2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Speed and Memory Efficient Dense RGB-D SLAM in Dynamic Scenes

Bruce Canovas[1], Michèle Rombaut[1], Amaury Nègre[1], Denis Pellerin[1] and Serge Olympieff[1]

Abstract— **Real-time dense 3D localization and mapping systems are required to enable robotics platforms to interact in and with their environments. Several solutions have used surfel representations to model the world. While they produce impressive results, they require heavy and costly hardware to operate properly. Many of them are also limited to static environments and small inter-frame motions. Whereas most of the state of the art approaches focus on the accuracy of the reconstruction, we assume that many robotics applications do not require a high resolution level in the rebuilt surface and can benefit from a less accurate but less expensive map, so as to gain in run-time and memory efficiency. In this paper we propose a fast RGB-D SLAM articulated around a rough and lightweight 3D representation for dense compact mapping in dynamic indoor environment, targeting mainstream computing platforms. A simple and fast formulation to detect and filter out dynamic elements is also presented. We show the robustness of our system, its low memory requirement and the good performance it enables.**

I. INTRODUCTION

Robust and dense simultaneous localization and mapping (SLAM) algorithms are a fundamental necessity for indoor mobile robots to operate in their environment. They have to provide sufficient accuracy and fast run-time performance coupled to low memory requirement so as to be implemented on less powerful devices. Low-cost RGB-D cameras are interesting in this context as they allow joint acquisition of dense depth and texture information at high framerate with reduced power consumption. Several real-time dense RGB-D SLAM solutions able to produce impressive results have been proposed. They usually perform dense direct frame-to-model registration to track the camera motion and integrate newly acquired RGB-D data into a 3D map of the observed scene. They rely on loop closure detection to correct drift accumulated due to sensor noise.

However various limitations are shared by many of these methods. They tend to operate with highly accurate but too expensive forms of 3D representation to model the map which forces them to rely on heavy hardware to run online and limits the scalability of the reconstruction. Furthermore the dense registration used to calculate the pose of the camera is often based on a coarse-to-fine iterative closest point (ICP) scheme failing in case of large shift between consecutive frames. Traditional dense RGB-D SLAM algorithms also assume static scenarios and thus performs poorly when working in dynamic environments.

In this paper we propose a novel online dense RGB-D SLAM system in indoor environment that intends to tackle some common bounds shared by traditional frameworks such as static scene and slow inter-frame motion assumptions as well as restricted scalability and reduced consistency. We promote execution and memory efficiency rather than highly accurate model reconstruction, which are decisive criteria to enable dense SLAM algorithms to run in real time on mobile robots. Our method integrates an approximate compact 3D representation, which we recently developed, for fast and lightweight mapping. The proposed SLAM relies on planar patches called supersurfels, generated from superpixels, to model the static part of the environment. We use both sparse feature-based visual odometry (VO) and dense frame-to-model registration for robust camera tracking. Sparse VO is able to deal with strong motion and flat textured area whereas the dense registration performs well in scenes with little texture but rich geometry. Map consistency is maintained with deformation-based loop closure. Besides the integration of a coarse form of 3D representation to a complete dense RGB-D SLAM pipeline, an other contribution of our paper is the addition of a dense moving object detection strategy combining egomotion compensation, depth information and dense optical flow for robustness against dynamic scenes.

II. RELATED WORK

Most of the actual dense 3D mapping systems are based on volumetric representation, popularized by KinectFusion [1]. They build a high-quality 3D model represented as a truncated signed distance function stored in a voxel grid. These methods produce accurate continuous surface reconstruction but suffer from huge memory footprint and are thus limited too small environments. To reduce the memory consumption [2] proposes to use voxel hashing. Kintinuous [3] extends the size of the reconstruction with a cyclical buffer to free voxels. Besides low scalability, volumetric representations lack flexibility: the resolution of the voxel is fixed limiting the adaptiveness and deforming them is too expensive. Therefore they are not suitable for live correction through loop closure.

Surfel-based methods are also widespread due to their lower computational complexity and better scalability. Surfels [4] are simply oriented disks. They offer better adaptivness compared to voxels since their resolution is bound to the accuracy of the sensor and better flexibility as they can be updated efficiently and independently. One of the first dense RGB-D mapping method to use surfels is the one of *Keller et al.* [5]. ElasticFusion [6] extends [5] by incorporating color information in the frame-to-model registration to be more robust in flat scenes and by employing a deformation graph to apply loop closure correction on the map instead of pose

[1]Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France {f_author, s_author}@gipsa-lab.grenoble-inp.fr

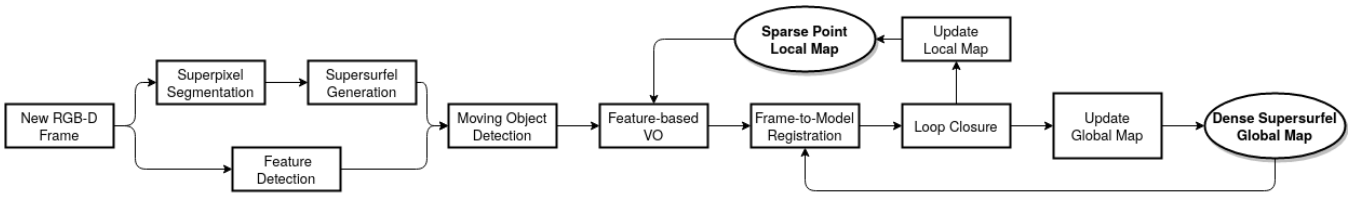

Fig. 1: Overview of the different steps of the proposed dense mapping RGB-D system.

graph optimization backend. Surfel representations are easy to generate, to fuse and to deform. However they are discrete and thus more prone to noise and redundancies. Surfel maps are often composed of millions of surfels, requiring extremly powerful GPU support to be processed online. [7] and [8] generate surfel-like primitives using superpixels for the mapping so as to enhance speed and memory efficiency but, unlike us, they rely on an external sparse visual SLAM to localize.

Previous methods are not designed to cope with dynamic elements that can corrupt the estimated camera trajectory and reconstructed map. They rely on the static world assumption and thus fail in dynamic scenes. To address this problem [9] and [10] exploit residuals and geometric constraints to detect and remove moving elements to build a dense map of the static part of the environment. However [9] does not run live and [10] uses QVGA resolution to reach online performance. Other dense approaches like [11] track and rebuild simultaneously the background and each dynamic object detected. [12] and [13] use neural networks for detection of the a priori dynamic objects and instance segmentation. Although they provide good accuracy, they are computationally too expensive to be used on low-cost platforms.

III. SYSTEM OVERVIEW

A. System Architecture

Figure 1 provides an overview of the proposed SLAM system. It takes as input live registered RGB-D frames (C,D), with C the color and D the depth, to simultaneously rebuild a dense global 3D map M of the static part of the environment and estimate the pose $T \in SE(3)$ of the sensor in global space. The camera pose T consists of orientation $R \in SO(3)$ and position $t \in \mathbb{R}^3$.

The global map M is modeled as a compact list of 3D primitives called supersurfels [7]. First a set F of supersurfels F^l is extracted from the superpixel segmentation of the current frame and sparse features are detected in a parallel thread. Fast moving object detection at a superpixel level based on robust ego-motion compensation and Dense Inverse Search (DIS) optical flow [14] is then used to extract and discard dynamic elements from the following camera tracking and mapping processes. The tracking step is initialized with a lightweight sparse feature-based VO and refined using a dense frame-to-model registration to get a robust and accurate estimate of the current camera pose. The sparse VO which aligns detected interests points to a small sparse local map of features is built on the implementation in [15] improved with ORB features and GMS matching [16]. Dense registration refinement attempts to align 3D points measurements from the current frame to supersurfels M^s from the dense global map M based on symmetric point-to-plane ICP [17]. Finally registered detected features and supersurfels F^l from the current frame are integrated respectively in the sparse local map and the dense global map M. Loop closures are applied in real time to correct local and global maps using non-rigid space deformation strategy, as done by ElasticFusion [6].

B. Supersurfel Representation

An approximate but coherent and lightweight representation [7] that we developed is used to model the environment. The dense global map M is rebuilt as an unordered list of supersurfels M^s, which are simply oriented elliptical planar patches. They can be seen as approximate 2D to 3D backprojections of superpixels. Superpixels [18] are groups of pixels that are homogeneous in color and geometry. Whereas surfel-based methods model per-pixel surfels to map the environment, supersurfels 3D primitives are generated from RGB-D superpixels, producing a lower resolution surface reconstruction. This easy to manage 3D data representation allows faster processing as well as reduced memory footprint and noise while preserving the important information (texture and depth discontinuities). Each map supersurfel M^s is described by the following attributes: $M^s_p \in \mathbb{R}^3$ its positions, $M^s_c \in \mathbb{R}^3$ its color, $M^s_R \in SO(3)$ its orientation, M^s_L and $M^s_l \in \mathbb{R}$ longitudinal and lateral elongations, $M^s_\Sigma \in \mathscr{M}_3(\mathbb{R})$ a covariance matrix describing its shape, $M^s_w \in \mathbb{R}_+$ a confidence weight to quantify its reliability and M^s_t, $M^s_{t_0} \in \mathbb{R}_+$, last update and initialisation timestamps. For convenience we also designate the normal of a supersurfel as M^s_n, corresponding to the third column of its orientation.

IV. SYSTEM DESCRIPTION

A. Supersurfel-based Mapping

1) Supersurfel Generation: The input frame is segmented into superpixels preserving as much as possible texture boundaries and depth discontinuities following a GPU implementation of the method presented in [19], which assigns to each segment a slanted plane in 3D. Then a supersurfel F^l is extracted for each superpixel segment (see Figure 2) by applying Principal Component Analysis (PCA) on the 2D to 3D backprojections of the pixels contained by the superpixels, as explained in [7].

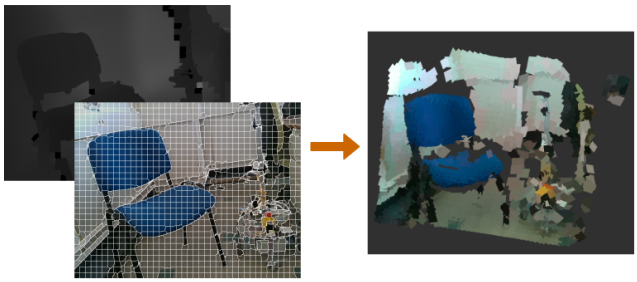

Fig. 2: Superpixels (left) with associated slanted planes and generated supersurfels (right), displayed as rectangular patches instead of elliptical patches for faster visualization.

2) Global Map Update: In this step, frame supersurfels are integrated into the global map M. Corresponding frame and model supersurfels are fused so as to refine the reconstruction and reduce redundancies, whereas frame supersurfels with no correspondence are simply added with a low confidence weight in the map. To find possible correspondences between supersurfels from the map M and from the current frame F, we use a projective data association strategy. The position M_p^s of each model supersurfel M^s in the field of view of the camera is projected into the superpixel segmented RGB-D frame and associated to the frame supersurfel F^l related to the superpixel containing the position of the projection. We then check similarity between M^s and F^l, ensuring that the distance between their colors in Lab space, the divergence angle between their normals and the euclidean distance between their centers are small, in order to avoid fusing different supersurfels. Similar supersurfels are merged together with weighted average and covariance intersection strategies in an update scheme similar to [7]. The confidence value M_w^s is incremented.

Finally, the model is cleaned by performing a fast free-space violation check to remove supersurfels in front of viable supersurfels, which are often related to dynamic objects. Supersurfels that stay in an unstable state (with low confidence value) for too long are also eliminated.

B. Camera Tracking

1) Sparse Feature-based Visual Odometry: First, sparse VO from ORB features [20] is applied to get an initial estimate of the camera pose. Unlike direct geometric registration methods, sparse feature-based VO performs well in flat textured environments and is also more robust in the case of strong inter-frame motion. We chose to rely on ORB features because they are fast to match and extract and possess good invariance to viewpoint properties. When a frame is captured, ORB features are extracted and the depth image served as a mask to reject points with invalid depth.

Detected features are then used to register the new frame against a limited sparse local map of features created over the past frames, by minimizing the reprojection error between newly extracted 2D features points and the matched 3D local map points:

$$\{R,t\} = \underset{R,t}{argmin} \sum_i \rho(\|x_i - \pi(RX_i + t)\|^2), \qquad (1)$$

with $x_i \in \mathbb{R}^2$ positions of the 2D features, $X_i \in \mathbb{R}^3$ the matched 3D map points and $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ the pinhole projection model, determined by the intrinsic camera parameters. ρ is the robust Cauchy cost function to deal with correspondence outliers. We based our local map management implementation on the lightweight visual tracking (LVT) system [15], a visual odometry algorithm designed for real-time operation with low computational overhead and memory requirements. The use of a local map allows to greatly reduce the accumulation of errors, with comparison to standard frame-to-frame methods. When a feature is no longer trackable it is cleared from the local map, so as to keep track of a limited number of interest points.

To perform fast and high quality matching between current frame 2D features and projected local map features, Grid-based Motion Statistics (GMS) [16] is applied after GPU brute force matching. GMS depends on a statistical formulation to distinguish false and true matches based on the number of neighboring matches. It presents good results even in weakly textured environment or strong motion scenarios. If the number of matches found is too small the initial estimate of the camera pose is only predicted following a constant velocity model based on the two previous poses.

2) Dense Frame-to-Model Registration: Next dense registration is employed to refine the initial pose estimate T We compute in global space the set of 3D points X_i associated to the input RGB-D frame pixels and assign to each of these points X_i a normal $n_i = RF_n^l$, with F_n^l the normal of the frame supersurfel F^l containing the point X_i and R the rotational part of the current pose estimate T. The set of 3D points X^i is then aligned to the supersurfels from the model that lie in the field of view of the camera.

The fast and reliable symmetric ICP variant presented in [17] is applied. It is a simple improvement to traditional point-to-plane ICP which produces faster and more reliable convergence. The rotation R_{rel} and translation t_{rel} parts of the relative transformation T_{rel} from the previous camera pose to the current one are computed using Gauss-Newton to minimize the following cost function:

$$E_{symm} = \sum_{\mathscr{A}_{i,s}} [(R_{rel}X_i - R_{rel}^{-1}M_p^s + t_{rel}).(n_i + M_n^s)]^2. \qquad (2)$$

$\mathscr{A}_{i,s}$ is the list of associations between the 3D points from the current frame and the supersurfels from the global model. The pose of the camera is updated after ICP: $T \leftarrow TT_{rel}$

Closest correspondences $\mathscr{A}_{i,s}$ are computed at each iteration of the ICP algorithm with fast projective data association. The centers M_p^s of model supersurfels are projected into the current frame and the 3D point X^i associated to the projection position is selected for the pairing. Correspondences with a high depth difference, a high color dissimilarity in Lab space, or a large angle between surface normals are rejected.

C. Moving Object Detection

1) Ego-motion Compensation: To detect dynamic elements, we chose to model the camera ego-motion in image space as a 2D perspective transformation matrix $H \in SE(2)$

because ego-motion compensation in 3D space is more computationally demanding and less robust due to noisy depth measurements. ORB features detected from the previous frame (C_{prev}, D_{prev}) and from the current one (C, D) are used in combination with RANSAC to calculate the 2D transformation H that relates the two images. We assume that static elements are predominant in the scene viewed by the camera, otherwise RANSAC might select as inliers a set of dynamic features and the estimated transformation H would not reflect the real camera motion. After that, the intensity image I associated to the color image C of the current RGB-D frame is computed as well as the previous intensity I_{prev} related to C_{prev}. The calculated perspective transformation H is then applied to I_{prev} and D_{prev} to generate two new images: the warped intensity I_{warp} and depth D_{warp} images that in case of a correct estimation and a fully static scene should look like I and D.

2) Static / Dynamic Segmentation: In a second step we calculate dense optical flow between the current image I and the predicted image I_{warp}. We use the DIS optical flow algorithm [14] to estimate the apparent motion for each pixel because it is able to run at up to 600Hz on a single CPU core and reaches state-of-the-art performance with large displacements. A 2D displacement vector Δx is then assigned to each superpixel of the current frame by averaging those of the pixels inside it. Using superpixels as rigidly moving elements allows to speed up the classification and to reduce the influence of erroneous values. We used adaptive thresholding on the magnitudes of the superpixel flow vectors and on the compensated depth differences, computed using D and D_{warp}, to detect moving segments. A superpixel with centroid x is classified as as dynamic if:

$$\|\Delta x\|_2 > \tau_f \qquad \text{and} \qquad |D_{warp}(x) - D(x)| > \tau_d. \tag{3}$$

τ_f and τ_d are two adaptive thresholds. τ_d takes into account the uncertainty of the depth sensor and is determined by:

$$\tau_d = k\sigma_d(x), \tag{4}$$

with k a constant set to 10 and $\sigma_d(x)$ the standard deviation of the depth noise for the pixel x. The threshold τ_f is defined in [21] as:

$$\tau_f = \alpha + \beta\sqrt{H(1,3)^2 + H(2,3)^2}, \tag{5}$$

where $\alpha = 1.0$ is set to compensate the destabilization caused by the resolution of the sensor or the precision of the optical flow and $\beta = 0.5$ weights the magnitude of the perspective transformation matrix elements $H(1,3)$ and $H(2,3)$ reflecting the speed of the sensor motion. To limit false detections due to inaccurate optical flow or depth data we set as static superpixels that have been labeled as dynamic but are only connected to static segments. A result of the moving object detection is shown in Fig. 3. Features and supersurfels associated to dynamic superpixels are rejected.

Fig. 3: An example of the moving object detection.

D. Loop Closure

1) Detection: We use randomized fern [22] for keyframe database management and to detect loop closure candidates since fern descriptors are really fast to compute and to compare. If an image similar to the current frame is detected in the keyframe database, ORB features from the current frame and the candidate keyframe are matched using GMS [16]. If there are enough matches, the transformation T_{LC} between the two images is calculated with the EPnP [23] solution in combination with RANSAC to remove outliers. If RANSAC gives less than 30% of inliers, the loop closure is discarded. The transformation T_{LC} is then refined with dense ICP registration between the candidate keyframe's supersurfels and the current frame (as in section IV-B.2).

2) Map Deformation: If the loop closure is accepted, a deformation graph embedded in the surface of the global model is constructed and optimised to non-rigidly deform the dense map so as to recover from accumulated drift. Similarly to ElasticFusion [6] we follow the embedded deformation formulation introduced by [24] and adapted it to our supersurfel-based representation.

Sumner et al. [24] define a deformation graph G as a set of nodes G^j and edges sparsely and uniformly distributed through a dense 3D model to deform: here the global map M. The nodes are sampled from the supersurfels M^s of the map. A node G^j is defined by a timestamp $G^j_{t_0} \in \mathbb{R}_+$, a position $G^j_p \in \mathbb{R}^3$, initialized with position M^s_p and initialization timestamp $M^s_{t_0}$ of its associated supersurfel M^s. It also stores an affine transformation composed of a rotation matrix $G^j_R \in SO(3)$ and a translation vector $G^j_t \in \mathbb{R}^3$ as optimization parameters. Each node is connected to its k neighbors $\mathcal{N}(G^j)$ nearest in time. In our implementation $k = 4$. Affine transformations of the nodes are computed from the optimization of an objective function that encourages smooth rigid deformation given a set of sparse positional loop closure constraints (see [6]).

After optimizing the deformation graph, the nodes are used to deform the supersurfels in the map M. Each supersurfel M^s in the global map is influenced by a set of neighboring deformation graph nodes $\mathcal{I}(M^s, \mathcal{G})$, collected as the k nearest nodes with regard to the euclidian distance, among a bigger set of α nodes having timestamps $G^j_{t_0}$ nearby supersurfel initialization timestamp $M^s_{t_0}$. The deformed position of a supersurfel M^s is calculated using the combination of optimized rigid transformations from its associated nodes $\mathcal{I}(M^s, \mathcal{G})$:

$$M^s_p \leftarrow \sum_{v \in \mathcal{I}(M^s, \mathcal{G})} w^v(M^s)[G^v_R(M^s_p - G^v_p) + G^v_p + G^v_t], \tag{6}$$

where $w^v(M^s) = (1 - \|M_p^s - G_p^v\|/d_{max})^2$, with d_{max} the distance to the k+1 closest node. The orientation and covariance of the supersurfel are updated according to:

$$M_R^s \leftarrow \bar{R} M_R^s \qquad \text{and} \qquad M_\Sigma^s \leftarrow \bar{R} M_\Sigma^s \bar{R}^T, \tag{7}$$

where $\bar{R}$ is the blended rotation calculated as the weighted average of quaternions representations of the rotations of the nodes $\mathscr{I}(M^s, \mathscr{G})$ influencing the supersurfel.

The deformation graph is also applied to deform the points from the local map used by the sparse tracking (Section IV-B.1). The current camera pose is corrected $T = T_{LC} T_{KF}$, where T_{KF} is the pose of the keyframe.

V. EXPERIMENTAL RESULTS

We perform quantitative and qualitative evaluation of our method and we compare it against other dense RGB-D SLAM systems: Co-Fusion (CF) [11], StaticFusion (SF) [10], ReFusion (RF) [9], all designed to deal with dynamic scenes and ElasticFusion (EF) [6] which is designed for static scenes. ElasticFusion, StaticFusion and Co-Fusion are surfel-based mapping methods while ReFusion uses voxel hashing volumetric representation. We use sequences from the TUM dataset [25], providing ground truth camera trajectories, to measure the camera tracking accuracy and the computational performance of the developed SLAM algorithm. The accuracy of the surface reconstruction is evaluated too on the sequences from the ICL-NUIM dataset [26].

The system is tested on a mainstream laptop with an Intel Core i5-6300HQ CPU at 2.3GHz, 6GB of RAM and an Nvidia GeForce GTX 950M GPU with 4GB of memory. It achieves around 15 FPS with VGA images. The segmentation in superpixels, the dense tracking and the global map update are performed on the GPU, whereas feature-based VO, moving object detection and loop closure are conducted on CPU. We use a superpixel size of around 256 pixels that delivers a good balance between efficiency and accuracy.

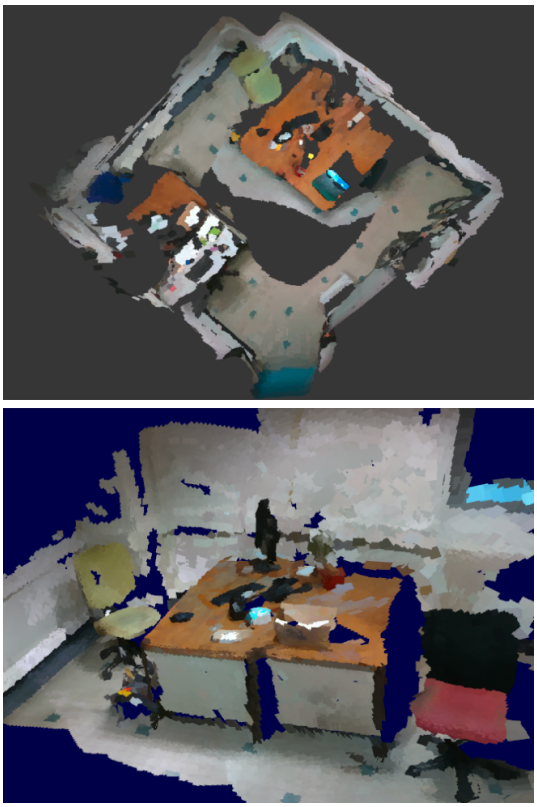

Fig. 4: Scans of an office environment captured in real-time.

A. Trajectory Estimation

TABLE I: Absolute Trajectory Error (RMS) on TUM dataset.

	Sequence	Trans. ATE RMSE (cm)				
		EF	CF	SF	RF	Ours
Static Env.	fr1/xyz	**1.2**	1.5	1.5	2.4	3.1
	fr1/room	**22.6**	26.9	51.8	76.5	27.3
	fr3/office	**2.3**	68.3	31.0	11.6	8.8
Dynamic Env.	fr3/sit_xyz	**2.4**	3.9	3.9	3.7	4.5
	fr3/walk_xyz	73.8	69.8	9.3	**8.7**	21.0
	fr3walk_halfsphere	55.0	82.0	68.0	**10.8**	16.7

Table I presents the results of the camera tracking accuracy evaluation. We use the Absolute Trajectory Error (ATE) [25] to measure the global quality of the estimated trajectories with regard to ground truth trajectories in 6 video sequences: three sequences in static environments (*fr1/xyz*, *fr1/room*, *fr3/office*), one in low dynamic environment (*fr3/sit_xyz*) and two in highly dynamic environment (*fr3/walk_xyz*, *fr3/walk_halfsphere*). Our system shows good overall results, on par with the others in static environment. It demonstrates consistency in long video sequences (*fr3/office*) and state of the art accuracy when encountering fast motions (*fr1/room*). The moving object detection strategy greatly enhances the stability of the system in highly dynamic scenarios compared to ElasticFusion, which can not handle dynamic elements and Co-Fusion whose accuracy heavily deteriorates when encountering objects moving too fast. No significant improvement is shown in low dynamic scenes.

B. Surface Estimation

TABLE II: Reconstruction accuracy on ICL-NUIM dataset.

System	Surface Accuracy (cm)			
	kt0	kt1	kt2	kt3
EF	**0.7**	**0.7**	**0.8**	2.8
Ours	1.3	5.2	1.1	**1.4**

Although accuracy is not the purpose of our method, the 3D map built has to be relevant. We compare the quality of the surface reconstructed by our approach to those of ElasticFusion on the living room scene of the ICL-NUIM dataset which provides four synthetic noisy RGB-D sequences and a ground truth 3D model. To evaluate the surface produced we convert our supersurfel-based global map to a dense point cloud by oversampling the surface of each supersurfel. We then compute the mean distance from each point of the obtained point cloud to the nearest surface of the ground truth 3D model. Results presented Table II show that even if our algorithm uses a rough form of 3D representation, a certain level accuracy can still be maintained in simple indoor

TABLE III: Mean runtime and maximal memory footprint of the dense static global map.

Sequence	Map Size (MB)					Runtime (ms)				
	EF	CF	SF	RF	Ours	EF	CF	SF	RF	Ours
fr1/xyz	14.4	13.2	4.5	166.5	**1.8**	72.6	83.0	64.2	297.6	**56.2**
fr3/office	57.8	42.5	18.6	785.6	**4.8**	84.9	125.0	77.3	480.6	**61.3**

environment. However it fails to well model thin details and important curves. Additional qualitative results are presented Figure 4.

C. Computational Performance

In Table III we show the computational efficiency evaluation of our method on two TUM sequences: *fr1/xyz*, a short video and *fr3/office* which is longer. The maximal memory footprint of the rebuilt static global map and the average execution time for the different systems are given. The use of a low resolution map representation makes our approach outperform the others in terms of memory consumption and speed, enabling extended scalability. Our map is more compact and thus easier to manage. StaticFusion achieves better performance compared to the other surfel-based methods because it processes downsized images while ReFusion demonstrates slow execution and huge memory footprint due to the volumetric representation.

VI. CONCLUSION

In this paper we described a nnew complete dense RGB-D SLAM pipeline. This system favours efficiency over highly detailed reconstruction, representing the environment as a set of patches extracted from superpixels. State-of-the-art camera tracking accuracy is reached thanks to a combination of sparse feature-based VO and dense frame-to-model registration, while robustness in dynamic scenes is ensured by a dense moving object detection framework using compensated optical flow and depth difference. Experiments showed the lower memory requirement and faster processing speed of our method compared to others, enabling it to run live on a wider range of computing platforms.

For future work we plan to port the designed system to a mobile robot embedding an Nvidia Jetson board and to exploit the supersurfel-based representation for navigation. We would also like to combine our approach with a lightweight deep learning strategy to enhance the moving object detection.

REFERENCES

[1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE ISMAR*. IEEE, October 2011.

[2] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, January 2013.

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

[4] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels-surface elements as rendering primitives," in *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 2000, pp. 335–342.

[5] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," 06 2013, p. 8.

[6] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," in *Proceedings of Robotics: Science and Systems*, July 2015.

[7] B. Canovas, M. Rombaut, A. Nègre, S. Olympieff, and D. Pellerin, "A coarse and relevant 3d representation for fast and lightweight rgb-d mapping," in *VISAPP 2019 - International Conference on Computer Vision Theory and Applications*, Feb. 2019.

[8] K. Wang, F. Gao, and S. Shen, "Real-time scalable dense surfel mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[9] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals," *arXiv*, 2019.

[10] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 3849–3856.

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

[12] T. Zhang and Y. Nakamura, "PoseFusion: Dense RGB-D SLAM in Dynamic Human Environments," in *2018 International Symposium on Experimental Robotics*, Nov. 2018.

[13] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Oct 2018, pp. 10–20.

[14] T. Kroeger, R. Timofte, D. Dai, and L. V. Gool, "Fast optical flow using dense inverse search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[15] M. Aladem and S. Rawashdeh, "Lightweight visual odometry for autonomous mobile robots," *Sensors*, vol. 18, p. 2837, 08 2018.

[16] J. Bian, W.-Y. Lin, Y. Matsushita, S.-K. Yeung, T. D. Nguyen, and M.-M. Cheng, "Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[17] S. Rusinkiewicz, "A symmetric objective function for icp," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019.

[18] Ren and Malik, "Learning a classification model for segmentation," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 10–17 vol.1.

[19] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *ECCV*, 2014.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," 11 2011, pp. 2564–2571.

[21] J. Huang, W. Zou, J. Zhu, and Z. Zhu, "Optical flow based real-time moving object detection in unconstrained scenes," 2018.

[22] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, pp. 571–583, 2015.

[23] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.

[24] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," *ACM Trans. Graph.*, vol. 26, p. 80, 2007.

[25] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[26] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, May 2014.