# Real-Time and Scalable Incremental Segmentation on Dense SLAM

Keisuke Tateno[1,2] and Federico Tombari[1,3] and Nassir Navab[1]

***Abstract*— This work proposes a real-time segmentation method for 3D point clouds obtained via Simultaneous Localization And Mapping (SLAM). The proposed method incrementally merges segments obtained from each input depth image in a unified global model using a SLAM framework. Differently from all other approaches, our method is able to yield segmentation of scenes reconstructed from multiple views in real-time, with a complexity that does not depend on the size of the global model. At the same time, it is also general, as it can be deployed with any frame-wise segmentation approach as well as any SLAM algorithm. We validate our proposal by a comparison with the state of the art in terms of computational efficiency and accuracy on a benchmark dataset, as well as by showing how our method can enable real-time segmentation from reconstructions of diverse real indoor environments.**

## I. INTRODUCTION AND RELATED WORK

Scene segmentation is one of the most important and researched topics in the field of robotic perception, being segmentation a typical pre-requisite for several robotic tasks such as object modeling and object recognition [1], autonomous grasping and manipulation of objects [2], object tracking [3], scene understanding and object discovery of unknown environments [4]. Within the robotic perception and computer vision communities, a great effort has been made to develop efficient 3D segmentation algorithms, i.e. real-time processing of depth maps obtained from RGB-D and 3D sensors. The focus on 3D data is motivated by the additional insight that geometry and shape provide with respect to texture and color only for the task of segmentation, as well as the opportunity to determine segments that lie in the 3D space in front of the robot and not just on the image plane. Fast real-time segmentation of depth maps has been recently investigated by the works of Uckermann et al. [5], [6], Pieropan et al. [7] and Abramov et al. [8].

Recently, 3D reconstruction methods which aim at registering together, in real-time, depth maps from multiple viewpoints obtained from a moving sensor are becoming increasingly exploited for higher level robotic perception tasks, since they offer additional information for the surrounding environment and are fundamental for robot navigation tasks: this is the case of Kinect Fusion[9], as well as dense SLAM [10], [11], [12]. While the former method yields a 3D mesh of the reconstructed environment, obtained by exploiting a specific data representation internally deployed, the output of SLAM methods is generally in the form of a 3D point cloud: in both cases, the size of such reconstruction tends to increase with the number of merged depth maps.

Fig. 1. Real-time incremental segmentation applied to the reconstruction of a kitchen (top) and a table-top scene (bottom). As witnessed by these examples, the proposed segmentation method can adapt to different scenarios and can yield accurate segmentation since it applies to scenes reconstructed by merging multiple views via SLAM, this usually providing more consistent, reliable and meaningful segments than the single-view case.

As a consequence, in addition to segmentation methods aimed at processing single depth maps, some work has recently addressed the problem of segmenting 3D reconstructions obtained via Kinect Fusion or SLAM. Toward this goal, segmentation methods specifically devised to work on 3D meshes [13] or point clouds [14], [15], [16] are generally deployed to yield a segmentation of such 3D representations, as proposed in the object discovery approach of [4]. One main limitation of such methods is clearly the computational cost, since they cannot run in real-time: this aspect strongly limits their use in those application scenarios characterized by real-time constraints, as it is the case in most of the aforementioned robotic perception tasks. In addition, their computational burden tends to increase with the size of the point cloud or 3D mesh. Hence, to limit the

[1] Chair for Computer Aided Medical Procedures (CAMP), TU Munich, Boltzmannstr. 3, 85748 Munich (Germany) {tateno, tombari, navab}@in.tum.de

[2] Canon Inc., Shimomaruko, Tokyo (Japan) tateno.keisuke@canon.co.jp

[3] Dipartimento di Informatica: Scienza e Ingegneria (DISI), University of Bologna, V.le del Risorgimento 2, 40136 Bologna (Italy) federico.tombari@unibo.it

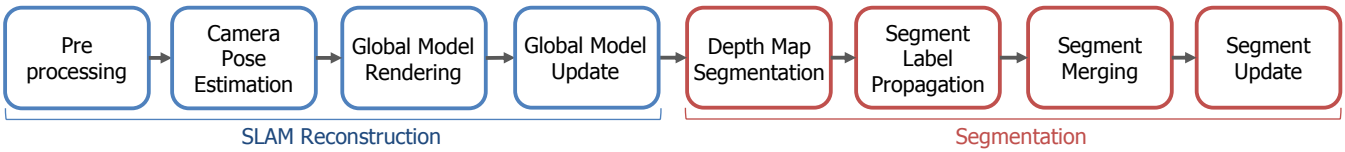



Fig. 2. Flow diagram of the proposed incremental segmentation pipeline applied at each input depth map.

overall computational requirements, up to a certain number of merged depth maps can be deployed with off-the-shelf hardware.

Aiming for the same goal, [17] proposes incremental object segmentation in a dense RGB-D SLAM framework. The method is based on Kintineous [10], a dense RGB-D SLAM method which builds upon KinectFusion [9], and relies on a 3D representation called Truncated Sign Distance Function (TSDF). In this method, newly merged depth frames in the TSDF are, from time to time, extracted in the form of "slices" (i.e., a 3D mesh) according to the estimated camera position, segmented via graph-based segmentation [13], and merged into a global segmentation map. Although yielding a much higher efficiency than previous approaches, the use of a segmentation method such as [13] on each "slice" (represented as a 3D mesh), as well as the fact that the segments extracted from each slice are successively merged in the global segmentation map, still does not allow this method to yield segmentation of the current input data in real-time, and, as stated in [17], yields an overall computational complexity that grows with the size of the global segmentation map.

Also related, the work by Salas-Moreno et. al. [18] aims at real-time plane segmentation and SLAM reconstruction. In this method, planes are segmented from each input depth map, then they are incrementally merged into a global model. The main limitation of such a method in the context of segmentation is the fact that it considers only planar surfaces, while curved surfaces are not segmented, posing a limit to the generality of the processed shapes especially in the presence of objects.

In this paper, we propose a method where segmentation is carried out incrementally within a SLAM framework. The proposed method separately segments each input depth image, then merges the obtained segments within a unified Global Segmentation Map (GSM) built on top of the SLAM 3D reconstruction. This merging procedure is carried out by projecting the GSM into the camera frame by means of the camera pose estimated via SLAM, then by merging newly observed segments on each input frame to those already present in the GSM. One important advantage provided by our approach with respect to existing ones is the ability to run in constant time regardless of the size of the GSM and the number of merged depth maps in the global 3D model: this makes our approach also particularly suited to deal with large-scale 3D reconstructions. Moreover, since both the frame-wise segmentation as well as the label merging are carried out by processing depth maps rather than the global 3D model, our method can process each depth frame in real-time. Additionally, the proposed approach is also general, since it can be used together with any frame-wise segmentation approach as well as SLAM algorithms. By means of quantitative and qualitative experimental results, we demonstrate that our approach can enable real-time, accurate segmentation of diverse indoor environments. This is also shown by Fig. 1, which reports two examples of segmentation attained by our method within, respectively, a kitchen and a table-top environment.

## II. SYSTEM OVERVIEW

This section briefly outlines the proposed framework for real-time point cloud segmentation. The flow diagram sketching the algorithm pipeline deployed at each input frame is shown in Fig. 2. In this flowchart, the stages regarding SLAM reconstruction are shown as blue boxes, while the stages regarding segmentation are shown as red boxes. We assume a stream of depth maps acquired from a moving 3D/RGB-D sensor, which we process one at a time.

The SLAM algorithm employed by our system is based on the point-based fusion method of Keller et al. [19]. Instead of standard voxel-based representations, such an approach uses a point-based representation with normal information referred to as *surfel*, i.e. a disk-shaped entity which describes locally planar regions without connectivity information. The advantage of using surfel-based representations is that common operations of map entities such as data association, insertion, averaging, and removal can be performed at a lower memory footprint compared to voxel-based representations like Kinect Fusion [9]. Throughout this approach, the camera pose of each input depth map is estimated and each depth map is merged into the global model.

As for the segmentation stages, as previously mentioned, our approach aims at incrementally - and synergically with respect to the SLAM reconstruction - building up a Global Segmentation Map (GSM) in real-time by properly propagating and merging segments extracted from each depth map. Toward this goal, each depth map is segmented first (Depth Map Segmentation stage). Although any segmentation method devised to work on depth maps can be employed at this stage, our approach uses an adapted version of the real-time method proposed in [6] (described in Sec. III-A). Successively, during the Segment Label Propagation stage, the segments building up the GSM and associated with each surfel on the global model are propagated to the current depth map by means of the estimated camera pose obtained via SLAM. In particular, propagated segments are only those whose surfaces overlap with the current geometry of the depth map: this allows us to assign a label to each segment

of the depth map, which is consistent with that of the GSM (detailed in Sec. III-B).

The goal of the successive Segment Merging stage is to detect and merge segments in the GSM that have the same correspondent in the current depth map, a circumstance that typically occurs when, in the previous views, the underlying surface had occluding foreground objects. We thus exploit new vantage points along the SLAM sequence to improve the consistency of the GSM (illustrated in SubSec. III-C).

Finally, in the Segment Update stage, the labels of the GSM are updated with the labels computed from the current depth map during the Segment Label Propagation stage. Since the frame-wise depth map segmentation might be noisy, it is undesirable to update the segment label directly on the GSM from the corresponding segment label on the depth map. Therefore, we assign each element of the GSM to a confidence based on the various label observations, so that the update process can be carried out only after each propagated label has reached a certain confidence level, thus removing noisy label associations. This is explained in Sec. III-D.

In the next Subsection, we briefly outline notation and the stages of our pipeline devoted to SLAM reconstruction, which deploy and adapt the approach proposed in [19]. The stages carrying out the incremental segmentation, which build up the core of our proposal (those depicted in red in 2), are described in Sec. III.

### A. SLAM framework

In this section, we briefly outline the SLAM stages involved in the proposed incremental segmentation framework. As proposed in [19], a *global model* (i.e., the output of the SLAM reconstruction) consists of an unorganized set of surfels $s_1, .., s_k \in \mathcal{S}$, where each surfel $s_k$ is characterized by a 3D position $v_k \in \mathbb{R}^3$, a normal $n_k \in \mathbb{R}^3$, a radius $r_k \in \mathbb{R}$, a confidence $c_k \in \mathbb{R}$ and a time stamp $t_k \in \mathbb{N}$. In the Preprocessing stage, a depth image $\mathcal{D}_t$ at current frame $t$ is transformed into a metric vertex map $\tilde{\mathcal{V}}_t(\boldsymbol{u}) = \boldsymbol{K}^{-1}\dot{\boldsymbol{u}}\mathcal{D}_t(\boldsymbol{u})$, with the camera intrinsic matrix $\boldsymbol{K}$, a depth map element $\boldsymbol{u} = (x, y)^\top$ in the image domain $\boldsymbol{u} \in \Omega \subset \mathbb{R}^2$ and its homogeneous representation $\dot{\boldsymbol{u}}$. The vertex map $\tilde{\mathcal{V}}_t$ is then smoothed by applying a bilateral filter [20] so as to generate a version of the vertex map with less noise, $\mathcal{V}_t$. The normal map of current frame $\mathcal{N}_t$ is simply generated from the vertex map $\mathcal{V}_t$ by central differences.

In the Camera Pose Estimation stage, the current 3D camera pose at frame $t$, composed of a $3\times3$ rotation matrix and a 3D translation vector, $\boldsymbol{T}_t = [\boldsymbol{R}_t, \boldsymbol{t}_t] \in \mathbb{SE}(3), \boldsymbol{R}_t \in \mathbb{SO}(3), \boldsymbol{t}_t \in \mathbb{R}^3$ is updated by incrementally aligning the filtered vertex map $\mathcal{V}_t$ with the global model in the form of the vertex map rendered from the previously estimated camera pose, $\mathcal{V}^m_{t-1}$ (hereinafter, we use the superscript "m" to indicate the rendered version of a 3D point cloud with respect to a particular camera pose). The alignment is obtained via dense ICP, with a point-to-plane error metric computed by means of the rendered normal map, $\mathcal{N}^m_{t-1}$, and the fast projective data association algorithm proposed in [9].

Fig. 3. Normal map (left), normal and geometrical edge map (middle) and resulting depth map segmentation with shading (right)

During the Global Model Rendering stage, to compute correspondences between the points on the current depth map and the surfels on the global model, a index map $\mathcal{I}$ is created which maps each surfel vertex of the global model $\boldsymbol{v} \in \mathcal{S}$ with its projection on the current depth map at element $\boldsymbol{u} = \pi(\boldsymbol{K}\boldsymbol{T}_t^{-1}\boldsymbol{v})$ via standard pin-hole projection function $\pi$ based on the updated camera pose $\boldsymbol{T}_t$. Additionally, the model vertex map $\mathcal{V}^m_t$ and model normal map $\mathcal{N}^m_t$ are rendered to be used for the camera pose estimation of the successive frame.

Finally, in the Global Model Update stage, the new surfel measurements $\boldsymbol{v}$ obtained from the current depth map are either added as *unstable* surfels, or they become merged with already present surfels. Merging $\boldsymbol{v}$ with a surfel already present in $\mathcal{S}$ increments the associated confidence $c$. After a certain number of stable measurements, unstable surfels change their status to stable: this occurs when the associated confidence grows above a threshold (set to 5 measurements). In specific temporal or geometric conditions, points are removed from the global model.

## III. INCREMENTAL SEGMENTATION FRAMEWORK

In this Section, we describe the stages of the proposed pipeline carrying out the incremental segmentation, which represent the core of our proposal: they are depicted in red in Fig. 2. These stages assume, as input, the current depth map $\mathcal{D}_t$ as well as the global model reconstructed via SLAM up to the current frame, $\mathcal{S}$, as described in Sec. II-A. Each depth map is also associated with a vertex map, $\mathcal{V}_t$, that stores the 3D vertices $\boldsymbol{v}(\boldsymbol{u})$ of each element $\boldsymbol{u}$ in the depth map $\mathcal{D}_t$. The goal is to incrementally build up and update a GSM $\mathcal{L}$, which has the same number of elements as the global model $\mathcal{S}$, in which each element represents a segment label. To this end, $\mathcal{L}$ is updated, at each new frame, with the segmentation information associated with the current depth map, as explained in the remainder of this Section.

### A. Depth map segmentation

To segment each input depth map $\mathcal{D}_t$, we employ a fast segmentation method inspired by the *normal edge* analysis carried out in [6], where, at each frame, a binary *edge map* is computed by comparing nearby normal angles, these edges representing the segment boundaries. In contrast to [6], we propose to extract only convex-shape segments by explicitly detecting concave boundaries. Indeed, this choice is motivated by recent works exploiting graph-based segmentation [4], [17], that introduced a penalty for concave regions based

on the assumption that real-world objects mainly consist of convex shapes.

Inspired by these methods, we adapt the concave region penalty to the normal edge-based segmentation deployed in our pipeline. First, for each element $\boldsymbol{u}$ in the depth map $\mathcal{D}_t$, we detect concave boundaries by computing the dot product between the normal at $\boldsymbol{u}$, $\boldsymbol{n}(\boldsymbol{u})$, and each normal of the 8-connected neighboring points of $\boldsymbol{u}$, $\boldsymbol{n}(\boldsymbol{u_i}), i = 1, .., 8$, In particular, we define an operator, $\Phi_i(\boldsymbol{u})$, as follows:

$$\Phi_i(\boldsymbol{u}) = \begin{cases} 1 & (\boldsymbol{v}(\boldsymbol{u_i}) - \boldsymbol{v}(\boldsymbol{u})) \cdot \boldsymbol{n}(\boldsymbol{u}) > 0 \\ \boldsymbol{n}(\boldsymbol{u}) \cdot \boldsymbol{n}(\boldsymbol{u_i}) & otherwise \end{cases} \tag{1}$$

$\boldsymbol{v}(\boldsymbol{u})$ and $\boldsymbol{v}(\boldsymbol{u_i})$ being the associated 3D vertices from the vertex map $\mathcal{V}_t$. This operator takes value 1 (i.e., its maximum value) when $\boldsymbol{n}(\boldsymbol{u})$ and $\boldsymbol{n}(\boldsymbol{u_i})$ lie on a convex surface, while it takes the dot product between such normals if they lie on a concave shape. Hence, the higher the concavity, the lower the value of $\Phi_i(\boldsymbol{u})$. We compute such an operator for all 8 neighbors, then take its minimum:

$$\Phi(\boldsymbol{u}) = \min_{i=1,..,8} \{\Phi_i(\boldsymbol{u})\}, \tag{2}$$

Such an operator highlights concavities along at least one of the eight directions around element $\boldsymbol{u}$. We thus compute our concavity-aware normal edge map by thresholding $\Phi(\boldsymbol{u})$ (we set the threshold $0.94$).

Another cue that we take into consideration is the distance between two vertices in the 3D space: indeed, a commonly deployed assumption is that depth borders define 3D segment borders. To reach this goal, we define another operator, $\Gamma(\boldsymbol{u})$, that takes into account the maximum 3D point-to-plane distance between an element $\boldsymbol{u} \in \mathcal{D}_t$ and its 8 neighbors:

$$\Gamma(\boldsymbol{u}) = \max_{i=1,..,8} \{ \left| \left( \boldsymbol{v}\left(\boldsymbol{u_i}\right) - \boldsymbol{v}\left(\boldsymbol{u}\right) \right) \cdot \boldsymbol{n}\left(\boldsymbol{u}\right) \right| \} \tag{3}$$

To threshold $\Gamma(\boldsymbol{u})$, we use an uncertainty measure $\sigma_d\left(\boldsymbol{u}\right)$ computed following the noise model proposed in [21]. Such a measure adaptively takes into account the noise level of each point of the depth map, assuming a noise model that increases with the distance from the sensor. By thresholding $\Gamma(\boldsymbol{u})$, we obtain a set of geometrical edges, which are added to the previously computed normal edges to yield the final edge map (shown in Fig. 3, middle). Finally, we apply a connected component analysis algorithm to the edge map obtained to yield a label map $\mathcal{L}_t$, where each element $\boldsymbol{u}$ is associated with a segment label $\mathcal{L}_t(\boldsymbol{u}) = l_j$ (shown in Fig. 3, right). In this map, the label $0$ (unlabeled segment) is assigned to all points lying within the detected concave regions and depth border regions.

### *B. Segment Label Propagation*

As anticipated, the goal of this stage is to propagate the segments of the GSM that are visible from the current camera viewpoint onto the label map of the current depth map. With this in mind, correspondences between the visible segments of the GSM, $l_i \in \mathcal{L}$ and those on the current depth map, $l_j \in \mathcal{L}_t$ are estimated by checking whether the underlying surface of both segments are the same. The label propagation procedure is illustrated in Fig. 4. As introduced in Sec. III, given the currently estimated camera pose at time $t$, we re-project the global model onto its image plane, yielding a vertex map $\mathcal{V}_t^m$ and a normal map $\mathcal{N}_t^m$. At the same time, we also compute the vertex map and the normal map associated with the current depth map, i.e. $\mathcal{V}_t$, $\mathcal{N}_t$ (note the absence of superscript "$m$" to distinguish them). In a similar fashion, we re-project the GSM, $\mathcal{L}$, on the same image plane, obtaining a label depth map $\mathcal{L}_t^m$. Note that this re-projection allows us to focus only on the elements of the GSM currently visible from the current camera viewpoint and to discard all GSM elements that are either occluded or outside of the camera's field of view, thus resulting in efficiency and scalability with respect to the global model size.

To efficiently determine segment correspondences between $\mathcal{L}_t$ and $\mathcal{L}_t^m$, first the number of corresponding points $\Pi(l_i, l_j)$ between all points with label $l_i \in \mathcal{L}_t^m$ and all points with label $l_j \in \mathcal{L}_t$ are determined. This is done by considering each pair of elements $l_i = \mathcal{L}_t^m(\boldsymbol{u}), l_j = \mathcal{L}_t(\boldsymbol{u}), \forall \boldsymbol{u} \in \mathcal{D}_t$, and by thresholding the distance of the corresponding vertices along the viewing ray and the angle between the corresponding normals:

$$\left| \left( \mathcal{V}_t\left(\boldsymbol{u}\right) - \mathcal{V}_t^m\left(\boldsymbol{u}\right) \right) \cdot \frac{\mathcal{V}_t\left(\boldsymbol{u}\right)}{\left|\mathcal{V}_t\left(\boldsymbol{u}\right)\right|} \right| < \sigma_d\left(\boldsymbol{u}\right) \tag{4}$$

$$\cos^{-1}(\mathcal{N}_t(\boldsymbol{u}) \cdot \mathcal{N}_t^m(\boldsymbol{u})) > \tau_{\mathcal{N}} \tag{5}$$

where $\sigma_d\left(\boldsymbol{u}\right)$ is the depth uncertainty measure previously used, and $\tau_{\mathcal{N}}$ is the normal angle threshold (in our experiments, $\tau_{\mathcal{N}} = 20°$). When both conditions are satisfied, $\Pi(l_i, l_j)$ gets incremented. We normalize this term by the size of the corresponding segment on $\mathcal{L}_t$:

$$\tilde{\Pi}(l_i, l_j) = \frac{\Pi(l_i, l_j)}{\#(l_j)} \tag{6}$$

(we refer to $\#$ as the cardinality operator for a segment).

Hence, $\tilde{\Pi}(l_i, l_j)$ represents the percentage of 3D overlap of segment $l_j \in \mathcal{L}_t$ with $l_i \in \mathcal{L}_t^m$, computed as the intersection between the two segment point sets and normalized by the magnitude of the point set of $l_j$. It can be regarded as a confidence measure for both segments to lie on the same 3D surface. We can thus easily associate each segment on $\mathcal{L}_t$ with the maximally-overlapping segment on $\mathcal{L}_t^m$ as follows:

$$\tilde{\Pi}_{max}(l_j) = \max_{l_i \in \mathcal{L}_t^m} \{\tilde{\Pi}(l_i, l_j)\} \tag{7}$$

We wish to note here that, since we take into account only those labels of the GSM that appear on the current label map $\mathcal{L}_t^m$, the complexity of the operation in formula (7) is independent from the size of the GSM.

Based on $\tilde{\Pi}_{max}(l_j)$, we build up a *propagated* label map $\mathcal{L}_t^p$ (depicted in Fig. 4, right) by applying to each element $l \in \mathcal{L}_t^p$ the following rule:

1) If $\tilde{\Pi}_{max}(l_j) \geq \tau_\Omega$, then $l = l_i$, i.e. we propagate the label of the GSM segment that yielded the highest overlap. $\tau_\Omega$ is set, in our experiments, to $0.3$ — i.e., we require at least $30\%$ segment overlap.

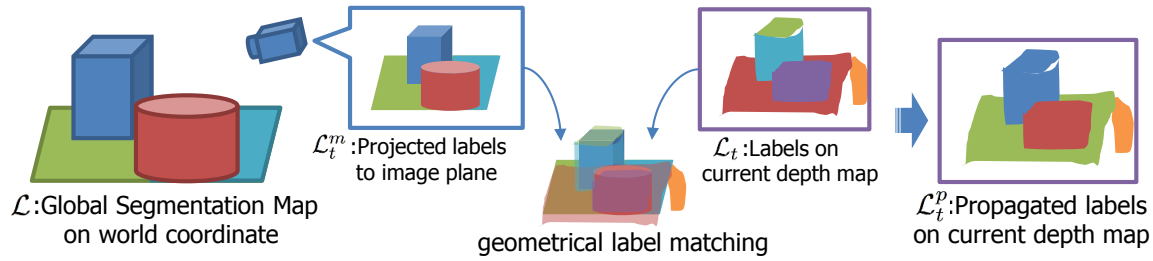



Fig. 4. Toy example explaining the proposed Segment Propagation stage: first segments from the GSM are re-projected onto the current camera plane; then, they are compared with those of the current depth map, so to identify corresponding segments between GSM and the camera plane.

Fig. 5. Benefits of the proposed Segment Merging procedure: an object initially segmented into two parts due to an occluding foreground surface (left), is then merged into the same segment when a different camera view reveals the right connectivity (middle and right)

2) otherwise $l = l_j$, i.e. we propagate the label $l_j$ directly from $\mathcal{L}_t$.

Finally, to avoid including in $\mathcal{L}_t^p$ new segments derived from possible data artifacts or bordering regions, we assign the label 0 to $l$ every time the newly propagated segment from $\mathcal{L}_t$ has a size smaller than a certain threshold (50 pixels in our experiments).

### C. Segment merging

During the label propagation stage, different segments on the GSM might correspond to the same segments on the depth map. This is typically the case for a surface which is initially separated into multiple segments due to occluding foreground objects: when the camera viewpoint changes, the occlusion might disappear to reveal that such multiple segments are indeed part of the same surface. An example of such a situation is depicted in Fig. 5. The goal of this stage is to identify and merge together these corresponding segments on the GSM.

To determine whether the underlying surface of the two segments is the same based on their 3D overlap, we rely on the same criterion used in the previous Segment Propagation stage. In particular, during the computation of term $\tilde{\Pi}(l_j)$, we identify all segments $l_i \in \mathcal{L}_t^m$ yielding an overlap higher than a certain threshold with $l_j \in \mathcal{L}_t$ (set to 0.2 in our experiments). Such segments form the label set $L_{l_j}$.

Since the segments obtained from the depth map are often noisy, it is not robust to perform such a merging procedure only from the observations derived from a single depth map. Therefore, we introduce a pairwise confidence that estimates to what extent a pair of segments are part of the same surface based on all input data seen so far, inspired by the confidence measure used on each surfel with the SLAM method [19]. Specifically, all possible label pairs $(l_a, l_b), l_a, l_b \in L_{l_j}, l_a \neq l_b$ are associated with a confidence $\Psi_t^m(l_a, l_b)$. If a pair $(l_a, l_b)$ is identified for the first time, its associated confidence is initialized as follows: $\Psi_t^m(l_a, l_b) = 0$ Instead, when such a segment pair has been already observed, its associated confidence is updated by incrementing it:

$$\Psi_t^m(l_a, l_b) = \Psi_{t-1}^m(l_a, l_b) + 1 \tag{8}$$

Finally, for all those segment pairs which are not observed at the current time $t$, but for which a confidence was already initialized in the previous frames, their confidence is updated as follows:

$$\Psi_t^m(l_a, l_b) = \max\left(0, \Psi_{t-1}^m(l_a, l_b) - 1\right) \tag{9}$$

Also when the confidence associated to a certain segment pair grows higher than a specific threshold (set to 3 in our experiments), the segment pair $(l_a, l_b)$ is merged by replacing the label $l_a$ with label $l_b$ in $\mathcal{L}$.

### D. Segment Update

The last step of the incremental segmentation procedure concerns updating the GSM with the label map obtained at the end of the Segment Label Propagation stage, i.e. $\mathcal{L}_t^p$. Analogously to the Segment Merging stage, in this case also it is not robust to directly modify our GSM based on the label indications contained in $\mathcal{L}_t^p$, since such a label map, being based on one single depth frame, usually contains noisy information. Hence, and similarly to the previous stage, we follow a confidence based approach, by associating each element of the GSM map $\mathcal{L}(v)$ with a confidence $\Psi_t^u(v)$.

To initialize and update such a confidence, for each element $\mathcal{L}_t^p(\boldsymbol{u})$, we compute the corresponding GSM element $\mathcal{L}(v)$ by means of the estimated camera pose, and follow these three cases:

1) If $\mathcal{L}(v)$ is unlabeled (e.g., the corresponding surfel has been just added due to the new observation), it is set to the label of the new observation: $\mathcal{L}(v) = \mathcal{L}_t^p(\boldsymbol{u})$, while the confidence is initialized as $\Psi_t^u(v) = 0$.
2) If the labels of $\boldsymbol{u}$ and $v$ are the same, the confidence is incremented:

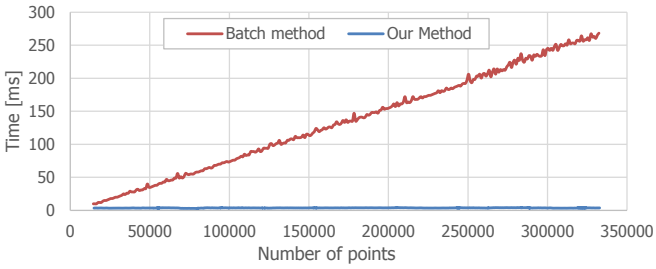
$$\Psi_t^u(v) = \min(\Psi_{t-1}^u(v) + 1, \Psi_{max}^u) \tag{10}$$

where $\Psi_{max}^u$ is the cap value for the confidence associated surfels (set to 10 in our experiments).

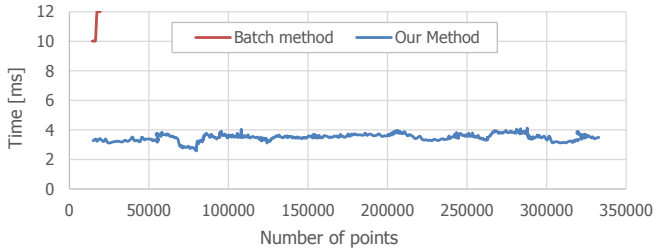
TABLE I

MEASURED EXECUTION TIMES OF EACH STAGE OF THE PROPOSED INCREMENTAL SEGMENTATION AVERAGED ON THE *fr1/room* SEQUENCE [22] AT DIFFERENT RESOLUTIONS OF THE INPUT FRAMES (ALL STAGES ARE IMPLEMENTED ON CPU)

| Number of frames | 739 | 739 | 739 |
|---|---|---|---|
| Resolution of depth map | 160×120 | 320×240 | 640×480 |
| Depth Map Segmentation [ms] | 1.81 | 7.63 | 32.39 |
| Segment Label Propagation [ms] | 1.5 | 5.58 | 29.2 |
| Segment Merging [ms] | 0.08 | 0.09 | 0.11 |
| Segment Update [ms] | 0.13 | 0.51 | 1.92 |
| Segmentation Total [ms] | 3.52 | 13.82 | 63.58 |
| Segmentation Frame-Rate [fps] | 284 | 72 | 15 |




(a) Time Comparison




(b) Close up Time Comparison of (a)

Fig. 6. Comparison between the proposed segmentation framework and [13] in terms of computational efficiency, tested at an increasing number of points of the reconstructed 3D data.

3) If the labels of $\boldsymbol{u}$ and $v$ are different, the label confidence is decreased:

$$\Psi_t^u(v) = \max\left(0, \Psi_{t-1}^u(v) - 1\right) \tag{11}$$

When the confidence associated with a point $\mathcal{L}(v)$ drops to 0, it means that the segment has changed, e.g. due to a change in the structure of the environment (in case of dynamic scenes). In such a case, the point is assigned the corresponding label on $\mathcal{L}_t^p$, i.e. $\mathcal{L}(v) = \mathcal{L}_t^p(\boldsymbol{u})$.

## IV. EXPERIMENTAL RESULTS

In this Section, we provide quantitative and qualitative experimental results to demonstrate how our approach can enable real-time, accurate segmentation of diverse indoor environments. An implementation of the proposed incremental segmentation framework is publicly available[1].

[1] campar.in.tum.de/Main/KeisukeTateno

### A. *Efficiency and scalability*

We compare the proposed incremental segmentation framework against the graph-based segmentation algorithm of Felzenszwalb and Huttenlocher [13], it being a widely used batch segmentation method for reconstructed 3D data [4], whose implementation is publicly available[2]. Since this approach relies on the 3D mesh topology in order to build the segment graph, to test it on our point clouds reconstructed via SLAM, we have employed the fast meshing algorithm proposed in [23], whose implementation is publicly available in the Point Cloud Library (PCL)[3].

For the comparison, we have used one sequence (*fr1/room*) from the public *TUM RGB-D SLAM* benchmark dataset [22]. Both algorithms have been tested on the same platforms, a standard laptop PC equipped with an Intel Core i7 CPU at 2.6GHz with 16GB of RAM. Both algorithms run totally on CPU processing (no GPU optimization of any part). The resolution of the input depth maps is converted to 160x120 to allow the dense SLAM framework employed to run in real-time on such CPU.

Fig. 6 shows the comparison in terms of efficiency between the two algorithms. For both algorithms, only the time required for segmentation is measured, while the time spent on SLAM is not taken into account. The efficiency is measured at an increasing size of the point cloud reconstructed with SLAM, also to compare the two methods in terms of scalability with the number of frames merged into the SLAM sequence. As shown in the Figure, while [13] exhibits a clearly linear complexity with the point cloud size, the proposed algorithm demonstrates a constant complexity with such size, thanks to the incremental nature of its approach. Also, the proposed method demonstrates real-time capabilities, running at an average of 3.5 ms per frame.

Also, we wish to point out here the advantage with respect to the incremental segmentation method in [17], which, as mentioned in Sec. I, also performs incremental segmentation of 3D reconstructions. As stated in Sect. IV of [17], the complexity of this method grows on the size of the map, and its run-time exhibits a linear dependency with the number of points of the 3D reconstruction, as reported by the results in Fig. 5 in [17]. Our method, instead, has a constant run-time with respect to the global model size, as shown in Fig. 6.

The execution time measured relatively to each stage involved in our incremental segmentation, averaged over the benchmark *fr1/room* sequence, is summarized in Table I, while the final segmentation of the whole sequence is shown in Fig. 7 (discussed in the next Subsection). As shown by the table, the whole segmentation framework can run in real-time at different resolutions of the input depth map, yielding 284 frame per second on the 160×120 resolution, and running at 72 and 15 frames per second at a resolution of, respectively, 320×240 and 640×480. We expect that even higher frame rates could be achieved if the segmentation stages were to

[2] cs.stanford.edu/people/karpathy/discovery

[3] www.pointclouds.org

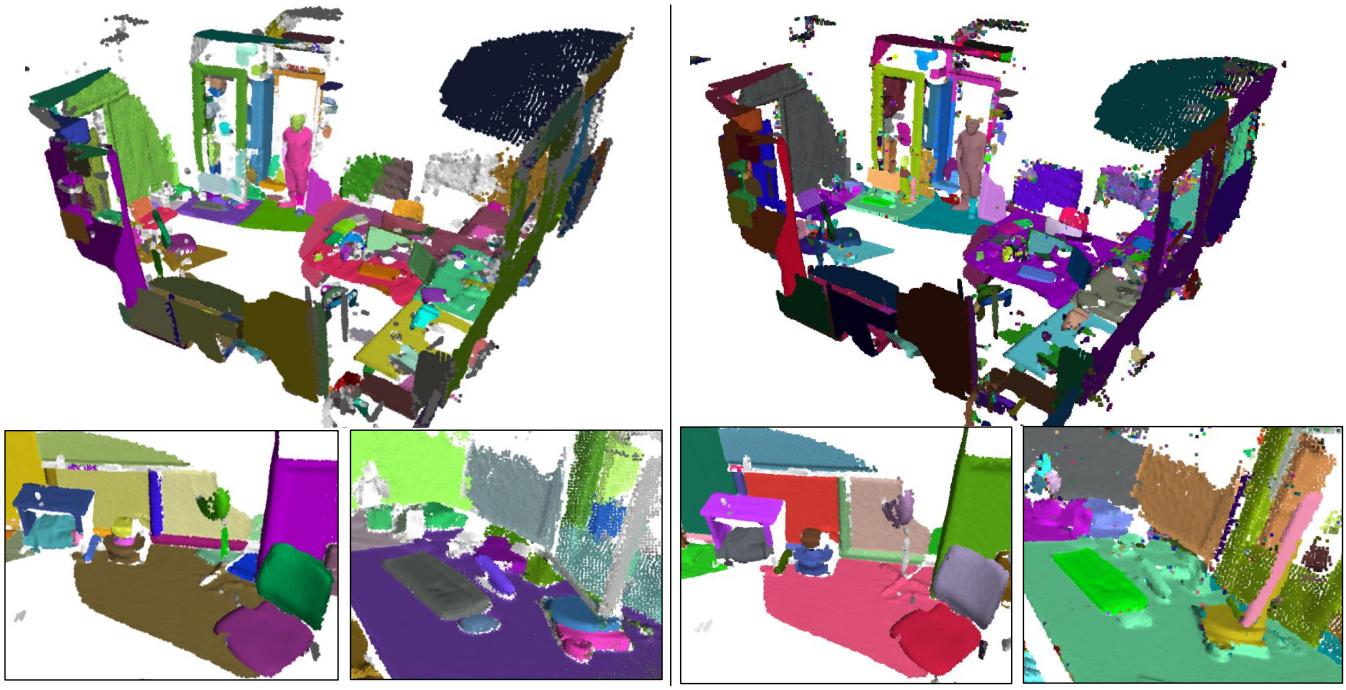


Fig. 7. Qualitative comparison between the segmentation obtained by the proposed method (left) and that obtained by the method in [13] (right).

be processed on GPU. Also interestingly, the Table shows that the most time consuming stage is the initial frame-wise segmentation step: hence, an even higher efficiency can be achieved by plugging in a different, more efficient frame-wise segmentation algorithm with respect to the one currently being deployed.

### B. Segmentation accuracy

In addition to the previous results, we also compare our method with [13] in terms of the segmentation accuracy of the same sequence used for the experiment in Subsec. IV-A (i.e., *fr1/room*) from the *TUM RGB-D SLAM* benchmark. Fig. 7 shows a qualitative comparison in terms of segmentation yielded by the proposed framework (left) and those by [13] (right). Overall, the segmentation accuracy appears to be comparable, although the proposed method seems able to better segment small objects on top of planar surfaces, as witnessed also by the right close-up snapshot shown at the bottom of the Figure.

Finally, we show some qualitative results of the segmentation reported by our method for reconstructed indoor environments acquired with our own setup based on a PrimeSense Carmine 1.09 RGB-D sensor. Two examples are shown in Figure. 1, which reports a kitchen-like scene and a typical table-top scene. In addition, we also acquired a video showing the incremental reconstruction carried out by the incremental segmentation framework in such scenarios, which is available as supplementary material with this submission. The overall reconstruction and some details relative to such sequences are reported in Fig. 8.

## V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an approach in which segmentation is carried out incrementally within a SLAM framework, where segments obtained from the depth map are merged within a unified GSM built on top of the SLAM 3D reconstruction. Conversely to available methods in literature, our method demonstrated its efficiency and to scale well with the number of frames of the SLAM reconstruction. We believe that real-time incremental segmentation of reconstructed environments can pave the way to new directions in the field of robotic perception: for example, the method can allow real-time object discovery in unknown environments (currently done offline [4]), as well as real-time object recognition from SLAM reconstructions (currently done on depth maps only [1]). As a future work, we plan to deploy our method in a SLAM framework with loop closure detection, so to exploit SLAM graph optimization to improve the accuracy of the incremental segmentation. Another interesting direction is represented by pairing the proposed framework with 3D mesh-based reconstruction algorithms, such as the Kinect Fusion[9].

## REFERENCES


[1] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. D. Stefano, and M. Vincze, "Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[2] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[3] T. Mörwald, M. Zillich, and M. Vincze, "Edge tracking of textured objects with a recursive particle filter," in *Proceedings of the Graphicon*, 2009.

Fig. 8. Overall reconstruction yielded by the proposed segmentation framework in an office scenario (top). Some zoomed in details from this reconstruction are reported in the bottom.


[4] A. Karpathy, S. Miller, and L. Fei-Fei, "Object discovery in 3D scenes via shape analysis," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[5] A. Uckermann, R. Haschke, and H. Ritter, "Realtime 3D segmentation for human-robot interaction," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2136–2143, 2013.

[6] A. Uckermann, C. Elbrechter, R. Haschke, and H. Ritter, "3D scene segmentation for autonomous robot grasping," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1734–1740, oct 2012.

[7] A. Pieropan and H. Kjellstrom, "Unsupervised object exploration using context,," *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, 2014.

[8] A. Abramov, K. Pauwels, J. Papon, F. Worgotter, and B. Dellen, "Depth-supported real-time video segmentation with the kinect - youtube," in *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, 2012.

[9] R. a. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," *10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, oct 2011.

[10] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. Mcdonald, "Real-time large scale dense RGB-D SLAM with volumetric fusion," *Intl. J. of Robotics Research, IJRR*, 2014.

[11] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D Mapping : Using Depth Cameras for Dense 3D Modeling of Indoor Environments," *The International Journal of Robotics Research*, vol. 31, pp. 647–663, 2012.

[12] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2100–2106, nov 2013.

[13] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, sep 2004.

[14] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, 2009, pp. 39–46.

[15] T. Rabbani, F. A. van den Heuvel, and G. Vosselman, "Segmentation of point clouds using smoothness constraint," in *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium 'Image Engineering and Vision Metrology'*, vol. 36, 2006, pp. 248–253.

[16] J. Strom, A. Richardson, and E. Olson, "Graph-based segmentation for colored 3D laser point clouds," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS'10)*, 2010, pp. 2131–2136.

[17] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Efficient Incremental Map Segmentation in Dense RGB-D Maps," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 5488–5494.

[18] R. Salas-Moreno and B. Glocken, "Dense planar SLAM," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014, pp. 157–164.

[19] 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 Vision (3DV)*. Ieee, 2013, pp. 1–8.

[20] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (ICCV)*, 1998.

[21] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3D reconstruction and tracking," *Proceedings - 2nd Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission, 3DIMPVT 2012*, pp. 524–530, 2012.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, oct 2012.

[23] Z. Marton, R. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," in *Proc. Int. Conf. on Robotics and Automation (ICRA)*, 2009.