3D Semantic Label Transfer in Human-Robot Collaboration

Dávid Rozenberszki[1,2] Gábor Sörös[1] Szilvia Szeier[2] András Lőrincz[2]

[1]Nokia Bell Labs
Budapest, Hungary

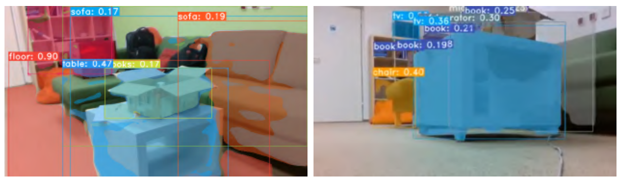
[2]Eötvös Loránd University
Budapest, Hungary

Abstract

We tackle two practical problems in robotic scene understanding. First, the computational requirements of current semantic segmentation algorithms are prohibitive for typical robots. Second, the viewpoints of ground robots are quite different from the typical human viewpoints of training datasets which may lead to misclassified objects from robot viewpoints. We present a system for sharing and reusing 3D semantic information between multiple agents with different viewpoints. We first co-localize all agents in the same coordinate system. Next, we create a 3D dense semantic model of the space from human viewpoints close to real time. Finally, by re-rendering the model's semantic labels (and/or depth maps) from the ground robots' own estimated viewpoints and sharing them over the network, we can give 3D semantic understanding to simpler agents. We evaluate the reconstruction quality and show how tiny robots can reuse knowledge about the space collected by more capable peers.

1. Introduction

Autonomous vehicles and robots are becoming ubiquitous in industrial applications and they can be expected to appear in our future homes and cities too. In order to improve their navigation capabilities and to collaborate with humans, robots need to well understand their spatial environment, a feature often referred to as spatial artificial intelligence (AI). Spatial AI can be divided into four distinct but interdependent layers: *spatial perception*, *pose tracking*, *geometry understanding*, and *semantic understanding*, which all can benefit from prior models of the world. Semantic 3D reconstruction focuses on capturing the shape and appearance of the physical space with both scene geometry and semantic classes. It has been a very active research area in the recent decade, because an accurate 3D map of the environment is a necessary component in industrial collaboration tasks, augmented reality, home robotics, autonomous driving, to name only a few application domains.



(a) Human perspective (b) Robot perspective

Figure 1: The accuracy of semantic segmentation drops when applied on image perspectives different from the ones in the typical human-view training sets. Segmentation models have difficulties recognizing objects from the bottom view of ground robots. Instead, we provide our robots with synthetic semantic labels rendered remotely from an existing or live reconstructed model of the space.

Of particular interest are methods that enable robots to map a previously unknown environment using their own sensors. The geometric reconstructions can be created sparsely with LiDAR sensors or densely with RGB or RGBD cameras. The semantic class labels are usually estimated from 2D RBG(D) images which are then backprojected onto the reconstructed geometry. Two significant practical problems in robotic semantic 3D reconstruction are the computational constraints of robotic platforms and the different view perspectives of the common robots. Humans see objects from the top while ground robots see everything from the bottom. It is a common challenge in image segmentation that the available training datasets (e.g., [37, 24, 5, 14, 15]) are annotated from human perspectives, and models generalize poorly to different perspectives. We illustrate in Figure 1 that the segmentation accuracy drops significantly with the change of perspective.

In this paper, we propose a system for multiple camera agents to share 3D metric-semantic information among each other. We first co-localize agents in the 3D space, then create a 3D metric-semantic map of the scene in close to real time. Finally, we generate synthetic semantic and/or depth images and deliver them to resource-constrained agents that would otherwise not be able to reach all levels of spatial AI on their own.

More specifically, we adapt an existing semantic reconstruction pipeline [32] to multiple agents, extend it with more precise localization and mapping, multiple semantic segmentation approaches, arbitrary number of classes, a map manager, and a new rendering component. One can run the reconstruction on resourceful agents, on a dedicated PC, or even on a cloud service, preferably using human viewpoint inputs that are ideal for existing semantic segmentation methods. One can store the accumulated metric-semantic knowledge about the space on a map manager instance. By re-rendering images of the reconstructed semantic model from the viewpoints of agents, we can provide the less capable agents with synthesized scene labels. We show in a number of synthetic and real experiments that our system can accurately reconstruct a 3D semantic model of a space and that smaller agents can effectively gain semantic information from the reconstructed model and can also detect changes in the environment. We release the source code of the reconstruction pipeline[1] for the research community.

2. Related Work

We are interested in a modular system which allows us to easily replace components and to combine existing solutions with our own contributions. We review the related literature corresponding to our modules of spatial AI.

2.1. Pose tracking

Assuming suitable sensors and basic spatial perception, the next layer of spatial AI is pose tracking. The goal here is to track the movement of robots robustly and accurately in a wide variety of different scenarios. In most cases, one needs to make compromises to find the most suitable tracking sensor and tracking method. In outdoor applications and changing weather conditions, the optimal choice is usually LiDAR-based sensor fusion, while for indoor environments (as in our case) visual SLAM and visual-inertial SLAM approaches based on sparse feature maps are preferred. The most popular approach for indoor sparse visual SLAM are the ORB-SLAM methods [22, 21, 3] being able to maintain real-time performance and robust tracking.

Another important design choice is the scale of the scene we aim to track the robot movement, for example BAD-SLAM [35] is able to jointly track the pose and also reconstruct the scene with high fidelity, but does not scale well to larger building-size environments. BundleFusion [6] is also a popular approach to track an RGBD sensor, but as it jointly reconstructs a dense model of the scene, it is not compatible with our modular architecture design and is also hardly able to optimize scenes of larger scale.

[1]https://github.com/RozDavid/semantic_mapping

2.2. 3D Reconstruction

In 3D reconstruction, one can distinguish real-time reconstruction methods and offline optimization methods. Offline 3D reconstruction methods are optimizing for a large unordered set of stereo images of the same scene, and minimizing the back-projection error by finding matching features among different view perspectives such as the works [1, 34] or multiple commercial products like Pix4D, AliceVision or RealityCapture. We are instead focusing on real-time reconstruction and sharing over the network, so that we are not dependent on prior reconstruction but can immediately share partially reconstructed areas with all agents within the space.

Real-time 3D reconstruction algorithms represent and store the model as either sparse point clouds, or in case of dense models either as surfels [35, 40] or as a structured grid of truncated signed distance field (TSDF) volume. To overcome the biggest challenge of the memory representation of large TSDF volumes, there exist many octree-like solutions such as the work of Reichl et al. [30] who store voxels in a binary grid or the work of Funk et al. [7] who represent the free space in a logarithmic grid instead to be directly used in path planning applications. Another solution for more efficient free-space representation is called voxel-hashing introduced in [28]. This representation is also used in the Voxblox [29] algorithm. The Kimera [32] method also utilizes Voxblox and is designed to maintain good performance with bounded computational power and highly efficient memory allocation.

There is also an emerging trend to represent a scene by neural network weights in so called neural radiance fields [18, 16] learnt from posed monocular RGB or RGBD images [2], but today these methods are too slow to build and even slower to render. Additionally, there are promising recent works [23, 39] that project CNN feature maps from posed monocular images from 2D to 3D to reconstruct a consistent sparse TSDF map close to real time, but as for other learning-based approaches, generalization to arbitrary scenes is still a challenge.

2.3. Semantic Segmentation

There are two families of approaches that segment the surrounding 3D geometry semantically either using complete 3D reconstructions as inputs [11] or projecting labels from 2D image segmentation [19] onto the 3D model.

While taking 3D geometry as input for inferring the semantics of objects in the space is certainly beneficial, and with learning common object shapes it is possible to infer even occluded regions or even to complete partial reconstructions, the biggest problem in these kind of methods is the lack of training sets with diverse sets of objects. Most benchmarks consist of the same few object classes and could easily overfit for certain types of objects shapes.

The other family of approaches extract labels from 2D views and project pixel-level predictions onto the 3D model such as in SemanticFusion [17] and also Kimera [32]. Annotated 2D image datasets of large scale are already available such as ImageNet [14] or CoCo [15] and it significantly easier to create new datasets for custom classes for new applications. This is the reason why we chose a 2D to 3D segmentation approach and project the labels onto the reconstructed mesh. Furthermore, in this case we can keep the reconstruction module and semantic segmentation module independent. With the 2D-to-3D approach, we are able to use off-the-shelf semantic segmentation networks [10, 36] that are easier to train for specific environments and where 2D segmentation datasets are widely available.

2.4. Semantic Label Transfer

Further related work include image recognition [8] and AR rendering [13] with edge cloud support. There is a natural tradeoff between accuracy and latency in systems that balance between device-based recognition (lower accuracy, little delay) and edge service-based recognition (high accuracy, large delay). However, none of these works address pixel-wise and 3D segmentation and the problem of low viewpoints.

While 2D-3D semantic information transfer is widely used, there exists only few works on pixel-level annotation of input images through 3D-2D semantic label transfer. The work of Xie et al. [41] uses a semi-supervised method where the authors first coarsely annotate a structure from motion (SfM) reconstruction of outdoor scenes and project labels onto camera images. Similarly, the ScanNet [5] dataset annotation images were created with human-annotated reconstructed scenes and back-projection of the information onto known camera poses. A similar tool by Nguyen et al. [27] was proposed to help 3D annotation in a self-supervised way, by a prior graph-based segmentation network post-pocessed by human annotators. As the virtual camera poses for the rendered semantic images cannot be precisely aligned with the real camera images, the authors use a Canny edge-detection based alignment to finalize the projection of the labels.

While these ideas are similar to ours in theory of 3D-to-2D semantic information transfer, our work is different because the robots are able to access semantic information without the need for on-board segmentation and even without the need for mounting specific visual sensors. Instead, our agents rely on another, more capable peer that performs 3D semantic reconstruction simultaneously.

3. Method

We propose a system that allows not only fast metric-semantic reconstruction of the scene, but also sharing information between multiple camera agents.

As the 3D reconstruction research advances quickly and the state of the art for various components in spatial AI is exceeded frequently, it is beneficial to focus on modular architectures instead of end-to-end solutions. By using a modular pipeline architecture, one could easily replace and improve certain components depending on the direct use case, conditions of the environment, and the available sensor setup. Additionally, when the architecture is modular and communication-oriented, we are able to outsource certain components to remote computers in the network or even to cloud services.

Our modular architecture was inspired by and is based on the Kimera [32] pipeline, which we modified and extended in many components for more robustness and more versatile usage. The proposed modules of the reconstruction pipeline can be seen in Figure 2. We implemented all modules within the Robot Operating System (ROS) framework that supports different programming languages, and communication between processes (called nodes in ROS) is managed by a network orchestrator. With ROS, it is possible to combine measurements from multiple robots into the same system, and reconstruct the scene in real time from various sources. Depending on hardware constraints, it is a design choice which modules we execute onboard and which are outsourced, in our case to a desktop PC.

A map manager component (on the PC) helps to bring multiple cameras into the same global coordinate system, and can share sparse maps and dense point clouds between participating agents. While the agents estimate their own pose from own (monocular, stereo, or RBGD) camera input and share it on the network, a map renderer component can synthesize semantic labels (or even depth maps) of the model from the agents' viewpoints. In the following, we introduce all modules in more detail and our own contributions for a robust and device/sensor-invariant setup.

3.1. Pose Tracking

In the original Kimera [32] work, the visual-inertial odometry (VIO) module utilizes inertial measurements that help to track the pose in structure-less areas. However, there is still significant pose drift which hinders the creation of accurate 3D models and ultimately results in false renderings for other peers in our system. Therefore, we replaced Kimera-VIO with a modified version of the state-of-the-art, vision-only UcoSLAM [20] method, which is a reimplementation of ORB-SLAM2 [21] extended with of ArUco [31] landmarks and several other improvements for better runtime performance. The main improvements include the way the algorithm tracks frame-to-frame the detected ORB features before projecting them back to the 3D space and a very efficient implementation of ORB feature matching during relocalization.

We extended UcoSLAM with a ROS wrapper and modi-

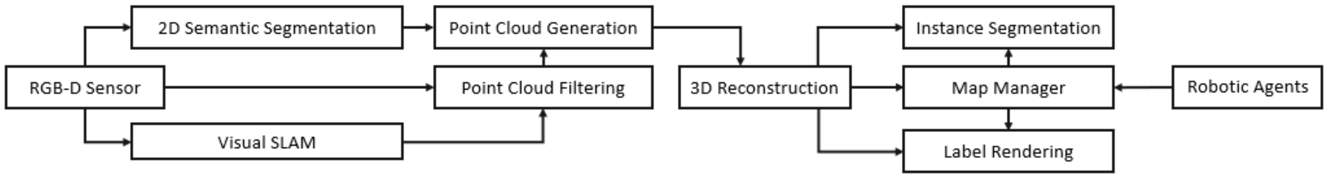


Figure 2: Our proposed reconstruction pipeline follows a modular design where individual components are easily swappable with newer ones for improvements and experimentation. Some components can be outsourced from less powerful agents.

fied it to be able to share the sparse SLAM maps with other agents for co-localization in the same physical space. To allow agents to localize in a SLAM map created by a different sensor setup (RGB, RGBD, stereo) and camera type (RealSense, Zed, Kinect), we feed input images into the SLAM framework with unified camera parameters both for the mapping phase and later for localization in the same coordinate system. By having unified camera parameters for all agents, it is possible to map the world with any of our cameras and to localize a different one later.

3.2. Semantic Segmentation

We can take any 2D semantic segmentation method that predicts pixel-wise semantic class labels for the input RGB or RGBD frame and register the segmentation masks on the input depth channel of our depth sensor. One should keep in mind two important aspects when choosing the segmentation method: computational complexity and segmentation accuracy. In our implementation, we tested for both aspects and integrated two state-of-the-art works for the different use cases. For fast inference and on-board calculation, we include the ESANet [36] model that takes aligned pairs of depth and color images as inputs. For higher accuracy but computationally more expensive segmentation, we outsource the task to remote a computer running MaskRCNN model with ResNet101 Feature Pyramid Network backbone. Both models were trained on the SUNRGBD [37] which contains annotated indoor scenes and hence suitable for our robotic reconstruction use case.

3.3. 3D Reconstruction

In the reconstruction module, we first generate a textured or semantically labelled point cloud per frame by projecting depth values of the image with known camera model and known pose to the 3D space. We integrate these per-frame point clouds into a TSDF volume and use a slightly modified version of the Kimera-Semantics [32] module that itself is based on the Voxblox [29] algorithm.

A common strategy to integrate new scans into TSDF is to cast a ray from the sensor origin to every point in the point cloud and update the surface distance along the ray. Over the casting process, every voxel crossed by the ray is updated with a weighting function. In the earlier KinectFusion [25] method, the weights were determined by the angle between the ray and the sensor origin of the surface normal at that point, or a constant weight of one was used for millimeter-size resolutions. The difference in Voxblox, and consequently in our work, are the more general equations for updating voxel weights that can be formulated as

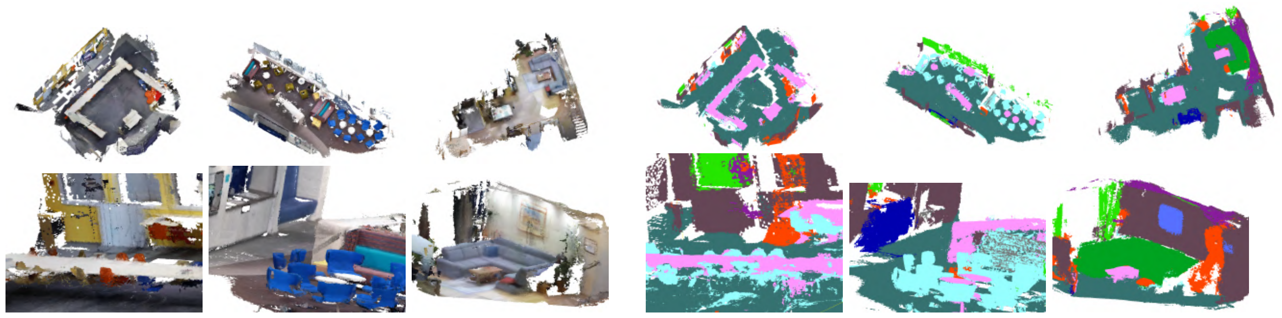
$$\begin{aligned} d(\mathbf{x},\mathbf{p},\mathbf{s}) &= ||\mathbf{p}-\mathbf{x}||sign\big((\mathbf{p}-x)\cdot(\mathbf{p}-\mathbf{s})\big) \\ w_{const}(\mathbf{x},\mathbf{p}) &= 1 \\ D_{i+1}(\mathbf{x},\mathbf{p}) &= \frac{W_i(\mathbf{x})D_i(\mathbf{x}) + w(\mathbf{x},\mathbf{p})d(\mathbf{x},\mathbf{p})}{W_i(\mathbf{x}) + w(\mathbf{x},\mathbf{p})} \\ W_{i+1}(\mathbf{x},\mathbf{p}) &= min(W_i(\mathbf{x}) + w(\mathbf{x},\mathbf{p}), W_{max}). \end{aligned} \tag{1}$$

where the existing distance and weight values are D and W, new values from point observations are the lowercase d and w, where d is the distance to the surface boundary and not the sensor-voxel distance. Given $\mathbf{x}$ voxel centroid position, $\mathbf{p}$ is the projected point position from the depth sensor measurement, and $\mathbf{s}$ is the sensor position in the scene.

The weighting model in Equation 1 is further improved in the work of Nguyen et al. [26] proving that if z is the depth of the camera measurement at the pixel value, the variance of the depth noise σ of a given ray varies predominantly with z^2 especially for structured light sensors. Following this, a simplified equation for estimating the measurement and behind-surface drop-off weight becomes

$$w_{quad}(\mathbf{x},\mathbf{p}) = \begin{cases} \frac{1}{z^2} & -\epsilon < d \\ \frac{1}{z^2}\frac{1}{\delta-\epsilon}(d+\delta) & -\delta < d < -\epsilon \\ 0 & d < -\delta. \end{cases} \tag{2}$$

Here, after empirical tests, we chose the truncation distance $\delta = 4v$ and $\epsilon = v$, where v is the voxel size. During the merging phase of the measurements, a so-called *grouped raycasting* approach is used, where the points of the measurement are projected into the voxel grid and grouped with the other points mapping to the same voxel. For dense measurements and large voxels, this increases performance without losing accuracy by calculating a weighted mean of points first and only casting the ray over the whole volume once. Since all measurements are taken into account, the noise is still reduced by the mean of multiple measurements, but shown to be almost exponentially



(a) Classroom (b) Office scene (c) Living room (d) Classroom (e) Office scene (f) Living room

Figure 3: Colored and semantic reconstruction results in three real scenes with different lighting conditions.

faster for dense RGB-D sensors and small indoor environments (depending on voxel sizes).

Until this point, our volume reconstruction is practically the same as the Voxblox approach, but instead of color values, one can also fuse semantic class labels into the voxel grid. We take the pixel-wise 2D semantic segmentations of each frame (produced by the segmentation module) and a pixel-wise semantic color that is included in the bundled raycasting to build an observed label frequency vector that can be propagated for all voxels along the bundled ray. Similar to other TSDF approaches, the updates are truncated behind the mesh surfaces for lower memory requirements and increased integration speed.

A Bayesian update rule is used for every voxel to determine posterior probabilities given all previous measurements. Every voxel is initialized with a uniform probability vector $P \in \mathcal{R}^n$, with values $l_i \in \mathcal{L}, l_i = log(1/n)$. Here n is the number of $\mathcal{L}$ labels in the used 2D semantic segmentation network. Voxel probabilities are updated iteratively along a ray $\mathcal{I}_t$ at timestep t that was projected by the registered semantic image and depth camera model. The posterior probability is calculated by

$$P(l_i, \mathcal{I}_1, \dots, k) = ZP(l_i, \mathcal{I}_1, \dots, k-1)P(\mathcal{O}_{\lfloor} = l_i, \mathcal{I}_k) \tag{3}$$

where $P(l_i, \mathcal{I}_1, \dots, k)$ is the probability for all labels and k measurements in voxels individually, $\mathcal{O}_{\lfloor} = l_i$ is the bundled prediction frequency vector for every measurement and Z is the normalizing likelihood parameter for measurement updates. Equation 3 was formulated as a combination of Kimera and SemanticFusion [17] ideas, as the original Kimera implementation was prepared for storing voxel semantics, but only with ground-truth labels and fixed size. With our modification, we can use any kind of segmentation module for segmentation masks, but still robustly reconstruct the volume with high confidence semantics.

Finally, a triangle mesh is extracted using the marching cubes algorithm, an example colored and semantic meshes of indoor scenes can be seen in Figure 3.

3.4. Point Cloud Filtering

In practice, the detection of low confidence or even false measurements is essential to successfully avoid reconstructions artifacts. In our 3D semantic reconstruction pipeline, we deal with three different error sources while integrating individual depth maps into the TSDF volume.

The first possible source of error stems from perception, where the pixel depth confidence calculated by stereo correspondence search is low. We ignore low-confidence values, because having noisy measurements in the map in low-textured areas results in low-quality reconstructions. Such pixel-wise confidence maps are automatically created with the score of a window-based stereo matching algorithm, for example in the Zed or Kinect SDKs.

The second source of error are dynamic objects in the assumed static scene. If we mask image areas where the segmentation module predicts dynamic objects such as humans, cars, or other robots, we are able to produce better reconstructions, where the artifacts of moving objects do not appear in the final model.

The third source of error is at pose estimation, especially pose drift and false relocalization. In real scenarios of larger areas, objects such as furniture are often moved, and the patterns on walls, floor or ceiling might be repetitive. Any of these can lead to false data associations in any visual SLAM method. To overcome the problem of localization with only sparse landmarks of the latest input image, one might use the whole reconstructed model in memory and jointly optimize pose and reconstruction, such as in the work BundleFusion [6]. Alternatively, to keep the modular approach and detach motion from the reconstruction, we use a motion model for detecting discontinuities in the camera trajectory. The proposed model validates the calculated pose with the smoothness of the predicted trajectory and stops point cloud integration when false data association could introduce artifacts in the reconstructed map. We check for both translational and rotational discontinuities of the camera pose. Whenever a discontinuity in the pose or its first derivative is detected, the reconstruction is paused until stable relocalization happens.

3.5. Object Instance Segmentation

For many AR or robotics tasks, having only a class-level semantic reconstruction of the scene is not enough, we need to know the pose and extent of individual objects. Such instance-level representation leads us to a higher level of understanding of the scene. A similar work to our reconstruction pipeline, Voxblox++ [9] detects instances of semantic objects directly from the prediction of a 2D instance segmentation network instead of a 2D semantic segmentation network.

We believe that extracting object instances in 3D is more robust than in 2D, because we can easier associate points belonging to the same instance and we can even extract objects whose class is unknown based purely on their geometry. Our proposed solution takes the surface point cloud of the 3D reconstruction, and clusters different objects together based on their spatial extent. We first filter possible cluster candidates with the difference of normals (DoN) [12] algorithm, and then extract the object instances and corresponding point cloud clusters with Euclidean clustering on the filtered cloud. Figure 4 illustrates how well the proposed instance segmentation module can identify objects in the reconstructed scene.

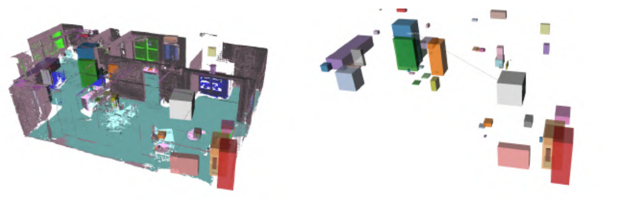


Figure 4: Instance-level bounding boxes extracted from the metric-semantic model (a scene from Habitat).

3.6. Semantic Label Transfer

When another agent localizes in the same SLAM map in which the model was made, it is possible to render semantic masks from the model to the new agent's view. The agents perform localization on their own, then request for semantics with their latest pose. The semantics can be generated continuously or on demand. While not implemented, agents could also request only bounding boxes of specific types of objects. The maps and the models are maintained by a simple map manager component.

To render segmentation labels remotely for arbitrary image perspectives, a simple model renderer based on PyRender is continuously running in the network. It can load a (partial) semantic mesh even during reconstruction and render masks with flat colors representing the semantic classes. Example rendered labels are shown in Figure 5 from human perspective, while transferred labels for low-perspective camera in Figure 9.

Note that given the camera model, one can generate semantic masks for any type of camera, and in fact if an agent is able to accurately localize itself by any other means (e.g., 2D LiDAR), it can still receive semantic information about its surroundings even without having its own camera. Similarly, we can also render depth maps from the model for agents without a depth sensor. Furthermore, our method can not only transfer semantic information to arbitrary image perspectives, but it also opens the way to generate pseudo-ground truth datasets for training object recognizers that need to deal with unusual viewpoints.

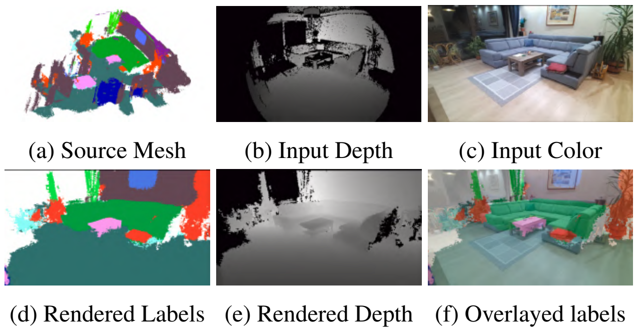


Figure 5: Top row: inputs; Bottom row: outputs of the label rendering module. We can generate semantic labels for arbitrary image perspectives and deliver them to agents when on-board image segmentation is not possible.

4. Experiments

We first evaluate our pipeline for robustness and precision in real scenes for a variety of possible environments qualitatively and in simulated environments quantitatively. Our agents are portable computers (laptops, Intel NUCs) with cameras (Intel Realsense, Microsoft Kinect, Stereolabs Zed mini, etc.) either handheld or mounted on Hercules or OpenBot robot bases.

Qualitative results in real scenes We demonstrated the quality of the semantic reconstruction pipeline in Figure 3, where three different scenes were scanned with a Kinect for Azure camera in real-time and later used as the source mesh for semantic label transfer. We show the quality of the reconstruction in the function of voxel size in Figure 6, where the trade-off between runtime performance for online applications and preserving details for object surfaces has to be balanced. We can see that while much more details are preserved with smaller voxel sizes, the quality is bounded by the sensor (in this case a Kinect for Azure) precision.

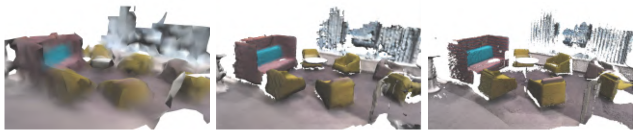

(a) $v = 20cm$ (b) $v = 5cm$ (c) $v = 1cm$

Figure 6: Reconstruction quality depending on voxel size.

Runtime measurements We did the corresponding runtime analysis for the computationally expensive steps of the pipeline. The memory and computation time requirements increase exponentially with the voxel size for integrating new point clouds, TSDF layer size and mesh extracting with the marching cubes algorithm. Other components such as point cloud generation or preprocessing cloud filtering remain independent from resolution. We empirically found the optimal balance between quality and performance at $2\,cm$ voxel size, where a full integration step takes $170\,ms$ of computation time on a laptop computer.

Quantitative results in synthetic scenes For quantitative evaluation, we use the Habitat [33] simulator with the Replica [38] dataset which contains photo-realistic representations of real scenes. It provides the highest geometric, texture, and semantic resolution among all datasets we know, yet still includes some noise from reconstruction errors. A screenshot of the simulator output can be seen in Figure 7. We tested our method's geometric and semantic accuracy, while also evaluating the robustness of the motion model for trajectory consistency.

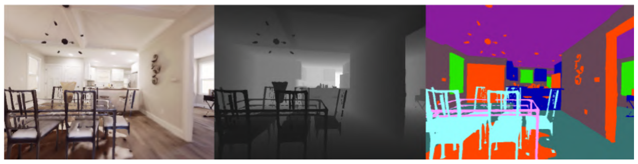


Figure 7: Example synthetic sensor input to our pipeline generated by the Habitat simulator and a Replica scene.

For geometric accuracy of the reconstruction, we compared the resulting mesh to the ground truth model for every extracted surface voxel in Table 1. Here we included a synthetic noise model [4] to evaluate our pipeline with more realistic commodity RGBD camera models. For comparing the reconstruction results with the ground truth model we run a variety of parameter combinations regarding the voxel resolution, input image resolution and simulated noise volume in both semantic- and textured-mesh modes. From these experiments, a few are shown in Table 1, where (1) shows the geometric accuracy with clean data and $2\,cm$ voxel size, (2) shows $10\,cm$ voxel resolution with 10x Redwood noise model [4] for simulated noise distribution, (3) shows a reconstruction with similar noise, but $2\,cm$ voxel size. The smaller error for larger voxel size for such noise distribution can be explained by the nature of the TSDF grid representation. Finally, (4) shows reconstructed scene without prior SLAM map and significant pose drift occurs during the reconstruction process.

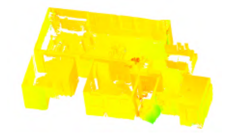
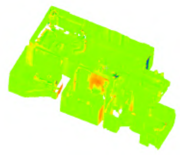
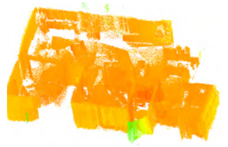
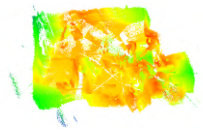
Nr.	Mean [m]	Std [m]	TSDF size [MB]	Error Heatmap
1	0.024	0.0790	197.8	
2	0.0613	0.0573	8.5	
3	0.1056	0.1830	641.7	
4	0.2952	0.5637	698.8	

Table 1: Reconstruction errors on a synthetic Habitat scene with varying depth noise level. Green symbolizes low, while red represents high reconstruction errors. The color code is not the same across measurements, but within the single reconstruction.

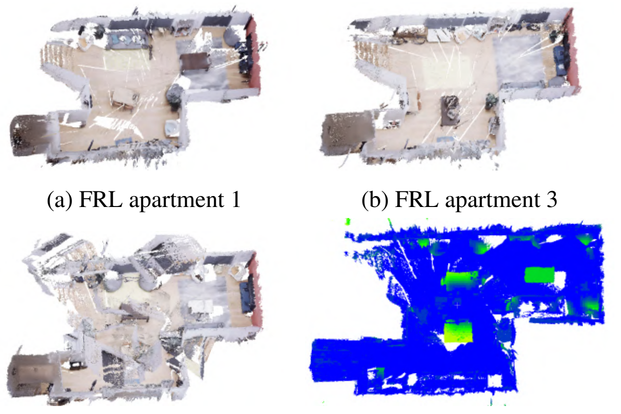
For evaluating the semantic accuracy of our proposed pipeline, we compare the reconstructed meshes with the ground truth for the closest voxel pair in Table 2. The rendered instance segmentation image has to be changed to class segmentation, where the classes are matched to the same color codes as in the previous implementation of SUNRGBD dataset with the Mask R-CNN model.

We illustrate the robustness of the localization node with our motion filtering component in Figure 8. It is able to localize in a rearranged room and reconstruct it again. We also showcase the possibility to detect and highlight furniture layout changes by simply comparing reconstructions of the same scene at different times.

We can conclude that our pipeline is able to reconstruct the scene robustly and accurately in close to real time, while keeping a modular approach so that we can outsource certain tasks to a remote PC.

Class Name	Precision	Recall	F1-Score
ceiling	22.8%	98.1%	0.371
chair	95.1%	68.3%	0.758
floor	93.6%	86.4%	0.899
furniture	35.5%	74.4 %	0.481
objects	19.3%	68.7 %	0.301
picture	2.7%	17.2%	0.047
sofa	61.2%	67.2%	0.640
table	31.7%	55.0%	0.402
wall	97.0%	40.3%	0.570
window	12.2%	72.3%	0.208

Table 2: Metrics evaluating the voxel-level predicted class labels compared to the ground truth labels.

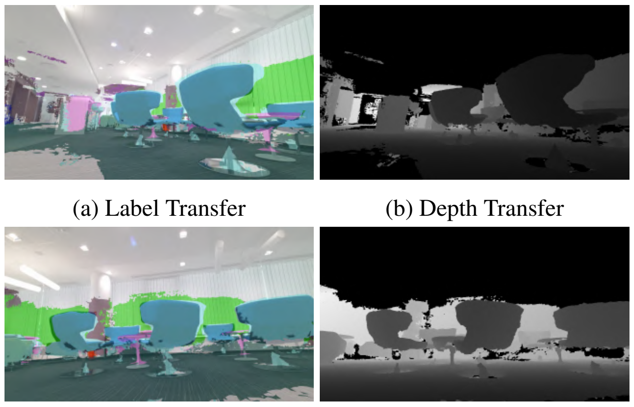

(a) FRL apartment 1 (b) FRL apartment 3

(c) No pose filtering (d) Change detection

Figure 8: Reconstructing the apartment when only a SLAM map and mesh from a previous furniture layout is available. (a)-(b) show previous and current furniture layouts. (c) shows the erroneous result when our pose filtering was off. (d) We can detect changes in the layout by comparing reconstructions at different times.

Object instance segmentation Figure 4 visualizes the output of our instance segmentation node, which generates instance-level bounding boxes for the class-based representation of a semantic mesh. The input mesh for the segmentation node was provided by the experiments conducted in the Habitat simulator with added synthetic noise.

Semantic label transfer After prior reconstruction from the human perspective, we can render semantic (and/or depth) images for arbitrary other perspectives and camera models. Figures 5 and 9 show transferred semantics and depth to the view of another agent with a different camera. The model was created by a Kinect for Azure RGBD camera, and in this example we localized a Zed mini stereo camera (the depth from stereo is not used). We see that the overlaid semantic labels match the image content to a large extent. For illustration, we also transferred the depth frames


(a) Label Transfer (b) Depth Transfer

(c) Label Transfer (d) Depth Transfer

Figure 9: Transferring semantics and depth from a model (same as Figure 3 (b)) generated with an RGBD camera to an agent localized with a stereo camera.

to the new viewpoint and we see that the objects are clearly recognizable. Most importantly, our depth transfer would allow depth perception even for a monocular agent.

5. Conclusions

We presented a modular pipeline for accurate semantic 3D reconstruction of indoor scenes with one or more agents. We also proposed an instance segmentation module that extracts object bounding boxes from semantic 3D maps. We showed how previous reconstructions of a scene can provide semantic information for robots who are unable to run onboard segmentation, who lack necessary sensors, or in cases where training data from the robot's viewpoint is not available. Our pipeline was tested in real scenes for qualitative and in photo-realistic simulated environments for quantitative evaluation. We showed that reconstruction is possible even in real time with detailed geometry and sufficiently accurate semantics.

We note that the same principles could apply to smartphone, smartglasses, and other agents too. The system would enable advanced semantic clues not only for simpler robots with monocular grayscale cameras, but even for 'blind' robots that navigate in the space with LiDAR or by other means once the coordinate systems are aligned, which is an interesting avenue for future work.

Acknowledgments

A. L. and Sz. Sz. were supported by the Hungarian Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Rick Szeliski. Building Rome in a day. *Communications of the ACM*, 54, 2011.

[2] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction, 2021.

[3] Campos, Carlos AND Elvira, Richard AND Gomez, Juan J. AND Montiel, Jose M. M. AND Tardos, Juan D. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *arXiv preprint arXiv:2007.11898*, 2020.

[4] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[6] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics*, 2017.

[7] Nils Funk, Juan Tarrio, Sotiris Papatheodorou, Marija Popović, Pablo F. Alcantarilla, and Stefan Leutenegger. Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning. *IEEE Robotics and Automation Letters*, 6(2), 2021.

[8] Stephan Gammeter, Alexander Gassmann, Lukas Bossard, Till Quack, and Luc Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 2010.

[9] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto. Volumetric instance-aware semantic mapping and 3D object discovery. *IEEE Robotics and Automation Letters*, 4(3), July 2019.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE International Conference on Computer Vision*, 2017.

[11] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. Deep learning based 3D segmentation: A survey. *CoRR*, abs/2103.05423, 2021.

[12] Yani Ioannou, Babak Taati, Robin Harrap, and Michael Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. In *Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, 2012.

[13] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, 2015.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 2017.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *European Conference on Computer Vision*, ECCV, 2014.

[16] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2021.

[17] J. McCormac, A. Handa, A. Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. *IEEE International Conference on Robotics and Automation*, 2017.

[18] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, ECCV, 2020.

[19] Shervin Minaee, Yuri Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

[20] Rafael Muñoz-Salinas and R Medina-Carnicer. UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers. *Pattern Recognition*, 101, 2020.

[21] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5), 2017.

[22] Mur-Artal, Raúl, Montiel, J. M. M. and Tardós, Juan D. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5), 2015.

[23] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3D scene reconstruction from posed images. In *European Conference on Computer Vision*, ECCV, 2020.

[24] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision*, ECCV, 2012.

[25] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[26] Chuong V. Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3D reconstruction and tracking. In *Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, 2012.

[27] D. Nguyen, Binh-Son Hua, Lap-Fai Yu, and S. Yeung. A robust 3D-2D interactive tool for scene segmentation and an-

notation. *IEEE Transactions on Visualization and Computer Graphics*, 24, 2018.

[28] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32, 11 2013.

[29] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS, 2017.

[30] Florian Reichl, J. Weiss, and R. Westermann. Memory-efficient interactive online reconstruction from depth image streams. *Computer Graphics Forum*, 35, 2016.

[31] Francisco Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76, 06 2018.

[32] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation*, ICRA, 2020.

[33] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *IEEE/CVF International Conference on Computer Vision*, ICCV, 2019.

[34] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

[35] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR, June 2019.

[36] Daniel Seichter, Mona Köhler, Benjamin Lewandowski, Tim Wengefeld, and Horst-Michael Gross. Efficient RGB-D semantic segmentation for indoor scene analysis. *arXiv preprint arXiv:2011.06961*, 2020.

[37] S. Song, S. P. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2015.

[38] Julian Straub, Thomas Whelan, and Lingni Ma et al. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[39] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *Conference on Computer Vision and Pattern Recognition*, 2021.

[40] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14), 2016.

[41] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3D to 2D label transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.