International Journal of Computer Vision (2020) 128:1220–1238
https://doi.org/10.1007/s11263-019-01187-z

Semi-supervised Semantic Mapping Through Label Propagation with Semantic Texture Meshes

Radu Alexandru Rosu[1] · Jan Quenzel[1] · Sven Behnke[1]

Received: 19 July 2018 / Accepted: 30 May 2019 / Published online: 13 June 2019

Abstract

Scene understanding is an important capability for robots acting in unstructured environments. While most SLAM approaches provide a geometrical representation of the scene, a semantic map is necessary for more complex interactions with the surroundings. Current methods treat the semantic map as part of the geometry which limits scalability and accuracy. We propose to represent the semantic map as a geometrical mesh and a semantic texture coupled at independent resolution. The key idea is that in many environments the geometry can be greatly simplified without loosing fidelity, while semantic information can be stored at a higher resolution, independent of the mesh. We construct a mesh from depth sensors to represent the scene geometry and fuse information into the semantic texture from segmentations of individual RGB views of the scene. Making the semantics persistent in a global mesh enables us to enforce temporal and spatial consistency of the individual view predictions. For this, we propose an efficient method of establishing consensus between individual segmentations by iteratively retraining semantic segmentation with the information stored within the map and using the retrained segmentation to re-fuse the semantics. We demonstrate the accuracy and scalability of our approach by reconstructing semantic maps of scenes from NYUv2 and a scene spanning large buildings.

Keywords Semantic mapping · Label propagation · Semantic textured mesh

1 Introduction

Robots acting in real-world environments need the ability to understand their surroundings, and know their location within the environment. While the problem of geometrical mapping and localization can be solved through SLAM methods (Zollhöfer et al. 2018), many tasks require knowledge about the semantic meaning of objects or surfaces in the environment. The robot should, for instance, be able to recognize where the obstacles are in the scene, and also understand whether those obstacles are cars, pedestrians, walls, or otherwise.

The problem of building maps has been extensively studied (Kostavelis and Gasteratos 2015). Most approaches can be grouped into the following three categories, based on map representation:

– Voxel-based: The scene is discretized into voxels, either using a regular grid, or an adaptive octree. Each voxel stores the binary occupancy value (occupied, empty, unknown) or the distance to the surface commonly referred to as Signed Distance Function (SDF).
– Surfel-based: The map is represented by small surface elements, which store the mean and covariance of a set of 3D-points. Surfels suffer from less discretization errors than voxels.
– Mesh-based: The map is represented as a set of vertices with faces between them. This naturally fills holes and allows for fast rendering using established graphics pipelines.

Communicated by Anelia Angelova, Gustavo Carneiro, Niko Sünderhauf, Jürgen Leitner.

✉ Jan Quenzel
quenzel@ais.uni-bonn.de

Radu Alexandru Rosu
rosu@ais.uni-bonn.de

Sven Behnke
behnke@ais.uni-bonn.de

[1] Autonomous Intelligent Systems Group, University of Bonn, Bonn, Germany

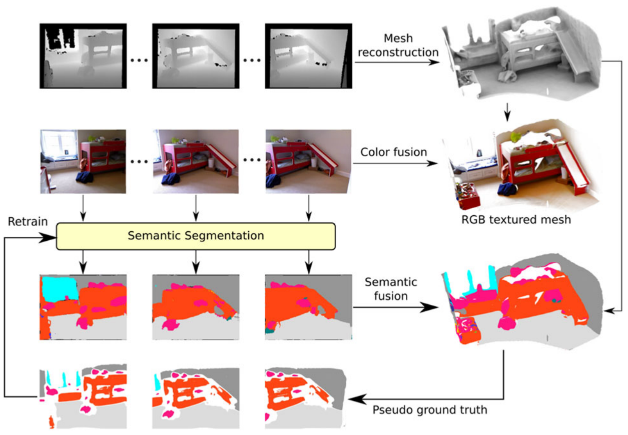

Fig. 1 Semantic Reconstruction: We generate a mesh with RGB texture and semantic annotations. The mesh enables us to ensure temporal and spatial consistency between semantic predictions and allows us to perform label propagation for improved semantic segmentation. Color coding of semantic labels correspond to NYUv2 dataset (Silberman et al. 2012)

Current semantic mapping systems treat the semantic information as part of the geometry, and store label probabilities per map element (voxel, sufel or mesh vertex/face). This approach has the intrinsic disadvantage of coupling the resolution of the geometrical representation to the semantics, requiring a large number of elements to represent small semantic objects or surface parts. This is an undesirable effect as it leads to unnecessary memory usage especially in man-made environments, where the geometry is mostly planar, and high geometrical detail would be redundant. Often, it suffices to represent the semantics relative to a rough geometric shape.

The key idea of our approach, visualized in Fig. 1, is to couple the scene geometry with the semantics at independent resolution by using a semantic texture mesh. In this, the scene geometry is represented by vertices and faces, whereas the semantic texture categorizes the surface with higher resolution. This allows us to represent semantics and geometry at different resolutions in order to build a large semantic map, while still maintaining a low memory usage. As our segmentation module we make use of RefineNet (Lin et al. 2017) to predict a semantic segmentation for each individual RGB view of the scene. These predictions are probabilistically fused onto the semantic texture that is supported by a coarse mesh representing the scene geometry. Having a globally persistent semantic map enables us to establish a temporal and spatial consistency that was previously unobtainable for individual-view predictor. To this end, we propose to propagate labels from the stable mesh by projection onto each camera frame, in order to retrain the semantic segmentation in a semi-supervised manner. Expectation Maximization (EM) is then carried out by alternating between fusing semantic predictions and propagating labels. This iterative refinement allows us to cope with view points which were not common in the training dataset. A predictor pretrained on street-level segmentation will not work well on images captured by a micro aerial vehicle (MAV) at higher altitudes or close to buildings. However, projecting confident semantic labels fused from street level onto less confident parts of views will enable to learn the semantic segmentation of new viewpoints (see Fig. 2).

We compare our method with SemanticFusion (McCormac et al. 2017), and evaluate the accuracy on the NYUv2 dataset (Silberman et al. 2012). We show that the increased resolution of the semantic texture allows for more accurate semantic maps. Finally, propagation and retraining further improve the accuracy, surpassing SemanticFusion in every class.

To showcase the benefits of textured meshes in terms of scalability and speed, we also recorded a dataset spanning multiple buildings, annotated with the 66 classes of the Mapillary dataset (Neuhold et al. 2017). We demonstrate that we are able to construct a large map using both RGB and semantic information in a time- and memory-efficient manner.

2 Related Work

The annotation of large datasets is a costly and time-consuming matter. Hence, the automation of annotation as well as the transfer of knowledge across different domains and datasets are active research topics. Most networks for image segmentation or their respective backbone (e.g. RefineNet (Lin et al. 2017) and Mask R-CNN (He et al. 2017) with ResNet-backbone (He et al. 2016)) are nowadays pretrained on large datasets like ImageNet (Deng et al. 2009) and only finetuned for a specific dataset or purpose.

Vezhnevets et al. (2012) classify super pixels in an automated manner using a pairwise conditional random field and request human intervention based on the *expected change*. Likewise, Jain and Grauman (2016) use a Markov Random Field for joint segmentation across images given region proposals with similar saliency. The resulting proposals are later fused to obtain foreground masks while supervision is requested based on an images influence, diversity and the predicted annotation difficulty. Instead, Yang et al. (2017) cluster unannotated data based on cosine similarity to other images and simply choose per cluster the one with most similar images for human labeling. Mackowiak et al. (2018) take a more cost-centric approach and train one CNN for semantic segmentation and one for a cost model that estimates the necessary clicks for annotating a region. The cost model predictions are then fused with the vote entropy of the segmenting networks activation and supervision is requested for a fixed number of regions. Castrejon et al. (2017) provide with Polygon-RNN a more interactive approach. Given

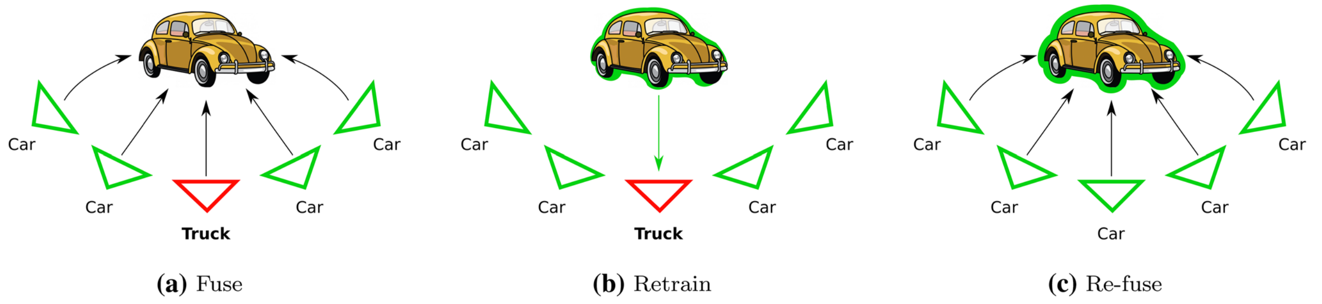

Fig. 2 Label propagation: Semantic segmentations are probabilistically fused in the scene (left). Frames with largest deviation (middle) from the stable mesh are used for retraining. Inference is repeated for all images and re-fused into the scene mesh to achieve a more accurate semantic map

a (drawn) bounding box around an object, the RNN with VGG-16 backbone (Simonyan and Zisserman, 2014) predicts an enclosing polygon around the object. The polygon can be corrected by a human annotator and fed back into the RNN to improve the overall annotation accuracy. Acuna et al. (2018) improve upon Polygon-RNN through architecture modifications, training with reinforcement learning and increased polygonal output resolution.

Most semantic segmentation methods are not real-time capable. Hence, Sheikh et al. (2016) proposed to use quad-tree based super pixels where only the center is classified by a random forest and labels are propagated to a new image if the super pixels location and intensity do not change significantly. This inherently assumes small inter frame motion but does not take spatial correspondences into account, yet runs on a CPU with up to 30 fps.

While image segmentation is fairly advanced, labeling point clouds still has a large potential for improvement. Voxel-based approaches like OctNet (Riegler et al. 2017) precompute a voxel grid and apply 3D- convolutions. Most grid cells are empty for sparse LIDAR point clouds. Hence, recent research shifts towards using points directly (Qi et al. 2017a, b), forming cluster of points (Landrieu and Simonovsky 2017), applying convolutions on local surfaces (Tatarchenko et al. 2018) or lifting points to a high-dimensional sparse lattice (Su et al. 2018). These methods do not enforce consistent labels for sequential data and would need to be recomputed once new data is aggregated while being strongly memory constrained. Nevertheless, Zaganidis et al. (2018) showed that semantic predictions can improve point cloud registration with GICP and NDT.

Semantic reconstruction and mapping received much attention in recent years. Civera et al. (2011), for example, paved the way towards a semantic SLAM system by presenting an object reasoning system, able to learn object models using feature descriptors in an offline step and then recognizing and registering them to the map at run time. However, their system was limited to a small number of objects and apart from the recognized objects, the map was represented only as a sparse point cloud. Bao and Savarese (2011) exploit semantics for Structure-from-Motion (SfM) to reduce the initial number of possible camera configurations and add a semantic term during Maximum-Likelihood estimation of camera poses and scene structure. Subsequently, Bao et al. (2013) use the estimated scene structure to generate dense reconstructions from learned class-specific mean shapes with anchor points. The mean shape is warped with a 3D thin plate spline and local displacements are obtained from actual details of the instance.

Instead, Häne et al. (2013) fuse single frame depth maps from plane sweep stereo to reconstruct a uniform voxel grid and jointly label these voxels by rephrasing the fusion as a multi-label assignment problem. A primal-dual algorithm solves the assignment while penalizing the transition between two classes based on class-specific geometry priors for surface orientation. Their method is also able to reconstruct and label underlying voxels and not only visible ones.

In subsequent work, more elaborate geometry priors have been learned, e.g. using Wolff shapes from surface normal distributions (Häne et al., 2014) and recently end-to-end-learned with a 3D-CNN (Cherabier et al., 2018). The data term in the optimization has been improved (Savinov et al. 2016), memory consumption and runtime reduced (Cherabier et al., 2016; Blaha et al., 2016), and an alignment to shape priors integrated (Maninchedda et al. 2016).

Schönberger et al. (2018) utilize the approach of Häne et al. (2013) for visual localization with semantic assistance to learn descriptors. An encoder-decoder CNN is trained on the auxiliary task of Scene Completion given incomplete subvolumes. The encoder is then used for descriptor estimation. Given a bag of words with a corresponding vocabulary one can thus query matching images for a given input frame.

For incremental reconstruction, Stueckler et al. (2014) presented a densely- represented approach using a voxel grid map. Semantic labels were generated for individual RGB-D views of the modeled scene by a random forest. Labels were then assigned projectively to each occupied voxel and fused using a Bayesian update. The update effectively improved the accuracy of backprojected labels compared to instantaneous segmentation of individual RGB-D views.

Similarly, Hermans et al. (2014) fused semantic information obtained from segmenting RGB-D frames using random forests but represented the map as a point cloud. Their main contribution was an efficient spatial regularizing Conditional Random Field (CRF), which smoothes semantic labels throughout the point cloud. Li and Belaroussi (2016) extended this approach to monocular video while using the semi-dense map of LSD-SLAM (Engel et al. 2014). Here, the DeepLab-CNN (Chen et al. 2018) was used instead of a random forest for segmentation.

Vineet et al. (2015) achieve a virtually unbounded scene reconstruction through the use of an efficient voxel hashed data structure for the map. This further allows them to incrementally reconstruct the scene. Instead of RGB-D cameras, stereo cameras were employed and depth was estimated by stereo disparity. Semantic segmentation was performed through random forest. The requirement for dense depth estimates is lifted in the approach of Kundu et al. (2014). They use only sparse triangulated points obtained through monocular Visual SLAM and recover a dense volumetric map through a CRF that jointly infers semantic category and occupancy for each voxel.

A different approach is used in the keyframe-based monocular SLAM system by Tateno et al. (2017) where a CNN predicts per keyframe the pixel-wise monocular depth and semantic labels.

Lianos et al. (2018) reduce drift in visual odometry via establishing of semantic correspondences over longer periods than possible with pure visual correspondences. The intuition is that the semantic class of a car will stay a car even under diverse illumination and view point changes while visual correspondences may be lost. However, semantic correspondences are not discriminative in the short term.

Tulsiani et al. (2017) perform single view reconstruction of a dense voxel grid with a CNN. During training multiple views of the same scene guide the learning by enforcing the consistency of viewing rays incorporating information from multiple sources like foreground masks, depth, color or semantics. Whereas, Ma et al. (2017) examined the use of warping RGB-D image sequences into a reference frame for semantic segmentation to obtain more consistent predictions. Sun et al. (2018) extend OctoMap (Hornung et al. 2013) with a LSTM per cell to be able to account for long term changes like dynamic obstacles.

Nakajima et al. (2018) segment surfels from the depth image and semantic prediction using connected component analysis and further refined incrementally over time. Geometric segments along with their semantic label are stored in the 3D map. The probabilistic fusion combines the rendered current view with the current frame and its low resolutional semantics.

Surfels are also used in the work of McCormac et al. (2017). The authors integrated semantics into ElasticFusion (Whelan et al. 2015) which represents the environment as a dense surfel map. ElasticFusion is able to reconstruct the environment in real-time on a GPU given RGB-D images and can handle local as well as global loop closure. Semantic information is stored on a per-surfel basis. Inference is done by an RGB-D-CNN before fusing estimates probabilistically. SemanticFusion fuses for each visible surfel and all possible classes which is very time- and memory-consuming since the class probabilities need to be stored per surfel and class on the GPU. Objects normally consist of a large number of surfels and share in reality a single class label even though semantic information within a surfel would only be required at the border of the object where the class is likely to change. Hence, many surfels store the same redundant information and since GPU memory is notoriously limited, memory usage becomes a problem for larger surfel maps. Furthermore, SemanticFusion tends to create many unnecessary surfels with differing scales and labels for the same surface when sensed from different distances.

Closely related to our approach is the work of Valentin et al. (2013). Their map is represented as a triangular mesh. They aggregate depth images in a Truncated Signed Distance Function (TSDF) and obtain the explicit mesh representation via the marching cubes algorithm. Afterwards, semantic inference is performed for each triangle independently using a learned classifier on an aggregation of photometric (color dependent) and handcrafted local geometric (mesh related) features. Spatial regularization is ensured through a CRF over the mesh faces. Their classifier infers the label with all visible pixels per face at once and is not designed to incrementally fuse new information. Furthermore, the pairwise potential of the CRF does not take the likelihood for other classes in to account. Especially around object borders this may lead to suboptimal results. The semantic resolution is tied to the geometry of the mesh, hence to have fine details the mesh resolution needs to be fine grained. Geometrically a wall can be described with a small number of vertices and faces, but to semantically distinguish between an attached poster and the wall itself the mesh would need a high resolution.

In comparison, we only store the likelihood for a small number of most probable classes and the meshing creates a single simplified surface while the texture resolution can be chosen independent of the geometry yet appropriate to the scene. During fusion we further include weighting to account for the sensor distance and in the case of color integration include vignetting and viewing angle.

3 Overview

In this paper, we present a novel approach to building semantic maps by decoupling the geometry of the environment from its semantics by using semantic textured meshes. This

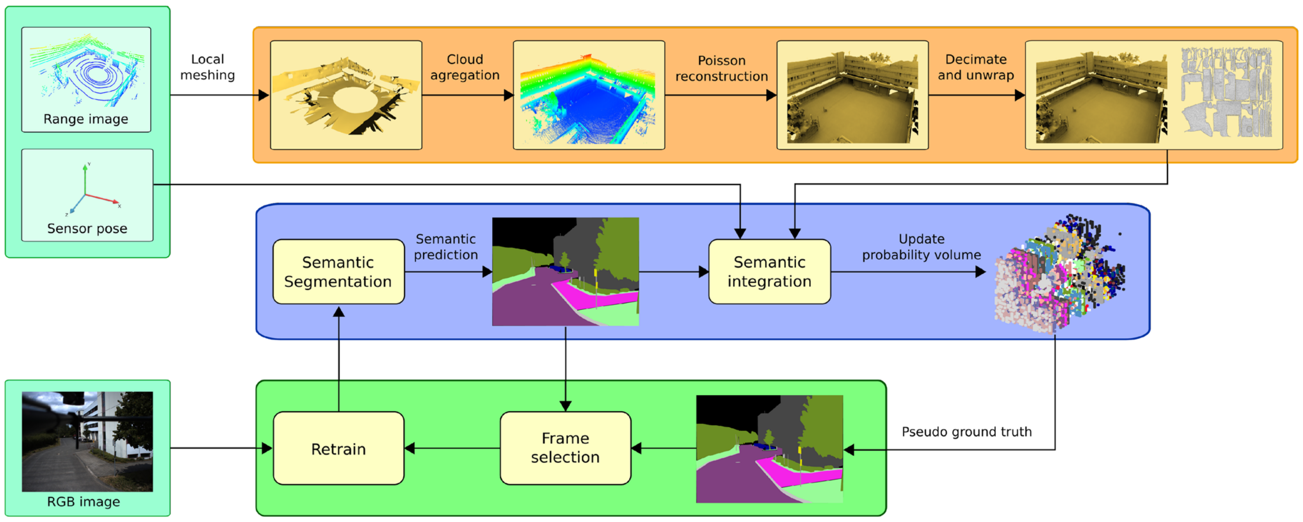

Fig. 3 System overview: Individual range images are used to create local meshes for fast normal estimation and cloud simplification. The resulting points, equipped with normals, are aggregated into a global point cloud. Poisson reconstruction is employed to extract a mesh, which is simplified and unwrapped to obtain a lightweight scene representation. In the next step, we perform semantic integration. Individual RGB views are segmented using RefineNet (Lin et al. 2017). The semantic labels are then probabilistically fused into a semantic texture. Finally, label propagation is performed by inferring pseudo ground truth views, and using them to retrain the predictor. The retrained semantic segmentation is used to re-fuse the labels into the mesh yielding a more accurate semantic map

decoupling allows us to store the geometry of the scene as a lightweight mesh which efficiently represents even city-sized environments.

Our method (see Fig. 3) operates in three steps: mesh generation, semantic texturing and label propagation.

In the *mesh generation* step, we create a mesh of the environment by aggregating the individual point clouds recorded by a laser scanner or an RGB-D camera. We assume that the scans are preregistered into a common reference frame using any off-the-shelf SLAM system. We calculate the normals for the points in each scan by estimating an edge-maintaining local mesh for the scan. Once the full point cloud equipped with normals is aggregated, we extract a mesh using Poisson reconstruction (Kazhdan and Hoppe 2013) and further simplify it using QSlim (Garland and Heckbert 1998). Our main contribution for 3D reconstruction is the proposal of a system capable of fast normal estimation by using a local mesh and also local line simplification which heavily reduces the number of points, therefore reducing the time and memory used by Poisson reconstruction.

In the *semantic texturing* step, we first prepare the mesh for texturing by parameterizing it into a 2D plane. Seams and cuts are added to the mesh in order to deform it into a planar domain. A semantic texture is created in which the number of channels corresponds to the number of semantic classes. The semantic segmentation of each individual RGB frame is inferred by RefineNet and fused probabilistically into the semantic texture. We ensure bounded memory usage on the GPU by dynamically allocating and deallocating parts of the semantic texture as needed. Additionally, the RGB information is fused in an RGB texture.

In the *Label Propagation* step, we project the stable semantics, stored in the textured mesh, back into the camera frames and retrain the predictor in a semi-supervised manner using high confidence fused labels as ground truth, allowing the segmentation to learn from novel view points.

Hence, the contribution presented in this article is fourfold:

– a scalable system for building accurate meshes from range measurements with coupled geometry and semantics at independent resolution,
– an edge-maintaining local mesh generation from lidar scans,
– a label propagation that ensures temporal and spatial consistency of the semantic predictions, which helps the semantic segmentation to learn and perform segmentation from novel view points,
– fast integration of probability maps by leveraging the GPU with bounded memory usage.

4 Notation

In the following, we will denote matrices by bold uppercase letters and (column-)vectors with bold lowercase letters. The rigid transformation $\mathbf{T}_{F_2 F_1}$ is represented as 4×4 matrix and maps points from coordinate frame F_1 to coordinate frame

F_2 by operating on homogeneous coordinates. When necessary, the frame in which a point is expressed is added as a subscript: e.g. $\mathbf{p}_w$ for points in world coordinates. A point $\mathbf{p}_w$ is projected into frame F with the pose $\mathbf{T}_F$ and the camera matrix $\mathbf{K}_F \in \mathbb{R}^{3\times 3}$. For the camera matrix, we assume a standard pinhole model with focal length f_x, f_y, and principal point c_x, c_y. The projection of $\mathbf{p}_w$ into image coordinates $\mathbf{u} = (u_x, u_y)^{\intercal}_F \in \mathbf{\Omega} \subset \mathbb{R}^2$ is given by the following mapping:

$$g_F(\mathbf{p}_w) : \mathbf{p}_w \to \mathbf{p}_F, \tag{1}$$

$$(\mathbf{p}_F, 1)^{\intercal} = \mathbf{T}_{F_w} \cdot (\mathbf{p}_w, 1)^{\intercal}, \tag{2}$$

$$\pi_F(\mathbf{p}_F) : \mathbf{p}_F \to \mathbf{u}_F, \tag{3}$$

$$(x, y, z)^{\intercal}_F = \mathbf{K}_F \cdot \mathbf{p}_F, \tag{4}$$

$$\mathbf{u}_F = (x/z, y/z)^{\intercal}. \tag{5}$$

An image or a texture is denoted by $I(\mathbf{u}) : \mathbf{\Omega} \to \mathbb{R}^n$, where $\mathbf{\Omega} \subset \mathbb{R}^2$ maps from pixel coordinates $\mathbf{u} = (u_x, u_y)^{\intercal}$ to n-channel values.

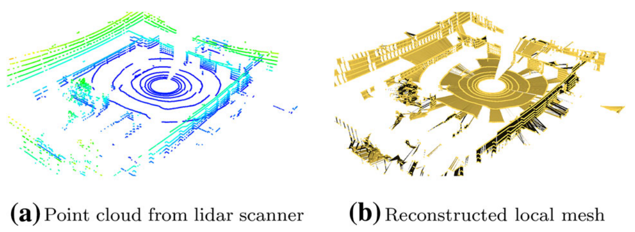
(a) Point cloud from lidar scanner
(b) Reconstructed local mesh

Fig. 4 Local mesh: We reconstruct an approximate local mesh from the given range measurements in order to estimate point normals needed for Poisson reconstruction

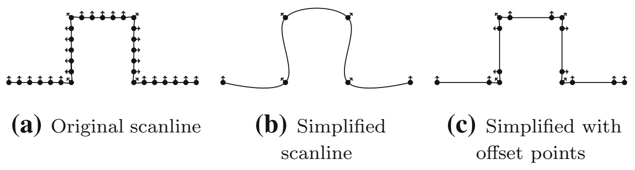

Fig. 5 Line simplification: The original scan line (left) is excessively dense in planar areas. The original simplification greatly reduces the number of points but creating a global surface using a method like Poisson reconstruction overly smoothes the edges (middle). Our extension simplifies the line and preserves hard edges by adding further constraints which allow Poisson reconstruction to maintain sharp features (right)

5 Method

The input to our system is a sequence of organized point clouds $\{\mathcal{P}^t\}$,[1] and RGB images $\{I^t\}$ (t indicates the time step). We assume that the point clouds are already registered into a common reference frame, and the extrinsic calibration $\mathbf{T}_{cd}$ from depth sensor to camera, as well as camera matrices, are given. The depth sensor can be an RGB-D camera or a laser scanner. The output of our system is threefold:

- a triangular mesh of the scene geometry, defined as a tuple $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ of vertices $\mathcal{V}$ and faces $\mathcal{F}$. Each vertex $\in (\mathbb{R}^3 \times \mathbb{R}^2)$ contains a 3D point and a UV texture coordinate, while the mesh face is represented by the indices $\in \mathbb{N}^3$ of the three spanning vertices within $\mathcal{V}$.
- a semantic texture S indicating the texels class probabilities,
- an RGB texture C representing the surface appearance.

After describing the necessary depth preprocessing in Sect. 5.1, we will explain in detail the mesh generation and parametrization (Sect. 5.2), before elaborating on the semantic (Sect. 5.3) and color integration (Sect. 5.4), sparse representation (Sect. 5.5) and label propagation (Sect. 5.6).

5.1 Depth Preprocessing

As previously mentioned, our system constructs a global mesh from the aggregation of a series of point clouds $\{\mathcal{P}^t\}$ recorded from a depth sensor. Many surface reconstruction algorithms require accurate per-point normal. One way to obtain these normals is by aggregating the full global point cloud, and using the k-nearest-neighbors to estimate the normals for each point. However, this would be prohibitively slow as it requires a spatial subdivision structure, like a Kd-tree, which can easily grow to a considerable size for large point clouds, limiting the scalability of the system. For fast normal estimation we take advantage of the structure of the recorded point cloud. Since depth from an RGB-D sensor is typically structured as an image, we can easily query adjacent neighboring points. Similarly, rotating lidar sensors can produce organized scans. A complete revolution of e.g. a Velodyne VLP-16 produces a 2D array of size $16 \times N$ containing the measured range of each recorded point, where N is determined by the speed of revolution of the laser scanner.

Given the organized structure, we could create a triangular local mesh with approximate normals as introduced by Holz and Behnke (2015). However, this would imply using all aggregated points despite the fact that a significant number of them are redundant, since they lay on a common plane. Hence, we propose a method for fast normal estimation and point cloud reduction which helps to reconstruct the mesh in a fast and memory efficient manner (Fig. 4). We first simplify each individual scan to obtain a reduced point cloud without sacrificing geometrical fidelity. For that, we start with the line simplification algorithm of Ramer-Douglas-Peucker (Douglas and Peucker 1973) which is applied on each scan ring. To maintain hard edges of the point cloud, we extend Ramer-Douglas-Peucker to create offset points

[1] An organized point cloud exhibits an image resembling structure, e.g. from commodity RGB-D sensors.

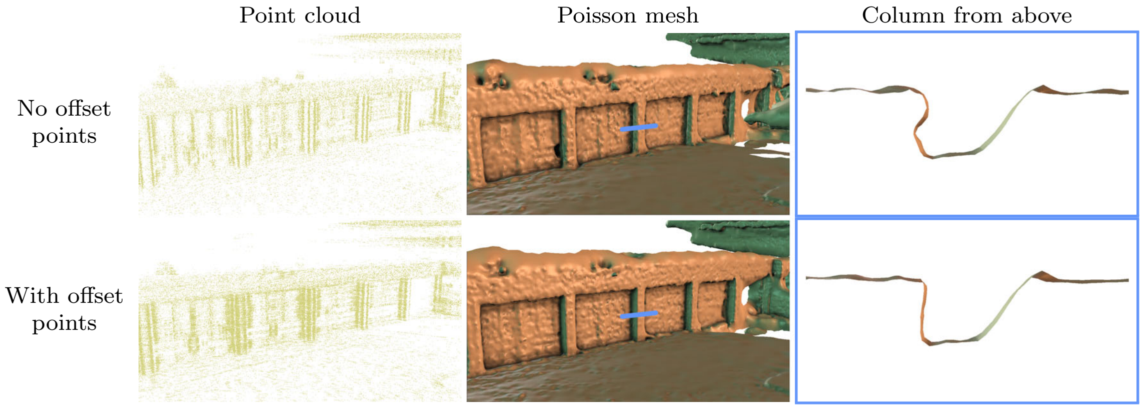

Fig. 6 Offset points: The impact of adding offset points during line simplification is evaluated on the courtyard dataset. The absence of offset points leads to a Poisson reconstruction that is overly smooth (first row). Adding offset points increases the density of the point cloud around the edges of the columns and results in a sharper reconstruction (second row). The smooth right hand side of the columns is due to undersampling from occlusion

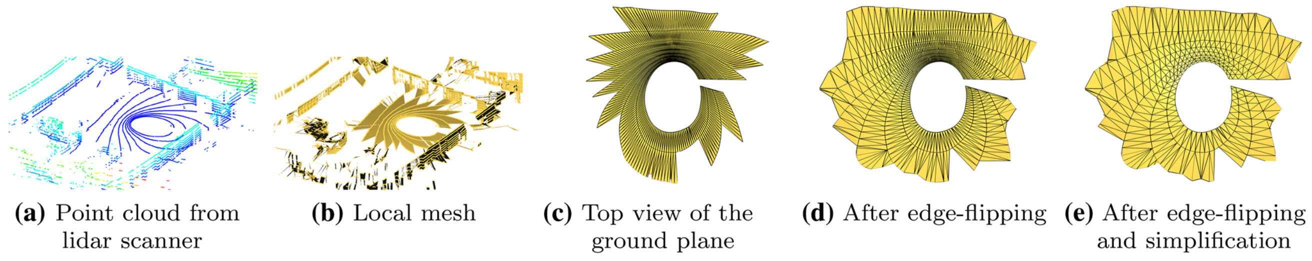
(a) Point cloud from lidar scanner
(b) Local mesh
(c) Top view of the ground plane
(d) After edge-flipping
(e) After edge-flipping and simplification

Fig. 7 Local mesh connections problem: During sudden movements of the laser scanner, the scan rings are compressed behind and expanded in front of the sensor. This creates many small and steep triangles which degrades normal estimation. We perform iterative edge-flipping in order to connect each vertex with their closest neighboring vertex, hence, improving the likelihood for estimating correct normals. Furthermore, we apply line simplification to each scan ring independently for data reduction without sacrificing mesh fidelity

around the simplified edges to add a further constraint on the normals of the points, and allow the subsequent mesh generator to recover sharp features as visualized in Figs. 5 and 6. The local mesh is created by unwrapping the scan in 2D using polar coordinates, and performing a constrained 2D Delaunay triangulation. Constrained edges are set to the ones obtained from the line simplification. This ensures that points that lie on the same scan ring will be connected together by triangles. After recovering a local mesh from the point cloud, normals are first estimated per face by calculating the cross product between two of the edges, and finally per vertex using a *Mean Weighted by Angle* (MWA) scheme (Thürrner and Wüthrich 1998) which weighs each triangle's contribution by the angle under which it is incident to the vertex:

$$\begin{aligned}
\mathbf{n}_f &= \frac{\mathbf{e}_{1,2} \times \mathbf{e}_{3,2}}{\left\|\mathbf{e}_{1,2} \times \mathbf{e}_{3,2}\right\|}, \\
\mathbf{n}_v &= \sum_{f \in AdyF(v)}^{n} \alpha_f \cdot \mathbf{n}_f, \\
\mathbf{n}_v &= \mathbf{n}_v / \left\|\mathbf{n}_v\right\|.
\end{aligned} \tag{6}$$

where $\mathbf{n}_f$ and $\mathbf{n}_v$ denote the face and vertex normals, respectively, and α_f is the angle between the two edge vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ that share the vertex. The angle between the two edge vectors can be computed using the dot product between them: $\alpha_f = \arccos\left(\frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\|\|\mathbf{e}_2\|}\right)$.

Until now, we have obtained fast normals using only the points contained in one revolution of the Velodyne. However, due to the anisotropy in the sampling of a laser scanner, the connections between points created by the Delaunay triangulation in 2D may not be optimal when lifted to 3D, as seen in Fig. 7. Since the connections between points are crucial for an accurate normal estimation, an iterative local mesh refinement is performed. The refinement ensures that each point will be connected to the neighbors that lie spatially close in 3D. Again, in order to avoid cumbersome and slow spatial subdivision structures we introduce an edge flipping algorithm which iteratively *flips* the edge shared between two triangles to increase the following quality measure:

$$q = \frac{4a\sqrt{3}}{h_1^2 + h_2^2 + h_3^2}, \tag{7}$$

where a denotes the area of the triangle and h is the length of the edge. We chose it such that it is monotonically increasing, and promotes more equilateral triangles. We perform edge flipping in a greedy fashion by choosing first the triangle which will experience the most quality increase. After performing the flip for a triangle, the quality of adjacent triangles may change and is updated. We continue flipping edges until the quality measure can no longer be increased.

5.2 Mesh Generation

After aggregation of the points with corresponding normals from all simplified scans, we perform Poisson reconstruction to recover a high quality mesh, despite having potentially noisy data and missing measurements. The resulting mesh can still be overly dense in areas which are geometrically simple, like the ground. Hence, we apply a second global simplification step following the QSlim method (Garland and Heckbert 1998). This approach iteratively collapses edges until a certain error threshold, or a predetermined number of faces, is reached.

The last step in the creation of the global mesh is to prepare it for texturing by parameterizing the mesh in the 2D domain in order to obtain UV coordinates for the vertices. For that, we make use of the *UV smart project* function provided within Blender.[2]

5.3 Semantic Integration

In this section we detail our approach on how to update the global semantic texture S using individual color images I^t. For semantic segmentation, we retrain RefineNet on the Mapillary dataset (Neuhold et al. 2017) for street level segmentation. The dataset contains 25,000 images densely labeled with 66 classes. Given the input image I_k, the output of the predictor can be interpreted as a per-pixel probability over all the class labels $P(O_\mathbf{u} = l_i|I_k)$, with $\mathbf{u}$ denoting pixel coordinates. One common approach to integrate semantic information is to perform a Bayesian update over the classes probability, fusing new observations into the global belief for the semantic labels. However, this scheme of updating becomes slow for a large number of classes since the belief for all labels needs to updated.

In our approach we choose to approximate the probability over the classes with only the *argmax* probability. Hence, a new observation will consist of only the *argmax* label and its probability instead of the full distribution. This enables us to use a fast integration scheme whose runtime is independent of the number of classes. We observe that in practice this approximation works well.

[2] https://www.blender.org.

We define the best class L and its corresponding probability P^* using:

$$\begin{aligned} L &= \underset{c}{\operatorname{argmax}}\, P(O_u = l_i|I_k), \\ P^* &= \max_c P(O_u = l_i|I_k). \end{aligned} \tag{8}$$

We perform a visibility check prior to updating the global semantic texture using individual segmentation results. Inspired by *shadow mapping* techniques in computer graphics, we first render a depth map D from the current camera view. In order to ensure that every texel is checked for visibility, we obtain for each texel x with UV coordinates $\mathbf{u}_x$ the 3D point $\mathbf{p}_x$ from the vertices of the corresponding mesh face by barycentric interpolation. The point $\mathbf{p}_x$ is then projected into the current view, and discarded if the depth d_x is larger than the stored value within the depth map $D(\pi(g_F(\mathbf{p}_x)))$, as it lies behind the visible part of the mesh. To indicate visibility, we use a per texel indicator variable $r_x \in \{0, 1\}$:

$$r_{x_i} = \begin{cases} 1, & \text{if } d_x \geq D(\pi(g_F(x_w))) \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

All remaining texels ($r_x > 0$) are fused with the current segmentation result by increasing the probability of the obtained classes:

$$S(\mathbf{u}_x, l)^t = S(\mathbf{u}_x)^{t-1} + r_{x_i} \cdot w_{x_i} \cdot p_{x_i}, \tag{10}$$

$$W(\mathbf{u}_x)^t = W(\mathbf{u}_x)^{t-1} + r_{x_i} \cdot w_{x_i}, \tag{11}$$

$$l_{x_i} = L(\pi(g_F(\mathbf{p}_x))), \tag{12}$$

$$p_{x_i} = P^*(\pi(g_F(\mathbf{p}_x))). \tag{13}$$

Additionally, we weigh the fused probability by the faces distance from the camera under the assumption that pixels are more difficult to recognize from farther away, due to the low resolution of semantic segmentation.

$$w_{x_i} = \begin{cases} 1, & \text{if } d_x \leq d_{min} \\ 1 - \frac{d_x - d_{min}}{d_{max} - d_{min}}, & \text{otherwise} \end{cases}, \tag{14}$$

where d_x denotes the depth of the current texel, and d_{min} and d_{max} are thresholds for the distance which define a linear fall-off for the weight. In our experiments we set them to 30 m and 100 m, respectively.

5.4 Color Integration

In addition to the semantics, we also fuse the raw images into a global color texture. The fusion is carried out by a weighted running average:

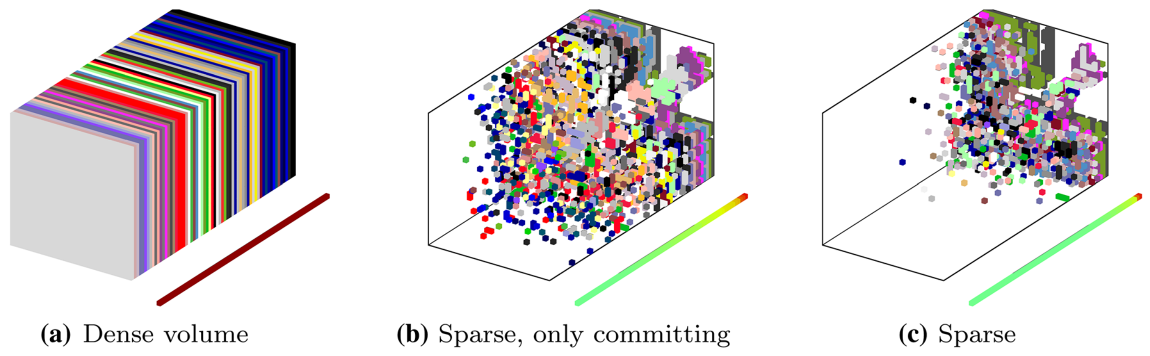

Fig. 8 Semantic probability volume for the courtyard dataset: The memory consumption of the dense 3D semantic texture (left) is prohibitive for most modern GPUs. Committing memory pages with non-negligible probabilities results in a sparse volume (middle) with only 12.29% allocated. Periodically removing low probability pages (right) further reduces the necessary memory (4.25%) and ensures bounded memory usage that easily fits into GPU memory

$$\begin{aligned} C(\mathbf{u}_x)^t &= \frac{W(\mathbf{u}_x)^{t-1} C(\mathbf{u}_x)^{t-1} + w_x I(\pi(g_F(\mathbf{p}_x)))}{W(\mathbf{u}_x)^{t-1} + w_x}, \\ W(\mathbf{u}_x)^t &= W(\mathbf{u}_x)^{t-1} + w_x. \end{aligned} \tag{15}$$

The weight w_x takes the distance, the radial intensity fall-off within an image, and the viewing angle into account:

$$\begin{aligned} w_x &= w_{dist} \cdot w_{vign} \cdot w_{view}, \\ w_{dist} &= (\|g_F(\mathbf{p}_{w,x})\|_2^2)^{-1}, \\ w_{vign} &= \cos(\theta_x)^4, \\ w_{view} &= (\mathbf{o}_w - \mathbf{p}_{w,x}) \cdot \mathbf{n}_x. \end{aligned} \tag{16}$$

Here, w_{dist} is the inverse distance from the texel to the camera, which promotes frames that are spatially closer to the mesh, improving the resolution of the fused colors. The viewing angle θ_x between reprojection of the texel and the principal axis of the camera is used to account for the radial decrease in intensity by following the $\cos^4$ law (Goldman and Chen 2005). The third term w_{view} increases the weight for texels that are imaged by the camera originating at $\mathbf{o}_w$ from a frontal perspective, further improving the quality of the fused texture (Fig. 8).

We chose different update schemes for color and semantics as their behavior is radically different. Firstly, the semantic segmentation is trained to be robust to illumination changes, hence the vignetting term is unnecessary. Secondly, it is not clear that the angle to the surface is a good indicator for confidence in semantic segmentation. In our experiments, we observed that the predictor learns to some extent the relative angle of surfaces with respect to camera view, thus weighting based on relative angle may be adversarial. For example during sudden camera movements tilting toward the ground, the semantic segmentation decreases in accuracy as the view is unfamiliar to the predictor. Thirdly, the distance weight assumes that the accuracy of the semantics is more confident for closer surfaces. However, this is not accurate for large semantic entities like buildings for which the accuracy decreases as we go closer, due to large untextured areas, and increases as we take a step back and observe the bigger picture.

5.5 Sparse Semantic Volume

The semantic 3D texture S contains for each texel the probability distribution over all the classes. However, for reasonable sized resolutions and number of classes, this volume can occupy more memory than is typically available in modern GPUs, rendering this process infeasible. For a texture with 8.192×8.192 pixels and the 66 classes of the Mapillary dataset, we would need to allocate a volume of 16.5 GB (assuming we store each element as a floating point number of 4 bytes). This problem will only become worse as we add more class labels or increase texture size. In order to overcome this issue, we propose to store the semantics into a sparse 3D texture in which we allocate and deallocate dynamically the memory, ensuring bounded memory usage.

In the first step, we divide our global semantic volume into pages of size $128 \times 128 \times 1$. Each page[3] stores the probability for only one class and can be either allocated in GPU memory or not. The volume starts initially with all pages in a deallocated state. Hence, it occupies no space on the GPU.

When fusing the semantic probability from the current frame into S, the corresponding pages that will be affected are computed and targeted for committing on the CPU. This ensures that we only add the parts that are actually relevant. After each frame, we also check for pages that have low probability and deallocate them

[3] Page size was chosen based on common supported values for multiple computers used during development.

from memory. The probability for a texel is computed as:

$$p_{x_l} = S(\mathbf{u}_x, l)/W(u_x). \tag{17}$$

If any texel inside a page is above a certain threshold, we will keep the corresponding page in memory, otherwise we target it for decommitting. This scheme of committing and decommitting portions of the memory can be seen as intrinsically *tracking* the modes of the distribution over the classes, ignoring parts with negligible probability.

Other schemes for memory reduction can be employed like quantization of the semantic 3D texture in which the stored probability is represented with a lower bit depth (reducing it to 2 bytes or even a mere 1 byte). However this approach is not scalable to bigger datasets with more classes and is prone to rounding errors in the case of very low bit depth. For this reason we deem that taking advantage of the sparsity in the semantic volume is a more appropriate method to deal with the memory issue.

5.6 Label Propagation

The fusion of semantic information from various view points into a global representation opens up possibilities to enforce temporal and spatial consistency of the semantic predictions by propagating the labels. The key insight is that if the majority of observations of the texture element predicts the correct class, the fused information S^* will be confident enough ($p_{x_l} \geq p_{min}$) in order to be used as ground truth. Hence, we can reproject the mesh into any camera frame, propagate the label $\widetilde{L}(\mathbf{u}_F)$ and retrain the semantic segmentation in a semi-supervised fashion to minimize discrepancy between image predictions L and mesh label S^*. Reprojecting the semantics into a camera frame is performed as follows:

$$\begin{aligned} \mathbf{u}_F &= \pi(g_F(\mathbf{p}_x)), \\ S^* &= \operatorname*{argmax}_c S, \\ S^*_F(\mathbf{u}_F) &= S^*(\mathbf{u}_x), \\ \widetilde{L}(\mathbf{u}_F) &= \begin{cases} S^*_F(\mathbf{u}_F), & p_{x_l} \geq p_{min} \\ Unlabeled, & \text{otherwise} \end{cases}. \end{aligned} \tag{18}$$

The result consists of the image $\widetilde{L}(\mathbf{u}_F)$ which we denote as *pseudo ground truth*. This semantic labeling, together with the corresponding RGB view I will be used to retrain the predictor.

The retraining ensures that the semantic segmentation will segment objects and surfaces more consistent as belonging to a certain class, regardless of different or extreme view points or even illumination changes. Furthermore, the label propagation and retraining stages can be applied iteratively in an Expectation Maximization scheme, e.g. for a certain number of iterations, or until all camera frames reach a consensus. This process is illustrated in Fig. 2. The obvious caveat are wrong predictions used for retraining since bootstrapping this has a self-reinforcing character. We have seen this behavior only in some rare cases where the initial single-frame predictions were already incorrect, e.g. the table in the bottom row of Fig. 11.

Due to the significant number of camera frames contained in a large-scale dataset, we perform a conservative frame selection in order to choose only a subset of frames on which to reproject the semantics for retraining. The frame selection is based on an inconsistency coefficient, which rates frames higher the more instantaneous semantic segmentations deviates from mesh semantics.

The inconsistency coefficient γ is calculated as:

$$\begin{aligned} \gamma &= \sum_i B(L_i, S^*_F(\mathbf{u}_F))P^*, \\ B(L_i, S^*_F(\mathbf{u}_F)) &= \begin{cases} 1, & L_i \neq S^*_F(\mathbf{u}_F) \\ 0, & L_i = S^*_F(\mathbf{u}_F) \end{cases} \end{aligned} \tag{19}$$

Given this coefficient for each frame, we select a restricted percentage of frames with the highest coefficient (in our experiments we choose 5% of the frames for NYU and 15% for the courtyard dataset) to be used for retraining. The intuition is that there is more information to gain by retraining on inconsistent frames as opposed to the ones which are already correct. Moreover, we also restrict the selected views to be at least 10 frames apart from one another to avoid adding redundant views that are too similar. In order to prevent forgetting during retraining, we add the original training set (in our case all the images from the NYU or Mapillary dataset, respectively) to the new views.

Additionally, we experiment with another type of view selection in which we select for retraining the ones with the lowest rather than highest inconsistency coefficient. The intuition behind this alternative scheme is that by reinforcing good frames (or the ones that are close to being good), the other frames that are "close" to them will also be improved. Therefore, performing the label propagation iteratively will eventually "lift" the less consistent frames to become better by reinforcing the ones that are already good or close to being good. An illustration of this process is depicted in Fig. 9. We ignore for this selection type the frames which have a too high consistency level as we saw that in the general case they provide too little new information. We rather focus on frames that are quite consistent but not fully. Hence, frames with more than 98% of consistent pixels are ignored. From the remaining set we select the same percentages as mentioned earlier.

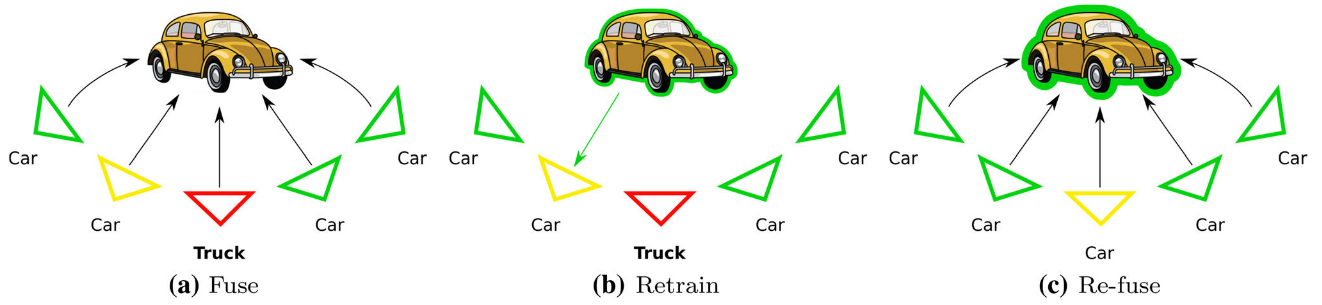

Fig. 9 Label propagation with best frames: Semantic segmentations are probabilistically fused in the scene (left). Frames with *low* (but not too low) deviation from the stable mesh are used for retraining (middle). Close frames are naturally improved thereby. Inference is repeated for all images and the semantic labels are re-fused into the scene mesh to achieve a more accurate semantic map (right)

6 Implementation

Our pipeline consists of three modules, the mesh generator, the semantic texture integrator and the segmentation retraining. The mesh generator module is fully implemented in C++ and integrated into ROS (Quigley et al. 2009) for ease of interaction with other ROS packages. The texturer was also developed in C++ with the addition of OpenGL for rendering and semantic integration using GLSL compute shaders.

We will describe in detail the optimization choices made for semantic integration as they are the main focus of this work. The segmentation map is initially precalculated from the predictor and stored to disk. An asynchronous module reads the RGB images and the corresponding segmentation maps and stores them in ringbuffers ready to be processed by the texturing module. The texturing module receives the images and transfers them to the GPU using double buffered Pixel Buffer Objects (PBO) in a method commonly known as *ping ponging*. This ensures that the transfer can be done on the GPU side using Direct Memory Access (DMA), freeing the CPU to do other tasks in the meantime. The semantic integration is performed fully on the GPU through efficient compute shaders.

In order to deal with the sparsity of our semantic 3D texture we make use of the *GL_ARB_sparse_texture* extension from OpenGL which provides functionality for committing and decommitting from sparse (partially resident) textures. However, the committing and decommitting of pages can only be performed from the CPU side, which requires synchronization and communication between CPU and GPU. We use two buffers for this communication, one for signaling to the CPU which pages require committing and one for confirmation to the GPU that they were committed. These buffers are updated asynchronously and are also double buffered to prevent stalling the pipeline.

While double buffering allows maximum usage of the available resources, it also implies a delay of one frame between the CPU-GPU communication which in our case does not pose a problem due to the high frame rate at which the camera images arrive.

7 Experiments

All tests were performed on a Intel Core i7-940 2.93 GHz CPU with an NVIDIA Titan GPU.

7.1 NYUv2 Dataset

For comparison against SemanticFusion (McCormac et al. 2017) we utilize the NYUv2 dataset (Silberman et al. 2012) and use 108 out of all sequences where ElasticFusion did not exhibit significant drift. For fairness comparability, we use the final surfel map for meshing and the segmentation module of Eigen and Fergus (2015) used within SemanticFusion that is trained on the 13 NYU classes. Furthermore, we store the semantics for Intersection-over-Union (IoU) calculation after the scene is completed and fused semantics are static.

The implementation of SemanticFusion provides two CNNs pretrained on the NYUv2 dataset, one that receives RGB only and another that receives RGB-D data from the Kinect. While the original work of SemanticFusion evaluates their approach using the RGB-D-CNN (including scene depth as an additional feature map) we use for our evaluation their RGB-CNN since we want our method to work well even in outdoor scenarios where dense depth may not be obtainable.

The network is retrained using both methods for view selection (worse or best frames) with a learning rate of 10^{-6}. The model is saved after each epoch and training is stopped when the model begins to overfit or the IoU for the epoch starts to decrease.

For retraining with the worse frames we add from each NYU scene 5% of the frames with the highest inconsistency coefficient. This percentage is chosen such that the amount of pseudo ground truth frames is comparable to the 795 originally used for training. The retraining then uses both the original training set and our pseudo ground truth. In the case of retraining with the best frames we choose the 5% frames with the *lowest* inconsistency coefficient, ignoring however those that have too few inconsistent pixels (in our case we choose those that have at least 2% of the pixels labeled as inconsistent).

7.2 Courtyard Dataset

The courtyard dataset (Droeschel and Behnke 2018) was captured using a DJI Matrice 600, with a horizontally attached Velodyne VLP-16 laser scanner. The lidar has 16 horizontal scan lines and a vertical field-of-view of 30° with a maximum range of 100 m. The color images were captured at 10 Hz using two synchronized global shutter Point Grey Blackfly-S U3-51S5C-C color cameras, with a resolution of 2448×2048 pixels. The MAV poses for this dataset were provided by Droeschel and Behnke (2018). The camera poses were calculated from the provided extrinsics and the continuous-time trajectory under consideration of an additional 40 ms time offset. The two cameras are mounted outward pointing on the left and right side of the copter to improve visual coverage. In total 13,458 frames were captured during the experiment.

We densely annotated 48 images spread throughout the area to conduct accuracy experiments.

RefineNet with ResNeXt 101 (Xie et al. 2017) was trained on the Mapillary dataset for 10 epochs with a learning rate of 10^{-5} and a batch size of 1. The final IoU achieved is 0.4546 placing the result on 3rd position on the Mapillary leaderbord.[4] The retraining with pseudo ground truth has to be done with a larger batch size of 16 to account for the decreased signal-to-noise-ratio introduced by the pseudo ground truth. We performed five epochs of further retraining. The retrained RefineNet achieves 0.4487 IoU on the Mapillary dataset. The slight reduction in accuracy is explained by the fact that the neural network is forced to learn from a new dataset which is different from the Mapillary one in which it is evaluated.

7.3 Accuracy Evaluation

We execute the two variants of Label Propagation for three iterations on the NYUv2 dataset. We observe that retraining on the worse frames yields a higher IoU hence we prefer this option for all further experiments. Furthermore, the highest increase in IoU is experienced after the first iteration, while subsequent ones yield a noticeably less improvement. We hypothesize that a random selection strategy would be placed somewhere in between both variants since some with high and some with low inconsistency coefficients would be chosen (Fig. 10).

[4] https://eval-vistas.mapillary.com/featured-challenges/1/leaderboard/1.

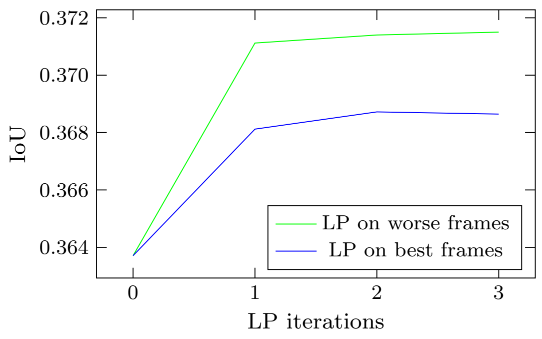

Fig. 10 LP variants: We evaluate the IoU increase by performing LP on the worse or the best frames, respectively. Propagating the labels towards the worse frames yields a higher IoU. Both LP variants converge quickly after the first iteration

We computed the IoU for different configurations on the NYUv2 dataset, including single-frame predictions and Semantic Fusion. Label propagation was performed using our approach to retrain the predictor. We denote in Table 1 the use of the retrained semantic segmentation as *with LP*. Table 1 shows that our method outperforms single-frame as well as SemanticFusion. Using label propagation further improves the IoU. A visual comparison is provided in Fig. 11 for four different scenes. Already SemanticFusion improves single-frame predictions e.g. on the TV (yellow, first row), the window (blue, third row) and wall (gray, last row), but the result is noisy and partially inconsistent. We attribute this mostly to surfels on different scales that are not correctly fused. In comparison, our mesh is more consistent, for example on the bath tub (second row), but smoother around the edges. Yet, the bed (third row) is still mostly classified as a sofa. Through our label propagation and subsequent retraining, we were able to correct the classification. In the last row, we show a failure case in which the Label Propagation decreases the accuracy as the table (green) gets segmented as furniture. This decrease in accuracy is due to the fact that most single-frame predictions are wrongly labeling the object and establishing consistency through LP reinforces this wrong labeling. Further complete reconstructed scenes from NYUv2 are visualized in Fig. 12.

We also conduct accuracy experiments on the courtyard dataset for which we densely labeled 48 frames around the scene. For fairness we labeled sky as background due to missing representation within the mesh. Figure 13 shows that retraining using label propagation greatly improves the accuracy for most classes. However, an interesting observation from this experiment is that the single-frame predictions have on average higher accuracy than the fused semantics from the mesh. This is due to the fact that both the camera poses and the mesh are imperfect, hence fusing the informa-

Table 1 NYUv2 results: We compare our method against single-frame predictions and SemanticFusion (McCormac et al. 2017)

Method	Bed	Bookshelf	Ceiling	Chair	Floor	Furniture	Objects	Picture	Sofa	Table	Tv	Wall	Window	Mean IoU
Single frame	0.46	0.17	0.13	0.25	0.68	0.34	0.28	0.33	0.22	0.15	0.12	0.51	0.29	0.302
Single frame with LP	0.52	0.20	0.14	0.27	0.68	0.35	0.30	0.35	0.26	0.14	0.14	0.53	0.31	0.322
SemanticFusion	0.47	0.15	0.18	0.30	0.65	0.36	0.30	0.35	0.24	0.15	0.20	0.53	0.33	0.324
SF with LP	0.52	0.18	0.21	0.31	0.65	0.38	0.31	0.38	0.28	0.16	0.20	0.54	0.36	0.343
Ours	0.54	0.17	0.23	0.35	**0.71**	0.40	0.33	0.39	0.28	**0.18**	0.21	0.56	0.37	0.363
Ours with LP	**0.56**	**0.20**	**0.25**	**0.35**	0.68	**0.40**	**0.34**	**0.41**	**0.31**	0.17	**0.23**	**0.57**	**0.38**	**0.372**

All cases are evaluated with and without Label Propagation (LP). For the case of single-frame, we exclude pixel without a valid depth measurement. All evaluations were performed at 320 × 240 resolution

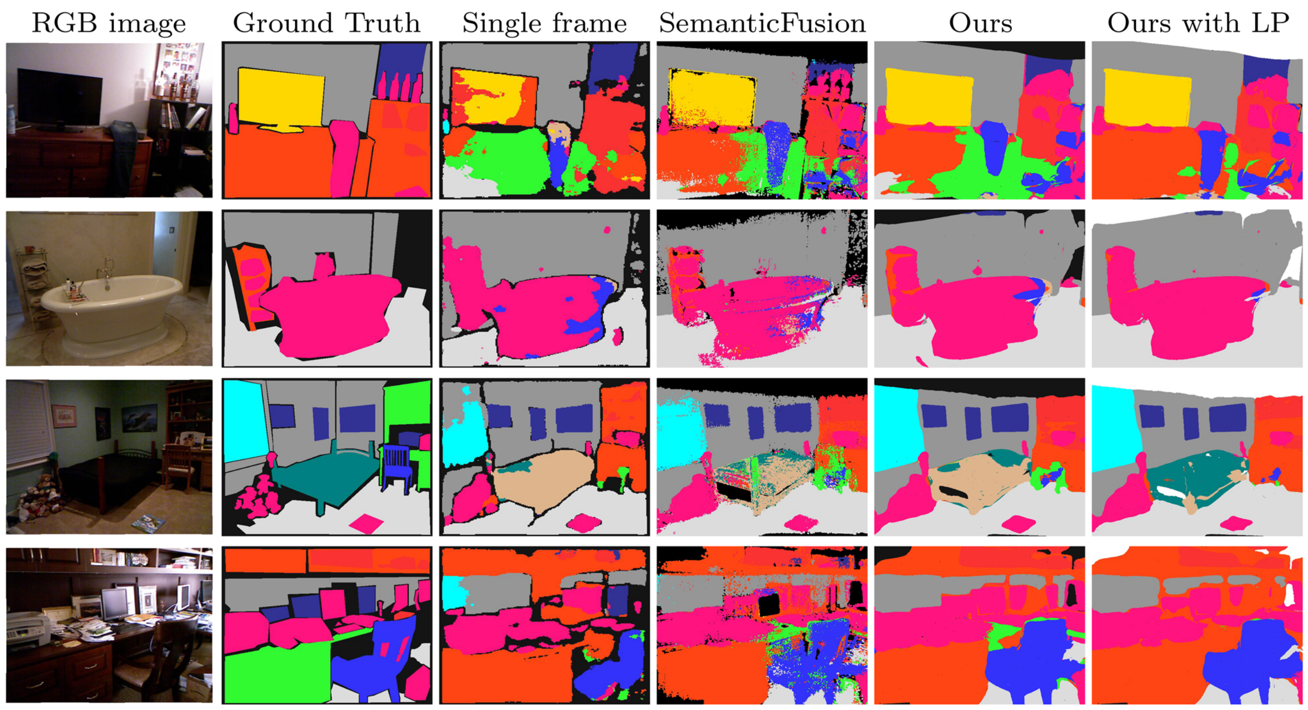

Fig. 11 NYUv2 qualitative results: We compare our method, with and without Label Propagation, against single-frame predictions and SemanticFusion. The first three rows show a clear improvement achieved through Label Propagation, as the predictor learns to segment the table, bathtub and bed more accurately. The last row shows a failure case in which the Label Propagation decreases the accuracy as the table represented in green gets segmented as furniture. This decrease in accuracy is due to the fact that most single-frame predictions are wrongly labeling the object and establishing consistency through LP reinforces this wrong labeling

tion from various points of view may lead to discrepancies. This limitation is further reinforced by the fact that the classes which experience a higher drop in accuracy from the fusing process are those which are spatially small (lane-markings, poles, and street lights), while broader classes like building and vegetation remain largely unaffected by errors from the scene reconstruction. For this reason we conduct further experiments to evaluate the impact of misalignments in the following Sect. 7.4. Nevertheless, we can conclude that label propagation grants a net improvement in the semantic accuracy, increasing the mean IoU for single-frame prediction by 7% and for the fused information by 3%. Figure 14 shows a visual comparison of the semantics using various view points from the courtyard. The copters landing gear and rotor arm

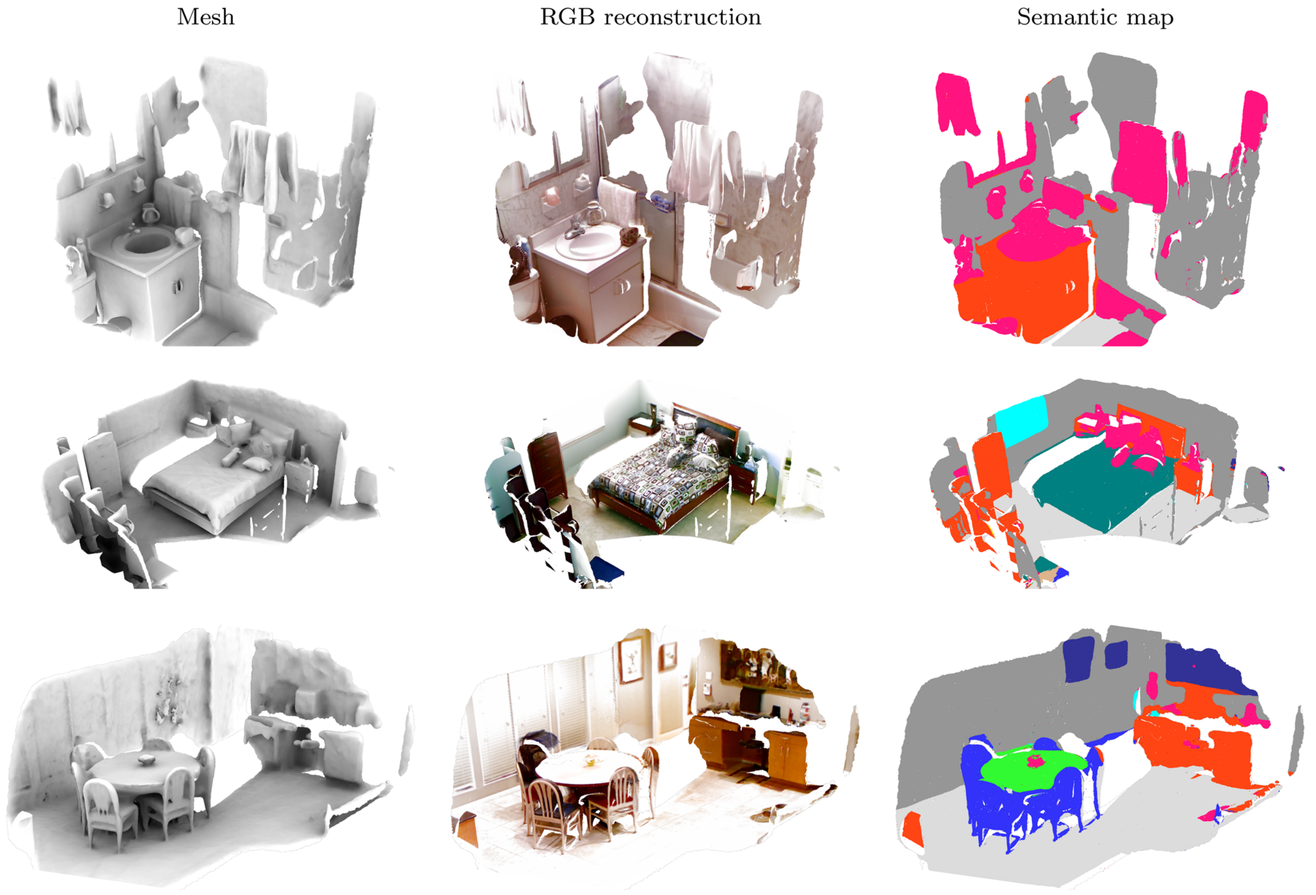

Fig. 12 Scenes from NYUv2: The mesh (left) is reconstructed from the surfel map of ElasticFusion and textured with RGB appearance (middle) and semantic labels (right)

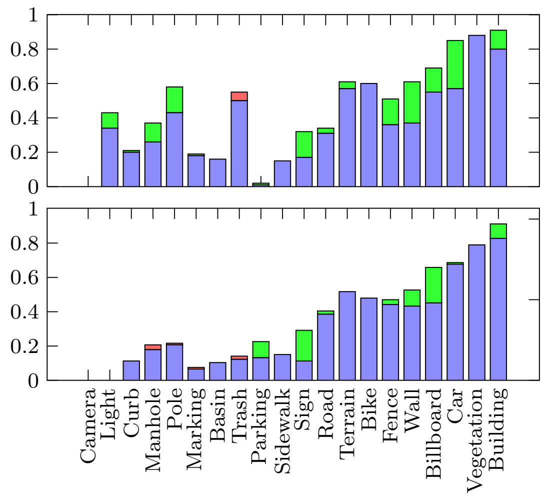

Fig. 13 Courtyard results: We compare our method (bottom) against single-frame predictions (top). The per class IoU is denoted by blue bars. An increase in IoU due to Label Propagation is marked in green, a decrease in red. For single-frame we mask out the landing gear of the MAV and the areas which are not covered by the mesh (Colour figure online)

visible within camera images are masked out prior to evaluation. A partial failure case is shown in the first row. The thin lane-markings are actually degraded through fusion and LP. We trace this back to inaccurate manual extrinsic and temporal calibration, since the corresponding single-frames continuously contained the lane-markings. Still, we observe improvements after LP on the container next to the service station. The second view was captured from behind the same service station. Insufficiency in the meshing process created only the top of the pole (front, left), which is correctly classified, but not connected to the ground plane reducing the overall IoU compared towards single-frame predictions. Further improvements through LP are especially visible in the last two rows. The left window is classified as a sign prior to retraining and a large portion of the sidewalk was incorrect. Also the building in the background is improved. The rooftop (third row) presents a unique novel view that is largely misclassified in the single-frame. Our mesh-based fusion improves the result as expected and allows successful retraining.

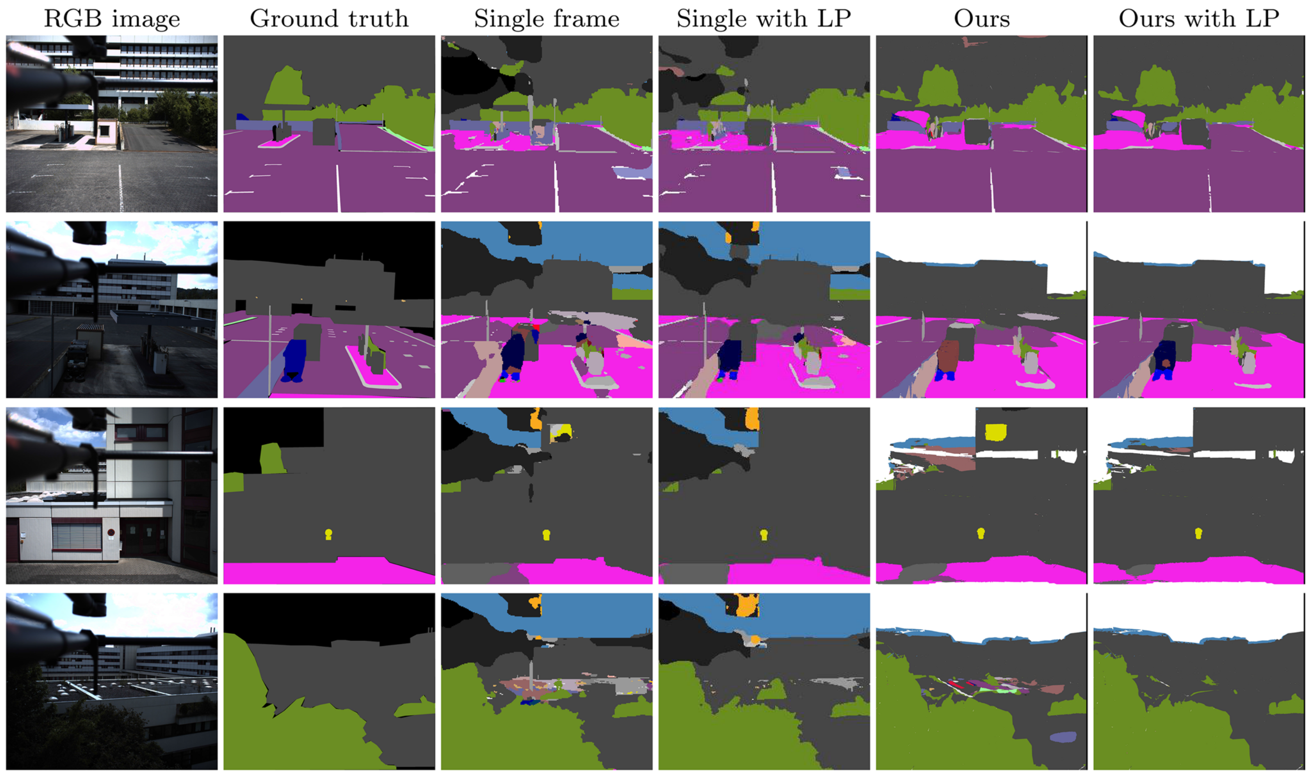

Fig. 14 Courtyard qualitative results: We compare our method against single-frame with and without the label propagated and retrained RefineNet

7.4 Registration Robustness

Mapping with known poses always raises the question of how robust the system is regarding misalignment. For this we perform experiments on the synthetic Synthia dataset (Ros et al. 2016) which provides ground truth poses, depth and semantics. We add random noise to the poses in order to observe the effect on the accuracy of the semantic map. We chose the *seq_4_summer* scene, due to the low number of dynamic objects. We aggregated the depth images and meshed the resulting point cloud (see Fig. 15). Synthia provides images from eight cameras arranged in groups of four to create an omnidirectional view-cone. For simplicity, we choose for the reconstruction only the front-facing camera of the left group. In order to analyze the behavior of the semantic map under incorrect poses or incorrect calibration between depth and color camera, the IoU is calculated for increasing amounts of noise on the translational (≤ 0.5 m) as well as rotational ($\leq 5°$) part of the camera poses. The IoU with increasing portion of noise is visualized per class in Fig. 16 with invisible or dynamic classes being disregarded. We choose to retain the cars as most of them were parked, and hence do not pose a problem for the reconstruction. As expected, the IoU decreases faster for smaller object classes, like poles, lights, and signs, than for buildings or the road. In conclusion, as the robotics community moves towards larger and bigger datasets with more semantic classes, the detail of the semantic maps will heavily depend on correct sensor poses.

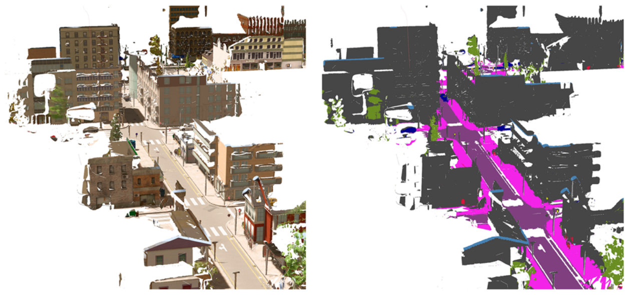

Fig. 15 Synthia semantic map: We reconstruct a semantic map from a subset of frames from the Synthia dataset (Ros et al. 2016). We use the ground truth data and apply noise to evaluate the impact of camera misalignment on the semantic map accuracy

7.5 Runtime Performance

We evaluate the runtime performance of our meshing and texturing modules separately, as they are performed sequentially with no overlap. Figure 17 shows the resulting meshes after Poisson reconstruction for our simplified cloud, and the naïvely aggregated full point set recorded by the Velodyne scanner. It can be observed that the reconstruction quality

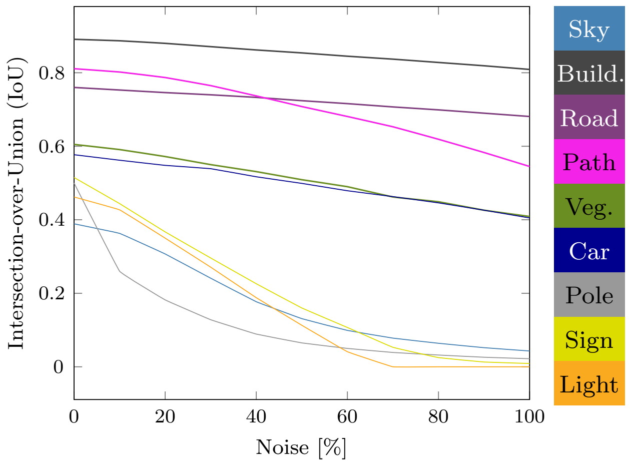

Fig. 16 Registration robustness: Incorrect sensor poses for semantic map creation affects the accuracy as measured by IoU of larger object classes like buildings less than light posts or signs. Increasing amounts of noise are applied on the translation (≤ 0.5 m) and rotation ($\leq 5°$) of the sensor poses

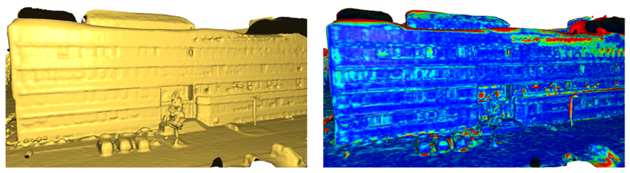

Fig. 17 Poisson reconstruction comparison: Reconstruction from the edge-aware simplification (left), and its difference (right) toward the full reconstruction. The colormap denotes the deviation between the two meshes where red equals a difference of 15 cm and blue shows no difference. The deviation is minimal in areas of interest while reconstruction after simplification is faster (see Table 2) (Colour figure online)

Table 2 Poisson reconstruction using the naïvely aggregated cloud and our edge-aware simplified cloud

Cloud	#Points	#Verts	Time (s)	Mem (GB)
Full	103M	4.5M	521.9	2.82
Simple	21M	3.3M	193.8	1.99

We report the number of points of the input cloud, the number of vertices of the reconstructed mesh, and the time and peak memory used by the reconstruction process

does not suffer while the runtime and memory consumption is significantly decreased (see Table 2).

The runtime of the semantic integration on the courtyard dataset is summarized in Fig. 18. We achieve real-time performance with an average texturing time of 27.1 ms per frame using a 8K texture. Decommitting the sparse volume is, however, a demanding functionality, and causes the average time per frame to increase to 90.1 ms. Nevertheless, the semantic integration achieves 38.6 ms per frame for a smaller texture resolution of 4K.

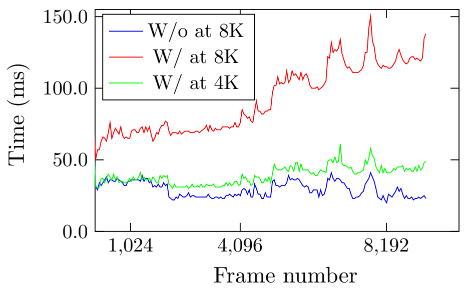

Fig. 18 Timing results for the courtyard dataset: Semantic integration using a texture resolution of 8K without decommitting (blue line) can be performed in real-time. Enabling the decommiting at the same texture resolution (red line) proves to be too slow for real-time usage. Lowering the texture resolution to 4K allows the semantic integration with decommittment (green line) to be performed at real-time speeds (Colour figure online)

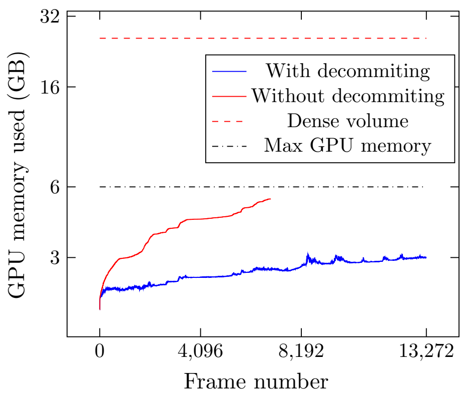

Fig. 19 GPU memory usage for 10K texture on the courtyard dataset: Dense allocation (red dashed line) would occupy more than 25.7 GB. Sparse allocation (red line) without decommitting quickly overburdens the available 6 GB causing a system failure. The memory usage drops with decommitting (blue line) below 3.1 GB at all times enabling the full reconstruction of the semantic map (Colour figure online)

7.6 Memory Consumption

Memory usage of the texturing system is also evaluated with and without decommitting of pages of the sparse texture. The results for the courtyard dataset are summarized in Fig. 19. We also analyze the relation between the decommitting threshold (the probability below which the pages in the sparse texture are deallocated) and the Intersection-over-Union (IoU) in Fig. 20. We evaluate this measure on the NYU dataset due to the presence of more labeled images than in the courtyard dataset which allows for a more accurate evaluation. We observe that while low values of the threshold greatly reduces memory usage, higher values cause the IoU to degrade as more valuable information from the semantic texture is disregarded. However the decrease is still minor ($\leq 0.6\%$), as we restrain from decommitting pages that contain the argmax class. For further experiments, we chose a threshold value of 0.1.

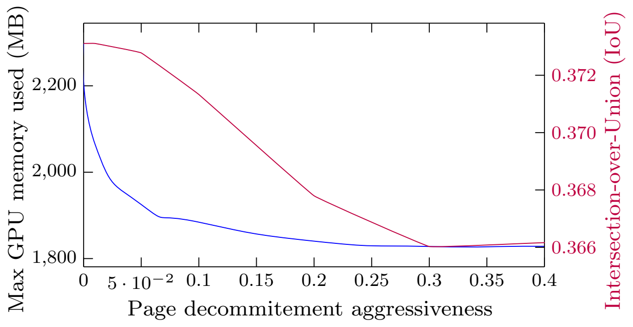

Fig. 20 Average GPU memory usage for different decommitting thresholds on NYUv2: Increasing the decommitting threshold quickly reduces the memory consumption (blue line), while the IoU decreases slowly. As a consequence, we typically fix the threshold to 0.1 (Colour figure online)

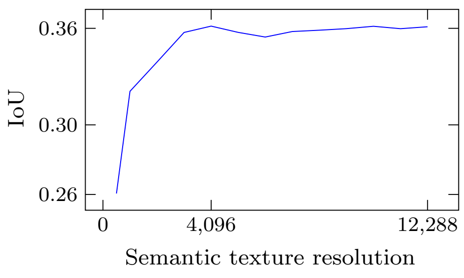

Fig. 21 Texture resolution and IoU: We evaluate the mean IoU on the courtyard dataset as we increase the resolution of the semantic texture. The accuracy quickly rises and converges at a resolution of around 4K

7.7 Texture Resolution and Semantic Accuracy

We evaluate the impact of the semantic texture resolution and the accuracy of the semantic map. We perform the evaluation on the courtyard dataset as it spans a larger area than the NYU dataset and therefore the impact of the texture size becomes more noticeable. Semantic texture integration is performed for a series of texture resolutions ranging from 512 to 12,288 pixels and IoU is evaluated for each one. We observe that the IoU steadily increased ans becomes stable at around a resolution of 4k (Fig. 21).

7.8 Distance Weighting and Semantic Accuracy

During semantic integration the only weighting applied while projecting from the 2D segmentation into the 3D scene is based on a linear fall-off between d_{min} and d_{max}. The impact of the weighting is evaluated on the courtyard dataset as it presents larger variability in terms of distance. Three different values for the distance thresholds are evaluated and compared to no weighting (full confidence in the semantic information regardless of distance). The results are gathered in Table 3. We observe that having no distance weighting results in the lowest IoU while the highest one was achieved by a linear fall-off between 0 m and 100 m. Small variations in the thresholds have little impact unless the d_{max} is severely restricted (for example to only 30 m) at which point distant parts of the scene are ignored and thus severely affecting the IoU.

Table 3 Distance fall-off: We experiment with various thresholds for d_{min} and d_{max} for the linear fall-off during semantic integration on the courtyard dataset

	No weight	30–100 m	0–100 m	0–30 m
IoU	0.294	0.3259	0.3289	0.309

Having no distance weighting (full confidence in the semantic information regardless of distance), achieves the lowest IoU. Modifying d_{min} has a marginal effect while restricting d_{max} is detrimental as too much distant information is discarded

8 Limitations

In the previous sections we have seen the impact of incorrect poses for semantic mapping as well as inadequate geometric meshes reduce the accuracy especially for small and thin classes. Another limitation arises from the Expectation Maximization strategy itself. The current belief is reinforced and thus not able to correct the estimate in all cases. Currently, dynamic objects are completely disregarded in our approach, but often present in collected data. This introduces artifacts in the reconstruction as well as the semantic map and reduces the accuracy for moving classes like pedestrians or cars. Furthermore, retraining from such sequences will bias the semantic segmentation towards static classes. Hence, incorporating object tracking could alleviate this problem. In terms of implementation the sparsity of the probability volume is greatly influenced by the quality of the UV parametrization. If areas of the mesh that are spatially close also lie nearby in the parametric space and are from the same class then they cover a page in GPU memory more efficiently. However, in the extreme case where each triangle is mapped to independent texture coordinates, the chance for decommitting reduces significantly. Luckily, most man-made environments are generally planar and their parametrization can be easily performed. Despite this, the limitation is an important one and it rises the question of a possibly semantic-aware parametrization in which the parametric area is influenced by the class, e.g. assigning lower resolution for vegetation and buildings and higher for cars and pedestrians.

9 Conclusion

We presented a novel semantic mapping system that uses textured meshes to store the semantic information and allows to couple semantic and geometric representation at independent resolution improving scalability and accuracy even for large datasets. The mesh-based representation allows to enforce spatial and temporal consistency over multiple observations as well as enabling semi-supervised retraining from novel view points by propagation of labels in an iterative manner. Quantitative and qualitative results on the NYUv2 dataset demonstrate the benefits of our approach compared to the state of the art method SemanticFusion.

Acknowledgements We would like to thank David Droeschel for his effort in providing accurate poses for the courtyard dataset. This work was supported by Grant BE 2556/7 of the German Research Foundation (DFG).

References

Acuna, D., Ling, H., Kar, A., & Fidler, S. (2018). Efficient interactive annotation of segmentation datasets with Polygon-RNN++. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 859–868).

Bao, S. Y., & Savarese, S. (2011). Semantic structure from motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Bao, S. Y., Chandraker, M., Lin, Y., & Savarese, S. (2013). Dense object reconstruction with semantic priors. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1264–1271).

Blaha, M., Vogel, C., Richard, A., Wegner, J. D., Pock, T., & Schindler, K. (2016). Large-scale semantic 3D reconstruction: An adaptive multi-resolution model for multi-class volumetric labeling. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3176–3184).

Castrejon, L., Kundu, K., Urtasun, R., & Fidler, S. (2017). Annotating object instances with a Polygon-RNN. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR).

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834–848.

Cherabier, I., Häne, C., Oswald, M. R., & Pollefeys, M. (2016). Multi-label semantic 3D reconstruction using voxel blocks. In *Proceedings of the international conference on 3D vision (3DV)* (pp. 601–610).

Cherabier, I., Schönberger, J. L., Oswald, M. R., Pollefeys, M., & Geiger, A. (2018). Learning priors for semantic 3D reconstruction. In *Proceedings of the European conference on computer vision (ECCV)*.

Civera, J., Gálvez-López, D., Riazuelo, L., Tardós, J. D., & Montiel, J. (2011). Towards semantic SLAM using a monocular camera. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1277–1284).

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 248–255).

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122.

Droeschel, D., & Behnke, S. (2018). Efficient continuous-time SLAM for 3D lidar-based online mapping. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)*.

Eigen, D., & Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 2650–2658).

Engel, J., Schöps, T., & Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 834–849).

Garland, M., & Heckbert, P. S. (1998). Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the IEEE VIS* (pp. 263–269).

Goldman, D., & Chen, J. (2005). Vignette and exposure calibration and compensation. In *Proceedings of the IEEE international conference on computer vision (ICCV)*.

Häne, C., Zach, C., Cohen, A., Angst, R., & Pollefeys, M. (2013). Joint 3D scene reconstruction and class segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 97–104).

Häne, C., Savinov, N., & Pollefeys, M. (2014). Class specific 3D object shape priors using surface normals. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 652–659).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE internatioinal conference on computer vision (ICCV)* (pp. 2980–2988).

Hermans, A., Floros, G., & Leibe, B. (2014). Dense 3D semantic mapping of indoor scenes from RGB-D images. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 2631–2638).

Holz, D., & Behnke, S. (2015). Registration of non-uniform density 3D laser scans for mapping with micro aerial vehicles. *Robotics and Autonomous Systems*, *74*, 318–330.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*(3), 189–206.

Jain, S. D., & Grauman, K. (2016). Active image segmentation propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2864–2873).

Kazhdan, M., & Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, *32*(3), 29.

Kostavelis, I., & Gasteratos, A. (2015). Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, *66*, 86–103.

Kundu, A., Li, Y., Dellaert, F., Li, F., & Rehg, J. M. (2014). Joint semantic segmentation and 3D reconstruction from monocular video. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 703–718).

Landrieu, L., & Simonovsky, M. (2017). Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Li, X., & Belaroussi, R. (2016). Semi-dense 3D semantic mapping from monocular SLAM. arXiv preprint arXiv:1611.04144

Lianos, K. N., Schönberger, J. L., Pollefeys, M., & Sattler, T. (2018). VSO: Visual semantic odometry. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 234–250).

Lin, G., Milan, A., Shen, C., & Reid, I. (2017). RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5168–5177).

Ma, L., Stückler, J., Kerl, C., & Cremers, D. (2017). Multi-view deep learning for consistent semantic mapping with RGB-D cameras. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 598–605).

Mackowiak, R., Lenz, P., Ghori, O., Diego, F., Lange, O., & Rother, C. (2018). CEREALS—cost-effective region-based active learning for semantic segmentation. arXiv preprint arXiv:1810.09726.

Maninchedda, F., Häne, C., Jacquet, B., Delaunoy, A., & Pollefeys, M. (2016). Semantic 3D reconstruction of heads. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 667–683).

McCormac, J., Handa, A., Davison, A., & Leutenegger, S. (2017). SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 4628–4635).

Nakajima, Y., Tateno, K., Tombari, F., & Saito, H. (2018). Fast and accurate semantic mapping through geometric-based incremental segmentation. arXiv preprint arXiv:1803.02784.

Neuhold, G., Ollmann, T., Bulo, S.R., & Kontschieder, P. (2017). The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 5000–5009).

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems* (pp. 5099–5108).

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). ROS: An open-source robot operating system. In *ICRA workshop on open source software*.

Riegler, G., Ulusoy, A.O., & Geiger, A. (2017). OctNet: Learning deep 3D representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., & Lopez, A. M. (2016). The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3234–3243).

Savinov, N., Häne, C., Ladicky, L., & Pollefeys, M. (2016). Semantic 3D reconstruction with continuous regularization and ray potentials using a visibility consistency constraint. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5460–5469).

Schönberger, J. L., Pollefeys, M., Geiger, A., & Sattler, T. (2018). Semantic visual localization. CVPR.

Sheikh, R., Garbade, M., & Gall, J. (2016). Real-time semantic segmentation with label propagation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 3–14).

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 746–760).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Stueckler, J., Waldvogel, B., Schulz, H., & Behnke, S. (2014). Dense real-time mapping of object-class semantics from RGB-D video. *Journal of Real-Time Image Processing (JRTIP)*, *10*, 599–609

Su, H., Jampani, V., Deqing, S. S., Maji, E., Yang, M. H., Kautz, J., et al. (2018). SPLATNet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Sun, L., Yan, Z., Zaganidis, A., Zhao, C., & Duckett, T. (2018). Recurrent-OctoMap: Learning state-based map refinement for long-term semantic mapping with 3D-lidar data. *IEEE Robotics and Automation Letters*, *3*(4), 3749–3756.

Tatarchenko, M., Park, J., Koltun, V., & Zhou, Q. Y. (2018). Tangent convolutions for dense prediction in 3D. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3887–3896).

Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. arXiv preprint arXiv:1704.03489.

Thürrner, G., & Wüthrich, C. A. (1998). Computing vertex normals from polygonal facets. *Journal of Graphics Tools*, *3*(1), 43–46.

Tulsiani, S., Zhou, T., Efros, A. A., & Malik, J. (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Valentin, J. P., Sengupta, S., Warrell, J., Shahrokni, A., & Torr, P. H. (2013). Mesh based semantic modelling for indoor and outdoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2067–2074).

Vezhnevets, A., Buhmann, J. M., & Ferrari, V. (2012). Active learning for semantic segmentation with expected change. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3162–3169).

Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V. A., Kähler, O., Murray, D. W., Izadi, S., Pérez, P., et al. (2015). Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 75–82).

Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., & Davison, A. (2015). ElasticFusion: Dense SLAM without a pose graph. In *Proceedings of robotics: science and systems*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5987–5995).

Yang, L., Zhang, Y., Chen, J., Zhang, S., & Chen, D. Z. (2017). Suggestive annotation: A deep active learning framework for biomedical image segmentation. In *international conference on medical image computing and computer-assisted intervention* (pp. 399–407).

Zaganidis, A., Sun, L., Duckett, T., & Cielniak, G. (2018). Integrating deep semantic segmentation into 3D point cloud registration. *IEEE Robotics and Automation Letters*, *3*(4), 2942–2949.

Zollhöfer, M., Stotko, P., Görlitz, A., Theobalt, C., Nießner, M., Klein, R., & Kolb, A. (2018). State of the art on 3D reconstruction with RGB-D cameras. In *Computer graphics forum* (pp. 625–652).