IEEE Robotics and Automation Letters (RAL) paper presented at the
2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Real-Time Fully Incremental Scene Understanding on Mobile Platforms

Johanna Wald[1], Keisuke Tateno[1,2], Jürgen Sturm[3], Nassir Navab[1], Federico Tombari[1]

***Abstract*— We propose an online RGB-D based scene understanding method for indoor scenes running in real-time on mobile devices. First, we incrementally reconstruct the scene via SLAM and compute a 3D geometric segmentation by fusing segments obtained from each input depth image in a global 3D model. We combine this geometric segmentation with semantic annotations to obtain a semantic segmentation in form of a semantic map. To accomplish efficient semantic segmentation, we encode the segments in the global model with a fast incremental 3D descriptor and use a random forest to determine its semantic label. The predictions from successive frames are then fused to obtain a confident semantic class across time. As a result, the overall method achieves an accuracy that gets close to most state-of-the-art 3D scene understanding methods while being much more efficient, enabling real-time execution on low-power embedded systems.**

I. INTRODUCTION

To advance the state of the art in robotic perception and augmented reality, the research community in computer vision and robotics has recently focused on 3D reconstruction and scene understanding algorithms with the goal of equipping mobile devices and mobile robots with full spatial and semantic awareness [1]. The goal of such algorithms is to extract semantic information from the environment and to automatically determine properties of the objects therein, to enable spatial reasoning for applications in robotics, gaming, and augmented reality. The availability of 3D sensors simplifies this task, as witnessed by many commonly employed 3D reconstruction and semantic mapping algorithms that rely on 3D data acquired from consumer depth cameras [2], [3], [4], [5], [6]. Nevertheless, scene understanding is still an open problem since the data acquired from these sensors is often noisy and incomplete, and the physical characteristics of the objects and scene elements (such as color and shape) greatly varies across different environments. However, current techniques mostly rely on computationally heavy algorithms for 3D reconstruction and object classification that are not yet real-time compatible with typical resources available on low-powered and memory-limited hardware platforms such as smartphones or embedded computers designed to be used with mobile autonomous robots. At the same time, real-time perception is required for many real-world applications in robotics and augmented reality.

[1] Computer Aided Medical Procedures and Augmented Reality, Technical University of Munich, Boltzmannstrasse 3, 85748 Garching, Germany johanna.wald@tum.de, tateno, tombari, navab@in.tum.de

[2] Canon Inc., Tokyo 146-0092, Japan tateno.keisuke@canon.co.jp

[3] Google Germany GmbH, Erika-Mann-Strasse 33, 80636 Munich jsturm@google.com

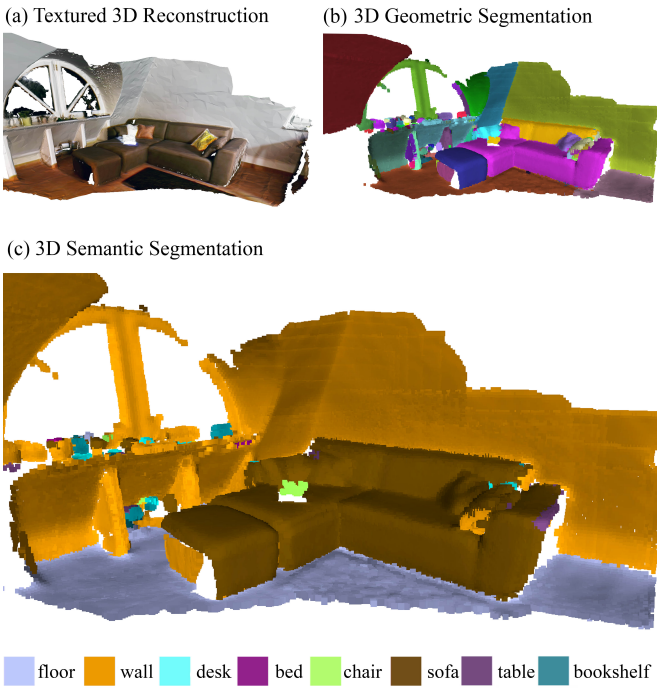

Fig. 1. Visualization of the semantic labels obtained by our approach on a sequence using a Google Tango device (see supplementary video). The image on the bottom shows the resulting semantic segmentation of the scene into sofa, walls and floor etc.

In this paper, we propose an algorithm for real-time incremental 3D scene understanding. Compared to the state of the art, it is designed to be very efficient and scalable, since its computation is fully incremental, so to be compatible with low-power, memory-constrained devices.

The main contribution of this work is the definition of a descriptor for 3D point clouds that can be updated incrementally as new data arrives. We demonstrate that our descriptor is rich enough for semantic classification, e.g., to predict whether a segment corresponds to a chair, table or wall. In particular, we trained a random forest on the descriptor to predict the semantic class of the corresponding segment. By integrating the descriptor and classifier with a point-based fusion approach [7], we obtain a real-time pipeline for fully incremental 3D reconstruction and scene understanding. In contrast to existing approaches on semantic segmentation, our descriptor recycles previous operations to take only constant time per frame regardless of the current size of the global map and therefore is able to encode 3D segments in constant time per frame. Since descriptors require to be updated only when new points get integrated,

Fig. 2. Outline of our approach. (c) During online reconstruction we generate a set of 3D segments with associated feature descriptors. (d) We use a random forest to predict semantic labels based on our feature descriptor. (e) Visualization of predictions.

the total computation time per frame depends only on sensor properties but is independent of the size of the map.

We evaluate the performance of the proposed framework on a benchmark dataset, ScanNet [8] showing that it achieves comparable accuracy to some state-of-the-art approaches while being significantly more efficient and scalable – in particular, it does not require a GPU and runs in real-time on mobile devices. To demonstrate this, we ported our approach to a Google Tango device as shown in Fig. 1 and in Sec. IV. In section II, we discuss related literature in the field of scene understanding, semantic mapping and 3D descriptors. We then describe our proposed method in section III, which is successively evaluated and compared in section IV.

II. RELATED WORK

Most algorithms that approach scene understanding can be categorized as either focused on semantic segmentation or object recognition. The former aims at densely classifying image pixels, 3D points or voxels into semantic classes, whereas the latter aims at localizing instances of object classes either on the image plane or in the 3D scene geometry. Some scene understanding approaches are designed in an incremental fashion so to update their internal semantic model at every new input frame (see Sec. II-A). We will refer to such approaches as *incremental* scene understanding, as opposed to those which batch process a pre-computed 3D reconstruction, which will be referred as *offline* scene understanding (see Sec. II-B). A few approaches do not deal with 3D data and reconstruction, and carry out scene understanding on the image plane, e.g. Choi et al. [9] which fuse different scene understanding tasks such as object detection, layout estimation and scene classification while operating on a single image. Hence, we will not review such avenue of related works as it falls outside the scope of this work.

A. Incremental Scene Understanding

Most incremental scene understanding algorithms rely on Simultaneous Localization and Mapping (SLAM) [4], [3] to obtain a 3D reconstruction of the environment together with a set of camera poses associated to each input frame. The geometric information of the map is then combined with semantic information extracted via geometric primitives, typically embedded as 3D descriptors (see subsection II-C for a brief review). In [10], a method is proposed to incrementally detect planes on the depth image and gradually merge them into a global model, while improving SLAM [10]. This method limits the semantic parsing to planes, not taking into consideration curved surfaces or non-planar objects. In SLAM++ [11], Salas-Moreno et al. combine the task of object detection with SLAM. Incremental reconstruction and object detection is proposed also by Tateno et al. [7] where the scene is incrementally segmented into geometric connected components and a descriptor is computed for each 3D segment. By matching descriptors to a database of 3D objects, object detection and 3D pose estimation is achieved. In [12] a method for real-time tracking of unknown objects with a shape defined by a surface of revolution (SoR) in an incrementally reconstructed environment is proposed. They can deal with cluttered scenes although they are restricted to symmetric objects that can be approximated by a SoR. Li et al. achieve incremental scene understanding on dense SLAM by fusing object predictions from different viewpoints [13] yielding a semantic segmentation map driven by object instances rather than classes. SemanticPaint [6] relies on user interaction to add semantic information to reconstructed scene parts via voxel hashing. The provided semantic information is also learned via an online procedure based on random forests. The methods yields real-time performance on a desktop PC via GPU optimization [6]. Finally, SemanticFusion [14] performs 3D semantic segmentation by fusing CNN-based semantic segmentation from independent frames into a SLAM reconstruction obtained via Elastic Fusion [15]. The method also runs in real-time on a desktop PC equipped with a powerful GPU.

B. Offline Scene Understanding

Algorithms that assume as input a given 3D reconstruction achieve state of the art accuracy, but are – by definition – not designed for real time execution [16], [17], [18]. As such, PointNet [16] conducts either segmentation, semantic annotation or object detection with a neural network to process a raw 3D point cloud. Qi et al. later proposed PointNet++,

a multi-scale network that uses PointNet to capture local features [19]. Similarly, [17] computes object labels in a given point cloud in contrast to [18] who introduce the so-called Deep Sliding Shapes, that output 3D object bounding boxes based on pre-existing 3D scenes. More recently, [8] semantically classifies voxel-based representations. Specifically, subvolumes are extracted from a voxel map and fed to a 3D Convolutional Neural Network (CNN) that predicts a class label based on the occupancy characteristics of the neighborhood of a voxel column. Due to the complexity of the networks used, a powerful GPU is required, and the voxel maps or point clouds need to be preprocessed offline. Compared to these offline scene understanding methods our focus is to run in real-time on a mobile device while acquiring comparable results.

C. 3D Descriptors in Scene Understanding

3D descriptors [20], [21], [22], [23] encode the geometrical (and sometimes color) properties of the neighborhood of a 3D point. They are commonly employed for tasks such as 3D object recognition, pose estimation and point cloud registration. In SLAM++ [11], Point Pair Features (PPF) descriptors [23] are used for 3D object detection. In [7], object detection and pose estimation is demonstrated by means of three different global descriptors, i.e. VFH [24], CVFH [25] and OUR-CVFH [26]. In many applications, 3D descriptors can hardly match real-time constraints as each descriptor requires to process the entire 3D neighborhood, which is often stored as an unorganized 3D representation (e.g. a point cloud or 3D mesh), hence requiring indexing techniques such as kd-trees or octrees. This is particularly true in case 3D descriptors are used for dense semantic segmentation, in which case the complexity for processing a scene is typically $\mathcal{O}(n^2)$ with the number of scene points. In contrast, our incremental descriptor stems from the need for efficient dense scene labeling.

III. Incremental Scene Understanding

As illustrated in Fig. 2, our method first applies geometric segmentation within a SLAM-based 3D reconstruction framework as discussed in Sec. III-A. Then, it utilizes a classifier to determine a corresponding semantic class for each 3D segment (see Sec. III-B and III-C). Our approach is based on the incremental segmentation of [27], which we ported and optimized for 3D sensor-enabled mobile devices such as the Google Tango. To inject contextual information into the geometric segments from [27], we introduce a semantic classification based on a novel class of incremental 3D descriptors. For the sake of efficiency, we only classify the segments (generally a small subset) that have been updated by the current input frame. For classification, we employ a random forest [28] that uses the incremental 3D descriptor as feature. Notably, we chose random forests because of their high efficiency and low memory requirements, which makes them suitable for embedded architectures.

A. SLAM-Based Reconstruction and Segmentation

We use Tango's robust internal Visual Inertial Odometry (VIO) to obtain the camera pose. It combines features tracked on the fish-eye camera image with IMU measurements in an Extended Kalman Filter framework. It also performs camera re-localization and pose refinement to reduce drift errors [29]. The incremental segmentation algorithm first computes vertices and normal maps using incoming depth frames. In order to segment the reconstructed 3D scene, connected components extracted from each depth map are fused into a global segmentation map [27]. This map consists of m segments $\mathbf{t}_1, \ldots, \mathbf{t}_m$. A segment $\mathbf{t}_i$ comprises n_i surfels, denoted by $\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_{n_i}}$. Each surfel $\mathbf{s}_i$ is described by its position $\mathbf{x}_i \in \mathbb{R}^3$, radius and confidence [3]. To this end, the geometric segments of the global model are matched with the ones obtained from the current depth map. This operation is particularly fast since we only need to evaluate regions that are currently visible. This results in the processing of a small number of segments; all remaining segments of the global map remain unaffected. If two separate regions overlap in consecutive frames the two corresponding segments are potentially merged into a single one. Its 3D descriptor, which will be illustrated in the following, is then given by the arithmetic addition of the merged features. As shown in Sec. III-B, this operation is computable in constant time $\mathcal{O}(1)$ regardless of the size of the merged segments. In a final step within the segmentation procedure all visible segments are updated. Each geometric label has a corresponding confidence value that is increased whenever the geometric segment has been successfully propagated. This is necessary since the depth measurements are noisy and quite unreliable, in particular when using Tango's mobile depth sensor. This happens together with the segment merging within the global model update loop and can therefore be computed very efficiently.

B. 3D Incremental Descriptor

The incremental 3D descriptor is the core of our semantic segmentation approach and the main contribution of this work. It incrementally encodes the surfels of each geometric segment $\mathbf{t}$ obtained in section III-A into a vector $\mathbf{d} \in \mathbb{R}^k$. The computational complexity of computing the descriptor is linear in the number of surfels n. As such, adding a 3D point to an existing descriptor is a constant operation. More generally, we can even define an arithmetic operation to add two descriptors $\mathbf{d}_1$ and $\mathbf{d}_2$ such that $\mathbf{d} = \mathbf{d}_1 \oplus \mathbf{d}_2$ in $\mathcal{O}(1)$ without the need of any re-computation. This comes especially handy for incremental updates and merging segments. Different incremental descriptors can be designed by encoding different 3D properties based on the available data types and modalities, such as geometry, color or texture. In our implementation, we rely on the scene geometry only which we embed by means of different distinctive 3D characteristics such as the distribution of the point coordinates and the normal vector distribution. Specifically, the following incrementally computable properties are used

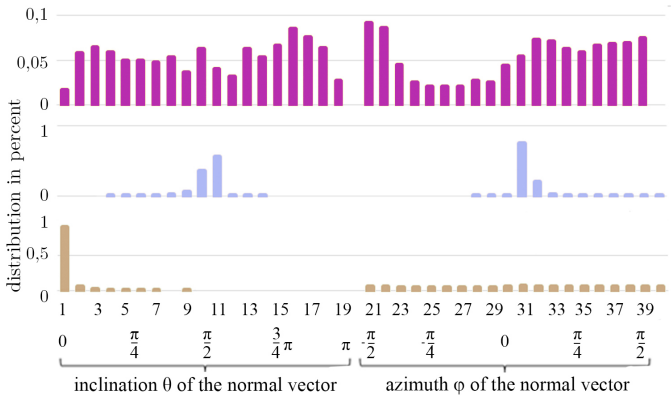

Fig. 3. Histogram of the normal vector distribution of three different segments in spherical coordinates with θ, Φ binned in 20 container. The top row corresponds to an object, the middle row to a wall component and the bottom row to a floor segment.

as features for our semantic classification.

Centroid: We refer to the centroid of each segment $\mathbf{t}_i = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ as $\boldsymbol{\mu}_i \in \mathbb{R}^3$ with

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{j \in \{1,\ldots,n_i\}} \mathbf{x}_j. \tag{1}$$

The centroid μ_i is computed for all surfels of each segment. There, we exploit the property that each point of the global model appears in one and only one segment , hence it should be accounted only once in the computation of the set of centroids. Note that a centroid can be trivially updated when a new surfel $\mathbf{s}_j$ gets added, e.g.,

$$\boldsymbol{\mu}_i \leftarrow \frac{1}{n_i + 1} \left(n_i \boldsymbol{\mu}_i + \mathbf{x}_j \right), \tag{2}$$

$$n_i \leftarrow n_i + 1. \tag{3}$$

Normal Vectors Distribution: The surface normal vectors $\mathbf{n} \in \mathbb{R}^3$ provide valuable information about the segment geometry. A normal vector is represented either with Cartesian coordinates $(x, y, z)^\top$ with $\|\mathbf{n}\| = 1$ or using spherical coordinates $(r, \theta, \Phi)^\top$ derived as

$$r = \|\boldsymbol{n}\|, \quad \theta = \operatorname{acos}\left(\frac{z}{r}\right), \quad \Phi = \operatorname{atan}\left(\frac{y}{x}\right). \tag{4}$$

Either representation is compatible with our incremental computation scheme. A more universal way to integrate normal information is the concatenation of the histograms of the normal vector coordinates either with Cartesian coordinates (x, y, z) or by using its spherical representation (r, θ, Φ). Please note that (a) the radius $r = 1$ can be omitted reducing the dimensionality of the histogram within the 3D descriptor by a factor of $\frac{1}{3}$ and (b) the computation of *acos* and *atan* can be avoided by storing its values in a lookup table. An example of such histogram with a bin size of $b = 20$ is given in Fig. 3. Whenever two segments are merged or a new point is added to an existing descriptor, their bin values are simply added which costs $\mathcal{O}(b) = \mathcal{O}(1)$.

Bounding Box: Due to the fact that our depth values are measured in absolute real-world space, it is straightforward to compute its physical size in the form of an axis aligned bounding box. For this, the descriptor keeps track of the minimum $x_{\text{min}} = \min(x_1, ..., x_n)$ and maximum value of each 3D coordinate $x_{\text{min}}, y_{\text{min}}, z_{\text{min}}$ and $x_{\text{max}}, y_{\text{max}}, z_{\text{max}}$ respectively. Whenever a point is added, we check if its coordinates are larger or smaller than any current extrema. The width, height and depth of a segment are derived from the absolute difference of the maximum and minimum $x_{\text{size}} = |\max(x_1, ..., x_n) - \min(x_1, ..., x_n)|$, which is incrementally computable as $x_{\text{size}} = |x_{\text{max}} - x_{\text{min}}|$ for each coordinate.

Height Distribution: Another component added to our descriptor is the height distribution of each segment stored in $b = 15$ bins. All values greater than 2 meters are clipped and therefore forced to end up in the last bin b. In ScanNet, reconstructions and their corresponding poses are normalized such that their bounds are within the positive octant of the coordinate system [8]. Using the height distribution to obtain semantic labels therefore naturally works on ScanNet sequences. For real world data acquired by, for example, a Tango device, the ground plane is extracted to normalize the descriptor in a similar way. It is worth mentioning that this assumption of course limits our method to only work on single floors.

The final configuration of the descriptor with its properties is a concatenation of the centroid, the mean normal, the physical size of the geometric segment, its normal vector distribution in spherical coordinates with a bin size of 15 as well as its height distribution with 35 bins. An evaluation of these properties is carried out in section IV.

C. Semantic Classification

The incremental descriptors are used as input feature to train and test a classifier. Specifically, we compute a 3D descriptor for each segment and determine its semantic class by means of a random forest, that is used to predict the semantic label of all segments that have changed in the last iteration due to the new measurements from the current depth map. We have decided to use random forests for the classification since they are designed for multi-class prediction, compatible with embedded architectures, stable towards outliers and also among the fastest classifiers. The aforementioned computation of the incremental descriptor has total linear-time complexity with respect to the number of segment points n. Once the semantic class for all segments is obtained, the 3D segmentation has incrementally turned into a semantic representation where continuous predictions are fused over time. This results in a probability distribution of the semantic classes for each segment. The semantic label of each segment is then given by the class with the highest probability.

D. Training Process

The random forest is trained using ScanNet [8]. It provides ground truth poses, depth maps as well as hand-annotated semantic annotation [8]. Dai et al. provide an annotation of up to 1163 different classes on 1513 scans together with a corresponding mapping to, for example, the popular NYU categories [30]. Those semantically annotated point clouds are the input of our training procedure together with the depth data and the ground truth camera trajectory. ScanNet's ground truth poses are used to take advantage of the aforementioned alignment. In doing so, the 3D data generated during testing recorded live on a Tango device and the sequences of the training data are transformed into the same coordinate system. This is important since the descriptor heavily relies on normal vectors and position information. The alignment is carried out on the device by utilizing IMU measurements available on all common smart-phones. The ground truth semantic labels given by ScanNet are mapped to our 3D segments acquired by the incremental segmentation of the training data. The segmentation of ScanNet's 1045 training sequences produces around $50,000$ annotated segments with corresponding semantic annotations. To obtain a balanced training set, 3D segments of low represented categories (refrigerator, toilet, bathtub, etc.) are augmented yielding around 10k descriptors per class. Segmented point clouds are randomly transformed and either selectively or randomly sampled simulating the incremental scanning procedure of real world data acquisition. This process is also efficient due to the previously mentioned incremental property of our descriptor. An illustration of the training procedure has been given in Fig. 2. There, in a first step, the rendered depth data is fed into the incremental segmentation. In this context, similarly to the testing phase, a 3D descriptor for each segment is incrementally computed. Since ground truth semantic annotation is given, a semantic class for each surfel can be obtained by projecting the 2D labels to the 3D global model. In doing so, we can derive the semantic class of each segment. Segments that exceed a minimum size and their underlying labels are not ambiguous are suitable for training. A segment is ambiguous if less than 80% of the points are annotated with a specific semantic class. Notably, the training of the random forest takes no longer than a few minutes.

IV. EXPERIMENTS

In the following section, we present experimental results of our incremental semantic classification by evaluating it on ScanNet [8] and NYUv2 [31]. First, we compare the runtime of our incremental descriptor with state-of-the-art global 3D descriptors. We evaluate against ScanNet, PointNet and PointNet++ using the voxel based semantic segmentation and against Hermans et al. [5], ScanNet and SemanticFusion on NYUv2. We further report the efficiency of the different stages of the semantic classification pipeline.

TABLE I

OVERALL ACCURACY OF THE FULLY INCREMENTAL PIPELINE USING DIFFERENT 3D DESCRIPTORS ON SCANNET TESTING SEQUENCES.

	FPFH	VFH	CVFH	OUR-CVFH	Ours
Point Accuracy	7.4%	59.5%	61.4%	60.9%	**63.8%**
Unweighted Voxel Accuracy	23.2%	25.9%	28.2%	28.0%	**36.1%**
Weighted Voxel Accuracy	20.4%	53.8%	55.3%	55.3%	**74.8%**

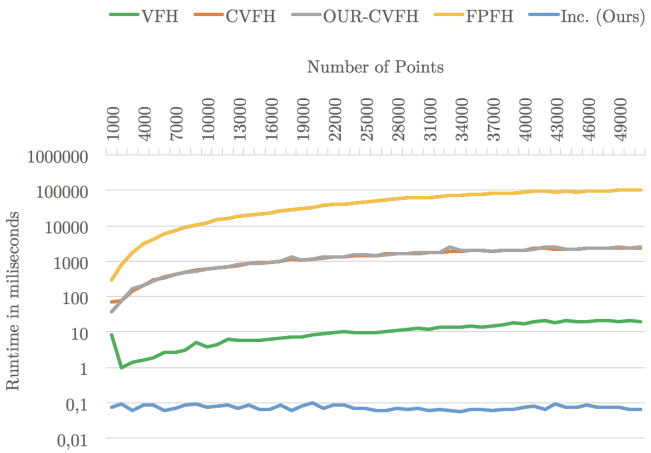

Fig. 4. Comparison of the runtime of the different global 3D descriptors VFH, CVFH, OUR-CVFH and FPFH for a 3D point cloud of increasing size. Note that the runtime is plotted using a logarithmic scale.

A. Incremental Descriptor

We compare our proposed incremental descriptor to the global descriptors VFH, CVFH, OUR-CVFH and FPFH. In our comparison, we used the implementation provided by the Point Cloud Library (PCL) [1]. As proposed by recent performance evaluations of 3D descriptors [32], [33], the parameters used were the default PCL ones [2]. In Table I, the average accuracy on test set of the ScanNet dataset [30] using different descriptor encodings is reported. Our descriptor achieves a weighted accuracy of 74.8% while the best global descriptor (CVFH) achieves 55.3%.

Subsequently, we compared the runtime of feature extraction. All tests were carried out on the same platform, a MacBook Pro with 16GB RAM and a 2.3 GHz Intel Core i5. In our experiment, 1000 points are iteratively added to an existing 3D segment. In Fig. 4, it can be observed that the incremental descriptor does not, in contrast to its non-incremental counterparts, scale with the size of the global map. Also, adding a point to a segment or merging two individual descriptors of an existing point cloud does not trigger a descriptor re-computation but rather is extremely fast running approximately 100 µs on average. This makes it around 120 times faster than VFH for typical segment sizes.

Therefore, by using the incremental descriptor we do not only achieve good performance but also obtain the best results within our semantic segmentation framework.

[1] http://pointclouds.org/

[2] http://pointclouds.org/documentation/tutorials/

TABLE II

EVALUATION OF DEDUCTING DIFFERENT DESCRIPTOR COMPONENTS

Without Descriptor Property	Weighted Accuracy	Mean Accuracy
Centroid	73.8%	**33.8**%
Normal Vectors Inclination	72.9%	29.8%
Normal Vectors Azimuth	73.9%	33.6%
Physical Size	68.4%	19.4%
Mean Euclidean Normal	73.8%	33.0%
Mean Inclination	73.9%	33.2%
Mean Azimuth	**74.2**%	33.4%
Height Distribution	71.9%	29.9%

TABLE III

AVERAGE RUNTIME OF THE SEMANTIC CLASSIFICATION PIPELINE ON TANGO PHONE AT DIFFERENT RESOLUTIONS OF THE INPUT DEPTH MAP.

Depth Map Resolution	224 x 117 Original	112 x 59 1x Subsampled	66 x 30 2x Subsampled
Pre-Processing	104.50 ms	28.13 ms	10.03 ms
Global Model Rendering	72.34 ms	21.91 ms	4.65 ms
Global Model Update	116.64 ms	28.73 ms	5.16 ms
Depth Map Segmentation	45.50 ms	10.86 ms	2.75 ms
Segment Label Propagation	11.47 ms	3.51 ms	0.66 ms
Semantic Classification	34.11 ms	10.49 ms	2.61 ms
Total	384.57 ms	103.64 ms	25.87 ms

B. Fully Incremental Pipeline

Since an online algorithm was required, the processing speed of the algorithm is particularly important. Table III shows its execution time on the mobile device Lenovo Phab 2 Pro. Due to the fact that the algorithm mainly depends on the resolution of the depth map the evaluation has been carried out for different pyramid levels. The segment update and segment merging operations as well as the descriptor computation, are carried out within the *global model update* and are therefore not specifically listed. The prediction of a semantic label usually takes less than 0.4 ms. Furthermore, only labels and descriptors for segments visible in the current view need to be updated and predicted. As stated, the semantic classification runs at 10 and 39 fps with depth maps of level 2 and 1, respectively. Please note that the input frame-rate of the depth data on Tango is only 5 fps. We compare this in Table IV to the runtime of other state-of-the-art methods. This overall proves the efficiency and high degree of scalability of the proposed method. Further, we report the voxel prediction accuracy per number of trees and tree depth respectively (see Fig. 6).

The semantic mapping classifies segments into one of 20 categories given by ScanNet considering mainly structure (wall, floor) and major furniture (bed, desk, bookshelf). To be able to evaluate our method against ScanNet, we uniformly sample the ground truth meshes as well as our segment- and point-based prediction with a sampling rate of 4.8cm, corresponding to the voxel size of ScanNets output. Table VI compares our incremental semantic segmentation with ScanNet's voxel based prediction. In the original evaluation from [8], the reported average accuracy is weighted by the size of each class: with such metric, our approach is able to yield a slightly higher accuracy than ScanNet. For completeness, we also compute the unweighted average accuracy for both ScanNet and our method, since the weighted average accuracy gives us a boost as our algorithm can handle big structures particularly well while struggling with clutter. In this case the performance is worse than ScanNet, which we believe is due to the segmentation quality and the limited descriptiveness of our features when applied to clutter. Figure 7 additionally compares our method to 3DCNN [35], PointNet [16] as well as the different PointNet++ [19] architectures. Please note that, in contrast to the state-of-the-art, our approach is much more efficient and can run on embedded hardware (Table IV).

TABLE IV

COMPARISION AND AVERAGE RUNTIME IN POINTNET, POINTNET++ AND SEMANTICFUSION

Method	Runtime	Hardware
SemanticFusion	39.5 ms	i7-5820K 3.30GHz, NVIDIA Titan Black
PointNet	25.3 ms	GTX 1080
PointNet++ (SSG)	82.4 ms	GTX 1080
PointNet++ (MSG)	163.2 ms	GTX 1080
PointNet++ (MRG)	87.0 ms	GTX 1080
Ours	25.9 ms	Mobile Phone (Lenovo Phab 2 Pro)

Finally, we report qualitative results by evaluating our incremental semantic mapping on a sequence acquired from a Tango device, as shown in Fig. 1. More results from this sequence are available in the submitted supplementary video. Note that the training in this case is still conducted using the ScanNet dataset. This demonstrates the domain generalization capabilities of our framework. Additional qualitative results are shown in Fig. 5 and in the supplementary video on a few test sequences from the ScanNet dataset together with their corresponding ground truth annotations. For what concerns failure cases, classification errors happen especially with cluttered regions, due to the aforementioned reasons. Furthermore, since our semantic annotation is based on the computation of geometric 3D segments, flat objects such as paintings on walls are geometrically combined together, and are therefore hard to correctly annotate.

V. CONCLUSION

We have presented an approach to incrementally segment a scene and to gain continuous high level semantic class annotations in real time. The incremental computation of different scene understanding tasks such as scene segmentation or semantic classification is due to its efficient nature, a very powerful approach and therefore well-suited for the mobile use case. The application of our incremental descriptor gives us the best computational performance as well as accuracy in such a framework compared to other state-of-the-art global descriptors. Our fully incremental pipeline produces semantically annotated point clouds in real time, while being almost on par with most state-of-the-art offline methods. We further want to point out the generality of the incremental descriptor,

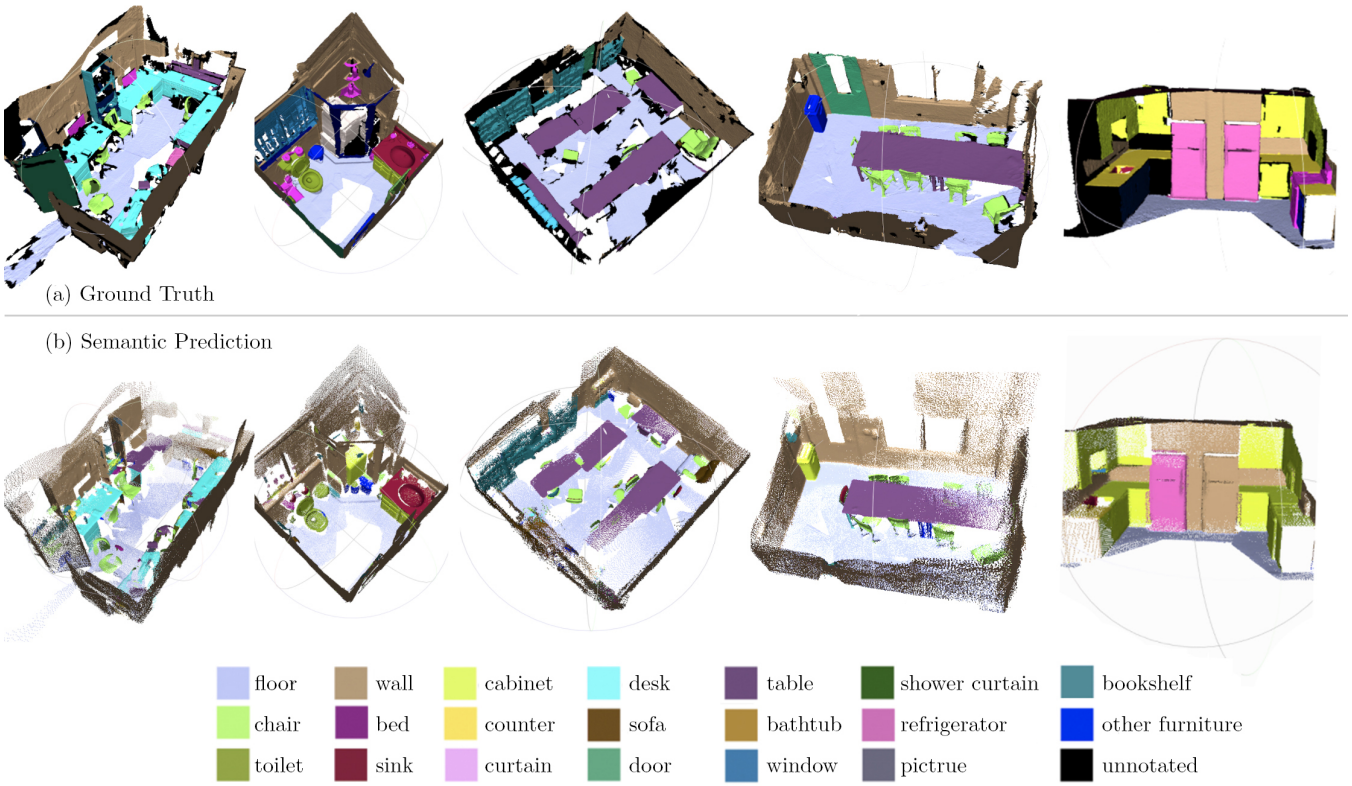

Fig. 5. Example results of the proposed semantic segmentation framework with its underlying segmentation on an example sequence of the ScanNet database. (a) semantically annotated ground truth mesh, (b) semantically annotated 3D point cloud. Please note that the 3D reconstruction of the ground truth annotation was acquired using BundleFusion [34] while our prediction relies on point-based fusion [3].

TABLE V

DENSE PIXEL CLASSIFICATION ACCURACY ON NYUV2 [31]. PLEASE NOTE THAT OUR APPROACH, TOGETHER WITH SCANNET, USE THE GEOMETRY ONLY, IN CONTRAST TO SEMANTICFUSION [14] AND HERMANS ET AL. [5] WHERE COLOR IS ADDITIONALLY UTILIZED.

	Floor	Wall	Chair	Table	Window	Bed	Sofa	TV	Objects	Furniture	Ceiling	Average
Hermans et al.	91.5	71.8	41.9	27.2	46.1	68.4	28.5	**38.4**	8.6	37.1	83.4	49.4
SemanticFusion	92.6	86.0	58.4	34.0	60.5	61.7	47.3	33.9	**59.1**	63.7	43.4	58.2
ScanNet	**99.0**	55.8	**67.6**	**50.9**	**63.1**	**81.4**	**67.2**	35.8	34.6	**65.6**	46.2	**60.7**
Ours	96.3	**94.3**	48.3	33.3	3.6	23.2	21.4	2.9	12.1	23.7	**87.0**	40.6

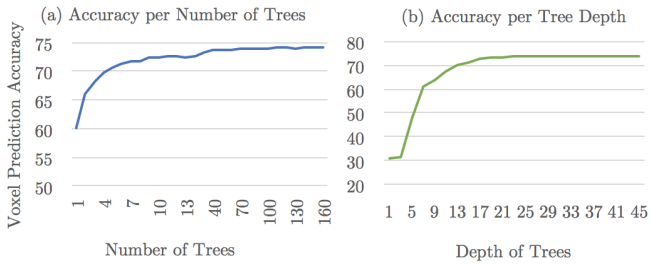

Fig. 6. Voxel Prediction Accuracy per Number of Trees t and with $d = 50$ and Depth of Trees d (with $t = 150$)

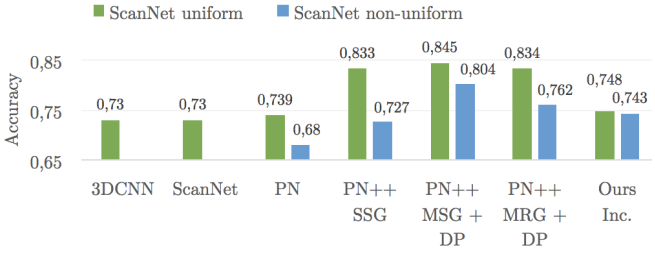

Fig. 7. Accuracy of our methods compared to 3DCNN [35], PointNet [16] and PointNet++ [19] on uniform and non-uniform point clouds.

which could be used for a wide range of applications. Our framework is, due to its efficient nature, suitable to be the basis for more sophisticated robotics and augmented/mixed reality algorithms and applications.

ACKNOWLEDGMENT

This work was funded by the Bavarian State Ministry of Education, Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B), as well as by Google Inc.

TABLE VI

SEMANTIC VOXEL LABEL PREDICTION ACCURACY ON SCANNET TEST SCENES. PLEASE NOTE THAT OUR % OF TEST SCENES DIFFERS SINCE A DIFFERENT RECONSTRUCTION ALGORITHM WAS USED.

Class	% of Test Scenes*	PN	PN++	ScanNet	Ours
Floor	35.7%	-	97.8%	90.3%	95.6%
Wall	38.8%	-	89.5%	70.1%	93.9%
Chair	3.8%	-	86.0%	69.3%	53.0%
Sofa	2.5%	-	68.3%	75.7%	71.6%
Table	3.3%	-	59.6%	68.4%	65.5%
Door	2.2%	-	27.5%	48.9%	7.1%
Cabinet	2.4%	-	39.8%	49.8%	36.8%
Bed	2.0%	-	69.7%	62.4%	50.4%
Desk	1.7%	-	66.7%	36.8%	30.6%
Toilet	0.2%	-	84.8%	69.9%	49.0%
Sink	0.2%	-	62.8%	39.4%	30.0%
Window	0.4%	-	23.7%	20.1%	1.3%
Picture	0.2%	-	0%	3.4%	0.9%
Bookshelf	1.6%	-	84.3%	64.6%	29.2%
Curtain	0.7%	-	48.7%	7.0%	6.1%
Shower Curtain	0.04%	-	85.0%	46.8%	6.9%
Counter	0.6%	-	37.6%	32.1%	19.3%
Refrigerator	0.3%	-	54.7%	66.4%	32.0%
Bathtub	0.2%	-	86.1%	74.3%	34.0%
Other Furniture	2.9%	-	30.7%	19.5%	8.9%
Mean (unweighted)		-	**60.2**%	50.8%	36.1%
Mean (weighted)		73.9	**84.2**%	73.0%	74.8%

REFERENCES

[1] A. J. Davison, "Futuremapping: The computational structure of spatial AI systems," *CoRR*, vol. abs/1803.11288, 2018. [Online]. Available: http://arxiv.org/abs/1803.11288

[2] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Int. Conf. on Robotics and Automation, (ICRA)*, 2014.

[3] 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 *Proc. Int. Conf. on 3D Vision*, 2013, pp. 1–8.

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

[5] A. Hermans, G. Floros, and B. Leibe, "Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images," in *Int. Conf. on Robotics and Automation*, 2014.

[6] J. P. C. Valentin, V. Vineet, M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. H. S. Torr, "Semanticpaint: Interactive 3D labeling and learning at your fingertips," *ACM Trans. Graph.*, 2015.

[7] K. Tateno, F. Tombari, and N. Navab, "When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam," *Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," 2017.

[9] W. Choi, Y. W. Chao, C. Pantofaru, and S. Savarese, "Understanding indoor scenes using 3D geometric phrases," in *CVPR*, 2013.

[10] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison, "Dense planar SLAM," in *Int. Symposium on Mixed and Augmented Reality*, 2014.

[11] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[12] L. Yang, H. Uchiyama, J. Normand, G. Moreau, H. Nagahara, and R. Taniguchi, "Real-time surface of revolution reconstruction on dense slam," in *Int. Conf. on 3D Vision (3DV)*, 2016.

[13] C. Li, H. Xiao, K. Tateno, F. Tombari, N. Navab, and G. D. Hager, "Incremental scene understanding on dense SLAM," in *Int. Conf. on Intelligent Robots and Systems, (IROS)*, 2016.

[14] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks," *CoRR*, 2016.

[15] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," in *Robotics: Science and Systems*, Rome, Italy, 2015, pp. 1697 – 1716.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *CoRR*, 2016.

[17] K. Lai, L. Bo, and D. Fox, "Unsupervised Feature Learning for 3D Scene Labeling," in *Int. Conf. on on Robotics and Automation (ICRA)*, 2014.

[18] S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1511.02300

[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[20] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conf. on Computer Vision (ECCV)*, Berlin, Heidelberg, 2010.

[21] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3D registration," in *Int. Conf. on Robotics and Automation (ICRA)*. Piscataway, NJ, USA: IEEE Press, 2009.

[22] A. E. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *Pattern Anal. Mach. Intell.*, 1999.

[23] M. Ulrich, B. Drost, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[24] R. B. Rusu, G. R. Bradski, R. Thibaux, and J. M. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram." in *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010, pp. 2155–2162.

[25] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. R. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *Int. Conf. on Computer Vision*, 2011.

[26] 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 *Pattern Recognition Proceedings*, 2012.

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

[28] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[29] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys, "3D modeling on the go: Interactive 3D reconstruction of large-scale scenes on mobile devices," in *Int. Conf. on 3D Vision*, 2015.

[30] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: 5M photorealistic images of synthetic indoor trajectories with ground truth," *CoRR*, 2016.

[31] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[32] L. Alexandre, "3D descriptors for object and category recognition: a comparative evaluation," in *IEEE International Conf. on Intelligent Robotic Systems - IROS*, vol. Workshop on Color-Depth Camera Fusion in Robotics, October 2012, pp. 1–6.

[33] J. P. S. do Monte Lima and V. Teichrieb, "An efficient global point cloud descriptor for object recognition and pose estimation," *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2016.

[34] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[35] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.