StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments

Raluca Scona[1,2], Mariano Jaimez[3,4], Yvan R. Petillot[1], Maurice Fallon[5], Daniel Cremers[3]

***Abstract*— In this paper we propose a method for robust dense RGB-D SLAM in dynamic environments which detects moving objects and simultaneously reconstructs the background structure. Dynamic environments are challenging for visual SLAM as moving objects can impair camera pose tracking and cause corruptions to be integrated into the map. While most methods employ implicit robust penalizers or outlier filtering techniques in order to handle moving objects, our approach is to simultaneously estimate the camera motion as well as a probabilistic static/dynamic segmentation of the current RGB-D image pair. This segmentation is then used for weighted dense RGB-D fusion to estimate a 3D model of only the static parts of the environment. By leveraging the 3D model for frame-to-model alignment, as well as static/dynamic segmentation, camera motion estimation has reduced overall drift — as well as being more robust to the presence of dynamics in the scene. Demonstrations are presented which compare the proposed method to comparable state-of-the-art approaches using both static and dynamic sequences. The proposed method achieves similar performance in static environments and improved accuracy and robustness in dynamic scenes.**

I. INTRODUCTION

State-of-the-art dense visual SLAM methods can produce impressive reconstructions of large indoor scenes. These approaches, however, often rely on certain assumptions such as the environment being static, the camera moving smoothly and there being sufficient geometry or texture in the scene for reliable tracking. We focus specifically on the assumption of a static scene. Robust operation in the presence of dynamic elements is an open problem. For example, most odometry methods perform registration between the current image and a previous reference and the estimated transformation between these images is assumed to originate from the camera motion. Dynamic elements violate this assumption and can cause failures in pose tracking. In addition, if not actively detected and segmented, dynamic objects can be fused into the map which can lead to irreversible corruptions.

This research is supported by the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh, and the ERC Consolidator Grant *3DReloaded*. M. Fallon is supported by a Royal Society University Research Fellowship.

[1]School of Engineering & Physical Sciences, Heriot-Watt University, UK. y.r.petillot@hw.ac.uk

[2]School of Informatics, University of Edinburgh, UK. raluca.scona@ed.ac.uk

[3]Department of Computer Science, Technical University of Munich, Germany. {jaimez, cremers}@in.tum.de

[4]Department of Systems Engineering and Automation, University of Málaga, Spain. marianojt@uma.es

[5]Oxford Robotics Institute, University of Oxford, UK. mfallon@robots.ox.ac.uk

Improving the performance of SLAM in dynamic environments is an important problem particularly for mobile robots. It is seldom the case that robots operate in strictly static environments and such a requirement would significantly limit the extent to which they could be successfully deployed. In example applications, co-bots such as the Rethink Baxter carry out assembly tasks among moving equipment and infrastructure. Mobile service robots face similar challenges, as their environments are also inhabited by people.

In this work we address this problem by jointly estimating the motion of an RGB-D camera and segmenting the scene it observes into static and dynamic parts. Camera tracking is performed by aligning incoming frames with a dense surfel-based model of the environment (similarly to ElasticFusion [1]). By decoupling the static and moving parts, we can build a background model which fuses only the static elements.

Effective detection and segmentation of moving objects typically requires temporal feedback or a multi-frame formulation. We demonstrate that background 3D reconstruction is an efficient way to propagate this temporal information without incurring significant runtime costs. Also, the resulting map is more meaningful in the sense that it only contains structural elements and the static objects present in the scene.

The contributions of the proposed work are as follows:

- A new formulation to simultaneously estimate the motion of the camera and to segment the static objects within the current frame.
- A dense mapping system which fuses only the temporally consistent data (i.e. it stores useful information of what was static in the past).
- An extensive evaluation demonstrating that the proposed algorithm outperforms state-of-the-art solutions in dynamic environments and achieves a very competitive runtime ($\sim$ 30 ms/frame).

The paper is organized as follows. We review state-of-the-art visual SLAM methods as well as several approaches which tackle dynamic environments in Section II. Section III describes the overall structure of our system, with Sections IV and V giving details about our proposed approaches for simultaneous estimation of camera motion and static/dynamic segmentation and weighted fusion. Section VI states specific details related to implementation and parameter tuning. We present our evaluation in Section VII where we compare our method against related state-of-the-art works. Section VIII concludes our work and states future directions for research.

The code and the demonstration video can be found here:

http://www.edinburgh-robotics.org/students/raluca-scona

II. RELATED WORK

Dense visual SLAM in indoor environments has achieved significant progress through the emergence of commodity 3D cameras such as the Microsoft Kinect or the Asus Xtion. A seminal work in this field is KinectFusion [2], the first system which used RGB-D data to perform real-time dense tracking and data fusion. The approach used a volumetric representation of fixed size integrated on a GPU. Ongoing research has produced systems with impressive performance, ranging from scalable extensions [3], [4] and loop closure capabilities [5], [6], to methods which consider run-time limitations [7], [8] in order to enable 3D scanning and mapping of large indoor scenes to be as robust and easy to use as possible.

While most existing approaches focus on scanning static environments, specific efforts have also been made to increase robustness in dynamic scenes.

Implicit Handling of Dynamic Elements: It is common to use a robust cost function within the visual odometry front-end which penalizes the contribution of high-residual points and implicitly increases the robustness of pose estimation to un-modeled effects. Gutierrez *et al*. [9] compare different robust functions focusing on the quality of the resulting pose estimate, while Kerl *et al*. [10][8] demonstrate robustness to the presence of small moving objects in the scene. These solutions however are insufficient and fail when moving parts occupy a significant portion of the image.

Reconstruction-focused approaches, such as ElasticFusion [1] as well as the method of Keller *et al*. [4], require that points be repeatedly observed through consecutive frames before becoming integrated within the 3D model. Similarly, in these methods dynamic elements are not explicitly detected and handled, resulting in robustness which is limited to small motions in the scene.

Outlier Rejection Strategies: A common strategy is to treat dynamically moving objects as noise which must be detected and filtered out.

For Keller *et al*. [4], input points with no close model correspondence are used to seed a region-growing procedure to segment the current image into static and dynamic parts. Subsequently, model points which are matched with dynamic input points are removed from the reconstruction. This approach can only be demonstrated once a confident reconstruction of the scene is in place.

Meanwhile, in DTAM [11], which is a monocular dense mapping system, the authors discard pixels with a photometric error higher than a specific threshold. For ORB-SLAM [12], [13] the authors enforce an effective survival-of-the-fittest strategy which judges the validity of keyframes and the points used for pose tracking. By being generous when spawning new keyframes and points within the system and by enforcing highly conservative culling strategies, they demonstrate impressive robustness and versatility.

Nevertheless, within these approaches, no spatial or temporal coherence is enforced among the detected dynamic points between consecutive frames.

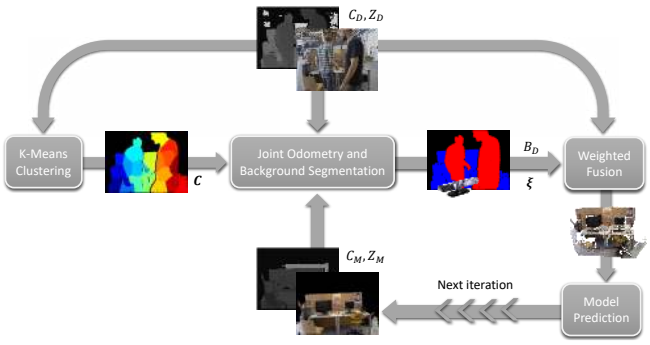

Fig. 1: System architecture: the process starts by receiving a new RGB-D image (C_D, Z_D) and grouping its pixels into geometric clusters $\boldsymbol{C}$. A prediction (C_M, Z_M) is rendered from the model and the last pose estimate $T(\boldsymbol{\xi})$ and used for joint alignment and background segmentation. Both results are then exploited for weighted fusion of the static clusters of C_D, Z_D with the map.

Methods Enforcing Spatial or Temporal Coherence: Jaimez *et al*. [14] introduces a joint visual odometry and scene flow estimation method. Similarly, BaMVO [15] is an odometry method which reconstructs the environment over the previous 4 frames by temporal propagation. As both are frame-to-frame methods, they will incur unbounded drift in the pose estimate over time. Rünz *et al*. [16] proposed a method to reconstruct and track each moving object with a separate 3D model, being one of the first real-time methods to perform dense tracking and fusion of multiple objects so as to explicitly handle dynamics and enforce both spatial and temporal coherence.

Finally, it is possible to robustify visual SLAM in dynamic environments through the use of motion priors using inertial measurement units [17] or a robot's odometry [18], but for many use-cases it is attractive to focus on improving the accuracy of pose tracking and map building using only a single camera.

III. FRAMEWORK AND NOTATION

We propose a new SLAM system for RGB-D cameras which focuses on background segmentation and filtering of dynamic objects in the foreground. This section provides a general description of its main components (Fig. 1); each individual component of the algorithm will be described in detail in the following sections.

The input to our system is a stream of registered RGB-D images. An RGB-D pair is represented as a colour image $C_D : \Omega \rightarrow \mathbb{R}^3$ and a depth image $Z_D : \Omega \rightarrow \mathbb{R}$, where $\Omega \subset \mathbb{R}^2$ is the image plane. We also compute an intensity image $I_D : \Omega \rightarrow \mathbb{R}$ from C_D for use in the algorithm.

First, every incoming pair (I_D, Z_D) is segmented into K geometric clusters $\boldsymbol{C} = \{C_i, i = 1, ..., K\}$ by applying K-Means on the 3D coordinates of the scene points (as described in [14]). In order to reduce the overall computational complexity, each cluster is assumed to behave as a rigid body, which allows us to solve the static/dynamic segmentation problem cluster-wise as opposed to pixel-wise. This is an acceptable approximation because we are not interested in estimating accurate motions of moving objects, but are rather

focused on building a *conservative* reconstruction of the static structures in the scene.

Second, an artificial image pair (I_M, Z_M) is rendered by placing a virtual camera at the previous camera pose estimate within the current map of the static scene constructed up to that point. Given the current images (I_D, Z_D) and the last prediction (I_M, Z_M), our novel step is to jointly obtain the camera motion $\boldsymbol{\xi} \in \mathfrak{se}(3)$ and a motion-based segmentation of the scene between the two time instances. Each cluster i is assigned a score $b_i \in [0, 1]$ which corresponds to the level of dynamism: $b \simeq 1$ corresponds to static clusters, $b \simeq 0$ to moving clusters and $0 < b < 1$ to intermediate levels of uncertainty.

After the solution to the joint estimation problem is calculated, the clusters and scores are used to compute a per-pixel segmentation image B_D for each point belonging to the background, which, together with the current colour and depth images (C_D, Z_D), is used for weighted 3D fusion.

IV. Joint Estimation of the Camera Motion and the Scene Segmentation

To estimate these two joint properties, we propose a new formulation based on the minimization of two energy terms:

$$\min_{\boldsymbol{\xi}, \boldsymbol{b}} \{D(\boldsymbol{\xi}, \boldsymbol{b}) + S(\boldsymbol{b})\} \quad s.t.\ b_i \in [0, 1] \quad \forall i \tag{1}$$

where $\boldsymbol{b}$ represents the full set of scores. The term $D(\boldsymbol{\xi}, \boldsymbol{b})$ encodes direct image alignment by enforcing photometric and geometric consistency only for pixels that belong to static clusters. The second term $S(\boldsymbol{b})$ complements $D(\boldsymbol{\xi}, \boldsymbol{b})$ by forcing clusters to be segmented as dynamic when their residuals are very high, and vice versa. It also includes spatial regularization to encourage a smooth segmentation of the clusters, and exploits prior geometric knowledge to help the optimization converge to the correct minimum. Next, we present the formulation of $D(\boldsymbol{\xi}, \boldsymbol{b})$ and $S(\boldsymbol{b})$ and describe how the overall minimization problem is tackled.

A. Camera Motion

For every new RGB-D pair, the incremental motion of the camera is computed by minimizing the geometric and photometric reprojection errors between the current RGB-D image and the last prediction obtained from the map. The respective reprojection errors (or residuals) are defined as:

$$r_Z^p(\boldsymbol{\xi}) = Z_M(\mathcal{W}(\boldsymbol{x}^p, \boldsymbol{\xi})) - \left| T(\boldsymbol{\xi})\, \pi^{-1}(\boldsymbol{x}^p, Z_D(\boldsymbol{x}^p)) \right|_z \tag{2}$$

$$r_I^p(\boldsymbol{\xi}) = I_M(\mathcal{W}(\boldsymbol{x}^p, \boldsymbol{\xi})) - I_D(\boldsymbol{x}^p)\ , \tag{3}$$

where $\boldsymbol{x}^p \in \Omega$ represents the coordinates of a given pixel p and $|\bullet|_z$ denotes the z-coordinate of a 3D point. The function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ projects 3D points onto the image plane according to the camera's pinhole model. $T(\boldsymbol{\xi}) \in \mathrm{SE}(3)$ is the homogeneous transformation associated to the twist $\boldsymbol{\xi}$. The warping function is given by:

$$\mathcal{W}(\boldsymbol{x}^p, \boldsymbol{\xi}) = \pi(T(\boldsymbol{\xi})\, \pi^{-1}(\boldsymbol{x}, Z_D(\boldsymbol{x}^p)))\ . \tag{4}$$

Similar procedures for minimizing these residuals are described in [14], [19].

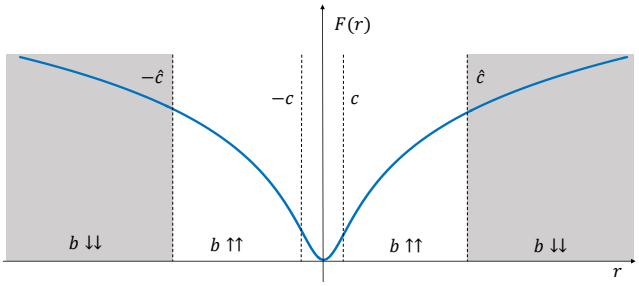

Fig. 2: Cauchy robust penalty and the different regions defined to distinguish between the clusters that are likely to be static ($b \uparrow\uparrow$) or dynamic ($b \downarrow\downarrow$).

The novelty of our formulation is to weight these residuals with the scores $\boldsymbol{b}$ so that only residuals associated to static parts of the scene have a high contribution:

$$D(\boldsymbol{\xi}, \boldsymbol{b}) = \sum_{p=1}^{N} b_{i(p)} \left[F(w_Z^p r_Z^p(\boldsymbol{\xi})) + F(\alpha_I w_I^p r_I^p(\boldsymbol{\xi})) \right]\ , \tag{5}$$

where N is the overall number of pixels and $b_{i(p)}$ refers to the score of the cluster i containing p. As the geometric and intensity terms compute errors with different units, the parameter α_I re-scales the intensity term so that it has a comparable effect in scale as the geometric term. The function $F(r)$ is the Cauchy robust penalty:

$$F(r) = \frac{c^2}{2} \log\left(1 + \left(\frac{r}{c}\right)^2\right)\ , \tag{6}$$

where c represents the inflection point of $F(r)$ and controls how robustly residuals are minimized. Lastly, w_Z and w_I weight the photometric and geometric residuals according to the noise of the measurements (σ_Z and σ_I) and also penalize occlusions and discontinuities observed through high spatial or temporal gradients:

$$w_Z = \frac{1}{k_\sigma^Z \sigma_Z^2 + |\nabla_x Z_D| + |Z_D - Z_M|}\ , \tag{7}$$

$$w_I = \frac{1}{k_\sigma^I \sigma_I^2 + |\nabla_x I_D| + |I_D - I_M|}\ . \tag{8}$$

In (7) and (8), the parameters k_σ^Z and k_σ^I control the relative importance of the noise against the derivatives.

B. Static / Dynamic Segmentation

The objective of the second term in (1) is to classify clusters with average high residuals as dynamic and those with low residuals as static. The underlying idea is that clusters with high residuals are the ones whose relative motion with respect to the camera does not coincide with the camera motion itself. In order to implement this concept we must quantify what a 'high residual' is.

Our assumption is that large residuals correspond to those significantly higher than the parameter c, i.e. those lying on the flatter sides of the function $F(r)$ (see Fig. 2). The following term sets this threshold within the overall minimization problem:

$$S_D(\boldsymbol{b}) = \sum_{i=1}^{K} 2(1 - b_i) F(\hat{c}) K_i \tag{9}$$

The total number of pixels in each cluster i is represented by K_i, and $\hat{c} > c$ is a heuristically selected threshold which defines the frontier between low and high residuals. The combination of this term with (5) basically encourages b_i to be as low as possible (to a minimum of 0) when the average residual of cluster i is higher than $\hat{c}$; otherwise it favours high values of b_i (to a maximum of 1).

Furthermore, we include a regularization term that encourages contiguous clusters to have a similar score:

$$S_R(\boldsymbol{b}) = \lambda_R \sum_{i=1}^{K} \sum_{j=i+1}^{K} G_{ij} \left(b_i - b_j\right)^2 \ . \tag{10}$$

In (10), G_{ij} is a connectivity map: it is equal to 1 when clusters i and j are contiguous in space and it is 0 otherwise. The parameter λ_R weights $S_R(\boldsymbol{b})$ with respect to the other terms.

Lastly, we add a geometric constraint that exploits the fact that moving objects do not appear in our map and therefore the depth differences between Z_D and Z_M will be significantly high for moving clusters. This constraint is expressed as a segmentation prior:

$$S_P(\boldsymbol{b}) = \lambda_P \sum_{i=1}^{K} \left(b_i - b_i^P\right)^2 \tag{11}$$

with

$$b_i^P = 1 - k_p \frac{\sum_{k=1}^{K_i} |Z_D(\boldsymbol{x}^k) - Z_M(\boldsymbol{x}^k)|}{K_i} \ , \tag{12}$$

where k_p controls how high depth differences should be to enforce a dynamic scoring and λ_P is the parameter that weights this constraint within the overall optimization. Admittedly, (11) has some degree of redundancy with (5), however, (12) computes depth differences directly without any pre-weighting as do (7) and (8). This provides additional evidence of the presence of moving objects.

The three terms described above only depend on $\boldsymbol{b}$. For the sake of clarity we group them into the combined term $S(\boldsymbol{b})$ which is used in (1):

$$S(\boldsymbol{b}) = S_D(\boldsymbol{b}) + S_R(\boldsymbol{b}) + S_P(\boldsymbol{b}) \ . \tag{13}$$

C. Solver

Since (1) involves direct image alignment, the whole minimization problem must be solved within a coarse-to-fine scheme. This implies building a pyramid of images and aligning them from the coarsest to the finest level. The segmentations obtained at the intermediate levels of the pyramid are stored and used to initialize the solver at the following level, thus allowing the algorithm to converge to the right segmentation at the different levels of the pyramid.

At each level, the term $D(\boldsymbol{\xi}, \boldsymbol{b})$ is nonlinear and non-convex with respect to $\boldsymbol{\xi}$. However, the combined optimization problem is convex and can be solved analytically with respect to $\boldsymbol{b}$. Therefore, we use iteratively re-weighted least squares (IRLS) to minimize (1) with respect to the camera motion $\boldsymbol{\xi}$ and obtain the close-form solution for $\boldsymbol{b}$ after every iteration of the IRLS algorithm. Decoupling $\boldsymbol{\xi}$ from $\boldsymbol{b}$ within the solver allows us to compute the solution for each efficiently, while the tight alternation of those two steps leads to good rate of convergence.

V. Surfel-based 3D Reconstruction

The 3D model is represented as an unordered list of surfels, as described in the work of Keller *et al.* [4] and made available through the open-source implementation of Whelan *et al.* [1]. A surfel is a 3D disk, with associated position and normal $\boldsymbol{p}, \boldsymbol{n} \in \mathbb{R}^3$, colour $\boldsymbol{c} \in \mathbb{N}^3$, radius $r \in \mathbb{R}$, viability $w \in [0, 1]$ (where $w \to 1$ means viable), initialization timestamp t_0, timestamp of latest update t and counter $h \in \mathbb{N}$ representing the number of times the surfel has been updated.

At every timestep, our system takes as input the last RGB-D pair (C_D, Z_D) as well as the per-pixel segmentation image B_D of each point belonging to the background. We maintain a fused colored model for visualization purposes but convert to intensity when rendering an image prediction.

We follow the same approach as [1], [4] for pre-processing and data association. The difference is we propose a strategy which judges the viability of each surfel in order to enable the model to remove those surfels that are matched with dynamic input points. Our fusion approach is listed in Algorithm 1 and explained below.

A. Surfel Viability

A surfel is considered *viable* if it is repeatedly observed and matched with static input points. These conditions ensure that a viable surfel is both not spurious and static. Only in this case do we impose that $w \to 1$. To achieve this, each new surfel is introduced into the model with low viability $w \to 0$. On subsequent observations, w is updated through a running sum of the log-odds probabilities of matching input points. This strategy is suitable to our application as viability only increases through repeated matches with static points ($B_D^i > 0.5$) and automatically decreases with dynamic matches ($B_D^i < 0.5$):

$$\text{sign}\left(\ln\left(\frac{w}{1-w}\right)\right) = \begin{cases} -1 & \text{if } w < 0.5 \\ 0 & \text{if } w = 0.5 \\ 1 & \text{otherwise} \end{cases} \ . \tag{14}$$

B. Fusion

During fusion, a weighted average scheme is used to update a surfel's position, color and normal. Besides B_D, we employ two additional weights to represent the quality of each input point:

1) For each pixel i, a Gaussian weight $\gamma_D^i \in [0, 1]$ biasing in favor of points close to the central pixel $\boldsymbol{x}_c$ [4]:

$$\gamma_D^i = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_c||}{2\sigma^2 ||\boldsymbol{x}_c||}\right) \ . \tag{15}$$

2) A velocity weight $v_D \in [0, 1]$ biasing in favor of points seen during slow motion:

$$v_D = \max\left(1 - \frac{1 - \max(||\boldsymbol{\xi}||, 0.15)}{0.15}, 0.5\right) \ . \tag{16}$$

Algorithm 1: Weighted Surfel Fusion

Input: C_D, Z_D, B_D
$s_D \leftarrow generate\_input\_surfels()$;
foreach s_D^i **do**
  $s_M^k \leftarrow search\_for\_model\_correspondence()$;
  if s_M^k *found* **then**
    $compute\_weights(\gamma_D^i, v_D)$;
    `//Compute input viability`
    $w_D^i \leftarrow \min(B_D^i, \gamma_D^i, v_D^i)$;
    `//Truncate to avoid early saturation`
    $w_M^i \leftarrow \max(0.01, \min(0.99, w_M^i))$;
    $w_D^i \leftarrow \max(0.01, \min(0.53, w_D^i))$;
    `//Update position, colour and normal through weighted average scheme`
    $\boldsymbol{k}_M^i \leftarrow \frac{h_M^i w_M^i \boldsymbol{k}_M^i + w_D^i \boldsymbol{k}_D^i}{h_M^i w_M^i + w_D^i}$;
    $\forall \boldsymbol{k} \in \{\boldsymbol{p}, \boldsymbol{c}, \boldsymbol{n}\}$;
    `//Update viability by sum of log-odds`
    $l \leftarrow \ln\left(\frac{w_M^k}{1-w_M^k}\right) + \ln\left(\frac{w_D^i}{1-w_D^i}\right)$;
    $w_M^k \leftarrow 1 - \frac{1}{1+\exp(l)}$;
    `//Update history counter`
    $h_M^k \leftarrow h_M^k + 1$;
  else
    if $B_D^i > 0.5$ **then**
      `//Add new surfels`
      $w_D^i \leftarrow \alpha$ where $\alpha \to 0$;
      $h_D^i \leftarrow 1$;

While the first term weighs points based on the assumption that measurements closer to the camera center are more accurate, the second penalizes the influence of points recorded during fast motion which would introduce blur within the model.

C. Surfels Removal

Finally, a cleaning stage removes surfels for which $w < 0.5$ for more than 10 consecutive frames. We also perform free-space violation checks to remove points remaining in front of viable surfels. This ensures that dynamic objects and noisy measurements are removed from the map and a clean representation of the environment structure is maintained in the long term.

VI. Implementation Details

A. Initialization

As we rely on a map for both odometry and static/dynamic segmentation of the scene, we require an initialization stream to generate the first reliable map. The initial frames (first 1-2 seconds) observed by the system should contain no more than 20-30% of moving elements in order to allow for a successful initialization of the map.

Our current formulation starts by aligning the first two RGB-D pairs read from the camera following the same procedure described in Section IV. By solving (1) we also obtain a segmentation of the last image (B_D) that we use to generate the first instance of the map. After this first step the algorithm always aligns the incoming RGB-D pairs with the last prediction obtained from the map.

B. Analyzing and processing residuals

Among the parameters presented in Section IV, the most important ones are c and $\hat{c}$ (see Fig. 2). The parameter c is commonly chosen as a linear function of the median or the median absolute deviation (MAD) of the residuals [9], [20]. Since this metric is computationally expensive, we sacrifice accuracy and compute the mean ($\bar{r}$) of the residuals instead:

$$\bar{r} = \frac{1}{2N}\sum_{p=1}^{N} |w_Z^p r_Z^p(\boldsymbol{\xi})| + |\alpha_I w_I^p r_I^p(\boldsymbol{\xi})| \,. \tag{17}$$

Note that $\bar{r}$ actually represents the mean of the pre-weighted residuals. This computation is performed before each iteration of the IRLS solver described in Section IV-C (for the first iteration $\boldsymbol{\xi}$ is assumed to be null). Afterwards, to provide robust estimates we set $c = 0.5\,\bar{r}$.

On the other hand, for $\hat{c}$ we select a lower value in order to segment out dynamic parts more aggressively. The reason to do that is that false positives (static regions segmented as dynamic) are preferable to false negatives (dynamic regions segmented as static). Moreover, we set this threshold to be even lower during the initialization phase to be sure that the initial map is built only using static parts of the scene:

$$\hat{c} = \begin{cases} \max(r_{min}, \bar{r}) & \text{During initialization} \\ 1.5\max(r_{min}, \bar{r}) & \text{Otherwise} \end{cases} . \tag{18}$$

The variable r_{min} sets the minimum residual value below which clusters should always be segmented as static. When images are perfectly aligned the mean residual $\bar{c}$ is very low and the threshold $1.5\,\bar{c}$ would also be very low, which would lead to having static clusters segmented as being dynamic — irrespective of how precise the alignment is. The introduction of r_{min} solves this problem.

VII. Evaluation

We perform a thorough evaluation of our approach in static and dynamic environments. First, results are presented for several sequences of the Freiburg dataset [21]. This dataset has a ground truth of the camera trajectory, which allows us to measure both relative and absolute drift. The Freiburg dataset contains a small number of sequences with high dynamics and, for that reason, we include additional sequences recorded with a hand-held camera to provide a more general evaluation with varied scenes and also on longer trajectories.

	Sequence	Trans. RPE RMSE (cm/s) VO-SF	EF	CF	BaMVO	SF	Rot. RPE RMSE (deg/s) VO-SF	EF	CF	BaMVO	SF
Static Env.	fr1/xyz	2.1	**1.9**	2.3	–	2.3	1.00	**0.91**	1.34	–	1.42
	fr1/desk	3.7	**2.9**	9.0	–	3	1.77	**1.48**	4.49	–	2.17
	fr1/desk2	5.4	7.2	9.2	–	**5**	**2.45**	4.07	4.79	–	3.39
	fr1/plant	6.1	**5.0**	8.9	–	10.4	2.00	**1.58**	3.02	–	3.16
Low Dynamic Env.	fr3/sit_static	2.4	**0.9**	01.1	2.4	1.1	0.71	**0.30**	0.44	0.69	0.43
	fr3/sit_xyz	5.7	**1.6**	2.7	4.8	2.8	1.44	**0.59**	1.00	1.38	0.92
	fr3/sit_halfsphere	7.5	17.2	**3.0**	5.8	**3.0**	2.98	4.56	**1.92**	2.88	2.11
High Dynamic Env.	fr3/walk_static	10.1	26.0	22.4	13.3	**1.3**	1.68	4.77	4.01	2.08	**0.38**
	fr3/walk_xyz	27.7	24.0	32.9	23.2	**12.1**	5.11	4.79	5.55	4.39	**2.66**
	fr3/walk_halfsphere*	24.8	16.3	31.1	–	**5**	5.49	5.70	8.45	–	**2.18**
	fr3/walk_halfsphere	33.5	20.5	40.0	**17.3**	20.7	6.69	6.41	13.02	**4.28**	5.04

TABLE I: Relative Pose Error

	Sequence	Trans. ATE RMSE (cm) VO-SF	EF	CF	SF
Static Env.	fr1/xyz	5.1	**1.2**	1.4	1.4
	fr1/desk	5.6	**2.1**	17.7	2.3
	fr1/desk2	17.4	5.7	16.8	**5.2**
	fr1/plant	7.8	**5.3**	12.6	11.3
Low Dynamic Env.	fr3/sit_static	2.9	**0.8**	1.1	1.3
	fr3/sit_xyz	11.1	**2.2**	2.7	4.0
	fr3/sit_halfsphere	18.0	42.8	**3.6**	4.0
High Dynamic Env.	fr3/walk_static	32.7	29.3	55.1	**1.4**
	fr3/walk_xyz	87.4	90.6	69.6	**12.7**
	fr3/walk_halfsphere*	48.2	48.6	75.6	**6.3**
	fr3/walk_halfsphere	73.9	63.8	80.3	**39.1**

TABLE II: Absolute Trajectory Error

We compare the accuracy of our method, StaticFusion (SF), against related state-of-the art approaches:

1) The joint visual odometry and scene flow of Jaimez *et al*. [14] (VO-SF). As an odometry method designed for dynamic scenes, a comparison is useful to investigate the benefits of reconstructing a 3D model of the static scene.
2) ElasticFusion [1] (EF) in order to investigate the performance of a state-of-the-art method designed for static environments within dynamic scenes.
3) Co-Fusion [16] (CF), as it is a state-of-the-art approach for tracking and reconstructing multiple moving objects.
4) The background model-based visual odometry of Kim *et al*. [15] (BaMVO), which is conceptually related to our approach but uses a multi-frame strategy instead of frame-to-model alignment. Since we were unable to replicate the published results using the open-source release of this method, we only include numerical results for the sequences evaluated in the original publication.

The experiments were performed on a workstation with an Intel(R) Core(TM) i7-3770 CPU at 3.40GHz and a GeForce GTX 1070 GPU using the Ubuntu 16.04 operating system. We used registered RGB-D images and maintained default parameters for the methods we compare against. Regarding the image resolution, we use QVGA (320 $\times$ 240) and keep the default resolution of the other methods included in the comparison (VO-SF, BaMVO also use QVGA, the remaining ones work with VGA).

A. Quantitative Evaluation

Regarding the accuracy of pose estimation, evaluation is performed through the metrics proposed by Sturm *et al*. [21]:

- Translational and rotational relative pose error (RPE) to measure the average local drift per second.
- Translational absolute trajectory error (ATE) to measure the global quality of the trajectory.

For completeness, sequences recorded in static, low dynamic and high dynamic scenes are included within this evaluation. Results are listed in Tables I and II. It can be seen that StaticFusion's performance in static environments is comparable to that of ElasticFusion regarding both ATE and RPE criteria. Table II demonstrates the advantage of frame-to-model alignment strategies over odometry methods in reducing overall drift. In highly dynamic sequences, our system outperforms the other approaches for the following reasons:

- EF is not designed to handle dynamics in the scene. Thus, moving objects corrupt its 3D reconstruction and, in turn, its pose estimate.
- VO-SF and BaMVO are odometry-based methods which cannot recover from poor alignment or segmentation.
- CF works well for slow camera motions but its performance deteriorates noticeably when the speed of the camera increases.

As mentioned in Section VI, StaticFusion requires sequences with limited dynamics during the initialization stage of the model and for this reason it produces high errors on *fr3/walking_halfsphere*. We include the additional sequence *fr3/walking_halfsphere**, which skips the initial 5 seconds with high dynamics, in order to illustrate the ability of the method to perform well for that same scene when the initial frames are not so challenging (note that the other methods still fail in this case).

In order to illustrate how our background model handles moving objects, Fig. 3 shows the temporal evolution of the map built during the sequence *fri3/walking_static*. It can be seen that the map evolves to reflect changes in the scene. It efficiently removes moving parts, keeping only the structure which represents a valuable prior for accurate segmentation of dynamics.

B. Qualitative Evaluation with a Hand-held Camera

In this section we evaluate our approach in scenes with varying levels of dynamism. To this end, we have recorded two sequences with a hand-held RGB-D camera. In the

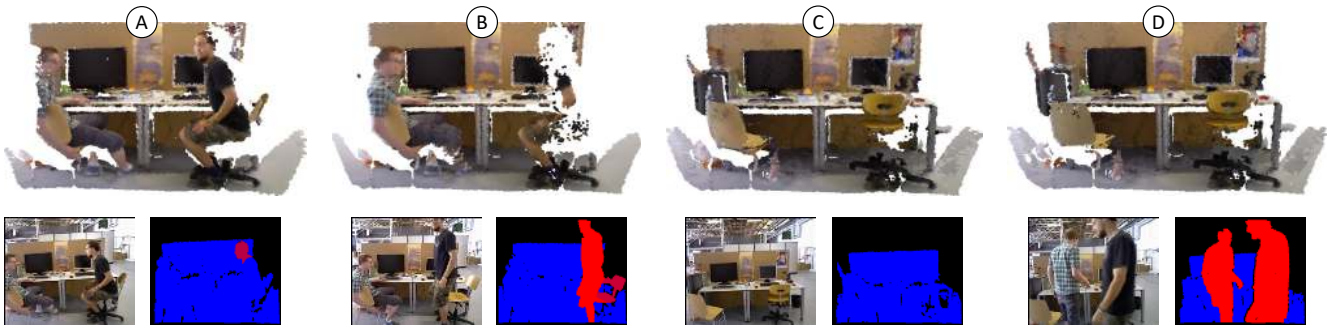

Fig. 3: **Top**: The background reconstruction evolves to reflect changes in the scene. **Bottom**: Corresponding RGB image (left) and computed segmentation (right) for each instance, where red means dynamic and blue means static. Between (A) and (C) the model adapts to remove dynamic objects based on a correct segmentation. The sequence at the instance of (D) contains significant dynamics and the model provides a valuable prior knowledge of the scene, allowing for correct segmentation and pose estimation.

first sequence the camera observes a person interacting with objects and performing varied motions at different speeds. Occasionally the person remains still and close to the camera, which poses a challenge for the tracking system because it must be able to segment out the person as soon as they move in order not to lose track of the sensor. For the second test we have recorded a 'selfie sequence' during which a person carries the camera while it points at them. This a very complex test because the person often occupies more than 50% of the image and looks quasi-static with respect to the camera (which represents a strong local minimum for the motion estimate of any method which does not segment moving parts explicitly).

Results for the first sequence are shown in Fig. 6. It can be seen that the person is initially inserted in the map because he does not move, but it is removed as soon as he moves. The segmentations estimated by StaticFusion are almost perfect during the whole sequence: the person is always segmented correctly, as well as the ball and the door when it gets closed.

The second sequence starts and finishes with the camera at the exact same position (see Fig. 5), which allows us to measure the overall drift for the full trajectory. It can be observed that our algorithm is able to provide good predictions (C_M, Z_M) even when the person covers most of the view of the camera. Moreover, it is able to track the camera accurately, which seems very complicated for a system based on direct image alignment.

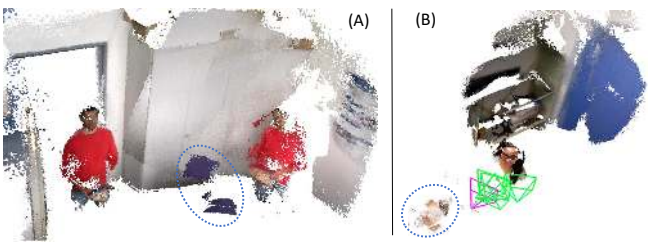

Fig. 4: (A) shows how ElasticFusion adds the person multiple times into the map and eventually fails to estimate the camera motion. In (B) Co-Fusion finds and tracks multiple independent objects which do not exist (note that the region surrounded by the blue circle is part of the person as well).

ElasticFusion [1] and Co-Fusion [16] are also tested with these sequences, and they both fail to provide good pose estimates. Figure 4 illustrates some of these failures. Figure 4 (A) shows the map reconstructed by ElasticFusion for the first sequence before it loses track of the camera (note the double chair and the misaligned walls). Figure 4 (B) shows how Co-Fusion finds too many moving objects during the second sequence and randomly keeps track of those parts of the scene, destroying the overall map. For that sequence, with an estimated trajectory lenght of 9.5 metres, the final drift of our method is just 1.5 centimetres, whereas the drifts of ElasticFusion and Co-Fusion are of 1.03 metres and 0.88 metres respectively.

VIII. CONCLUSIONS

In this paper we have described a new approach to accurately estimate the motion of an RGB-D camera in the presence of moving objects. It is based on building and exploiting a 3D background model that only contains static parts of the scene, hence its name – StaticFusion. Quantitative and qualitative results demonstrate that our approach outperforms state-of-the-art techniques when the tested sequences include several moving objects, and achieves that level of accuracy with a very competitive runtime (30 milliseconds).

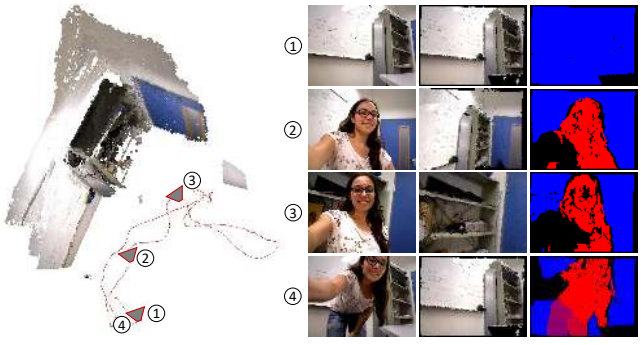

Fig. 5: **Left:** trajectory estimated by StaticFusion for the second experiment in Section VII-B. **Right:** Some of input images of this sequence (first column), together with our predictions for those same views (second column) and the estimated segmentations (third column).

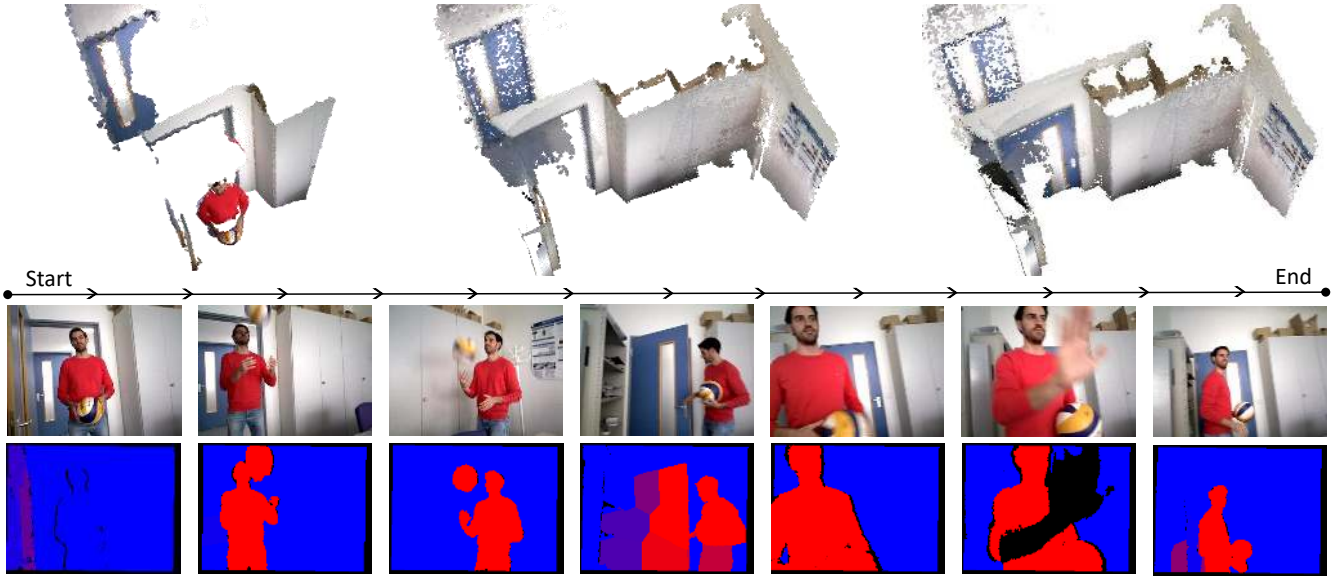

Fig. 6: **Top**: Evolution of the map built during the first sequence of Section VII-B. **Bottom**: Some of the sample input images and the segmentations that StaticFusion provides for them.

Given the rising interest in developing visual odometry and SLAM algorithms that work in very dynamic environments, in the future we plan to create a new RGB-D dataset for this purpose. This dataset should contain sequences with longer trajectories, fast and slow camera motions and varying degrees of dynamic elements in the scene. Concerning our approach, we are interested in investigating multi-frame strategies that allow us to initialize our background model even when many moving objects are observed. More efficient strategies to fuse and remove data from the map will also be explored.

REFERENCES

[1] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," *Robotics: Science and Systems (RSS)*, 2015.

[2] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE/ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[3] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

[4] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *International Conference on 3D Television (3DTV)*, 2013, pp. 1–8.

[5] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[6] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *Intl. J. of Robotics Research*, 2012.

[7] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 2021–2028.

[8] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.

[9] 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 *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 83–89.

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

[11] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, 2017.

[14] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 3992–3999.

[15] D. H. Kim and J. H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1565–1573, 2016.

[16] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 4471–4478.

[17] D. H. Kim, S. B. Han, and J. H. Kim, "Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment," in *Proc. Int. Conf. Robot. Intell. Technol. Appl.*, 2015, pp. 11–26.

[18] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon, "Direct visual SLAM fusing proprioception for a humanoid robot," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

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

[20] A. Concha and J. Civera, "An evaluation of robust cost functions for RGB direct mapping," in *European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–8.

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