%. For all the tested sequences, we compare our method with DIFODO [3], DVO [1] and the semi-rigid scene flow (SR-Flow) [16] (which is the only method, apart form ours, providing both odometry and scene flow). The accuracy of each method is measured with the root mean square (RMS) translational and rotational drifts per seconds, as proposed in [25]. Besides this quantitative evaluation, and since only pixels with valid depth can be used to estimate the camera motion, we compute two additional statistics per sequence: the average percentage of pixels with valid depth, and the minimum percentage of pixel with valid depth among all the frames (i.e. the RGB-D image with the highest number of null depth measurements).

Results are shown in Table II. It can be noticed that, for the static scenes, DIFODO shows the lowest translational drift and our approach has the lowest rotational error. The worst results are provided by the SR-Flow, being on average between 2 and 4 times less accurate than the other methods. As far as the dynamic sequences are concerned, the best results are always obtained with our approach. However, the drift is quite high in all cases, a fact that can be explained by analyzing the number of valid/used points in each sequence. While static sequences present, on average, 75% of valid depth measurement, this number drops to 55% in the non-static sequences. Moreover, within the non-static sequences, some RGB-D frames contain even less than 20% of valid depth measurements which, together with the fact that some of them are observing moving objects, renders the odometry problem extremely complicated. For the "*Freiburg3/walk static*" sequence, the percentage of used pixels never goes below 45%, and therefore our approach can still provide reasonable results. In the other two cases, the presence of RGB-D pairs with very low percentages of valid depth leads to much higher RMS errors.

TABLE I
PERCENTAGE OF PIXELS WRONGLY SEGMENTED AS BACKGROUND OR AS UNCERTAIN REGIONS IN THE *Freiburg1* SEQUENCES.

Sequence	Moving objects	Uncertain
Freiburg1/desk	1.95%	3.73%
Freiburg1/desk2	2.53%	4.44%
Freiburg1/teddy	1.2 %	1.58%

We also show some of the segmentations provided by our method (Fig. 2) and, for the static sequences, we compute the percentage of pixels wrongly segmented as background, and also those considered as uncertain (Table I). Results show that these percentages are quite low for the three sequences considered.

B. Scene Flow Evaluation

In this section we compare our approach with three of the most recent works on scene flow estimation: primal-dual scene flow (PD-Flow) [15], the semi-rigid scene flow (SR-Flow) [16] and the smooth piecewise rigid flow (MC-Flow) presented in [18]. Given the lack of RGB-D datasets with ground truth for the evaluation of scene flow, we have selected a set of RGB-D pairs observing different objects with varied motions. Some of the tested images have been used in previous works [16] [18], while others are new and will be published together with the code. Half of these RGB-D pairs were recorded with a moving camera, while the camera stood still in the other half. The accuracy of the different methods is assessed by warping the RGB-D pairs according to the estimated scene flow and measuring the RMS residuals (geometric and photometric) after alignment. The only inconvenient associated to this procedure is the fact that occluded pixels always generate high residuals even if the images are perfectly aligned, and therefore distort the error metric. To prevent this, for every method we discard pixels with geometric residuals below a certain threshold, i.e., a pixel x is disregarded if $r_Z(x) < -\Delta Z_{occ}$ after warping. Thus, the resulting RMS residuals become a more faithful metric of precise image alignment.

The testing images, together with the segmentations estimated by our approach, are shown in Fig. 3. It can be observed that the segmentations are precise but not perfect, mostly because pixels are not segmented independently but in clusters, and clusters sometimes contain points of different objects (e.g. in the "cleaning whiteboard" image, the hand with the eraser is segmented as a moving object together with a small part of the whiteboard). Quantitative results are presented in Table III. It can be noticed that MC-Flow provides the best results for almost every RGB-D pair, followed by our method. This is to be expected because MC-Flow uses as strategy similar to the one described here to estimate motion, but it also optimizes for the clusters in an alternating scheme. Consequently, its results are more precise

TABLE II

ODOMETRY - SEQUENCE STATISTICS AND TRANSLATIONAL/ROTATIONAL DEVIATIONS PER SECOND.

Sequence	Average %	Min %	Translational RMSE (cm/s)				Rotational RMSE (deg/s)			
	valid depth	valid depth	DIFODO	DVO	Ours	SR-Flow	DIFODO	DVO	Ours	SR-Flow
Freiburg1/desk	74.49%	60.07%	**3.66**	4.08	3.79	11.6	2.56	2.18	**1.88**	7.44
Freiburg1/desk2	74.52%	60.95%	**5.28**	6.45	5.33	12.7	3.31	3.55	**2.51**	9.96
Freiburg1/teddy	77.53%	64.73%	**5.18**	9.67	6.51	17.7	2.77	2.46	**2.04**	13.47
Freiburg3/walk static	54.59%	45.20%	45.4	31.2	**11.1**	23.7	10.2	4.56	**1.83**	5.42
Freiburg3/walk xyz	52.77%	17.67%	69.4	48.1	**30.4**	50.1	12.49	8.45	**5.69**	11.43
Freiburg3/walk halfsphere	58.59%	30.46%	70.0	41.2	**34.1**	46.5	19.21	7.22	**6.77**	13.8

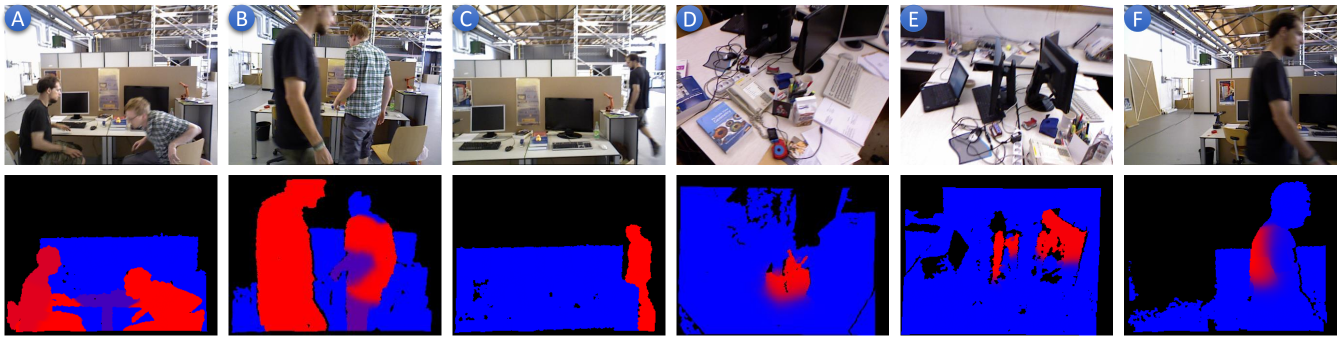

Fig. 2. Segmentations obtained for some of the RGB-D frames in the tested sequences. Clusters segmented as background are depicted in blue and moving objects in red. It can be noticed that different moving parts can be segmented properly: (A) is segmented perfectly and image (B) is not perfect but quite reasonable. (C) shows that, despite the geometric prior (15), our method can also detect moving objects which are at the background, far from the camera. (D) and (E) are examples of clusters wrongly segmented as moving parts. This sometimes occurs with clusters which observe non-smooth surfaces with scattered points and depth discontinuities, but they do not have any negative effect on the odometry estimation as long as there are enough clusters segmented as background. Last, (F) shows one of the mentioned cases where more than half of the image has null depth and, among the pixels with valid depth, only half of them are observing static parts. In this case, our method fails to segment the scene properly and hence also fails to estimate the camera motion.

but its computational burden is much higher. The average runtimes of the compared methods are:

- PD-Flow: 40 milliseconds on GPU or 2 seconds on CPU.
- SR-Flow: 105 seconds on a single CPU core.
- MC-Flow: 20 seconds running on CPU and GPU.
- Our approach: 80 milliseconds on multiple CPU cores.

In summary, our approach provides the second best scene flow estimates after MC-Flow (residuals are 12% higher on average), and it is 50% and 21% more accurate than PD-Flow and SR-Flow, respectively. Moreover, it is 250 times faster than MC-Flow and 1300 times faster than SR-Flow (only PD-Flow on GPU is faster than our method).

X. CONCLUSIONS

In this paper we have presented a method to estimate both odometry and scene flow with RGB-D cameras. Results demonstrate that our method performs similarly or better than other state-of-the-art approaches, which normally focus either on odometry or on scene flow, but do not estimate both. The only method which also addresses the joint estimation problem (SR-Flow [16]) is significantly less accurate with the camera motion and 20% less accurate with scene flow, as well as more than 1000 times slower. The main strength of our approach is that it provides accurate results with a very low runtime (80 milliseconds). To the best of our knowledge, this is the first method providing precise odometry and scene flow at a frame rate (>10Hz) that is sufficiently high to work in real-world scenarios.

Nevertheless, some aspects can still be improved. The K-Means clustering sometimes mix different objects in the same cluster, which leads to inaccurate scene flow estimates. As a solution, we will consider ways to exploit color (superpixels) or orientation in the segmentation process to obtain clusters that are more meaningful and still fast to compute. On the other hand, the temporal propagation of the background segmentation could be improved by introducing probabilistic/weighted models that are more consistent through longer periods of time. Last, we want to optimize the implementation of our algorithm so that it reaches real-time performance.

REFERENCES

[1] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. Int. Conference on Robotics and Automation (ICRA)*, 2013, pp. 3748–3754.

[2] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2013, pp. 1449–1456.

[3] M. Jaimez and J. Gonzalez-Jimenez, "Fast visual odometry for 3-D range sensors," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809–822, 2015.

[4] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 652–659.

[5] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects," in *Proc. Int. Conference on Robotics and Automation (ICRA)*, 2013, pp. 5003–5010.

[6] F. Steinbruecker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proc. Int. Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 719–722.

TABLE III

SCENE FLOW - RMS PHOTOMETRIC AND GEOMETRIC RESIDUALS.

RGB-D pair	Camera moving	RMS photometric residuals				RMS geometric residuals (m)			
		PD-Flow	SR-Flow	Ours	MC-Flow	PD-Flow	SR-Flow	Ours	MC-Flow
Two People walking	No	0.0368	0.0309	0.0284	**0.0251**	0.0511	0.0468	0.036	**0.0336**
Person standing	No	0.0532	0.0494	0.0303	**0.0256**	0.1385	0.1282	0.0829	**0.0814**
Robot	No	0.0518	0.0357	0.0251	**0.023**	0.0733	0.0688	0.0643	**0.0601**
Hand	No	0.0135	0.0077	0.0066	**0.0065**	0.0179	**0.0171**	0.0182	0.0182
Tree	No	0.0661	0.0335	0.0303	**0.0258**	**0.0143**	0.0259	0.0242	0.0196
Two People moving	Yes	0.0387	0.0301	0.0245	**0.0241**	0.1355	0.096	0.0928	**0.0654**
Cleaning Whiteboard	Yes	0.034	0.0233	0.0214	**0.0185**	0.0666	0.0561	0.0377	**0.0315**
Opening door	Yes	0.0231	0.0214	0.0204	**0.0153**	0.1681	0.1317	0.1014	**0.0551**
Person sitting	Yes	0.0753	0.0558	0.0452	**0.0428**	**0.0576**	0.0601	**0.0576**	0.0599

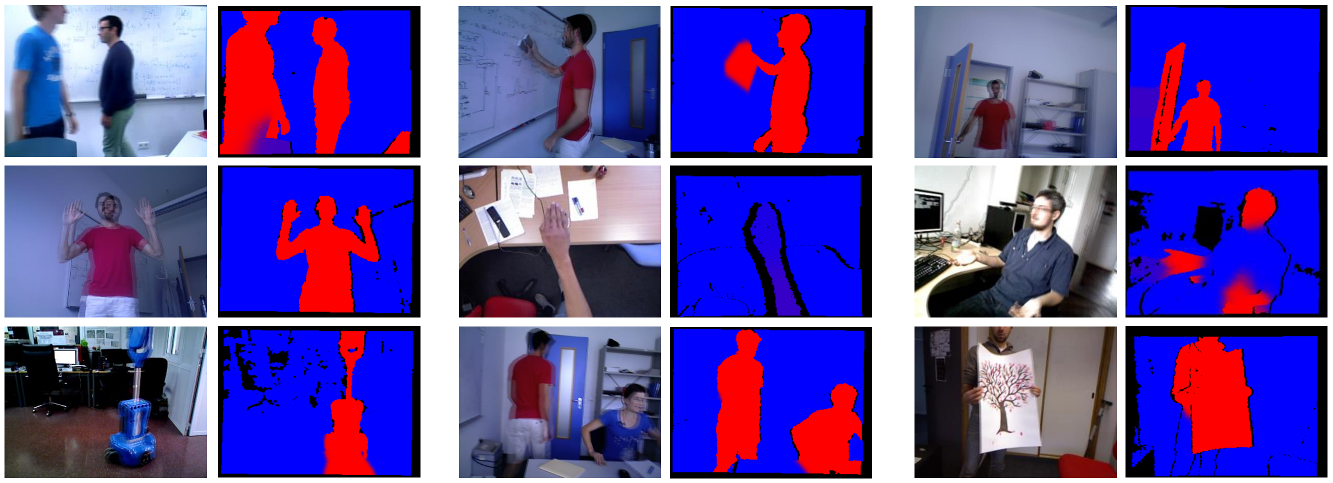

Fig. 3. Selected images to evaluate scene flow, together with the segmentation provided by our method. Clusters segmented as background are depicted in blue and moving objects in red. It can be observed that the moving objects are segmented precisely in most of the images. The main exception is the "hand RGB-D pair", where the hand was not segmented as a moving object but as a region with uncertainty (slightly purple in the picture) due to its very small motion. Other inaccuracies can be observed in small parts of the backgrounds wrongly segmented as moving objects (which does not have a very detrimental effect on the motion estimates).

[7] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry," in *Proc. Int. Conference on Robotics and Automation (ICRA)*, 2015, pp. 83–89.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux *et al.*, "Kinect-Fusion: Real-time dense surface mapping and tracking," in *Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[9] J. Stückler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras," in *Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012, pp. 162–167.

[10] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. Int. Conference on Robotics and Automation (ICRA)*, 2013, pp. 5724–5731.

[11] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers, "Stereoscopic scene flow computation for 3D motion understanding," *Int. Journal of Computer Vision*, vol. 95, no. 1, pp. 29–51, 2011.

[12] A. Letouzey, B. Petit, and E. Boyer, "Scene flow from depth and color images," in *Proc. British Machine Vision Conference (BMVC)*, 2011, pp. 46.1–46.11.

[13] E. Herbst, X. Ren, and D. Fox, "RGB-D flow: Dense 3-D motion estimation using color and depth," in *Proc. Int. Conference on Robotics and Automation (ICRA)*, 2013, pp. 2276–2282.

[14] J. Quiroga, F. Devernay, and J. L. Crowley, "Local/global scene flow estimation," in *Proc. Int. Conference on Image Processing (ICIP)*, 2013, pp. 3850–3854.

[15] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual framework for real-time dense RGB-D scene flow," in *Proc. Int. Conference on Robotics and Automation (ICRA)*, 2015, pp. 98 – 104.

[16] J. Quiroga, T. Brox, F. Devernay, and J. L. Crowley, "Dense semi-rigid scene flow estimation from RGBD images," in *Proc. European Conference on Computer Vision (ECCV)*, 2014, pp. 567–582.

[17] D. Sun, E. B. Sudderth, and H. Pfister, "Layered RGBD scene flow estimation," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] M. Jaimez, M. Souiai, J. Stückler, J. Gonzalez-Jimenez, and D. Cremers, "Motion Cooperation: Smooth piece-wise rigid scene flow from RGB-D images," in *Int. Conf. on 3D Vision (3DV)*, 2015, pp. 64–72.

[19] M. Hornacek, A. Fitzgibbon, and C. Rother, "SphereFlow: 6 DoF scene flow from RGB-D pairs," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3526 – 3533.

[20] H. A. Alhaija, A. Sellent, D. Kondermann, and C. Rother, "Graphflow–6D large displacement scene flow via graph matching," in *German Conference on Pattern Recognition*, 2015, pp. 285–296.

[21] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 343–352.

[22] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "VolumeDeform: Real-time volumetric non-rigid reconstruction," *arXiv:1603.08161*, 2016.

[23] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson *et al.*, "Fusion4D: real-time performance capture of challenging scenes," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4.

[24] C. Vogel, K. Schindler, and S. Roth, "3D scene flow estimation with a piecewise rigid scene model," *Int. Journal of Computer Vision*, vol. 115, no. 1, pp. 1–28, 2015.

[25] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Int. Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.