When 2.5D is not enough: Simultaneous Reconstruction, Segmentation and Recognition on dense SLAM

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

***Abstract*— While the main trend of 3D object recognition has been to infer object detection from single views of the scene — i.e., 2.5D data — this work explores the direction on performing object recognition on 3D data that is reconstructed from multiple viewpoints, under the conjecture that such data can improve the robustness of an object recognition system. To achieve this goal, we propose a framework which is able (i) to carry out incremental real-time segmentation of a 3D scene while being reconstructed via Simultaneous Localization And Mapping (SLAM), and (ii) to simultaneously and incrementally carry out 3D object recognition and pose estimation on the reconstructed and segmented 3D representations. Experimental results demonstrate the advantages of our approach with respect to traditional single view-based object recognition and pose estimation approaches, as well as its usefulness in robotic perception and augmented reality applications.**

I. Introduction and related works

Object recognition and pose estimation in cluttered scene from 3D data is one of the most relevant topics in 3D computer vision and robotic perception, with applications such as robotic manipulation and grasping, industrial automation, augmented reality, industrial quality control and defect detection. The state of the art is roughly subdivided into two main classes. One class of methods relies on template matching carried out on depth data [1], [2], [3]. In general, these approaches can deal with poorly descriptive shapes and texture-less objects, and are efficient to compute, but are generally limited in terms of scalability with the number of models being matched and presence of occlusions.

The other class matches together 3D descriptors [4], [5], [6] computed on the 3D model and the 3D scene. While inherently more scalable and robust to occlusions, the computation of a set of descriptors on a single scene is usually not compatible with real-time constraints. Typically, 3D descriptors are either *global* [4] or *local* [5], [6]: the former represent the whole model shape with one single vector, while the latter compute multiple descriptors at each 3D keypoint extracted from the model surface, each descriptor taking into account a small neighborhood around the keypoint. While global descriptors can generally describe also poorly informative shapes (e.g., household objects or primitive shapes), they rely on 3D scene segmentation [7], [8] as a pre-requisite step, which might fail in the presence of clutter and occlusions.

Fig. 1. The proposed framework incrementally reconstructs 3D scenes while simultaneously performing 3D segmentation, 3D object recognition and 3D pose estimation. Bottom row, left to right: one input RGB frame, output 3D reconstruction, output 3D segmentation.

One common aspect of template matching-based and descriptor-based approaches is that they match templates and descriptors computed on single views of the model and the scene, i.e. 2.5D data or range images. This is due to the fact that scenes are acquired via 3D sensors, which provide 3D representations from specific viewpoints, i.e. in the form of range images. To avoid matching 2.5D scenes with fully 3D model representations, a high number of model views are usually rendered from virtual cameras placed around each model, so to allow matching 2.5D model views against 2.5D scene views. Particularly for template matching approaches, 2.5D is the way to go since it provides depth measurements over an organized 2D grid structure, which is better suited to compute patch-based representations. By relying on a single observation from a specific viewpoint, 2.5D data often reports noisy and distorted versions of the real 3D surface, in particular (but not only) in presence of dark and reflective surfaces: this is especially the case of recent, low-cost consumer RGB-D cameras such as Microsoft Kinect, Kinect 2.0 and Asus Xtion. As a consequence, 3D object recognition methods based on single depth maps have to particularly robust to such nuisances.

[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

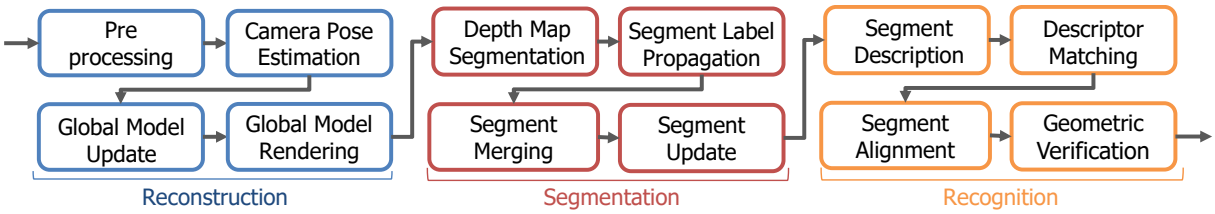


Fig. 2. Flow diagram of the proposed simultaneous reconstruction, segmentation and recognition pipeline.

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 on 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]. In this scenario, specific methods [13], [14] have been proposed to leverage the recognition of specific 3D models within the current scene to improve the 3D reconstruction of the scene via SLAM (and vice-versa), e.g. by providing additional 3D matches within Bundle Adjustment. However, these methods still carry out 3D object recognition following the traditional paradigm, i.e. by matching model views against the current scene frame (2.5D vs 2.5D).

Intuitively, by comparing the 3D scene reconstructions against the fully-3D model representations, the robustness of 3D object recognition as well as the accuracy of the estimated 3D object pose can be improved. Based on this, our work proposes a framework that deploys simultaneously a SLAM algorithm to reconstruct the environment, an incremental segmentation algorithm to obtain 3D segments of such reconstruction in real-time, and an incremental 3D object recognition algorithm that carries out descriptor matching out of such segments. The goal is to demonstrate that we can exploit multiple viewpoints around the same scene to achieve robust recognition and extremely stable 3D poses in presence of heavy clutter and occlusion at a high efficiency. At the same time, by matching segments that encode the fully-3D shape of our object directly against the fully 3D model, we can be robust towards noise and artifacts that affect the current scene frame. Moreover, being the number of global descriptors that need to be computed equal to the number of such scene segments, the overall process is remarkably efficient, yielding near real-time performance. We validate our proposal by evaluating qualitatively its performance on a benchmark dataset for 3D object recognition in clutter. In addition, we also test our framework within an augmented reality application, so as to demonstrate its usefulness and effectiveness in real scenarios.

Notably, object recognition methods from SLAM reconstructions have been already proposed [15], [16], [17], [18]. [17] proposes 3D object recognition utilizing temporal multi-view information. [18] proposes semantic labeling method using learning feature from point clouds incrementally estimated from SLAM reconstruction. Although they rely on fully 3D information to aid 3D object recognition, their reconstruction and recognition assumes an offline or non real-time framework. Interestingly, [19] proposes 3D object recognition from incrementally reconstructed scene via dense SLAM in real-time, by means of keypoints extracted nearby 3D corners and matched. It achieves a higher efficiency than 2.5D based approaches. However, since the recognition method relies on such keypoints, it fails on smooth surfaces where keypoints can hardly be repeatably extracted (e.g., spherical objects), conversely to the global descriptors employed by our approach, that can deal also with poorly descriptive shapes. In addition, the idea of inferring semantic information out of SLAM-based reconstruction has been recently deployed with the goal of semantic segmentation rather than object recognition [20], [21], respectively in the case of indoor and outdoor scenes.

II. Simultaneous Reconstruction, Segmentation and Recognition Framework

This section describes the proposed framework for simultaneous reconstruction, segmentation and object recognition. The flow diagram sketching the algorithm pipeline deployed at each input frame is shown in Fig. 2. In this flowchart, the stages regarding 3D reconstruction, segmentation and recognition are shown as blue, red and orange boxes, respectively. We assume to have a stream of range images from a moving 3D/RGB-D sensor, which we process one at a time.

The 3D reconstruction algorithm employed by our system is based on the Kinect Fusion approach [9], a real-time method which estimates the camera pose of a moving sensor and relies on a volumetric surface representation called Truncated Signed Distance Function (TSDF): it is described in Sec. II-A. Successively, the incremental segmentation algorithm is based on the recent approach proposed in [22]. Contrary to [22], which merges segments within a point cloud (the Global Segmentation Map), since our reconstruction approach deploys Kinect Fusion and a TSDF-based representation, we propose to incrementally merge the 3D segments within a specific voxel-based representation, which we call as Label Volume. Details of the segmentation stages are illustrated in Sec. II-B.

Finally, the 3D object recognition part is inspired from the global descriptor pipeline proposed in [5] to yield 3D correspondences between a range map of the scene and a set of rendered views of each 3D model. Differently, our framework computes the 3D descriptor directly on each 3D segment derived from the incremental segmentation stage, and match it with the single 3D descriptor computed on

the fully 3D object model. To increase the robustness of this stage, we propose a novel method, called as *Geometric Verification*, which is carried out after descriptor matching and is able to recognize the correct model with a small number of available viewpoints. Details of the recognition stages are explained in Sec. II-C.

The offline stage consists of computing, for each model in the object database, $m_k \in \mathcal{M}$, a 3D descriptor from the model point cloud, which can be obtained directly from a CAD model of the object (if available), or via dense SLAM reconstruction. During the offline stage, we also store the model descriptors in a kd-tree: this yields the important property that the scalability of descriptor matching is log-linear with respect to the number of models in the database.

A. Reconstruction

As proposed in [9], a *global model* (i.e., the output of the Kinect Fusion reconstruction) consists of a volumetric representation by a TSDF, a data structure where each voxel stores the distance to the closest surface, as well as a weight that measures the uncertainty of the surface measurement, computed as the capped number of measurements fused so far in that voxel.

1) Pre-processing: In the Pre-processing stage, the range image $\mathcal{D}_t$ at the current time interval t is transformed into a metric vertex map $\tilde{\mathcal{V}}_t$ as:

$$\tilde{\mathcal{V}}_t(\boldsymbol{u}) = \boldsymbol{K}^{-1}\dot{\boldsymbol{u}}\mathcal{D}_t(\boldsymbol{u}) \tag{1}$$

where $\boldsymbol{K}$ is the camera intrinsic matrix, $\boldsymbol{u} = (i, j)^\top$ is the generic range image element belonging to the image domain $\boldsymbol{u} \in \Omega \subset \mathbb{R}^2$, and $\dot{\boldsymbol{u}}$ is its corresponding representation in homogeneous coordinates. A vertex map stores the 3D vertex $\boldsymbol{v}(\boldsymbol{u})$ associated to each element $\boldsymbol{u}$ of the range image $\mathcal{D}_t$. To generate a version of the vertex map with less noise, $\mathcal{V}_t$, the vertex map $\tilde{\mathcal{V}}_t$ is smoothened by applying a bilateral filter [23] Then the normal map $\mathcal{N}_t$ is simply generated from the vertex map $\mathcal{V}_t$ by computing, at each element, the normal via central differences.

2) Camera Pose Estimation: In the successive Camera Pose Estimation stage, the current 3D camera pose at frame t, composed of a 3×3 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 TSDF Volume and the previously estimated camera pose by ray casting algorithm, $\mathcal{V}^m_{t-1}$ (computed in the Global Model Rendering stage, later). The alignment is obtained via dense Iterative Closest Point (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].

3) Global Model Update: Then, in the Global Model Update stage, the depth measurements associated with the current vertex map, $\mathcal{V}_t$, are integrated into the previous TSDF volume, $\mathcal{T}_{t-1}$, as follows. First, a correspondence between each element of the TSDF, $\mathcal{T}_{t-1}(\boldsymbol{p})$ and an element on $\mathcal{V}_t$, $\mathcal{V}_t(\boldsymbol{u})$, is obtained by projecting $\mathcal{T}_{t-1}(\boldsymbol{p})$ on $\mathcal{V}_t$ by means of the camera pose $\boldsymbol{T}_t$, where $\boldsymbol{p} = (x, y, z)^\top$ is the generic 3D element on the global coordinate system belonging to the volume domain $\boldsymbol{p} \subset \mathbb{R}^3$. Then, the Truncated Signed Distance between these two elements, $\bar{d}\left(\mathcal{T}_{t-1}(\boldsymbol{p}), \mathcal{V}_t(\boldsymbol{u})\right)$ is used to update the TSDF element by averaging:

$$\mathcal{T}_t(\boldsymbol{p}) = \frac{\omega_{t-1}(\boldsymbol{p})\mathcal{T}_{t-1}(\boldsymbol{p}) + \omega_t(\boldsymbol{p})\bar{d}\left(\mathcal{T}_{t-1}(\boldsymbol{p}), \mathcal{V}_t(\boldsymbol{u})\right)}{\omega_{t-1}(\boldsymbol{p}) + \omega_t(\boldsymbol{p})} \tag{2}$$

where the new weight associated to the current TSDF element, $\omega_t(\boldsymbol{p})$, is updated as:

$$\omega_t = \begin{cases} \omega_{t-1} + 1 & \text{if } \omega_{t-1} < \omega_{max} \\ \omega_{t-1} & \text{otherwise} \end{cases} \tag{3}$$

where ω_{max} is the cap value for the confidence weights of the TSDF (set to 128 in our experiments).

4) Global Model Rendering: Finally, during the Global Model Rendering stage, the model vertex map $\mathcal{V}^m_t$ and the model normal map $\mathcal{N}^m_t$ are rendered from the TSDF Volume using the estimated camera pose of the current frame, $\boldsymbol{T}_t$, via raycasting. During raycasting, the extraction of a predicted surface from the TSDF is achieved by detecting the zero crossings of the surface with respect to the TSDF elements. The newly obtained vertex map $\mathcal{V}^m_t$ is used in the Camera Pose Estimation stage for the next time interval, $t + 1$.

B. Incremental Segmentation

Incremental segmentation is carried out via the four stages depicted in red in Fig. 2. As input, it assumes the current depth map $\mathcal{D}_t$ and its associated smooth vertex map $\mathcal{V}_t$, as well as the TSDF reconstructed up to the current frame, $\mathcal{T}_t$, as described in Sec. II-A. Analogous to [22], the goal is to incrementally build up and update a Label Volume $\mathcal{L}$, where each voxels is associated to a segment's label and to a label confidence. To achieve this, $\mathcal{L}$ is updated, at each new frame with the segmentation information associated to the current depth map. In contrast to [22], $\mathcal{L}$ is a voxel map and not a point cloud, so to better match the 3D representation used by the deployed Kinect Fusion algorithm.

1) Depth Map Segmentation: In the Depth Map Segmentation stage, to segment each input depth map $\mathcal{D}_t$, we employ a fast segmentation method based on the *normal edge* analysis, where, at each frame, a binary *edge map* is computed by comparing nearby normal angles, these edges representing the segment boundaries. Under the assumption that real-world objects are mainly consisting of convex shapes, the concave boundary map is obtained by computing the angle between a point's normal and the normals at its neighboring points. Then, a connected component analysis algorithm is used to yield a segmentation map $\mathcal{L}_t$, where each element $\mathcal{L}_t(\boldsymbol{u})$ is associated with a segment label l_j. In this map, the label 0 (unlabeled segment) is assigned to all points laying on a concave boundary.

2) Segment Label Propagation: In the Segment Label Propagation stage, a *propagated* label map, $\mathcal{L}^p_t$, is determined (depicted in Fig. 3, right), that contains all the 3D segments of the Label Volume that correspond to the 3D segments on the current range image. First, a rendered label map, $\mathcal{L}^m_t$ is

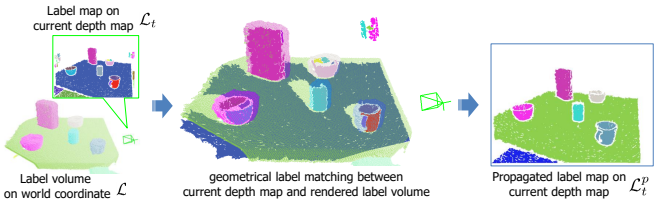


Fig. 3. Toy example explaining the proposed Segment Propagation stage: corresponding segments between Label Volume and the current depth map are identified based on 3D overlap percentage of the current depth map and Label Volume.

computed by rendering the Label Volume from the currently estimated camera pose, $\boldsymbol{T}_t$. This map stores only the visible segments from the current camera pose. Then, to efficiently determine segment correspondences between $\mathcal{L}_t^m$ and $\mathcal{L}_t$, we compute the percentage of 3D overlap of every segment $l_j \in \mathcal{L}_t$ with every segment $l_i \in \mathcal{L}_t^m$. By maximizing such percentage, we can easily compute, for each segment l_j, its maximally-overlapping segment $\tilde{l_i} \in \mathcal{L}_t^m$. If the overlap percentage yielded by $\tilde{l_i}$ is higher than a threshold, we propagate its label to the label map $\mathcal{L}_t^p$. Otherwise, we propagate the label l_j directly from $\mathcal{L}_t$, which means that such segment has not yet been seen in the Label Volume.

3) Segment Merging: Then, in the Segment Merging stage, the multiple segments which are part of the same surface are detected and merged by means of the same criterion used in the previous Segment Propagation stage, i.e. determine whether the underlying surface of the two segments is the same based on their 3D overlap. In particular, 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): all such segments are then assigned the same label in $\mathcal{L}$.

4) Segment Update: Finally, in the Segment Update stage, the Label Volume $\mathcal{L}$ is updated by means of the propagated label map $\mathcal{L}_t^p$. To withstand the noise of each depth map, we adopt a confidence based approach, by associating each element $\mathcal{L}(\boldsymbol{p})$ with a confidence $\Psi_t(\boldsymbol{p})$. Such a confidence is initialized and updated by computing the corresponding element on the propagated label map $\mathcal{L}_t^p(\boldsymbol{u})$ by means of the similar way to the Global Model Update stage (see Sec. II-A.3). In particular, the confidence on the voxel which is located within truncated distance from surface in the TSDF volume is updated. If $\mathcal{L}(\boldsymbol{p})$ is unlabeled, it is set to the label of $\mathcal{L}_t^p(\boldsymbol{u})$ while the confidence is initialized as $\Psi_t(\boldsymbol{p}) = 0$. Then, $\Psi_t(\boldsymbol{p})$ is incremented or decremented depending on whether the labels of $\mathcal{L}(\boldsymbol{p})$ and $\mathcal{L}_t^p(\boldsymbol{u})$ are either the same or different. When $\Psi_t(\boldsymbol{p})$ drops to 0, this means that the segment has changed, e.g. due to noise or a change in the environment structure (in case of dynamic scenes). In this case, the point is assigned the corresponding label on $\mathcal{L}_t^p$, i.e. $\mathcal{L}(\boldsymbol{p}) = \mathcal{L}_t^p(\boldsymbol{u})$.

C. Recognition Framework

This subsection describes the stages carrying out 3D object recognition from fully-3D data: they are depicted in orange in Fig. 2. Along these stages, each segment extracted from the currently reconstructed scene as part of the Label Volume $\mathcal{L}$ is deployed as a potential object candidate and matched against the pre-trained database of 3D descriptors computed from full-3D models. This is inspired from the 3D object recognition pipeline based on global descriptors in [5], where each segment obtained from a frame-wise segmentation of the scene is matched against the database of model views. Importantly, at each new frame, we avoid to match those segments that have not been updated by the current depth map. Since it avoids recomputing the 3D descriptors, this notably speeds-up the overall efficiency. In addition, we introduce a fast Geometric Verification stage: this is a novel approach to be applied after descriptor matching, aimed at increasing the robustness of the recognition process by verifying the geometric consistency of each object hypothesis with respect to the TSDF surface of the matched segment.

1) Segment Description: First, in the Segment Description stage, the subset of indices of all segments that need to be matched are extracted from the Label Volume. As anticipated, this subset only contains those segments belonging to both the Label volume and the propagated label map ($l_i \in (\mathcal{L} \wedge \mathcal{L}_t^p)$), to avoid recomputing 3D descriptors for segments that have not changed in the current frame. The 3D vertices corresponding to this subset of indices is in turn extracted from the TSDF volume, by detecting the zero-crossing voxels. Each vertex is also associated to a normal, by calculating the derivative of the TSDF values over neighboring voxels.

Successively, a 3D global descriptor is used to describe the geometry of the current segment. Since the 3D representation of the current segment is a set of 3D vertices, we can directly employ 3D descriptors proposed for point clouds such as VFH [24], CVFH [25] and OUR-CVFH [4]. In particular, the advantage of a descriptor such as OUR-CVFH is that, by explicitly relying on a repeatable Local Coordinate Frames (LRF), it provides also a 3D alignment between two segments. This is particularly useful for those applications that require the 3D pose of the object in the scene, e.g. for augmented reality or robotic manipulation applications.

As a result of the Segment Description stage, the set of 3D descriptors $f(l_i)$ and associated LRF-based transformation $\boldsymbol{T}_{l_i}$ (when available) is computed on the segments $l_i \in (\mathcal{L} \wedge \mathcal{L}_t^p)$. It is worth pointing out that simple heuristics can be deployed at this point to avoid computing 3D descriptors on segments that are clearly not part of the object model database. A simple approach that we adopt is to discard 3D segments whose number of vertices is bigger than the maximum amount or smaller than the minimum amount of the objects in the model database:

$$|l_i| < (1-\beta) \min_{m_k \in \mathcal{M}} |m_k| \tag{4}$$

$$|l_i| > (1+\beta) \max_{m_k \in \mathcal{M}} |m_k| \tag{5}$$

where $|\cdot|$ is the cardinality operator for the set of points of a segment or a model, while β is a percentual margin value for segment discarding. If both conditions (4) and (5) hold, then

the current segment l_i is not considered as a possible scene object. In our experiments, β is set to 0.2, this means that every segment whose number of vertices is smaller than 80% of the amount of vertices of the smallest model, or bigger than 120% of the amount of vertices of the biggest model is discarded.

2) Descriptor Matching : During the Descriptor Matching stage, object hypotheses at each segment are generated by matching the associated descriptor to the database of model descriptors. The best matched candidates associated to each segment l_i on current observation is computed by performing a Nearest Neighbor Search (NNS) across the pre-built kd-tree database by computing the descriptor distance $d(f(l_i), f(m_k))$ between the descriptors of the current segments and the descriptors of each model $m_k \in \mathcal{M}$ in the database, according to the L_2 metric.

$$\tilde{m}_{l_i} = \underset{m_k \in \mathcal{M}}{\arg\min}\, d\left(f\left(l_i\right), f\left(m_k\right)\right) \tag{6}$$

When $d\left(f(l_i), f(\tilde{m}\left(l_i\right))\right)$ is below a matching threshold, an observed object hypothesis is generated associating segment l_i to the respective best matching model $\tilde{m}_{l_i}$.

3) Segment Alignment: In the Segment Alignment stage, if an LRF associated to each descriptor is available, the object's 6DoF pose can be estimated by computing the transformation that aligns the two LRFs [4]. In particular, the object pose for segment l_i can be computed as:

$$\boldsymbol{T}_{l_i\tilde{m}} = \boldsymbol{T}_{l_i}^{-1}\boldsymbol{T}_{\tilde{m}_{l_i}} \tag{7}$$

where $\boldsymbol{T}_{l_i}$ is the LRF associated to the 3D descriptor of segment l_i, while $\boldsymbol{T}_{\tilde{m}_{l_i}}$ is the LRF associated to the best matching model descriptor $f(\tilde{m}_{l_i})$.

4) Geometric Verification : Finally, in the Geometric Verification stage, the obtained object hypotheses are verified by computing the residuals in the 3D space between the model and the reconstructed scene by means of the TSDF volume $\mathcal{T}_t$. To recognize the object that has grown the highest confidence throughout multiple views, a current residual $R_t\left(l_i\right)$ and current model $m_t\left(l_i\right)$ are initially associated to each matched scene segment l_i at the first frame $t = 0$, and updated at each new frame. To update it, an *observed* residual $R_t^o\left(l_i\right)$ is computed for each visible segment of the Label Volume $l_i \in (\mathcal{L} \wedge \mathcal{L}_t^p)$ and best matched model $\tilde{m}\left(l_i\right) \in \mathcal{M}$ as the (truncated) L_1 distance between the 3D model surface and the segment surface:

$$R_t^o\left(l_i\right) = \frac{1}{|\tilde{m}_{l_i}|}\sum_{i=1}^{|\tilde{m}_{l_i}|}|\mathcal{T}_t(\boldsymbol{T}_{l_i\tilde{m}} \cdot \tilde{\boldsymbol{m}}_{\boldsymbol{l_i}}(i))| \tag{8}$$

where the index i loops over all 3D points of current model $\tilde{m}_{l_i}$. As it can be seen from (8), we can obtain a good estimate of such residual, which provides a measure of the 3D fitting between segment and model, directly by indexing the elements of the TSDF where the transformed model points falls according to the estimated pose. This provides a great benefit in terms of efficiency, since we can skip the costly NNS operation at each 3D point of the model, differently from the state of the art geometric verification methods [5], [6].

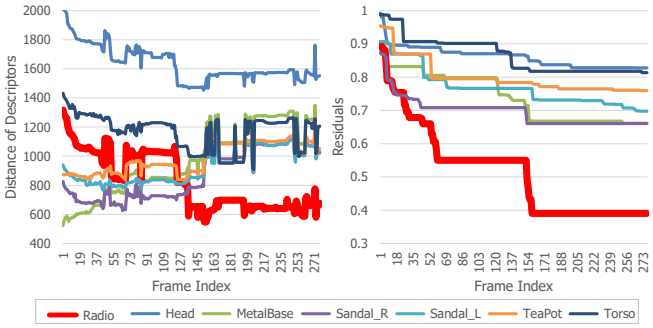


Fig. 4. Comparison between the use of the descriptor distance (left chart) and the geometric residual values (right char) for object recognition, on the scene segment corresponding to the object *Radio* over the whole sequence of Fig. 9.

Then, the current residuals $R_t\left(l_i\right)$ and current matched model $m_t\left(l_i\right)$ on segment l_i are updated by means of the observed residuals $R_t^o\left(l_i\right)$ as:

$$R_t\left(l_i\right) = \begin{cases} R_t^o\left(l_i\right) & \text{if } R^o\left(l_i\right) < R_{t-1}\left(l_i\right) \\ R_{t-1}\left(l_i\right) & \text{otherwise} \end{cases} \tag{9}$$

i.e., if the observed residual is lower than the previous value, the residual and model are updated to new observation: $m_t\left(l_i\right) = \tilde{m}_{l_i}$, otherwise the matched model $\tilde{m}_{l_i}$ is discarded. Due to this scheme, only the object hypotheses yielding the lowest residuals are recognized throughout the multiple views. A final check is applied, so that if $R_t(L_t)$ is greater than a certain threshold it is not assigned to any model, this meaning that no database model matches that segment. This is particularly useful in case of cluttering objects that are not part of the object database.

To demonstrate the usefulness of the proposed Geometric Verification approach in increasing the robustness of object recognition, we provide the experiment in Fig. 4, which shows the measured descriptor distance (left chart) and residual values (right chart) among all models and the scene segment corresponding to the object *Radio* over the whole sequence of Fig. 9. As we can see, by means of the proposed Geometric Verification stage we can already recognize the correct model after 40 frames rather than 130, the residual case yielding overall a remarkably bigger gap between the best model and the second-best compared to the descriptor distance case.

III. Experimental Results

In this section, we provide quantitative experimental result by means of a comparison, on a 3D benchmark dataset, between our recognition pipeline and the corresponding object recognition based on matching 2.5D data. In addition, we also provide qualitative results where our framework is tested under challenging conditions in terms of clutter, occlusions object model shapes. Finally, we show an application of our approach in an augmented reality context, to show how well our proposal can suit such scenario.

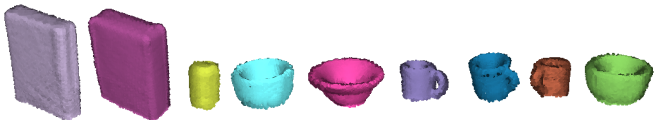


Fig. 5. The 9 full 3D models of the dataset reconstructed via Kinect Fusion.

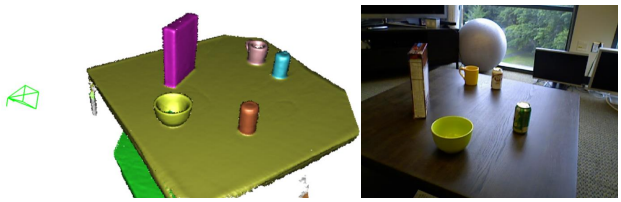


Fig. 6. An example of scene reconstruction and segmentation provided by our method on the *scene_02* sequence of the RGB-D Scenes Dataset v2[18].

A. Object recognition accuracy

We compare the proposed recognition framework against the single-view recognition pipeline based on global descriptor computed on 2.5D data [5]. As for this pipeline, each model is rendered into different views, while the scene is segmented using a segmentation algorithm for range maps. Then, each scene segment and model view is associated to a 3D descriptor, which is the matched to retrieve object correspondences. For fairness of comparison we employ, as the range map segmentation algorithm of this pipeline, the same algorithm used in the Depth Map Segmentation stage of our framework. To normalize the point cloud density as in [5], we use a uniform sampling with the same resolution as the one used for data pre-processing within our framework.

For the comparison, we have used the public *RGB-D Scenes Dataset v2* [18]. Five sequences from the dataset (scenes number 1, 3, 9, 11 and 12) were used for reconstructing via Kinect fusion the 3D models, so to obtain all 9 object models that are present on the dataset (shown in Fig. 5). The 3D model descriptors are then computed directly on these reconstructed models. Then, 7 sequences (number 2, 4, 5, 6, 7, 8, 10) were used for testing the two pipelines in terms of 3D object recognition, i.e. all remaining sequences except for scenes number 13 and 14, which consists of only partial 3D observations. Each of these sequences contain from 3 to 5 objects on a table-top scenario, with each object undergoing strong occlusions. Fig. 6 shows an example of a reconstructed and segmented scene (*scene_2* sequence) as provided by our framework.

Both pipelines have been tested on the same platforms, a desktop PC equipped with an Intel Xeon CPU at 2.4GHz with 16GB of RAM, and a Nvidia Quadro K5200 GPU with 8GB of VRAM. The resolution of the input depth maps is 640×480, the resolution of the TSDF volume is 512×512×512, and the voxel scale is 5 mm. The Kinect Fusion Reconstruction stages and Depth Map Segmentation stage are implemented on GPU, while the remaining segmentation stages and all object recognition stages are implemented on CPU.

Fig. 7 shows the Precision-Recall curve concerning the comparison between our method and the single view pipeline in terms of 3D object recognition using three different global descriptors, i.e. VFH[24], CVFH[25] and OUR-CVFH[5], whose implementations are publicly available in the Point Cloud Library (PCL)[1]. The Precision-Recall curve is calculated by changing the threshold of the final check on geometric verification (See Sec. II-C.4). As for the single-view pipeline, results are averaged over all views. The charts also report the theoretical curve yielded by selecting, for each sequence, the single view that yields the best results (*best view*), so to observe the ideal performance of a single view approach that selects the most advantageous viewpoint.

The results show the remarkable advantage of directly matching full 3D models against reconstructed 3D scenes, as opposed to single view-based recognition, even if the dataset contains very similar 3D shapes such as the models of the cup shown in Fig. 5. This situation, although very common in practice, indeed represents a typical failure case for single view recognition pipelines. Interestingly, our approach also outperforms the single view pipeline in case of selection of the best viewpoint, this motivating further the use of such representations. The same trend is exhibited for all the 3 global descriptors used, the best performance obtained by using the OUR-CVFH descriptor.

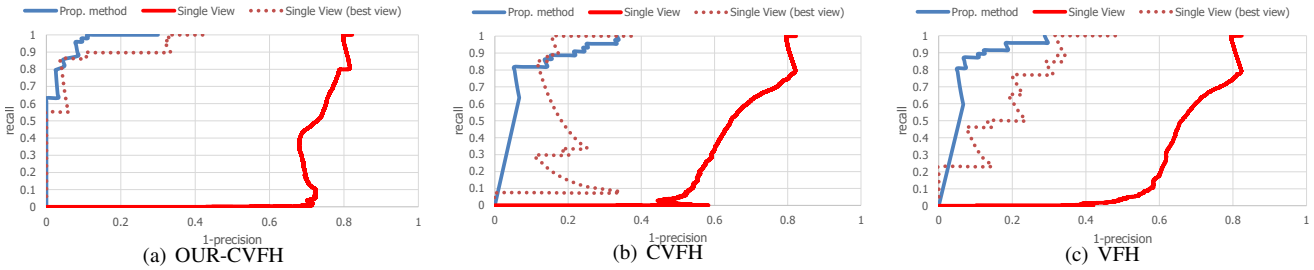
In addition, Fig. 8 illustrates the Precision-Recall curves obtained using different view point coverage for reconstruction, segmentation and recognition, averaged on all test sequences and obtained with the OUR-CVFH descriptor. In particular, we compare full 3D reconstruction (blue line) against using only a limited number of views (respectively, only 75%, 50% and 25% of continuous frames of each sequence). As expected, the result shows that the recognition

TABLE I

MEASURED EXECUTION TIMES OF EACH STAGE INVOLVED IN THE PROPOSED PIPELINE AVERAGED ON THE RGB-D SCENES DATASET V2 [18]. ALL REPORTED EXECUTION TIMES ARE IN MS.

DATA STATISTICS		
Number of frames	5586	
Depth Map Resolution	640×480	
Resolution of TSDF volume	512×512×512	
RECONSTRUCTION		
Preprocessing	7.49	
Camera Pose Estimation	7.06	
Global Model Update	9.65	
Global Model Rendering	4.48	
Total	**28.68**	
SEGMENTATION		
Depth Map Segmentation	4.37	
Segment Label Propagation	3.72	
Segment Merging	0.86	
Segment Update	1.29	
Total	**10.17**	
RECOGNITION	OUR-CVFH	VFH
Segment Description	213.62	20.64
Segment Matching	0.91	0.03
Segment Alignment	0.01	—
Geometric Verification	2.18	—
Total	**216.72**	**20.68**

[1] www.pointclouds.org



(a) OUR-CVFH (b) CVFH (c) VFH

Fig. 7. Precision-Recall curve for 3D object recognition (no pose estimation) comparing the proposed framework with the single-view pipeline. Each chart is relative to a different global descriptor (VFH, CVFH and OUR-CVFH) whose implementation is available in the Point Cloud Library (PCL). The charts also report the theoretical curve yielded by selecting, for each sequence, the view that yields the best results (*best view* curve).

performance improves as more viewpoints are processed. As a complement, the chart also shows the result yielded by our method without the proposed Geometric Verification stage (red line). The result illustrates the usefulness brought in by the use of this stage in terms of recognition performance.

In terms of qualitative results, Fig. 1 shows the result of our framework in a highly challenging scene, typical for indoor robotic perception, composed of similar household objects, characterized by poor geometrical shapes and placed over different metallic planes (noisy to acquire with low-cost RGB-D sensors). We point the reader to the supplementary material, which includes additional qualitative results within challenging indoor scenarios.

Finally, Table I shows the measured execution times for each stage of the proposed framework averaged on the whole test sequences. From the table, we can see that the computational burden associated to the reconstruction and segmentation stages is sufficiently low to allow real-time performance. The main bottleneck is currently represented by the feature description stage: in particular, OUR-CVFH descriptor has a high computational cost, as witnessed by the reported time of the Description Stage on the Table. For this reason, we use two different CPU threads, one devoted to reconstruction and segmentation, the other carrying out object recognition, so to allow the entire framework to run in near real-time, with a reported frame rate of 4.6 fps. On a single CPU core, the reported frame rate gets down to 3.9 fps. Conversely, by using the VFH descriptor, the whole pipeline reports a frame rate of 25.7 fps by using two CPU threads as previously explained, while can still run at 16.8 fps on a single CPU thread.

B. Augmented Reality application

As a complement to previous results, we show an application of our framework to 3D object recognition and 3D augmented reality in indoor environments. In particular, we acquired some RGB-D sequences with our own setup based on a PrimeSense Carmine 1.09 RGB-D sensor. A few examples of the input data, as well as the output in terms of recognition and augmented reality, are shown in Fig. 9, while the complete video is included as Supplementary Material. The setup refers to a highly cluttered table-top scenario, with 10 models used for object recognition. In particular, the left column on Fig. 9 shows, from top to bottom, the input

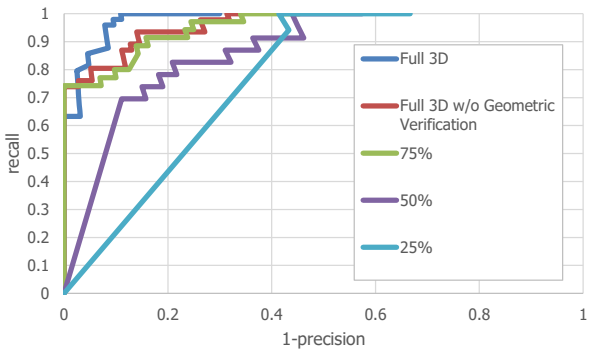


Fig. 8. Precision-Recall curves reported by the proposed framework using different number of viewpoints (from 25% of the entire viewpoints to full 3D) and using the OUR-CVFH descriptor. The chart also shows the recognition performance obtained without the proposed Geometrical Verification stage.

RGB image and the normal map computed from the input depth image, the TSDF reconstruction and the segmentation result. In the middle, the figure shows the output in terms of object recognition and pose estimation. As shown by the central image, our method can recognize the *Helmet* object, demonstrating how it can deal also with smooth and spherical shapes where keypoints can hardly be extracted. Finally, images on the right side show real-time augmentation of a black hat with 3D content of a recognized object. Notably, in this case, the augmented content is aligned with respect to the recognized object by means of its estimated pose, and can be shown also coherently with respect to occlusions due to the scene geometry (including 3D self-occlusions), as shown in the bottom right image of Fig. 9. Once the object is recognized, disadvantageous viewpoints where the object gets occluded by foreground do not deteriorate the augmented content, as visible also from the video included in the supplementary material.

IV. Conclusions and Future Work

We analyzed and experimentally evaluated how full 3D object recognition is advantageous with respect to standard single view-based approaches. We have proposed a framework for Simultaneous Reconstruction, Segmentation and Recognition, which incrementally segments and recognizes full 3D objects out of a Kinect Fusion reconstruction, and yields robust object recognition and 3D pose estimation,

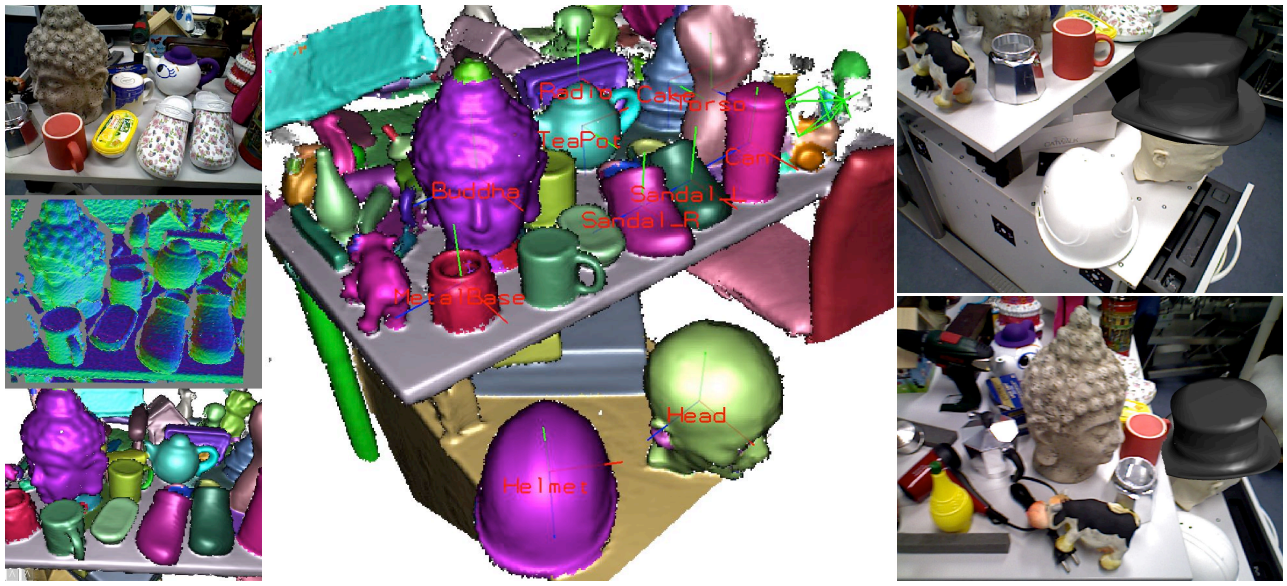


Fig. 9. Application of the proposed framework to 3D augmented reality. Left column, from top to bottom: input RGB frame, normal map, TSDF reconstruction and segmentation result. Middle: output of the object recognition and pose estimation from a database of 10 models. Right side: real-time augmentation of a black hat with 3D content of a recognized object.

useful for a number of applications such as grasping, manipulation and augmented reality. Future direction regards the use of the color cue in the recognition stages of our framework, so to be able to recognize 3D objects with similar geometry and distinctive texture. Another interesting direction will be the extension of our framework to object discovery applications and to large scale environments.

REFERENCES

[1] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient Response Maps for Real-Time Detection of Textureless Objects," *TPAMI*, vol. 34, no. 5, pp. 876–888, 2012.

[2] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3D object detection: A real time scalable approach," in *Int. Conf. Computer Vision (ICCV)*, 2013.

[3] A. Tejani, D. Tang, R. Kouskouridas, and T.-k. Kim, "Latent-class hough forests for 3D object detection and pose estimation," in *European Conf. Computer Vision (ECCV)*, 2014.

[4] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "Our-cvfh: Oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation," in *Joint DAGM-OAGM Pattern Recognition Symposium*, 2012.

[5] 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.

[6] Z. Xie, A. Singh, J. Uang, K. S. Narayan, and P. Abbeel, "Multimodal blending for high-accuracy instance recognition," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[7] 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.

[8] A. Pieropan and H. Kjellstrom, "Unsupervised object exploration using context,," *Int. Symp. Robot and Human Interactive Communication (RO-MAN)*, 2014.

[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," *IEEE Int. Symp Mixed and Augmented Reality*, 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," *IEEE Int. Conf. on Intelligent Robots and Systems*, 2013.

[13] R. Salas-Moreno, R. Newe, H. Strasdat, and P. Kelly, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[14] N. Fioraio and L. Di Stefano, "Joint detection, tracking and mapping by semantic bundle adjustment," in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[15] L. Nan, K. Xie, and A. Sharf, "A search-classify approach for cluttered indoor scene understanding," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2012)*, vol. 31, no. 6, 2012.

[16] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an rgbd camera," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 136:1–136:11, nov 2012.

[17] S. Pillai and J. Leonard, "Monocular SLAM Supported Object Recognition," in *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[18] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3d scene labeling," in *Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[19] Y. Li, A. Dai, L. Guibas, and M. Niessner, "Database-assisted object retrieval for real-time 3d reconstruction," in *Eurographics*, 2015.

[20] J. Valentin, V. Vineet, M. M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Niessner, A. Criminisi, S. Izadi, and P. Torr, "Semanticpaint: Interactive 3d labeling and learning at your fingertips," *ACM Transactions on Graphics*, 2015.

[21] V. Vineet, O. Miksik, M. Lidegaard, M. Niessner, S. Golodetz, V. A. Prisacariu, O. Kaehler, D. W. Murray, S. Izadi, P. Perez, and P. Torr, "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction," in *Int. Conf. Robotics and Automation (ICRA)*, 2015.

[22] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense slam," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[23] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Int. Conf. on Computer Vision (ICCV)*, 1998.

[24] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 1–6.

[25] A. Aldoma, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, M. Vincze, and G. Bradski, "Cad-model recognition and 6 dof pose estimation using 3d cues," in *Proc. ICCV Workshop on 3D Representation and Recognition (3DRR)*, 2011, p. 585592.