Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Master Thesis

Learning Semantic Map Refinement from Novel-View Synthesis

Author:
Jan Nogga

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Dr. Jens Behley

Advisor:
Radu Alexandru Rosu

Date: February 25, 2023

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

Semantic mapping yields a persistent model of the environment facilitating any downstream task associated with purposeful environment interaction. Such models are typically obtained by fusing measurements from several sensor modalities into a volumetric structure and filtering datapoints assigned to a common voxel. However, the results are afflicted by artifacts caused by inaccurate sensor pose, measurement noise, discretization errors, object mislabeling or regions of the scene occluded in the input data. In this work, we propose a scheme for refinement of 3D semantic maps based on the proxy task of novel view synthesis using differentiable volume rendering. Furthermore, we introduce a flexible and user-friendly tool built on sl-cutscenes to generate realistic bin picking scenes, which is used as a testbed for this work. We demonstrate that our system is capable of leveraging a learned prior to refine and improve unseen semantic maps over a wide range of scenes with various degrees of sensor noise.

Acknowledgments

First of all, I want to thank my supervisor Alex for helping me navigate the intricacies of volumetric rendering. Without your expertise, I would certainly have taken a wrong turn at one of countless design decisions on the road to completing this thesis. Also, thank you Alex for providing invaluable input to the writing process! Furthermore I would like to thank my friend and former supervisor Peer for teaching me how to generate realistic synthetic datasets. In addition, my father Andreas for proofreading this thesis and, together with my mother Astrid, for their unwavering support throughout my (admittedly somewhat longer) studies. Moreover, my grandparents Hanne, Anita, Lothar and Gerd for representing an infinite source of encouragement all this time. Finally, I want to extend my gratitude to Professor Behnke, who has provided me with a wonderful learning environment for many years and was instrumental in helping me find a thesis topic which combined my goals to familiarize myself with volumetric rendering and simulation of dynamic scenes.

Contents

1 Introduction **1**
- 1.1 Semantic Mapping . . . 1
- 1.2 Renderable Semantic Maps . . . 2
- 1.3 Our Proposed Approach . . . 2
- 1.4 Contributions . . . 3

2 Related Work **5**
- 2.1 Semantic Mapping . . . 5
  - 2.1.1 Semantic Mapping Using Object-Class Segmentation of RGB-D Images . . . 5
  - 2.1.2 Real-Time Multi-Modal Semantic Fusion on Unmanned Aerial Vehicles . . . 6
  - 2.1.3 Recurrent-OctoMap . . . 6
- 2.2 Semantic Scene Segmentation . . . 6
  - 2.2.1 3DMV . . . 7
  - 2.2.2 Bidirectional Projection Network . . . 7
- 2.3 Novel View Synthesis . . . 8
  - 2.3.1 NeRF . . . 8
  - 2.3.2 Semantic NeRF . . . 8
  - 2.3.3 Direct Voxel Grid Optimization . . . 9
  - 2.3.4 Plenoxels . . . 9
  - 2.3.5 Light Field Networks . . . 9
  - 2.3.6 NeSF . . . 10

3 Method **11**
- 3.1 Semantic Mapping as Voxel Grid Fusion . . . 11
- 3.2 Classic Volume Rendering . . . 12
- 3.3 Differentiable Semantic Volumetric Renderer . . . 13
- 3.4 Semantic Map Refinement Model . . . 15

4 Dataset Generation **17**
- 4.1 Binpicking Scenes . . . 17
- 4.2 Synthetic Cluttered Binpicking Scenes . . . 18

4.3 Data Augmentation for Annotated Point Clouds 19
4.3.1 Data Augmentation for Depth Measurements 19
4.3.2 Data Augmentation for Semantic Segmentation 22
4.3.3 Data Augmentation for Camera Parameters 25
4.4 Labeled RGB-D Data as Annotated Point Clouds 25

5 Experiments **27**
5.1 Experiment Configuration 27
5.1.1 Precomputed Annotated Point Clouds 27
5.2 Training 28
5.3 Evaluation 29
5.4 Implementation Details 31

6 Results **33**
6.1 Quantitative Results 33
6.1.1 Novel-View Synthesis 33
6.1.2 Backprojection to Sensor Frames 35
6.1.3 Disambiguation of Fusion Steps 37
6.2 Qualitative Results 41
6.3 Performance 47

7 Conclusion and Outlook **49**

1 Introduction

1.1 Semantic Mapping

Traditionally, robots have relied on maps to permit autonomous interaction with an environment in a goal-oriented fashion. These capabilities are supplemented by denoting affiliation to known object classes for entities in the map, giving rise to a semantic map.

"A semantic map for a mobile robot is a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes." - Andreas Nüchter [1]

A common type of semantic map is stored in 3D voxel grids (Stückler et al., Bultmann et al. [2, 3]). The map can be populated by probabilistically fusing information from multiple sensor measurements or multiple sensor modalities, such as annotated RGB-D images or segmented 3D LiDAR scans into these voxels. Maintaining such a semantic map yields a persistent representation of all sensor data which has been collected so far. This is very desirable in robotics applications for a multitude of reasons. In navigation and localization, moving objects can be respected in path planning, and the robustness of the robot localization increases given knowledge about which map entries are potentially dynamic. Furthermore, interaction in the form of purposefully manipulating the environment is expedited by a semantic map. Importantly, visualizing a semantic map allows humans to appraise what an agent knows about the environment, which provides insights into its decisions and is crucial for their interpretability.

On the flip side, there is ample potential for defects which can arise due to flawed sensor pose estimates, sensor noise, discretization errors, inaccurate semantic labels or occluded volumes. Here, it is difficult to attribute mapping artifacts to specific root causes, so an explicit correction is often unfeasible. Unfortunately, traditional machine learning techniques which could address this problem are contingent on ground truth labeled data, which is expensive to obtain densely in 3D. Thus, learned refinement of semantic maps is frequently not possible, either.

1.2 Renderable Semantic Maps

Recent advances in neural rendering, particularly the advent of techniques for novel-view synthesis involving explicit representations of radiance fields used in conjunction with a differentiable rendering scheme (Yu et al., C. Sun et al. [4, 5]) offer exciting new possibilities in the context of semantic map refinement; if we can interpret a semantic map not only as a persistent representation of sensor data, but rather as an explicit form of a plenoptic function, we are able to apply aforementioned methods to create a learning framework supervised only with 2D images wherein novel-view synthesis serves as a proxy task targeting semantic map refinement. The necessity of densely labeled 3D data could then be relaxed in favor of densely annotated 2D data, which is substantially more straightforward to obtain. This idea is not far-fetched, since the semantic label observed at a certain point should be constant for all observable rays which terminate there. Thus, a minimal bridge between 3D semantic map and a basic radiance field describing such a Lambertian scene is built by infusing the semantic map with a notion of density.

1.3 Our Proposed Approach

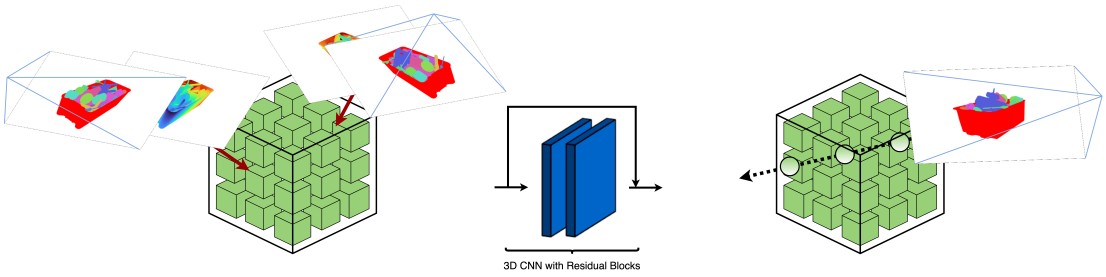

Figure 1.1: An overview of the core idea. *Left*: Fuse RGB-D measurements in a voxel grid. *Center*: Refine the grid with a 3D CNN. *Right*: Render 2D views from it and backprop rendering loss to 3D CNN.

We propose to explore the concept outlined throughout the previous section in manner illustrated in Figure 1.1. A semantic map represented by a 3D voxel grid is created by fusing one or several RGB-D sensor measurements, where the color channels are superseded by semantic labels. Jointly with the fusion process, density emerges proportionally to a voxels hit counter and, in conjunction with the map, an explicit radiance field is formed. Then, a simple neural refinement model is applied to the incomplete or flawed radiance field, producing a corrected version which is then differentiably rendered to multiple novel sensor views. In

its drive to improve upon these synthesized novel semantic views, encouraged by following the gradient of the error between rendered and ground truth sensor data, the refinement model must complete the given semantic map and correct artifacts appropriately. Thereby, semantic map refinement is reformulated in terms of novel-view synthesis, and the proxy task put forward above is implemented. Note that both the refined as well as the input radiance field remain semantic maps in their own right, with density corresponding to occupancy and the voxel entries representing class distributions. In particular, the unrefined map can also be rendered and constitutes a persistent representation of the fused sensor data.

1.4 Contributions

1. A framework for GPU-accelerated and batch-capable fusion of semantic point clouds obtained from RGB-D sensors with known pose.

2. A volume renderer suited for rendering the semantic information fused in the map.

3. A fast refinement module that operates on the 3D semantic map and is trained only with the task of novel view synthesis.

4. A large synthetic dataset of binpicking scenes with realistic sensor artifacts and semantic predictions errors.

2 Related Work

In this chapter we provide an overview of methods which relate to one or multiple aspects of our proposal. Since we draw on techniques from mapping, 3D semantic segmentation and novel-view synthesis, this chapter is subdivided accordingly.

2.1 Semantic Mapping

In robotics applications there are several ways to represent a 3D map such as surfels, point clouds or meshes. In this work, we focus on maps which are represented as voxel grids, because their structure is efficiently processed in parallel with convolutional networks and allows representation of continuous functions via interpolation. This section illustrates techniques to construct and maintain voxel maps containing semantic labels.

2.1.1 Semantic Mapping Using Object-Class Segmentation of RGB-D Images

Stückler et al. [2] propose a Bayesian 3D semantic fusion which composes a voxel-based map from RGB-D images. Random decision forests process the RGB channels to produce per-pixel class probabilities which can be projected to an allocentric coordinate frame using the depth measurements and their registration in the context of SLAM. The resulting grid contains a discrete distribution over class labels at each voxel, which can be back-projected to the original camera frames to improve their accuracy. In the sense that back-projection can be considered a special case of novel-view synthesis, the input/output behavior of this method is very similar to our proposed approach. However, once mapped, there is no mechanism to account for 3D priors like plausible object arrangements or shapes to correct the scene geometry.

2.1.2 Real-Time Multi-Modal Semantic Fusion on Unmanned Aerial Vehicles

To allow for building a comprehensive allocentric map onboard a UAV, Bultmann et al. [3] combine RGB-D images segmented using lightweight CNNs, person detections based on semantic information extracted from thermal camera data and segmented LiDAR scans. The RGB-D and thermal data is converted to annotated point clouds, which are probabilistically combined with the LiDAR scans in a voxel grid, demonstrating 3D semantic fusion across multiple sensor modalities. The resulting map improves the UAV localization system by discounting map entries associated with dynamic classes and facilitates scene understanding by denoting labels such as people, different vehicles, buildings or trees. Additionally, the semantic segmentation of the LiDAR scans can be retrained. This takes into account pseudo-labels sourced from the contributions of the camera to the semantic map. The performance of the LiDAR scan segmentation component is increased in this semi-supervised manner. Moreover, the importance of numerical stability in the fusion step is highlighted. To prevent the collapse of map entries to discrete delta distributions, the Bayesian filter is implemented in log space, and the normalization step is additionally improved using a stabilized log-sum-exp trick.

2.1.3 Recurrent-OctoMap

Instead of updating the voxel state explicitly, L. Sun et al. [6] store an LSTM state summarizing prior observations along with a high dimensional hidden state encoding the class probabilities in each grid cell. When new observations represented by a point cloud annotated with semantic features fall into the vicinity of a voxel, the features are pooled and mapped together with the local voxel state by an LSTM with global weights to the novel voxel state. To convert the hidden state to logits per object class, it must be interpreted by a further network and finally normalized via softmax to yield a probability distribution over class labels. This approach relaxes the Markov assumption required to justify other Bayesian frameworks to improve the maps robustness against incorrect measurements in the fusion step.

2.2 Semantic Scene Segmentation

In this section we present a family of methods which operate directly in 3D space instead of fully relying on 2D images. To highlight the importance of taking into account geometric features alongside appearance to decide a voxels class label, we

focus on processes which can jointly utilize 2D and 3D data. In that sense, rather than being treated primarily as a byproduct of fusion, 3D geometry plays a crucial role in the following methods.

2.2.1 3DMV

3DMV(Dai et al. [7]) represents the 3D map as an occupancy grid together with a series of posed images. They propose a joint 2D-3D approach that combines features from both 2D images and 3D in an end-to-end manner. To extract 3D geometry features, a series of 3D convolutions is used. It is strided over windows bounded in the horizontal xy plane of a voxel map obtained from RGB-D scans, thus splitting it into overlapping chunks for processing. In the map column that the network is currently anchored to, these local features are mapped to class probabilities per voxel. At the same time, 2D features are obtained by selecting RGB frames which overlap the respective map chunk and encoding them using a 2D CNN. These features are fused into the map using a differentiable neural backprojection layer. There, they are merged using per-voxel max pooling and further 3D convolutions to supplement the 3D geometry features. To aid the 2D CNN in extracting features meaningful for semantic segmentation, the authors employ a proxy loss which maps these features to class scores and then compares these with ground truth segmentation masks. Thus, 3D and 2D ground truth is required to train this system, which learns to predict and refine scene labels, but not geometry. At the same time, the backprojection layer would not be suitable for novel-view synthesis. Nevertheless, *3DMV* illustrates a mechanism to and highlights the potential of taking into account 2D data when annotating 3D maps.

2.2.2 Bidirectional Projection Network

By jointly segmenting 3D scenes and multiple 2D views in structurally symmetrical 2D and 3D U-Nets, Hu et al. [8] capitalize on the high-resolution 2D textural information in 2D images and the 3D geometric information presented by voxelized point clouds by treating these modalities as complementary. At each level of the U-Net decoders, a *bidirectional projection module* connects the corresponding 2D and 3D feature maps. It is realized by a *link matrix* using the sensor parameters to associate eligible voxels and pixels, transporting 2D feature maps into the 3D map and 3D features into the 2D masks. Similar to 3DMV, 2D semantic segmentation ground truth is successfully exploited to enable learned 3D semantic segmentation. However, this approach also relies on labeled 3D data.

2.3 Novel View Synthesis

Another recent family of methods for representing 3D scenes was presented by Mildenhall et al. [9], proposing to represent a 3D scene as a radiance-density field encoded in an neural network. These methods are characterized by their ability to learn 3D geometry from 2D supervision. In this section, we provide some context regarding the evolution of such methods from implicit to explicit representations as well as their ability to target 3D priors and semantic labels.

2.3.1 NeRF

Mildenhall et al. [9] represent the continuous radiance field of a scene using an MLP, called *neural radiance field* (*NeRF*) which maps the 3D coordinate location to a density σ and the 5D combination of coordinate location and viewing direction to a color value c. Rays are cast into the scene and evaluated at multiple sample points by querying the NeRF network given the sample position and ray direction. Using a quadrature of the classic volume rendering equations, the densities and color values along the ray are mapped to an expected color value. This result can be compared via squared error with ground truth images of the scene, a loss which is backpropagated to correct the outputs of the MLP. Since the density is nescient to the camera ray viewing direction, the trained neural scene representation is inherently multi-view consistent and can render novel views into the scene. The network is supported by concatenating a positional encoding similar to those used in language models to the input coordinate locations, which helps resolve high frequency spatial detail.

2.3.2 Semantic NeRF

Semantic NeRF (Zhi et al. [10]) builds on NeRF to encode additional semantic channels alongside geometry and appearance. Since semantics are independent of the viewing direction, they are represented by an additional classification head attached to the NeRF network which interprets the features that also give rise to the density channel. Since semantic NeRF is capable of learning even when only a subset of the RGB training data is annotated and can subsequently render to the previously unlabeled views, the method can implicitly perform key-framing. Moreover, the training process is robust to noisy semantic labels, which shows that the priors learned by the network are capable of denoising semantics in a 3D fusion process.

2.3.3 Direct Voxel Grid Optimization

By storing the density directly in a trainable voxel grid, C. Sun et al. [5] propose a hybrid form of implicit and explicit coordinate-based neural rendering schemes. The implicit color component is also lightweight compared to the aforementioned NeRF, consisting of another voxel grid which stores abstract feature vectors. These are interpolated for given 3D coordinate, and interpreted by a color MLP together with the coordinate and a viewing direction. Overall, limiting the reliance on implicit scene representations vastly improves the convergence time while securing high-quality novel-view synthesis.

2.3.4 Plenoxels

Yu et al. [4] demonstrate that neural components representing the or parts of the plenoptic function are not necessary to produce realistic novel-view synthesis. Their system uses *plenoptic voxels*, a sparse grid structure which stores a density channel and color weights in each entry. To retrieve color values, the interpolated color weights at a location x are used in a weighted sum of spherical harmonics evaluated at the desired viewing direction. Density is directly interpolated for points on a ray, and used in conjunction with the color values in the same volume rendering method used by (Mildenhall et al. [9]). This highlights the importance of a differentiable volumetric renderer rather than intricate neural networks when capturing 3D scene representations.

2.3.5 Light Field Networks

So far, this section has covered the spectrum from implicit to explicit coordinate-based learned radiance fields latching onto a differentiable volume rendering formula. Sitzmann et al. [11] propose to skip this step and estimate the color along a ray in a single inference step from an MLP which interprets a 6D Plücker Vector representation of the ray. This MLP represents a *light field*, which describes the flow of light in a static scene with fixed illumination. Untethered from the beneficial constraints of classical volume rendering, the light field network cannot be expected to produce multi-view consistent outputs of its own accord. To address this, a meta-learning scheme is leveraged, which employs a hypernetwork to output the weights parameterizing a light field network given a latent code which uniquely encodes a given scene. The lightfield network then renders the training views, and the resulting losses are used to optimize the hypernetwork. Presented with a large amount of scenes, the hypernetwork is expected to capture a prior over multi-view consistent light fields. When presented with a novel scene at test time,

the hypernetwork is frozen and the error signal with respect to a single view of the scene is passed to the latent code instead, thereby localizing the scene on a manifold of multi-view consistent light fields. While the datasets trialed by Sitzmann et al. [11] are large and unrealistic, and the performance of the method is shown to break down for smaller numbers of training scenes, the results exhibit that it is possible to learn priors over plausible plenoptic functions, which encourages our attempt to learn priors over reliable semantic maps.

2.3.6 NeSF

Priors over plenoptic functions are also accessible in a setup based on classic volume rendering and can also address semantic channels. This is shown by Vora et al. [12]. To train the proposed system, they iterate over multiple scenes viewed from RGB cameras with available class labels, training a NeRF on each of them. The NeRF density component is then sampled to obtain a grid estimating scene density. This density grid is consecutively converted by a 3D U-Net to a grid which stores a feature vector per voxel. At a location x, the feature grid yields an interpolated feature vector which is mapped by a scene-independent MLP to class score vector $s(x)$. Together with interpolated queries into the density grid, classic volume rendering is used similarly to semantic NeRF, producing class labels. The combination of 3D U-Net and semantic feature MLP trained in this manner is able to generalize to processing NeRFs trained on novel scenes of similar nature, producing 3D semantic maps which can be rendered to 2D semantic labels, on top of the 3D geometry provided by the input NeRF. The factor limiting practical application is the necessity of a pretrained NeRF, which requires capturing novel scenes from several points of view for the time-consuming NeRF training process. We intend to improve upon this aspect and increase interpretability by providing an input semantic map obtained by fusing multiple RGB-D measurements and utilizing only explicit representations of the density and semantic fields. On the flip side, we require depth measurements for the fusion step, while 3D geometry is extracted from appearance in the context of NeSF.

3 Method

To recapitulate, our proposed method takes as input RGB-D measurements with semantic labels, then

- fuses them into a 3D voxel grid,
- refines the grid
- and renders from it towards 2D views.

Next, we detail each of the components of our system.

3.1 Semantic Mapping as Voxel Grid Fusion

In the context of this thesis, we assume that RGB-D data with semantic labels is available from a sensor j for a given scene i. In the following, we treat this data as annotated point clouds ${}^{i}_{j}\mathcal{X}$ produced by projecting the semantic segmentation using the corresponding depth image and sensor parameters. This data is independent of its original sensor coordinate frame and thus readily combined over multiple measurements in a common semantic map using a voxel grid ${}^{i}\mathcal{V}$ with a probabilistic fusion scheme as described by McCormac et al. [13]. As suggested by Bultmann et al. [3], we utilize a logarithmic formulation to prevent numerical instability which can occur when the prior and a new measurement are contradictory but individually display a high confidence. Specifically, for the semantic channels $V_{\mathcal{Y}}$ of a voxel $V \in {}^{i}\mathcal{V}$ we apply

$$V_{\mathcal{Y}} \leftarrow V_{\mathcal{Y}} + \sum_{l \in {}^{i}_{j}\mathcal{X}_{V}} l \tag{3.1}$$

for ${}^{i}_{j}\mathcal{X}_{V}$, the set of object class distributions in the vicinity of V and consecutively normalize using a numerically stabilized logarithm of summed exponentials

$$V_{\mathcal{Y}} \leftarrow V_{\mathcal{Y}} - \left(\max_{c} V_{\mathcal{Y}}[c] + \log \left(1 + \sum_{c \neq \operatorname*{argmax}_{c} V_{\mathcal{Y}}[c]} e^{V_{\mathcal{Y}}[c] - \max_{c} V_{\mathcal{Y}}[c]} \right) \right). \tag{3.2}$$

This yields a voxel grid in which each volume element stores a log probability per class label. Upon initialization, these are considered to be uniformly distributed.

Since we intend to use $\mathcal{V}$ in a volume rendering setting, we incorporate a notion of density by appending a channel V_μ which is updated according to

$$V_\mu \leftarrow V_\mu + F_{\text{volume}} \cdot \left|{}^{i}_{j}\mathcal{X}_V\right| \tag{3.3}$$

in fusion steps. The *density factor* F_{volume} is a manually selected hyperparameter scaled according to the volume of each voxel, which depends on the world size and the grid resolution. A further channel $V_{N_{\text{hits}}}$ is implemented to ensure that the number of observations which have contributed to each voxel is retained. Note that $V_{N_{\text{hits}}}$ is redundant given F_{volume}, but helpful down the line when a refinement model applied to $\mathcal{V}$ may uncouple V_μ from it. Overall, this process manipulates a semantic map shaped $(N_{\text{classes}}+1+1)\times H_\mathcal{V}\times W_\mathcal{V}\times D_\mathcal{V}$. To summarize, $H_\mathcal{V}\times W_\mathcal{V}\times D_\mathcal{V}$ are the dimensions of the voxel grid and each voxel stores a vector with channels corresponding to the logarithm of an object label distribution, one density channel and one measurement hit counter channel.

3.2 Classic Volume Rendering

As stated in Section 2.3.1, techniques from classic volumetric rendering play a central role in current methods for novel view synthesis. This is vested in their beneficial properties; the gradients of the corresponding equations are trivially computed and information propagates well through space along rays, which are the focal objects in volumetric rendering scenarios. Specifically, a plenoptic function mapping a ray $r(t) = o + td$ passing through the scene and bounded by t_n, t_f to the expected color observed along that ray is given by the well-known volume rendering integral (Kajiya et al. [14])

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma\left(r(t)\right) c\left(r(t), d\right) dt \tag{3.4}$$

with

$$T(t) \coloneqq e^{-\int_{t_n}^{t} \sigma(r(s))ds}. \tag{3.5}$$

Here, the density $\sigma\left(r(t)\right) \in [0, \infty)$ can be understood as the differential probability of the ray terminating at a point $r(t)$ (Mildenhall et al. [9]). The transmittance $T(t) \in [0, 1]$ is thus the probability of the ray reaching $r(t)$ without terminating

beforehand. In particular, when the transmittance along a ray is 0, the ray passes only zero-density space, which we consider unoccupied. Thus, the pixel location corresponding to such a ray would be considered part of the background.

3.3 Differentiable Semantic Volumetric Renderer

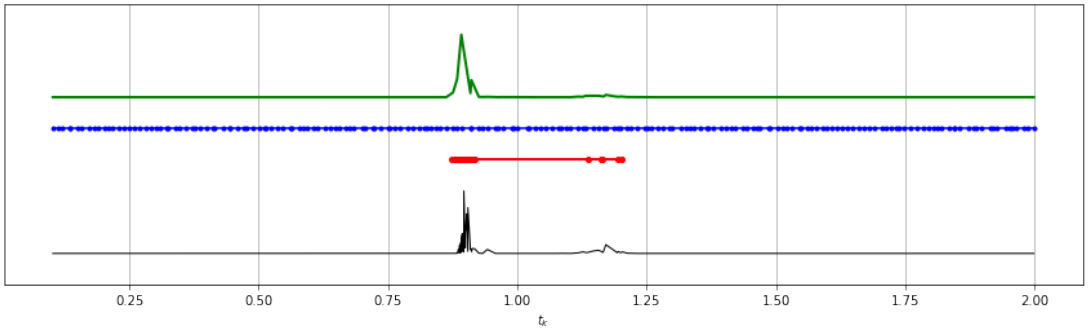

Figure 3.1: An example of hierarchical sampling of points along a ray. *Blue*: Stratified samples. *Green*: Weights calculated according to blue samples along ray cast through down-sampled density grid $\tilde{\mathcal{V}}_\mu$. *Red*: Points resampled using green weights as distribution. *Black*: Weights calculated using blue and red samples along ray in $\mathcal{V}_\mu$.

Our objective of learning a prior conducive to repairing common artefacts in voxel based semantic maps differs from the most closely related direct volume rendering approaches (Yu et al., C. Sun et al. [4, 5]) in the sense that we are necessarily concerned with more than a single scene. Training using only individual scenes one after the other is expected not only to result in unacceptably slow training times at scale due to inadequate GPU utilization, but also yield suboptimal estimates of the objective function gradient. For reference, Sitzmann et al. [11] employ batch sizes of up to 300 scenes to capture a prior over multi-view consistent plenoptic functions. Keeping this in mind, we anticipate that the ability to render batches of voxel grids to targets posed by batches of scenes containing dozens of sensors along a multitude of rays is crucial to succeed in producing a viable training pipeline. Thus, we implement a differentiable volumetric renderer by combining elementary concepts from Mildenhall et al. [9] with explicit components inspired by C. Sun et al. [5] while generalizing these approaches to multiple concurrent scenes.

In analogy to the previous section, we parameterize rays passing from the camera origin through a pixel location as $r = (r_{\text{origin}}, r_{\text{direction}})$ with

$$r_k \coloneqq r_{\text{origin}} + t_k r_{\text{direction}}. \tag{3.6}$$

Then, we calculate the raw class scores along a ray r using a numerical approximation of 3.4 (Max [15])

$$C_{\mathcal{Y}}(r) = \sum_{k=1}^{N} T_k \alpha_k \mathcal{V}_{\mathcal{Y}}^{\exp}(r_k) \tag{3.7}$$

where the transmittance is defined as

$$T_k \coloneqq e^{-\sum_{l=1}^{k-1} \mathcal{V}_\mu(r_l)\delta_l} \tag{3.8}$$

and the alpha values are given by

$$\alpha_k \coloneqq \left(1 - e^{-\mathcal{V}_\mu(r_k)\delta_k}\right). \tag{3.9}$$

Here, we use $\mathcal{V}_{\mathcal{Y}}^{\exp}$, the exponential of the grid semantic channels containing log probabilities because the query $\mathcal{V}_{\mathcal{Y}}^{\exp}(r_k)$ of the voxel grid at location k along ray r is implemented as trilinear interpolation. This is not an issue when querying the corresponding density $\mathcal{V}_\mu(r_k)$. Note that we consider $C_{\mathcal{Y}}(r)$ a vector of raw class scores because the weights $w_k \coloneqq T_k\alpha_k$ are not inherently normalized.

Just as in (Mildenhall et al. [9]), we sample

$$t_k \sim \mathcal{U}\left[t_n + \frac{k-1}{N_S}(t_f - t_n), t_n + \frac{k}{N_S}(t_f - t_n)\right] \tag{3.10}$$

where the near and far points t_n, t_f are hyperparameters of the renderer. In contrast to simply using the N_S bin centers placed equidistantly in $[t_n, t_f]$, sampling in this stratified manner treats the voxel grid representation of the scene as continuous. In our implementation, this sampling occurs individually for each ray.

In addition, we also follow the recommendation of Mildenhall et al. [9] and compute $\tilde{w}_k$ for these N_s sample points. Where (Mildenhall et al. [9]) employs a coarse network, we downsample $\mathcal{V}_\mu$ by a factor of 2 to obtain $\tilde{\mathcal{V}}_\mu$ and query it, yielding $\tilde{w}_k$, which is normalized and then treated as a distribution according to which N_H additional samples are drawn using inverse transform sampling. This ensures that the intervals along the ray which are expected to contribute most to the resulting class scores are queried densely while using a limited amount of samples. The samples referred to in Equation 3.7 resemble the union of these two sample sets, with $N = N_S + N_H$ and $\delta_k \coloneqq t_{k+1} - t_k$. An example of sample points and weights obtained using this process are pictured in Figure 3.1. Finally, we query $\mathcal{V}_\mu(r_k)$ before $\mathcal{V}_{\mathcal{Y}}^{\exp}(r_k)$ as probabilities $\left\{\mathcal{V}_{\mathcal{Y}}^{\exp} \middle| \mathcal{V}_\mu(r_k) = 0\right\}$ do not contribute to $C_{\mathcal{Y}}(r)$ and are safely skipped.

To access scene geometry by rendering, the distance of sample points t_k along

rays can be used in

$$C_{\mathcal{D}}(r) = \sum_{k=1}^{N} T_k \alpha_k t_k \tag{3.11}$$

to estimate the expected depth of the scene along a ray.

3.4 Semantic Map Refinement Model

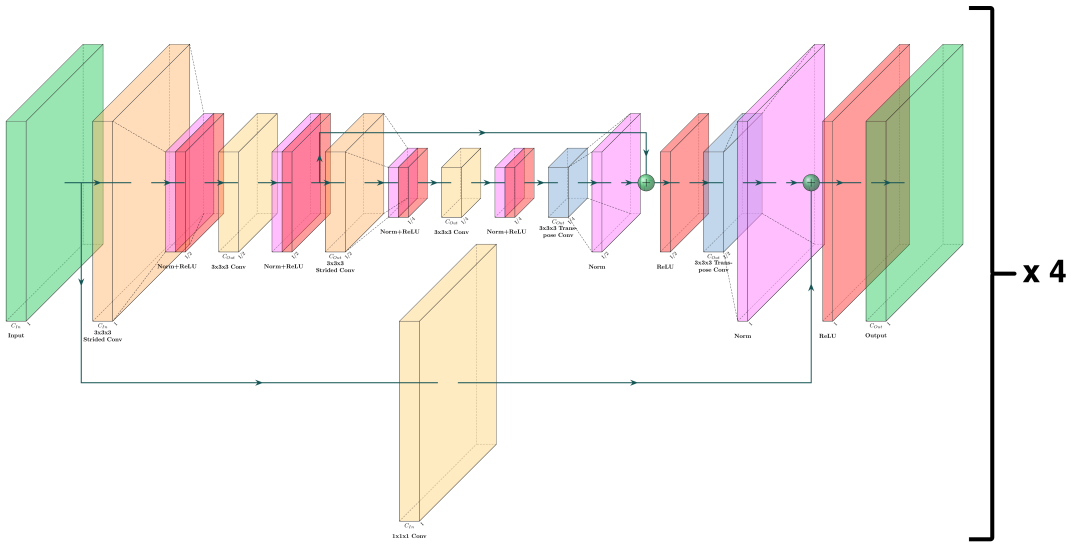

Figure 3.2: The semantic map refinement architecture consist of several modified stacked hourglass blocks (Chang et al. [16]) which are drop-in replacements for 3D CNN residual blocks. The layout of an individual block is depicted above. Our complete model consists of 4 stacked hourglass blocks. Diagram generated using [17].

When the rendering scheme from the previous section is applied to semantic voxel grids as described in Section 3.1, the resulting renderings are afflicted by missing class labels from incomplete data and other inaccuracies explained by the data augmentation applied before constructing the annotated point clouds fused to these maps. Our core idea can now be formulated concisely; we are looking to implement a neural refinement model $\mathcal{M}_{\mathcal{V}}$ which repairs a given semantic map via

$${}^{i}\mathcal{V} \mapsto \mathcal{M}_{\mathcal{V}}({}^{i}\mathcal{V}^{\exp}) =: {}^{i}\mathcal{V}_{\mathcal{M}}. \tag{3.12}$$

Just as for rendering, we process the class label distribution channels of the voxel grid as probabilities rather than log probabilities. Switching to this perspective

keeps the corresponding exponential out of the gradient graph at training time. Notwithstanding, an important objective is that the nature of the voxel grids is preserved by refinement, i.e. that ${}^{i}_{\mathcal{M}}\mathcal{V}$ entails density and semantic channels ${}^{i}_{\mathcal{M}}\mathcal{V}_{\mu}$ and ${}^{i}_{\mathcal{M}}\mathcal{V}_{\mathcal{Y}}$ which are suitable for use in a volume rendering setting as described in Equation 3.7. It stands to reason that a simple 3D CNN consisting of several residual blocks using $3 \times 3 \times 3$ convolution layers would be a good fit for this task. However, initial experiments seeking to overfit such a network to an individual scene demonstrated that the performance of this architecture is unlikely to meet our needs. This is plausibly due to the circumstance that the small kernels employed here struggle to interpret context information unless the network is unreasonably deep. To address this issue without increasing the network size, we borrow from (Chang et al. [16]) where *stacked hourglass* modules are used to replace and improve upon 3D residual CNNs in the context of disparity estimation. Such blocks as adjusted to our use case are shown in Figure 3.2. The expectation is that the sequence of encoder-decoder groups using strided and then transposed convolutions can ideally aggregate context information efficiently due to the low spatial resolution of the intermediate feature maps without requiring a deeper refinement network or larger kernels. At the same time, notions of spatial detail are passed forward as residuals at full grid and also at half feature map resolutions.

Notable differences with respect to the architecture showcased by Chang et al. [16] are that we do not use intermediate supervision to constrain the outputs in between stacked hourglass blocks because we do not employ a preceding series of feature extractors and because we want to omit the costly rendering operations required to provide these terms. Furthermore, we initialize the parameters of all transposed convolution blocks such that they represent upsampling via interpolation at the beginning of training (Odena et al. [18]).

Referring back to our goal of voxel grid interpretability after refinement, we use ReLU activation functions to constrain the density to its valid range $[0, \infty)$. For the resulting class channels, we are compelled to treat them as class scores in $[0, \infty)$ rather than probabilities, as their normalization could destabilize the training process. The most self-evident alternative non-linearity is a shifted softplus as proposed by Barron et al. [19], but C. Sun et al. [5] demonstrate that this option is most effectively applied after the tri-linear interpolation step, a principle coined *post-activation*. This would further loosen the ranges of values in our refined voxel grid, so we forgo applying post-activation in this thesis even though it is likely preferable in some settings, for example a single-scene semantic novel-view synthesis problem where the interpretability of the voxel grid is less consequential. Finally, the hit counter channel $V_{N_{\text{hits}}}$ can be read, but not written to by our model.

4 Dataset Generation

4.1 Binpicking Scenes

Throughout this thesis, we examine semantic maps corresponding to binpicking scenes, a scenario featuring tote boxes cluttered with a variety of objects. This application is particularly suitable because the associated tasks, such as picking up objects from the boxes, benefit from comprehensive knowledge about the scene, like the objects which are presented and the poses in which they lie in the tote. In this sense obtaining reliable semantic maps is a crucial step in scene understanding and any subsequent manipulation task. These problems are also of considerable practical importance, as evidenced for example by the Amazon Robotics Challenge 2017, which focused on identification and manipulation of warehouse goods. Moreover, this scenario is particularly suitable for our method, as the scenes are effectively constrained by the tote boxes. This allows representing semantic maps as dense voxel grids with high resolution, an assumption which otherwise requires finding the bounding box of occupied space in a preliminary step per scene (C. Sun et al. [5]). Furthermore, there is a limited variety of objects which can be present in such scenes, so strong scene priors are accessible through our learning scheme, enabling the refinement of the semantic maps. Finally, sensor poses in these scenarios are usually very precise, as the sensors are mounted at fixed locations or to robotic manipulators with accurate kinematics.

Similarly to other work concerned with scene and object priors (Sitzmann et al., Müller et al. [11, 20]), learning depends on large datasets specific to a particular context. Thus, we train on synthetic binpicking scenes. For this purpose, there is a preexisting dataset and dataset generator *SynPick* (Periyasamy et al. [21]), which uses the renderer *Stillleben* (Schwarz et al. [22]) and physics simulation NVIDIA PhysX[1] to create synthetic binpicking scenes. However, the objects in these scenes are dropped concurrently, which limits the amount and realism of clutter. To address this, we make use of *sl-cutscenes* (Boltres et al. [23]), a framework on top of *stillleben* designed to generate realistic indoor scenes. It features many

[1] `https://developer.nvidia.com/physx-sdk`

helpful tools, such as the ability to generate plausibly furnished rooms, specify and manipulate custom object sets and is flexible with regards to simulating multiple sensors within the scenes. We utilize these to implement a binpicking scenario generator which is inspired by *SynPick* and adapted to our needs.

4.2 Synthetic Cluttered Binpicking Scenes

(a) (b) (c) (d) (e) (f)

Figure 4.1: Examples from an annotated simulated scene created by our cluttered bin generator. **(a)**: RGB, **(b)**: depth, **(c)**: semantic segmentation, **(d)**: object poses, **(e)**: bounding boxes and **(f)**: bounding boxes shrunk to object visibility.

A simulated binpicking scene ${}^{i}\mathcal{S}$ with scene index i is initialized by setting up an empty tote box on a table located in the center of a furnished room. The

decoration in the room, including potential wall cabinets, shelving or chairs, the wall and floor textures as well as the table itself are sampled from a variety of options and arranged in a plausible fashion. Then, a random set of up to N_{objects} possible objects is sampled without replacement. The maximum total number of object classes is referred to as N_{classes}, including the object classes as well as the box class, but not the label for the scene background. With the PhysX simulation running, the objects are spawned and released one by one every second of simulated time with a random orientation at a random location in the region above the box. One simulated second after the final object is dropped, the scene is checked for objects which are still moving or are situated outside of the bounds of the tote. Such objects are removed and subsequently dropped again, a process which is repeated until each object is resting within the box.

Once the scene is set up in this manner, sensors ${}^i_j\mathcal{C} = ({}^i_jP, {}^i_jK)$ with sensor index j are placed at uniformly sampled positions on the upper hemisphere of an ellipsoid appropriately elongated to encompass the tote. They are oriented towards the center of the tote and then capture the scene. The terms ${}^i_jP \in \mathbb{R}^{4\times4}$ and ${}^i_jK \in \mathbb{R}^{3\times3}$ refer to the sensor pose and intrinsics; the latter is recovered from the projection matrix used by *stillleben*. The resulting synthetic measurements ${}^i_jz = ({}^i_j\mathcal{I}, {}^i_j\mathcal{Y}, {}^i_j\mathcal{D}, {}^i_j\mathcal{W})$ are comprised of an RGB image ${}^i_j\mathcal{I}$, a semantic segmentation mask ${}^i_j\mathcal{Y}$ denoting object, tote and background classes, a depth map ${}^i_j\mathcal{D}$ and a camera-specific surface normal map ${}^i_j\mathcal{W}$ represented by $|\cos(\theta_{\vec{n},\vec{v}})| = |\vec{n} \cdot \vec{v}|$. Here, $\theta_{\vec{n},\vec{v}}$ is the angle between surface normals $\vec{n}$ and sensor viewing direction $\vec{v}$.

Note that the decorated room in which the tote is located is masked when this data is utilized in the context of our work. While implemented for this thesis, the cluttered bin generator described above is designed for a wider range of applications. It is customizable with regards to object sampling set, sensor pose sampling techniques and output (meta-)data types, amongst others, and available as a self-contained package[2]. Figure 4.1 depicts examples of the available data annotation for a simulated scene.

4.3 Data Augmentation for Annotated Point Clouds

4.3.1 Data Augmentation for Depth Measurements

In practice, depth sensors suffer from artifacts such as measurement noise, limited sensor resolution and outright missing values. Examples of this are illustrated using an Intel RealSense [24] in Figure 4.2 for generic scenes, while Figure 4.3

[2] `https://git.ais.uni-bonn.de/nogga/simple-cluttered-bin-generator`

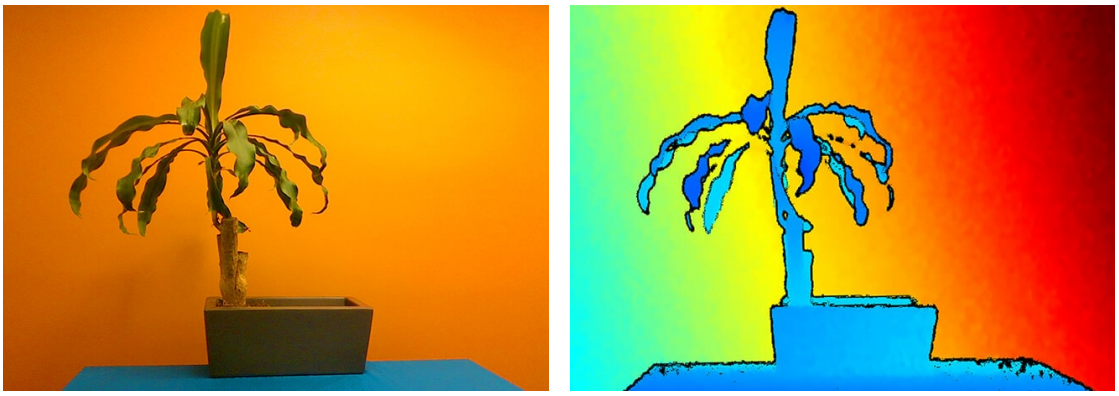

Figure 4.2: Measurements of a generic scene using an Intel RealSense L515 (from [24]). Missing depth measurements are visible on the tabletop and along edges. The background displays measurement noise and quantization due to limited sensor resolution.

compares measurements using the same sensor of a binpicking scene with its synthetic counterpart from a simulated scene. Of course, these defects are specific to the operating principle and model of the deployed depth sensor. For flexibility in this regard, we attempt to decompose multiple flavors of errors into their root causes to obtain a data augmentation framework that can represent a variety of common depth sensors when appropriately parametrized. This includes the quantization of depth values to N_{quant} bins partitioning the sensor measurement range $[R_{min}, R_{max}]$ to simulate limited sensor resolution and depth-dependent additive Gaussian noise $\sim \mathcal{N}(\mu = 0, \sigma = s_\sigma \cdot \sigma(\mathcal{D}))$, where

$$\sigma(\mathcal{D}) = (9\mathcal{D}^2 - 26.5\mathcal{D} + 20.237)10^{-3} \tag{4.1}$$

is noise typical for Kinect depth measurements (Mallick et al. [25]) and s_σ is a scaling factor used to control the severity of the resulting static. Furthermore, we model the probability of dropping a measurement due to the sensor limits as binary

$$p_{\text{range}}(\mathcal{D}) = \begin{cases} 1, & \mathcal{D} \in [R_{min}, R_{max}] \\ 0, & \text{otherwise} \end{cases}, \tag{4.2}$$

and the probability of dropping a measurement close to significant edges in the scene as

$$p_{\text{edge}}(\mathcal{D}) = f(||\nabla\mathcal{D}||_2) * g(\sigma_{\text{edge}}), \tag{4.3}$$

where f clips and normalizes the depth gradient norm image, which is then

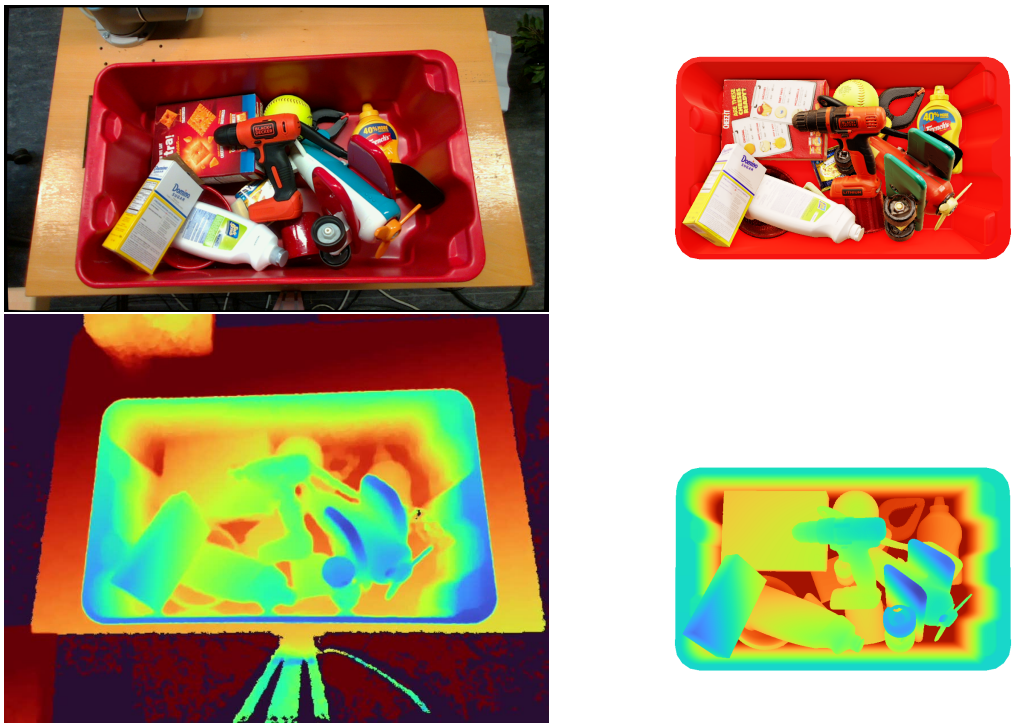

Figure 4.3: *Left*: A binpicking scene captured using an Intel RealSense L515. *Right*: Simulated measurements of a similar scene.

blurred via convolution with $g(\sigma_{\text{edge}})$, an 11×11 Gaussian filter. Finally, the probability of missing measurements due to small viewing angles is modeled by applying the Fresnel equations quantifying the reflection of light as a probability distribution

$$p_{\text{angle}}(\mathcal{W}) = \left(\frac{\left| \frac{n_1}{n_2}\mathcal{W} - \sqrt{1 - \left(\frac{n_1}{n_2}\sqrt{1-\mathcal{W}^2}\right)^2} \right|}{\left| \frac{n_1}{n_2}\mathcal{W} + \sqrt{1 - \left(\frac{n_1}{n_2}\sqrt{1-\mathcal{W}^2}\right)^2} \right|} \right)^2, \tag{4.4}$$

which can be tuned to a specific sensor using the correspondence

$$\theta_{\text{Total}} = \arcsin \frac{n_2}{n_1} \tag{4.5}$$

between the angle of total reflection θ_{Total} and the ratio of refractive indices $\frac{n_2}{n_1}$. Thus θ_{Total} is the parameter which determines the severity of this effect, as illustrated in Figure 4.4. Overall, the measurement dropout probability is calculated as

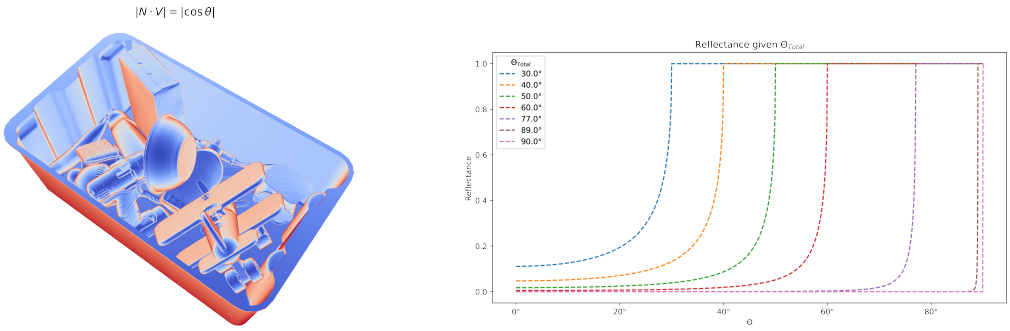

Figure 4.4: *Right*: Compact representation of the surface normal map relative to sensor viewing direction. *Left*: Distributions of measurement dropout probability at different viewing angles given the angle of total reflectance θ_{Total}.

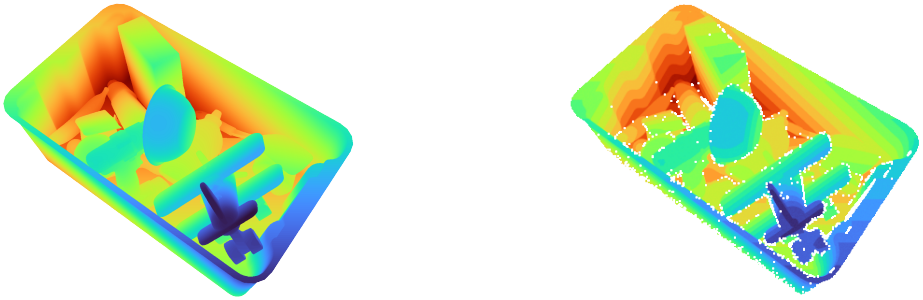

Figure 4.5: *Right*: Ground truth simulated depth measurement. *Left*: Depth map after applying data augmentation. White pixels represent missing measurements.

$$p_{\text{missing}}(\mathcal{D}, \mathcal{W}) = 1 - (1 - p_{\text{range}}(\mathcal{D}))(1 - p_{\text{edge}}(\mathcal{D}))(1 - p_{\text{angle}}(\mathcal{W})), \tag{4.6}$$

under the assumption that the aforementioned effects occur independently. Figure 4.5 illustrates the effects of applying the aforementioned effects to simulated depth measurements.

4.3.2 Data Augmentation for Semantic Segmentation

Naturally, we cannot expect the semantic segmentation labels $\mathcal{Y}$ to be flawless, either. To simulate realistic artifacts, we train a semantic segmentation model $\mathcal{M}_{\mathcal{Y}}$ on 8000 cluttered bin scenes viewed by 5 cameras each to classify an input image $\mathcal{I}$ in terms of the object and tote box classes in the foreground of the scene, producing N_{classes}-channel log probabilities per pixel location

$$\hat{\mathcal{Y}}_{\mathcal{M}}(\mathcal{I}) \coloneqq \mathrm{LogSoftmax}(\mathcal{M}_{\mathcal{Y}}(\mathcal{I})). \tag{4.7}$$

The dataset is generated for this purpose exclusively. Here, $\mathcal{M}_{\mathcal{Y}}$ is represented by a U-Net (Ronneberger et al. [26]) using a b1-sized feature extractor from SegFormer (Xie et al. [27]) which has been pretrained on ImageNet (Deng et al. [28]) but remains frozen throughout the training process on our cluttered bin data. This way, we achieve a good tradeoff between training time and segmentation performance. Sample predictions from the synthetic segmentation dataset validation split are depicted in Figure 4.6.

Ironically, an undesirable consequence of this approach is that the learned segmentation masks are of consistently good quality. For any data augmentation pipeline, control over the severity of applied effects is desirable. Thus, we experimented with dropout layers inserted between the decoder layers of $\mathcal{M}_{\mathcal{Y}}$. However, the dropout probability for low values has little effect on the output of the segmentation model, but for increasing probabilities the output segmentation masks suddenly deteriorate catastrophically. This proved too difficult to balance, so we introduce a probability p_{shuffle} to apply a random permutation to the semantic channels at each spatial location in $\hat{\mathcal{Y}}_{\mathcal{M}}$ instead. On the other hand, when we do not wish to apply any augmentation via $\mathcal{M}_{\mathcal{Y}}$ to $\mathcal{Y}$, we one-hot encode these class labels, yielding $\mathcal{Y}_{\delta}$ and then convert to ε-soft log probability distributions

$$\hat{\mathcal{Y}}_{\varepsilon}(\mathcal{Y}_{\delta}) \coloneqq \begin{cases} \log(1-(N_{\text{classes}}-1)\varepsilon), & \mathcal{Y}_{\delta}=1 \\ \log(\varepsilon) & \text{otherwise} \end{cases}. \tag{4.8}$$

Setting $\varepsilon \ll 1$, this hardly detracts from the ground truth semantic mask but ensures that the probabilistic filter used further along in our pipeline does not collapse for these inputs. The background class is omitted and its class vectors set arbitrarily in this step as those locations are excluded later.

Overall, the parameterized data augmentation pipeline introduced in this section can be summarized as

$${}^{i}_{j}z = ({}^{i}_{j}\mathcal{I}, {}^{i}_{j}\mathcal{Y}, {}^{i}_{j}\mathcal{D}, {}^{i}_{j}\mathcal{W}) \xmapsto{(N_{\text{quant}},R_{min},R_{max},s_{\sigma},\sigma_{\text{edge}},\theta_{\text{Total}},\mathcal{M}_{\mathcal{Y}},p_{\text{shuffle}})} ({}^{i}_{j}\hat{D}, {}^{i}_{j}\hat{\mathcal{Y}}) =: {}^{i}_{j}\hat{z} \tag{4.9}$$

with the understanding that any effect controlled by $(N_{\text{quant}}, \ldots, p_{\text{shuffle}})$ can be omitted if desired.

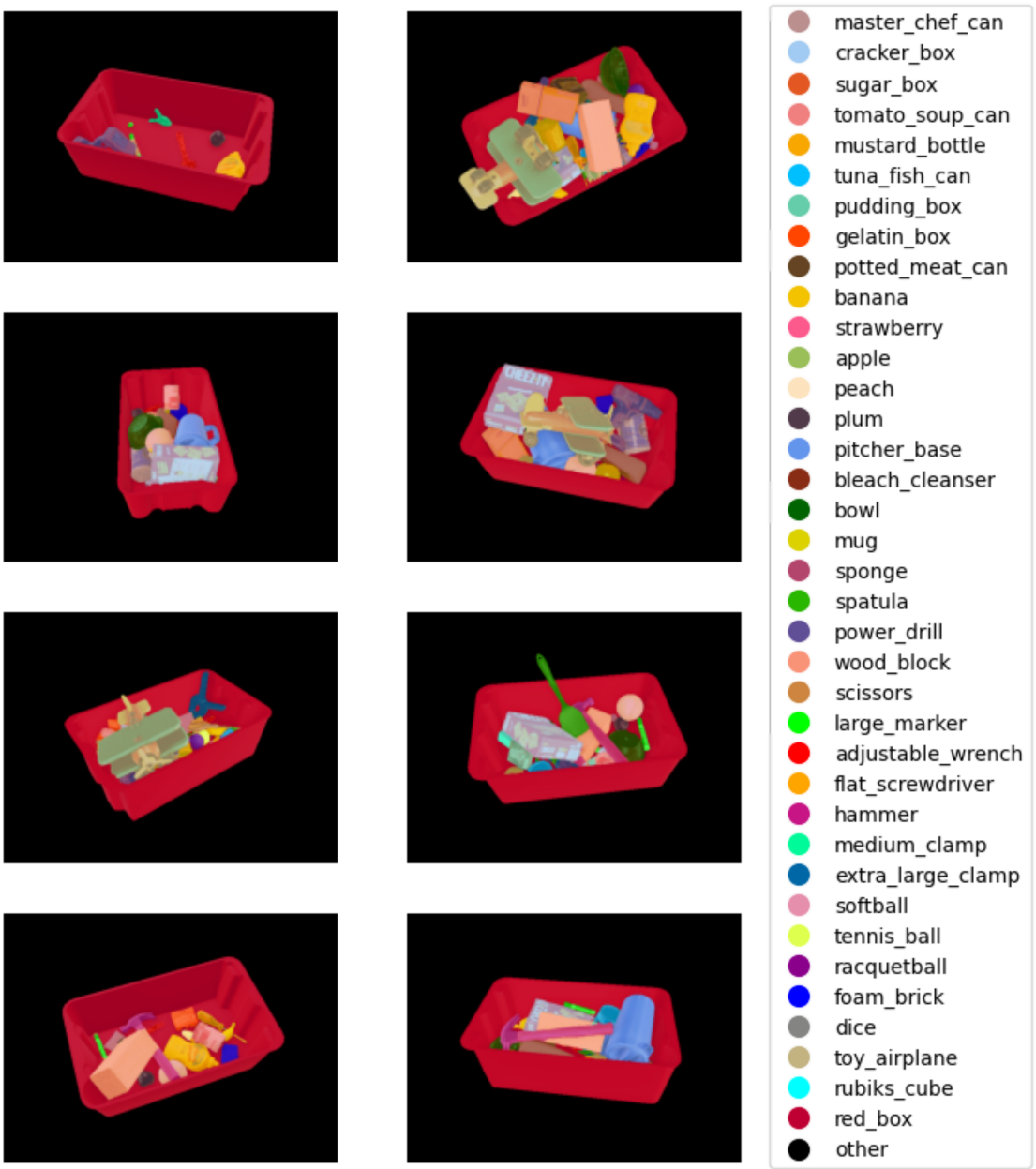

Figure 4.6: Example predictions of our semantic segmentation model $\mathcal{M}_{\mathcal{Y}}$ for validation data.

4.3.3 Data Augmentation for Camera Parameters

In addition to artifacts in the sensor measurement, inaccurate estimates of the sensor pose and intrinsic parameters can also influence the quality of semantic maps resulting from ${}^{i}_{j}\hat{z}$. Therefore, we also implemented the functionality

$$ {}^{i}_{j}\mathcal{C} = ({}^{i}_{j}P, {}^{i}_{j}K) \xmapsto{(\sigma_{\text{rot}},\sigma_{\text{pos}},\sigma_f,\sigma_c)} ({}^{i}_{j}\hat{P}, {}^{i}_{j}\hat{K}) =: {}^{i}_{j}\hat{\mathcal{C}}, \tag{4.10} $$

rotating the sensor orientation around the coordinate axes with random angles $\sim \mathcal{N}(0, \sigma_{\text{rot}})$ as well as incrementing the sensor position, camera focal length and principal point offset with noise $\sim \mathcal{N}(0, \sigma_{\text{pos},f,c})$ respectively. Noise concerning ${}^{i}_{j}K$ is only sampled once every ${}^{i}\mathcal{S}$, reflecting the assumption that each scene is viewed by the same sensor in different poses.

4.4 Labeled RGB-D Data as Annotated Point Clouds

Before semantic mapping, it is necessary to convert the 2.5 dimensional depth maps with their corresponding semantic label distributions $({}^{i}_{j}\hat{D}, {}^{i}_{j}\hat{\mathcal{Y}})$ to an adequate 3D form in an allocentric reference frame for each pixel coordinate u, v. We apply

$$ \begin{pmatrix} {}^{i}_{j}x_{\text{World}}(u,v) \\ {}^{i}_{j}y_{\text{World}}(u,v) \\ {}^{i}_{j}z_{\text{World}}(u,v) \\ 1 \end{pmatrix} = {}^{i}_{j}\hat{D}[u,v]\, {}^{i}_{j}\hat{P} \begin{pmatrix} {}^{i}_{j}\hat{K}^{-1} & 0 \\ \vec{0} & 1 \end{pmatrix} \begin{pmatrix} u+0.5 \\ v+0.5 \\ 1 \\ \frac{1}{{}^{i}_{j}\hat{D}[u,v]} \end{pmatrix} \tag{4.11} $$

to obtain world coordinate locations from $\hat{C}$ and $\hat{D}$, then concatenate with $\hat{Y}$ and filter background pixels to obtain

$$ {}^{i}_{j}\mathcal{X} = \left\{ \begin{pmatrix} {}^{i}_{j}x_{\text{World}}(u,v) \\ {}^{i}_{j}y_{\text{World}}(u,v) \\ {}^{i}_{j}z_{\text{World}}(u,v) \end{pmatrix} \oplus {}^{i}_{j}\hat{\mathcal{Y}}[u,v] \,\middle|\, {}^{i}_{j}\mathcal{Y}[u,v] \neq c_{\text{Background}} \right\} \tag{4.12} $$

an ${}^{i}_{j}N_{\text{Foreground}} \times (3 + N_{\text{classes}})$ point cloud with semantic labels per pairing of ${}^{i}_{j}\hat{\mathcal{C}}$ and ${}^{i}_{j}\hat{z}$.

5 Experiments

5.1 Experiment Configuration

To train our refinement models, we use the cluttered bin generator described in 4.2 to generate 13400 binpicking scenes ${}^{i}\mathcal{S}$ using $N_{\text{objects}} = 38$ different object classes. In each scene, the tote contains between 8 and 32 objects which are viewed from 32 simulated sensors ${}^{i}_{j}\mathcal{C}$ at a resolution of 640×480. This dataset is split into 12800 scenes for training, 500 for validation and 100 for testing, referred to by indices $i \in \mathcal{S}_{\text{Train}}$, $\mathcal{S}_{\text{Val}}$, and $\mathcal{S}_{\text{Test}}$ respectively. At the same time, the sensors are split into 10 sensors for semantic fusion and 22 for novel-view synthesis. Conversely, these splits are referred to by indices $j \in {}^{i}\mathcal{C}_{\text{Map}}$ and $j \in {}^{i}\mathcal{C}_{\text{NVS}}$. This dataset using these splits gives rise to all experiments described in this section.

5.1.1 Precomputed Annotated Point Clouds

Precomputed Dataset Configurations *apc_*												
Param	N_{quant}	R_{min}	R_{max}	s_σ	σ_{edge}	θ_{Total}	$\mathcal{M}_\mathcal{Y}$	p_{shuffle}	σ_{rot}	σ_{pos}	σ_f	σ_c
original	✗	✗	✗	✗	✗	✗	✗	0%	0°	0°	0	0
moderate	150	$0.25m$	$9m$	1	5	86°	✓	0%	$\frac{1°}{3}$	$\frac{5mm}{3}$	0	0
heavy	75	$0.25m$	$9m$	2	8	77°	✓	25%	$\frac{4°}{3}$	$\frac{4cm}{3}$	$\frac{1}{3}$	$\frac{0.5}{3}$

Table 5.1: Overview of data augmentation configuration settings for our datasets.

Applying the augmentation scheme from 4.3 on the fly uses up significant GPU resources, as it must be applied 13300 times to 10 sensors per epoch. Therefore, we prebrake annotated point clouds using different settings to form the point cloud datasets for use during training. Specifically, we apply depth and semantic segmentation effects

$$ {}^{i}_{j}z \xmapsto{(N_{\text{quant}}, R_{min}, R_{max}, s_\sigma, \sigma_{\text{edge}}, \theta_{\text{Total}}, \mathcal{M}_\mathcal{Y}, p_{\text{shuffle}})} {}^{i}_{j}\hat{z} \tag{5.1} $$

and sensor parameter noise

$$ {}^{i}_{j}\mathcal{C} \xmapsto{(\sigma_{\text{rot}},\sigma_{\text{pos}},\sigma_f,\sigma_c)} {}^{i}_{j}\hat{\mathcal{C}} \tag{5.2} $$

and then calculate the annotated point clouds

$$ ({}^{i}_{j}\hat{z}, {}^{i}_{j}\hat{\mathcal{C}}) \mapsto {}^{i}_{j}\hat{\mathcal{X}} \tag{5.3} $$

for $j \in {}^{i}\mathcal{C}_{\text{Map}}$.

We also calculate uncorrupted point clouds ${}^{i}_{j}\mathcal{X}_{GT}$ for $j \in {}^{i}\mathcal{C}_{\text{Map}} \cup {}^{i}\mathcal{C}_{\text{NVS}}$, which are available downstream to compute pseudo-groundtruth semantic maps for visualization and evaluation purposes. Finally, we reserve ${}^{i}_{j}\mathcal{C}$ and ${}^{i}_{j}z$ for $j \in {}^{i}\mathcal{C}_{\text{NVS}}$ as training targets.

To showcase the interaction of our semantic map refinement with increasing degrees of sensor errors, we create the datasets *apc_original*, *apc_moderate* and *apc_heavy* as described above. Here *apc* abbreviates *annotated point cloud* and the suffix summarizes the severity of artifacts. The corresponding settings producing each dataset

$$ \left\{ \left(\{ {}^{i}_{j}\hat{\mathcal{X}} | j \in {}^{i}\mathcal{C}_{\text{Map}} \}, \{ ({}^{i}_{j}z, {}^{i}_{j}\mathcal{C}) | j \in {}^{i}\mathcal{C}_{\text{NVS}} \}, \{ {}^{i}_{j}\mathcal{X}_{GT} | j \in {}^{i}\mathcal{C}_{\text{Map}} \cup {}^{i}\mathcal{C}_{\text{NVS}} \} \right) \right\}_{i \in \mathcal{S}_{\text{Train}} \cup \mathcal{S}_{\text{Val}}} $$

are detailed in Table 5.1. It should be noted that the obvious drawback to precomputing corrupted annotated point clouds is that the diversity of sensor defects is thereby limited. We are nonetheless confident that 12800 scenes providing 10 separate point clouds provide a sufficient amount of training data while massively speeding up training epochs. Note that a random subset of these point clouds is combined in semantic mapping before refinement, contributing to map diversity per scene, further curtailing the negative impact of precomputation.

5.2 Training

With annotated point clouds, targets and refinement model at the ready, a training step falls into place as follows:

1. Select scene batch $\{i\} \subset \mathcal{S}_{\text{Train}}$, random subsets $\{j\} \subset {}^{\{i\}}\mathcal{C}_{\text{Map}}$, $1 \leq |j| \leq 8$
2. Fuse point clouds ${}^{\{i\}}_{\{j\}}\hat{\mathcal{X}}$ in voxel maps ${}^{\{i\}}\mathcal{V}$
3. Apply refinement model ${}^{\{i\}}\mathcal{V}_{\mathcal{M}} \leftarrow \mathcal{M}_{\mathcal{V}}({}^{\{i\}}\mathcal{V}^{\text{exp}})$

4. Sample rays $\{r\}$, N_{rays} each for all ${}^{\{i\}}_{j}\mathcal{C}$, $j \in {}^{\{i\}}\mathcal{C}_{\text{NVS}}$
5. Render $C_{\mathcal{Y}}(\{r\})$, $C_{\mathcal{D}}(\{r\})$, T_{t_f} using ${}^{\{i\}}\mathcal{V}_{\mathcal{M}}$
6. Compute loss L_{Train}, backprop to weights of $\mathcal{M}_{\mathcal{V}}$

Note that only a maximum of 8 out of 10 available annotated point clouds are used in step 1. This is because we want to examine how the network responds to 9 or 10 fusion steps in evaluation, i.e. whether it generalizes to more comprehensively mapped scenes.

In the loss term, there is an option to include a signal supervising the estimated depth. One ablation was conducted without depth supervision, which yielded equivalent semantic novel-view synthesis performance but about 50% increase in losses measuring depth estimation. Therefore, we use depth supervision in all other experiments, but it is not strictly required to do so. More precisely, the full training loss is

$$\begin{aligned} L_{\text{Train}} = 0.5 \cdot L_{\text{CE}}\left(C_{\mathcal{Y}}(\{r\}), \mathcal{Y}[u_r, v_r]\right) + \\ 0.5 \cdot L_{\text{LS}}\left(C_{\mathcal{Y}}(\{r\}), \mathcal{Y}[u_r, v_r]\right) + \\ L_{\text{L1}}\left(C_{\mathcal{D}}(\{r\}), \mathcal{D}[u_r, v_r]\right) + \\ 2 \cdot L_{\text{BCE}}(T_{t_f}, BG) + \\ 5\text{e}{-4} \cdot L_{\text{Reg}}(\mathcal{V}_{\mathcal{M}}, \mathcal{V}^{\text{exp}}) \end{aligned} \tag{5.4}$$

where L_{CE} is the cross entropy loss for multi-class segmentation. The Lovász-Softmax loss (Berman et al. [29]) L_{LS} is also responsible for segmentation quality and alleviates class imbalance and small object recovery errors. The term L_{L1} provides depth supervision and L_{BCE} treats the rendered transmittance as a foreground/background segmentation problem, encouraging correct labeling of empty pixels. Finally, L_{Reg} is another $L1$-term regularizing changes in density between $\mathcal{V}^{\text{exp}}$ and $\mathcal{V}_{\mathcal{M}}$.

5.3 Evaluation

In contrast to training, evaluation requires rendering full images, book-keeping the number of fusion steps contributing to voxel grids before refinement and also rendering from the unrefined as well as the pseudo-ground-truth voxel grids to collect baseline results. Moreover, the quality of the input data must be measured as well.

Therefore, for each scene $i \in \mathcal{S}_{\text{Test}}$, we:

1. Fuse pseudo ground truth annotated point ${}^{i}\mathcal{X}_{GT}$ clouds to voxel grids ${}^{i}\mathcal{V}_{GT}$

2. Render ${}^{i}\mathcal{V}_{GT}$ to all known and novel views $({}^{i}\mathcal{Y}_{GT}, {}^{i}\mathcal{D}_{GT})$
3. Fuse $N = 1 \dots 10$ point clouds to increasingly dense semantic maps ${}^{i}_{N}\mathcal{V}$
4. Refine ${}^{i}_{N}\mathcal{V}$ individually to ${}^{i}_{N}\mathcal{V}_{\mathcal{M}}$
5. Render ${}^{i}_{N}\mathcal{V}$ to all sensors that contributed to them $\left({}_{MAP\leq N}{}^{i}\hat{\mathcal{Y}}^{N}, {}_{MAP\leq N}{}^{i}\hat{\mathcal{D}}^{N}\right)$
6. Render ${}^{i}_{N}\mathcal{V}$ to all novel views $\left({}_{NVS}{}^{i}\hat{\mathcal{Y}}^{N}, {}_{NVS}{}^{i}\hat{\mathcal{D}}^{N}\right)$
7. Render ${}^{i}_{N}\mathcal{V}_{\mathcal{M}}$ to all sensors that contributed to them $\left({}_{MAP\leq N}{}^{i}\hat{\mathcal{Y}}^{N}_{\mathcal{M}}, {}_{MAP\leq N}{}^{i}\hat{\mathcal{D}}^{N}_{\mathcal{M}}\right)$
8. Render ${}^{i}_{N}\mathcal{V}_{\mathcal{M}}$ to all novel views $\left({}_{NVS}{}^{i}\hat{\mathcal{Y}}^{N}_{\mathcal{M}}, {}_{NVS}{}^{i}\hat{\mathcal{D}}^{N}_{\mathcal{M}}\right)$
9. Calculate metrics ${}^{i}M_{GT}$ for $({}^{i}\mathcal{Y}_{GT}, {}^{i}\mathcal{D}_{GT})$
10. Calculate unrefined backprojection metrics ${}_{MAP}{}^{i}\hat{M}^{N}$ for $\left({}_{MAP\leq N}{}^{i}\hat{\mathcal{Y}}^{N}, {}_{MAP\leq N}{}^{i}\hat{\mathcal{D}}^{N}\right)$
11. Calculate unrefined novel-view synthesis metrics ${}_{NVS}{}^{i}\hat{M}^{N}$ for $\left({}_{NVS}{}^{i}\hat{\mathcal{Y}}^{N}, {}_{NVS}{}^{i}\hat{\mathcal{D}}^{N}\right)$
12. Calculate refined backprojection metrics ${}_{MAP}{}^{i}M^{N}$ for $\left({}_{MAP\leq N}{}^{i}\hat{\mathcal{Y}}^{N}_{\mathcal{M}}, {}_{MAP\leq N}{}^{i}\hat{\mathcal{D}}^{N}_{\mathcal{M}}\right)$
13. Calculate refined novel-view synthesis metrics ${}_{NVS}{}^{i}M^{N}_{\mathcal{M}}$ for $\left({}_{NVS}{}^{i}\hat{\mathcal{Y}}^{N}_{\mathcal{M}}, {}_{NVS}{}^{i}\hat{\mathcal{D}}^{N}_{\mathcal{M}}\right)$
14. Calculate input data metrics ${}_{MAP}{}^{i}M_{z}$ for ${}_{MAP}{}^{i}\hat{z}$

Note that it is inconsequential to distinguish between novel and known views with regards to the pseudo ground truth data used in steps 2 and 9, as all available ground truth sensor information is fused to these grids.

As metrics, we use mIoU to judge segmentation quality and *mIoU Foreground* to judge segmentation quality restricted to the foreground classes. An L1 term evaluates the estimated depth, which is always restricted to the foreground pixels. Finally, a completeness score calculates the fraction of foreground pixels which are present. Evaluating the results in 2D is justified because this perspective is not susceptible to discretization issues regarding the voxel grids. By calculating scores for rendered pseudo ground truth grids as well, we provide a baseline which relates our refined 3D results to the complete 3D maps expected for perfect measurements at the given grid resolution.

5.4 Implementation Details

We train our model using 5 synchronized NVIDIA A6000 handling 6 scenes each. GPU synchronization is implemented using *DistributedDataParallel* [1]. Nevertheless, training and validation for 13300 scenes requires 40 minutes per epoch and our experiments ran between 22 and 35 hours depending on early stopping conditions on a high performance node. During training, instead of rendering full images, we only cast a stochastic sample of possible rays, this is a common practice (e.g. $N_{\text{rays}} = 5000$ (Yu et al. [4]), $N_{\text{rays}} = 8192$ (C. Sun et al. [5]), $N_{\text{rays}} = 1024$ (Zhi et al. [10])) to speed up training. We use $N_{\text{rays}} = 832$ per scene for each of the 22 sensors available as targets. Effectively, this corresponds to a ray batch size of $N_{\text{total rays}} = 832 \cdot 22 \cdot 6 \cdot 5 \approx 550k$ during training. Allocating voxel grids with spatial dimensions $140 \times 112 \times 100$ covering volumes of $1m \times 0.8m \times 0.7m$, we sample $(N_S = 192) + (N_H = 48)$ points per ray as described in 3.3. The refinement model processing these maps uses 4 stacked hourglass blocks.

[1] `https://pytorch.org/docs/stable/distributed.html`

6 Results

This chapter interprets the results of the experiments described in the previous chapter. All values and images presented in the following originate from the test sets of the annotated point cloud datasets *apc_original*, *apc_moderate* and *apc_heavy*, which are defined in Table 5.1 in the previous chapter. To conclude this chapter, we describe the performance of the components of our pipeline.

6.1 Quantitative Results

In principle, our trained refinement system can be judged not only by its novel-view synthesis performance, that is, the quality of the rendered novel views given a refined semantic map, but also by its ability to preserve or improve the sensor information which was used to create the semantic maps in the first place. This second task is commonly referred to as *denoising* (Zhi et al. [10]), but can also be described as *backprojection* (Stückler et al. [2]). We will use the term backprojection from now on. However, we first examine the novel-view synthesis performance, since this is the primary indicator of the quality of the refined semantic maps.

6.1.1 Novel-View Synthesis

Metric	mIoU↑	mIoU FG↑	L1↓	Completeness↑
Input Data	**1**	**1**	**0**	**1**
Pseudo-GT	0.7732	0.7702	0.0268	0.9997
Unrefined	0.7570	0.7496	0.0487	0.9658
Refined	0.8708	0.8727	0.0258	0.9998

Table 6.1: Test metrics for novel-view synthesis on *apc_original*.

First, we examine the results on the dataset *apc_original*, which presented challenges primarily resulting from incomplete semantic maps, as the depth and object label distributions used to form the annotated point clouds were perfect and not augemented with any noise. The computed novel-view synthesis metrics for rendering from refined grids ${}^{i}_{NVS}M^{N}_{\mathcal{M}}$, unrefined grids ${}^{i}_{NVS}\hat{M}^{N}$ and pseduo ground

truth grids ${}^{i}M_{GT}$ are listed in Table 6.1 as an average over all 10 fusion steps and 100 scenes. Naturally for this experiment, the input data cannot be outperformed, as it is perfect anyhow. Nevertheless, our refinement model outperforms the pseudo ground truth renderings by a large margin, conceivably because it as learned not only to complete the given semantic maps, but also to minimize artefacts which arise as symptoms of grid discretization issues. While the refined renders are not perfect, an mIoU of ~ 0.87 is nonetheless a significant improvement over the unrefined grid. The unrefined grids perform only slightly worse than the pseudo ground truth grids because they are equivalent save for the number of sensor measurements which are fused into them. The main difference is reflected in the completeness scores, which alleges that the unrefined grids render to a smaller fraction of the foreground pixels which is correctly attributed as such. The foreground mIoU was included because the background class is present in many sensor frames and represents a large percentage of the pixels in some. That there is no significant difference between these and the normal mIoU scores means that our model has not simply learned to solve the foreground/background segmentation problem correctly, but actually provides adequate foreground object labels. What stands out, however, is that the rendered depth estimates deviate significantly from the ground-truth depth for all rendered outputs as measured by the L1-error (units in m). We do not believe that this is a voxel grid resolution issue, as the cells there are about 1cm^3 small. Instead, it is plausible that this might be an issue with volume rendering in general, as the weights w_k along a ray determining depth are not normalized. However, enforcing such normalization during training destabilizes the process and would also be ill-posed when all weights along a given ray are zero. Also rendered depth is inherently ambiguous near object edges, which could conceivably contribute to the problem.

Metric	mIoU↑	mIoU FG↑	L1↓	Completeness↑
Input Data	0.6328	0.6234	**0.0067**	0.9829
Pseudo-GT	0.7732	0.7707	0.0268	**0.9997**
Unrefined	0.5802	0.5897	0.0670	0.9678
Refined	**0.8041**	**0.7985**	0.0273	0.9994

Table 6.2: Test metrics for novel-view synthesis on *apc_moderate*.

When we apply a semantic segmentation model to obtain object label distributions and moderately corrupt depth measurements and camera parameters in the dataset *apc_moderate*, we obtain the results depicted in Table 6.2. In this scenario, the proposed refinement system shows great potential, outperforming both baselines as before, but also improving on the input data. The achieved mIoU is

worse than in the previous experiment, reflecting that this dataset is more difficult than the one discussed in the previous section. Once more, foreground and background scores are very close, which indicates that the foreground objects are correctly learned. However, there is a large gap in between pseudo ground truth and unrefined mIoU scores, which speaks to the degree of input data noise. Since errors in depth and errors in semantic labels are conflated by the fusion process, which is also clearly reflected by this score, this means that our model is inherently at a disadvantage when comparing to the input data, which is not affected by the corresponding depth measurement's defects. Notably, the input data completeness has dropped below the refined and pseudo ground truth values because of missing depth measurements which are counted towards this metric. For the depth metric, we again find the rendered estimates worse than the input data quality. However, these values are very similar to those from the previous experiment, suggesting that this issue is related to the method itself.

Metric	mIoU↑	mIoU FG↑	L1↓	Completeness↑
Input Data	0.1604	0.1389	**0.0134**	0.9297
Pseudo-GT	**0.7754**	**0.7748**	0.0268	**0.9997**
Unrefined	0.0741	0.0696	0.1083	0.9536
Refined	0.7638	0.7474	0.0334	0.9985

Table 6.3: Test metrics for novel-view synthesis on *apc_heavy*.

With further increased levels of input data noise, we find the results for *apc_heavy* summarized in Table 6.3. The input object class distributions are very unreliable in this setup, reflected by their poor mIoU of only 0.1604. The unrefined voxel grid performs even worse, with an mIoU of only 0.741, because it is affected by the errors in sensor parameters and pose. Nevertheless, the refined grid renders to predictions of semantics which are very competitive compared to the pseudo ground truth and not much worse than the values from the previous experiment, a strong indication that geometric features play a large role in the refinement process. While the depth error in the input data does not seem very severe, this score does not take into account the significant dropout of input depth measurements reflected by the completeness metrics, which drops below the value for the unrefined grids for this dataset.

6.1.2 Backprojection to Sensor Frames

Testing our system in the context of rendering back to views which contributed to the semantic map for the dataset *apc_original*, we compare the unrefined back-

Metric	mIoU↑	mIoU FG↑	L1↓	Completeness↑
Input Data	**1**	**1**	**0**	**1**
Pseudo-GT	0.7732	0.7702	0.0268	0.9997
Unrefined	0.7949	0.7994	0.0243	0.9999
Refined	0.9014	0.9028	0.0246	0.9999

Table 6.4: Test metrics for backprojection on *apc_original*.

projection metrics ${}_{MAP}^{i}\hat{M}^{N}$ to the baselines posed by the refined backprojection metrics ${}_{MAP}^{i}M^{N}$, the pseudo ground truth renders and the input data. The resulting scores are averaged over fusion steps and scenes and summarized in Table 6.4. Unsurprisingly, the results are better than in the novel-view synthesis problem because scene completion is not necessary, especially when the input sensor data is not noisy. This is also reflected by the completeness scores, which is high for the renders from unrefined grids. The most surprising aspect is that the unrefined grids outperform the pseudo ground truth in terms of segmentation quality. This hints at an issue with accumulation of grid discretization errors which we will examine in a subsequent section. However, the refined renders reach a very good mIoU of 0.9014, and since scene completion is not asked here, this confirms that the refine model successfully diminishes the impact of the limited semantic map resolution.

Metric	mIoU↑	mIoU FG↑	L1↓	Completeness↑
Input Data	0.6328	0.6234	**0.0067**	0.9829
Pseudo-GT	0.7732	0.7707	0.0268	**0.9997**
Unrefined	0.6230	0.6496	0.0365	0.9996
Refined	**0.8225**	**0.8166**	0.0261	0.9995

Table 6.5: Test metrics for backprojection on *apc_moderate*.

The corresponding scores for the dataset *apc_moderate* are displayed in Table 6.5. As for novel-view synthesis, the refinement model can successfully improve upon the semantic segmentation provided by a segmentation model that uses only 2D input. The scores are also better than for the previous problem because scene completion is less difficult for sensor frames which have been fused into the scene themselves. In contrast to the corresponding results for *apc_original*, the unrefined semantic map renders now perform much worse than than the ones for pseudo ground truth maps. This is probably because map resolution errors are secondary as soon as sensor noise is involved. Notably, the unrefined grids compete with the input data in segmentation quality, even outperforming it slightly when the

semantic segmentation is restricted to the foreground. That the input data performs slightly differently in mIoU and mIoU restricted to the foreground is because the semantic segmentation model used to generate it only outputs labels for foreground objects and takes the background class label from the ground truth. For this reason, the input data is always at an advantage when correctly identifying the background is also judged. Since this dataset was created with a realistic semantic segmentation model, the competitive performance of the unrefined render in this scenario demonstrates that the semantic mapping scheme developed for this thesis does not degrade our input data under realistic circumstances and can thus serve as a valid persistent representation of the contributing sensor data on its own, even without refinement.

Metric	mIoU↑	mIoU FG↑	L1↓	Completeness↑
Input Data	0.1604	0.1389	**0.0134**	0.9297
Pseudo-GT	0.7754	**0.7748**	0.0268	**0.9997**
Unrefined	0.0814	0.0812	0.0798	0.9856
Refined	**0.7794**	0.7623	0.0324	0.9985

Table 6.6: Test metrics for backprojection on *apc_heavy*.

Finally, the test results for backprojection of heavily corrupted input data are summarized in Table 6.6. These values present much like those in the novel-view synthesis case. The refine model in this case barely inches out on the pseudo ground truth renders when also accounting for the background because it does not need to perform as much scene completion as for novel-view synthesis, but also because the pseudo ground truth renders tend to show thicker objects which extend into the background, slightly worsening segmentation scores there.

6.1.3 Disambiguation of Fusion Steps

Of course, the metrics discussed so far are only summaries of the test set results and hide effects which unfold over varying numbers of fusion steps. We have previously seen signs of their existence, as for example the advantage of unrefined grids over pseudo ground truth semantic maps when backprojecting in the context of *apc_original*. Also, the unrefined completeness scores seem very high across the board. This is quickly cleared up by examining the saturation of completeness scores for novel view synthesis, exemplified by data from *apc_original* in Figure 6.1, which illustrates that semantic maps of binpicking scenes rapidly saturate and are most often complete after around 7 fusion steps. Furthermore, this observation

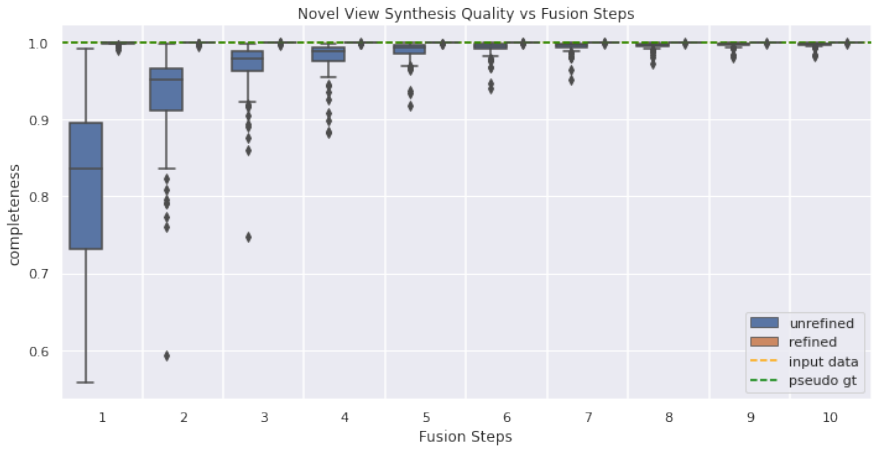

Figure 6.1: The completeness score as plotted against the number of fusion steps. Each box plot summarizes the score of the 100 test scenes at a specific number of fusion steps. A quick progression towards map saturation is evident.

justifies our use of only 10 sensor measurements for fusion, and only 32 to generate the pseudo ground truth.

Knowing this, we can interpret the superiority of fewer fusion steps over pseudo ground truth for backprojection experiments by looking at the relationship between fusion steps and the backprojection mIoU, depicted in Figure 6.2. On the dataset *apc_original*, where unrefined and pseudo ground truth grids are related, a convergence of the unrefined render mIoU scores towards the pseudo ground truth baseline is implied. This confirms our suspicion that this effect is related to the number of fused measurements. This, in turn, points to grid resolution issues as the most likely culprit. On the other hand, we can also observe the refined scores for this dataset remaining completely unaffected, as scene completion is not necessary here and confirming that the refinement can address problems arising from the map resolution. For the more noisy datasets, the performance of the unrefined grid also decreases with increasing fusion steps. This is expected, as a larger number of erroneous map entries will impact the rendered segmentation negatively. However, the refinement scores remain stable for different fusion steps, especially the final two, which were not trained for. This shows that the refinement process responds in a stable manner to semantic map saturation and also that the scores reported in the earlier sections were not distorted by the averaging scheme used to summarize them.

Of course, the impact of fusion steps should also be examined for the novel-view synthesis case, which is graphically depicted in analogy to the backprojection problem in Figure 6.3. For novel views, there is a clear trend showing that the refinement model benefits from increasing fusion steps, but the gain in performance

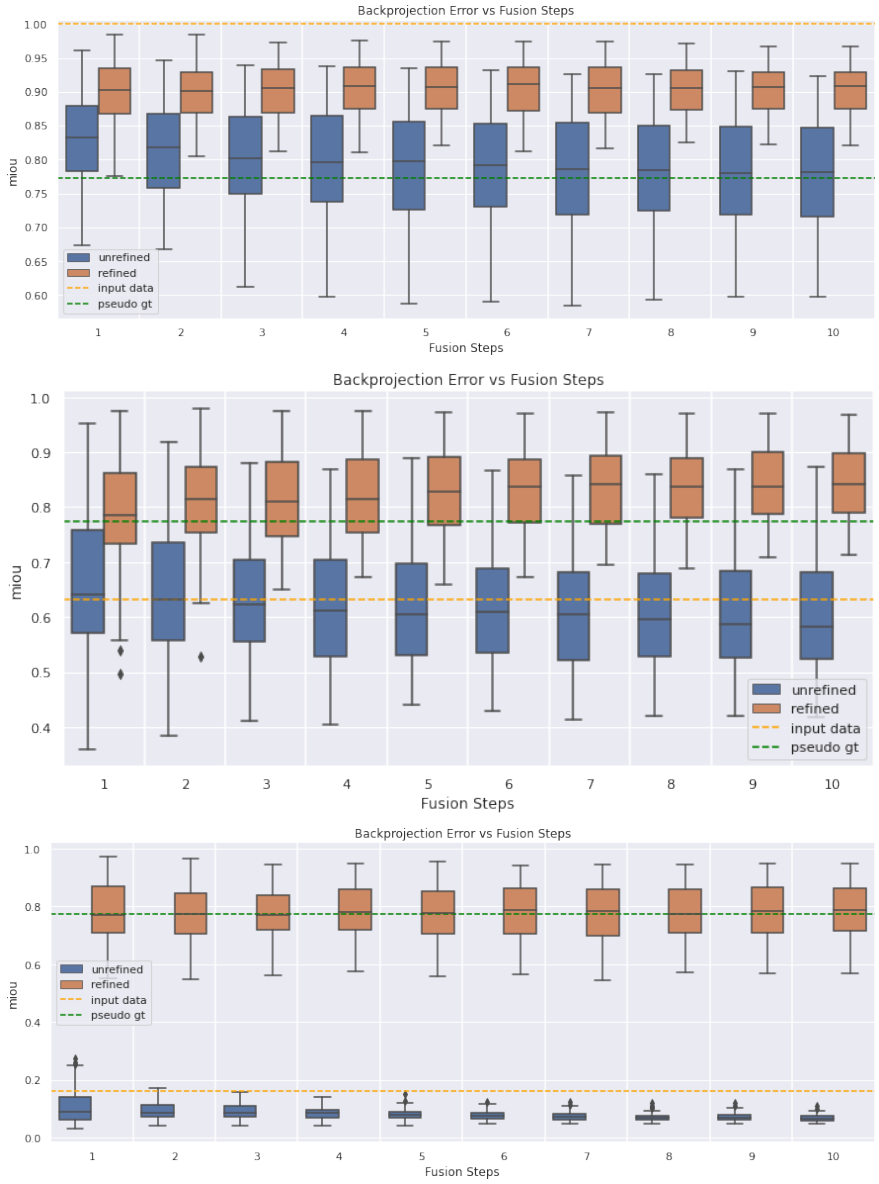

Figure 6.2: Backprojection mIoU plotted against fusion steps for refined and unrefined renders. In the top row, convergence of the scores for unrefined data towards the pseudo ground truth baseline is visible. For the other datasets, performance for unrefined data also decreases with additional fusion steps. The refined data does not rely heavily on additional fusion steps for this problem. *Datasets from top to bottom*: *apc_original*, *apc_moderate*, *apc_heavy*

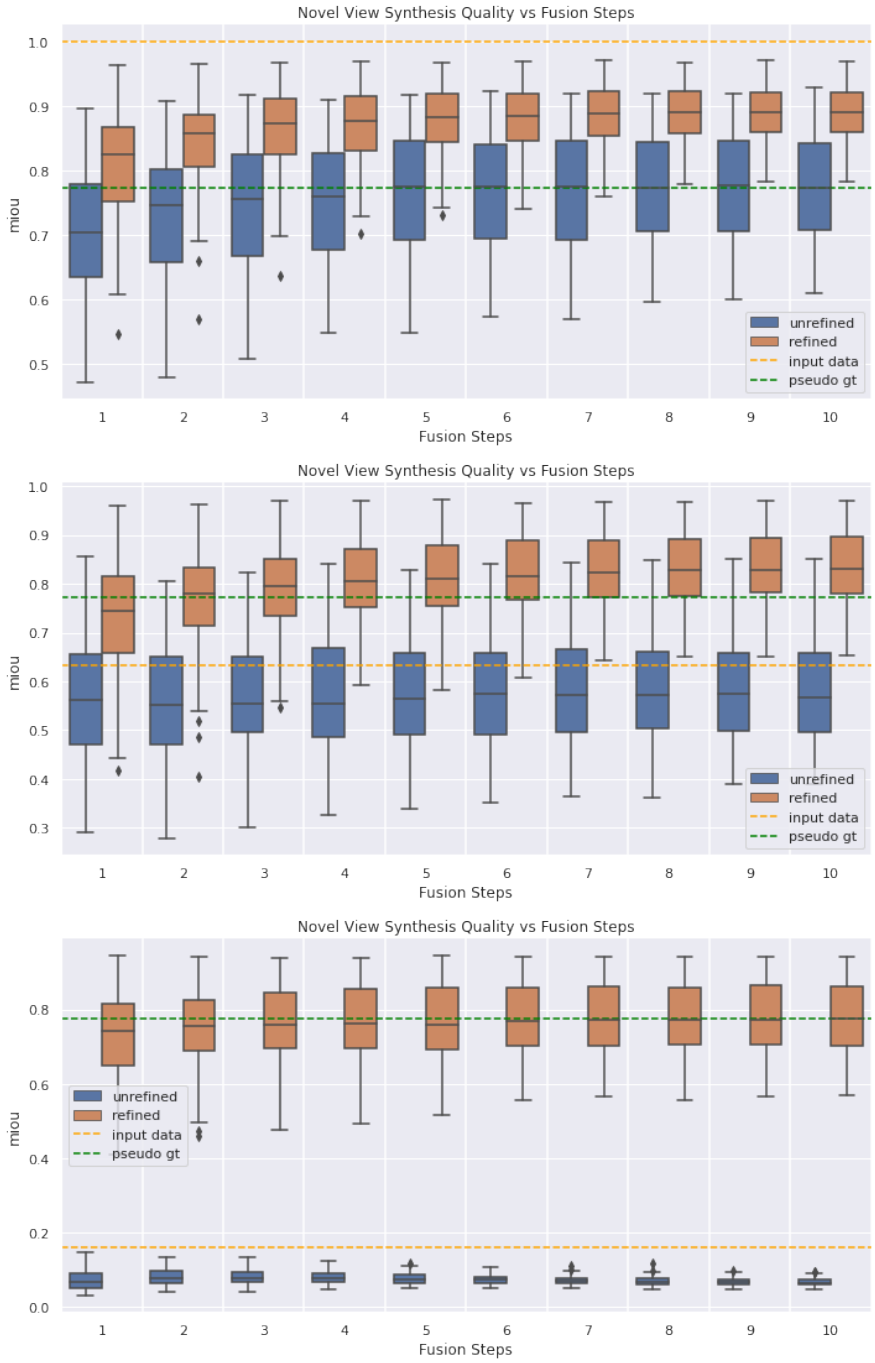

Figure 6.3: Novel-view mIoU plotted against fusion steps for refined and unrefined renders. In this case additional fusion steps are clearly beneficial when the data is not noisy. The map refinement exploits additional map entries for all degrees of data corruption. *Datasets from top to bottom*: *apc_original*, *apc_moderate*, *apc_heavy*

stagnates as the maps saturate. For the unrefined maps, there is also a benefit to fusing more data, but it is only pronounced when the sensor measurements are accurate.

6.2 Qualitative Results

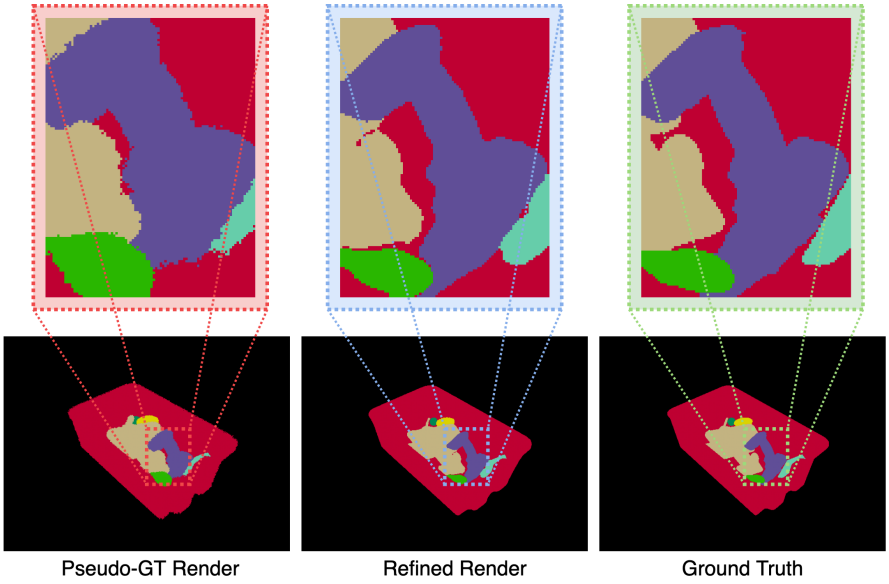

Figure 6.4: Rendering a pseudo ground truth grid reveals that it suffers from voxel grid resolution artefacts which make objects appear larger than they are. Refined semantic maps do not suffer from this issue and produce renders which are visually more similar to the ground truth.

It is important to relate differences in semantic map refinement quality expressed by scores in the previous section to the perceived visual quality of the corresponding outputs. Therefore, we dedicate this section to illustrating the views rendered by our method in different data corruption scenarios. For instance, the suboptimal segmentation scores for pseudo ground truth are corroborated visually, as depicted in Figure 6.4.

On the following pages, we depict one novel view into one scene from each dataset. Since the impact of the number of fusion steps into the semantic maps before and after refinement was previously demonstrated, we encourage the reader to take into account which objects the input data, depicted at the top of the figures, contribute to the semantic maps. Also, we refer to the figure captions for detailed commentary on the depicted scenes.

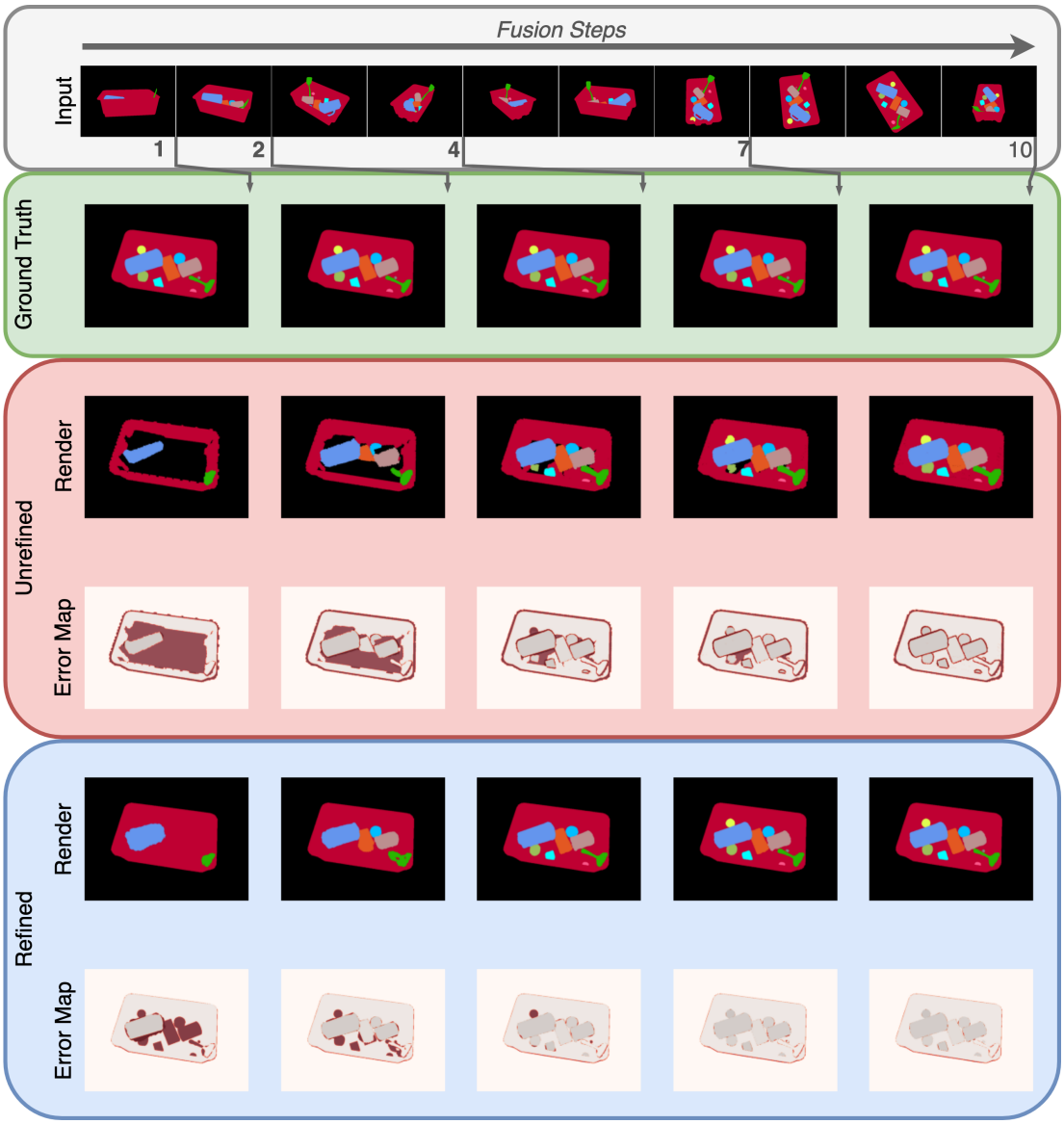

Figure 6.5: The progression of semantic rendering of one novel view into the same scene in *apc_original* over multiple fusion steps. The topmost box shows the input data which contributed to each fusion step. The rows depict the ground truth, the unrefined and refined render for the given view. Error maps highlight deviation from ground truth semantic labels in the renders. The columns show the progression of this data over multiple fusion steps. Note how the model is able to complete the tote when receiving a very sparse initial semantic map. The spatula in the lower right box corner is initally incomplete, as its orientation cannot be determined until the next fusion step. The lowermost row of error maps show how the refinement model adds objects as they appear in the input data, even when the entity is partially occluded. This highlights that our proposed method has learned reasonable priors.

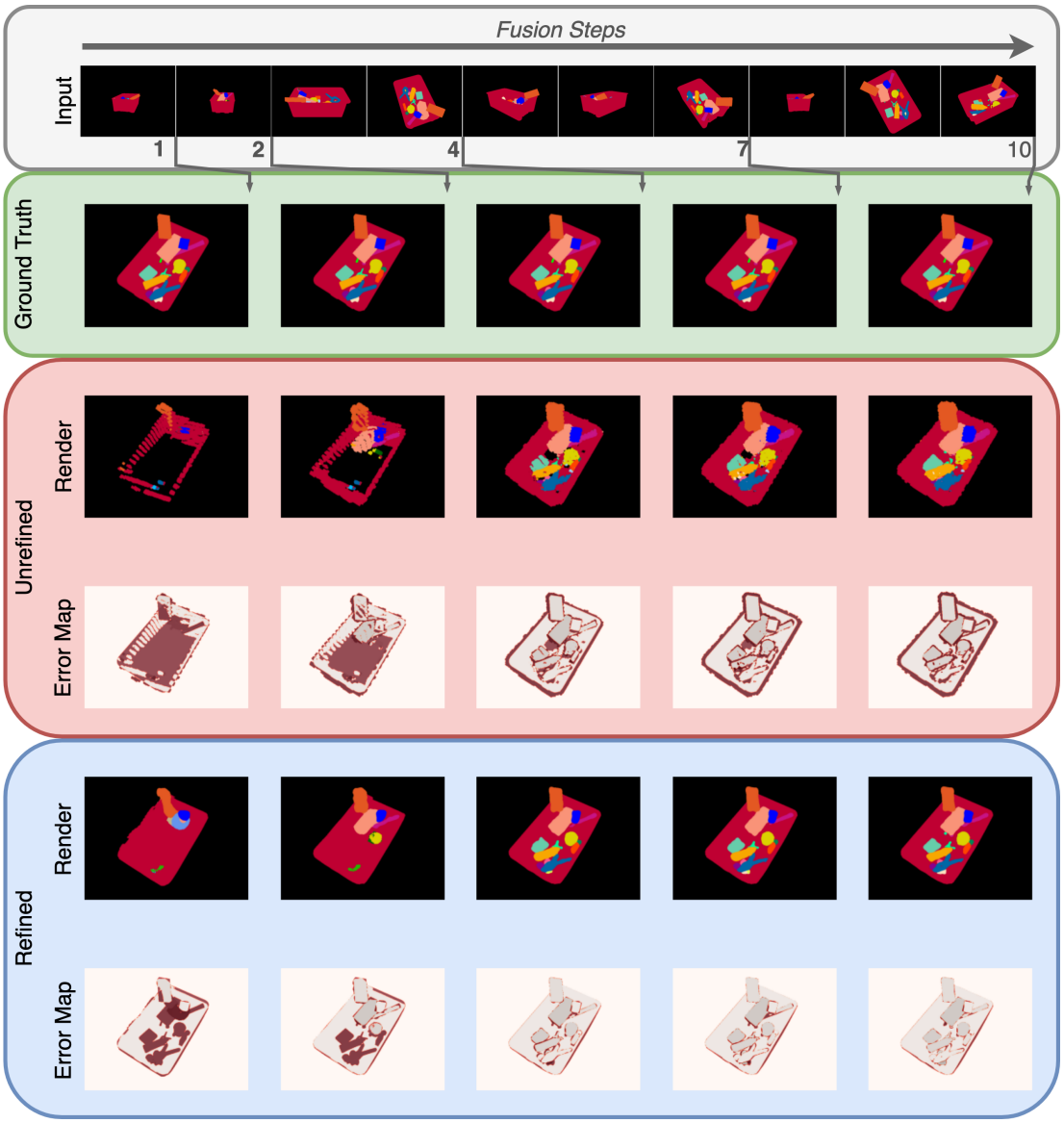

Figure 6.6: The progression of semantic rendering of one novel view into the same scene in *apc_moderate* over multiple fusion steps. The topmost box shows the input data which contributed to each fusion step. The rows depict the ground truth, the unrefined and refined render for the given view. Error maps highlight deviation from ground truth semantic labels in the renders. The columns show the progression of this data over multiple fusion steps. In the initial step, the semantic map is extremely incomplete, and an object which is not present is added to the tote, potentially to explain the floating blue object which actually exists. At the next fusion step, the prediction improves as new data indicates which objects are present in the far side of the box. At the 4th fusion step, an unobstructed view into the box is fused to the map. Immediately, various objects are completed in a plausible manner even though this region of the tote is cluttered.

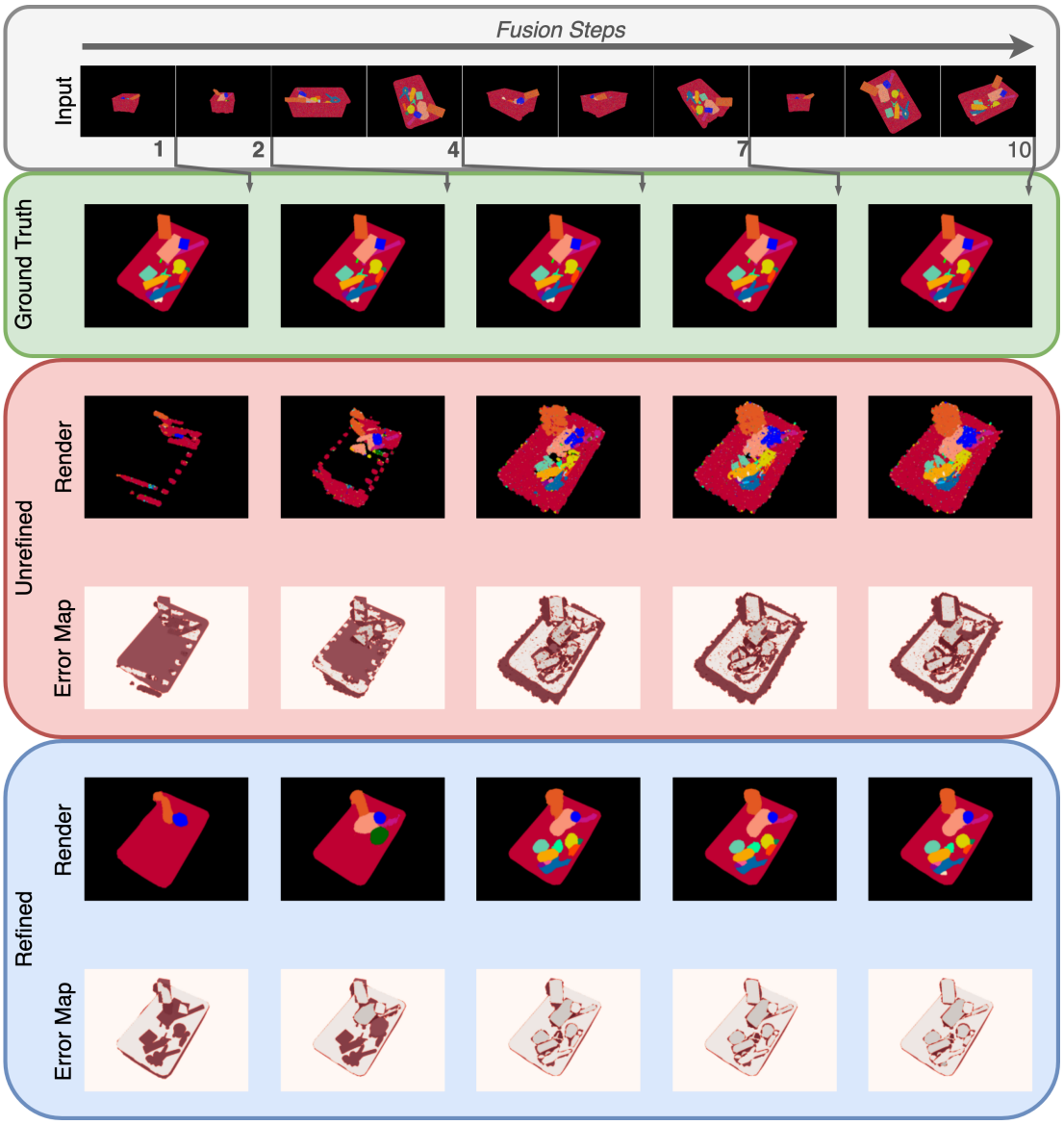

Figure 6.7: The progression of semantic rendering of one novel view into the same scene in *apc_heavy* over multiple fusion steps. The topmost box shows the input data which contributed to each fusion step. The rows depict the ground truth, the unrefined and refined render for the given view. Error maps highlight deviation from ground truth semantic labels in the renders. The columns show the progression of this data over multiple fusion steps. This scene was specifically chosen to be the same as the one depicted in Figure 6.6. Note that while the refined semantic map is still rendered to servicable novel views, the objects are noticeably more blurry than in the previous example. This shows how overwhelming sensor noise can impede learning of sharp object shapes, which is also not specifically targeted by our pixel-wise training loss function.

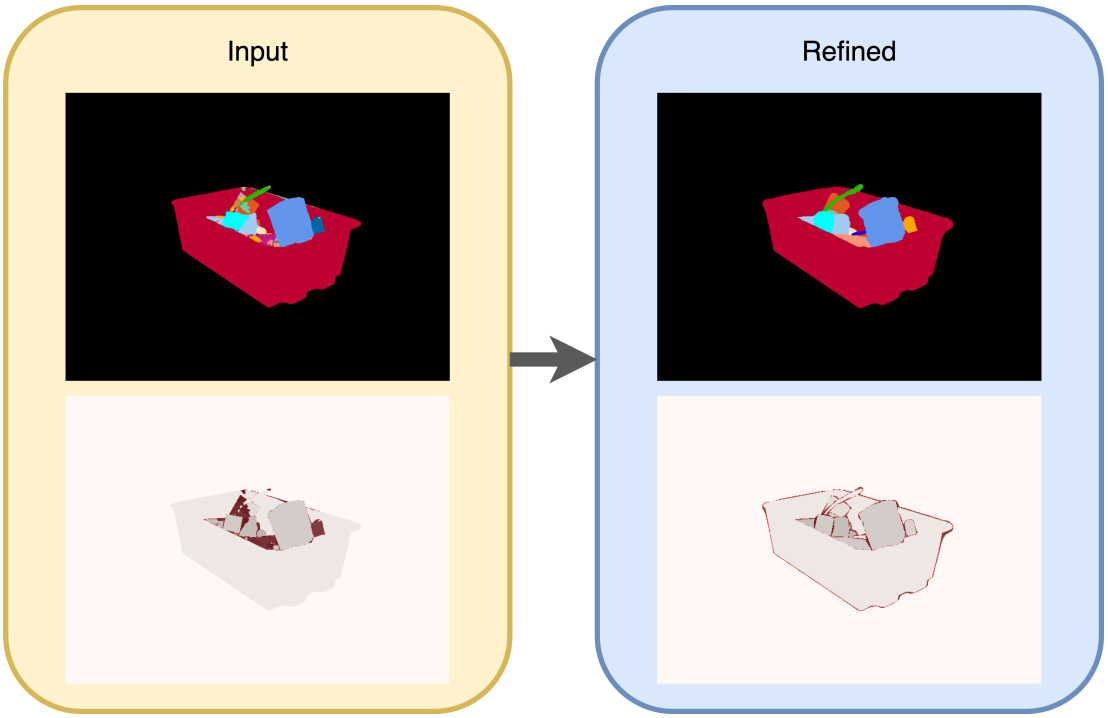

Figure 6.8: *Left*: The input semantic segmentation features an entire mislabeled object and several wrong labels where the scene is cluttered. *Right*: When rendered to the same view from a refined semantic map, these defects are corrected.

The backprojection task is interesting to visualize as well. Figure 6.8 shows an example of an input semantic segmentation which contains many mislabeled pixels in a cluttered tote and an object which has been completely misidentified. After semantic map fusion, a refinement step and rendering to the original view, the semantic segmentation is corrected.

For a more comprehensive set of examples in the backprojection setting, Figure 6.9 illustrates an extremely difficult case from *apc_heavy* featuring multiple small objects and very damaged input data. The backprojection for multiple input frames which are successively fused to the scene's semantic map is depicted for the unrefined and refined semantic maps. Also, the input semantic maps are compared to the ground truth frames to illustrate their noisy labels and how they are corrected.

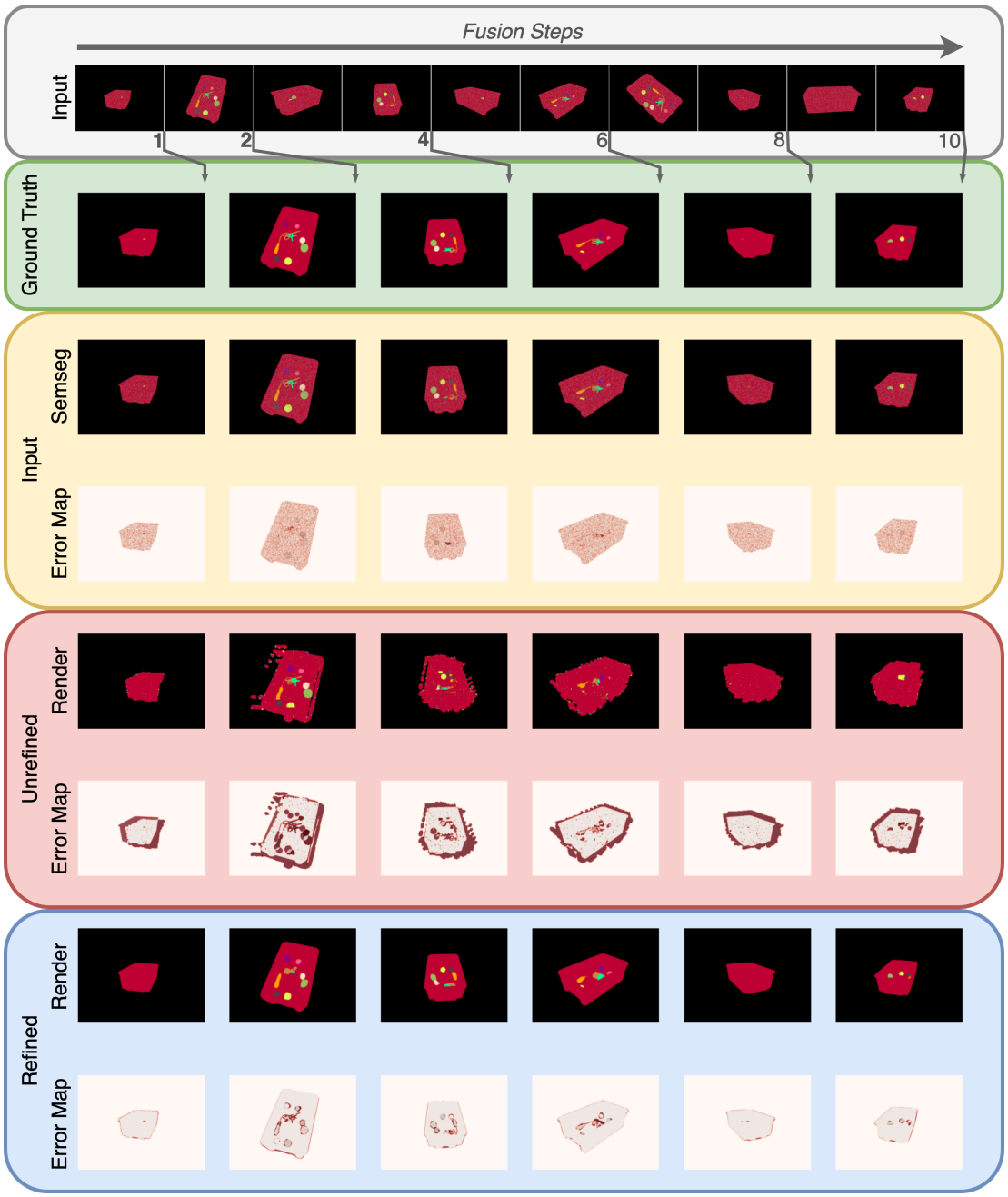

Figure 6.9: Backprojection to cumulatively fused input sensor frames performed for a scene in *apc_heavy*. The unrefined semantic maps are barely comprehensible, but refinement nonetheless brings forth plausible object arrangements and denoises the input semantic segmentation labels. These are depicted alongside their own error maps in the rows marked by the yellow box.

6.3 Performance

Op	Fuse	Refine	Render	Backprop
Inference Step (640 x 480)	0.00278s	0.31104s	0.11411s	✗
Train Step (22 x 832 Rays)	0.00268s	0.31025s	0.01090s	0.26501s

Table 6.7: Performance.

The performance of the individual operations in our pipeline is presented in Table 6.7. At inference time, our system can render novel views at an interactive $\sim$ 9 fps on an NVIDIA RTX3090. The fusion step in particular is very efficient because it relies almost exclusively on highly optimized sparse update operations. When an annotated point cloud corresponding to one sensor measurement is fused, $\sim$ 20million points per second can be processed. However, as for ray casting, the performance increases drastically if batched processing is considered. When all 32 measurements corresponding to an individual scene are fused, for example to obtain the pseudo ground truth voxel grids, the number of fused points per second increases to $\sim$ 120million per second per scene in the batch. However, refinement is quite slow at only $\sim$ 3 fps. Together, this suggests that a performant practical implementation would attempt to fuse at high frequency but refine much less often than new observations arrive. This could easily be realized by, in addition to a refined semantic map, maintaining an unrefined voxel grid on the side which fuses all incoming measurements. When a specified time span or amount of novel observations have been collected, the refinement model can be called to overwrite the current refined map. Also, there is no requirement to render the semantic map in the same manner as done here doing training. If batched processing is no longer important, sparse grid structures could be initialized from the semantic map in the single-scene setting, a strategy employed by Yu et al. [4].

7 Conclusion and Outlook

Throughout this thesis, we introduce a 3D semantic map fusion and refinement pipeline which is supervised by 2D semantic segmentation and optionally depth measurements. We demonstrate the viability of our approach in a synthetic binpicking scenario with realistic problems such as complex object clutter, flawed sensor poses, inaccurate semantic segmentation and incomplete depth measurements.

To explain our method, we first examine related work in the domains of semantic mapping, semantic scene segmentation and novel-view synthesis. Taking into account the advantages of the reviewed semantic mapping techniques, we implement a highly optimized fusion process for point clouds annotated with object class distributions into voxel grids, which henceforth serve as our semantic maps. Using examples from the field of semantic scene segmentation, we explore the joint processing of 3D geometry and 2D semantics. However, we eliminate the requirement of 3D annotated ground truth by drawing from a plethora of techniques originating in novel-view synthesis. In this regard, we proceed with recent explicit formulations which offer efficiency, but most importantly, also interpretability. By integrating a simple notion of density into our semantic maps, we are immediately able to render them, bridging the gap between 3D and 2D.

In the following, we detail the exact steps necessary to represent these semantic maps, to fuse multiple sensor measurements to, and then finally render them. We propose a suitable neural refinement model and briefly justify its architecture. Next, we motivate using synthetic binpicking scenes to trial this pipeline and set up a flexible simulator to produce a variety of output data. To increase realism, we propose several avenues of data augmentation for depth measurements, train a semantic segmentation model and apply noise to sensor parameters. Overall, this results in input data meant to closely resemble the difficult sensory challenges a binpicking robot might encounter.

Next, we detail our experiments and describe the extensive testing process designed to evaluate the performance of our proposed method while relating it to the completeness of the input semantic maps. We touch on the inner workings of our implementation and describe the compute resources required to run our large scale experiments.

Finally, we provide a quantitative comparison of our results with a strong baseline of rendering from a pseudo ground truth semantic map and additionally measure the improvement versus the quality of the input data and the unrefined semantic maps. We follow up on these scores and show visualizations that corroborate the metrics well. Then, we briefly discuss the performance of different components in the pipeline.

Overall, our end result is a semantic map refinement system which is successfully trained to leverage 3D priors without using 3D ground truth. It easily outperforms the weak baseline of an unrefined semantic map, improves upon the challenging baseline of realistic input semantic segmentation and even surpasses or is at minimum competitive when compared to rendering from pseudo ground truth semantic maps, which were originally expected to represent an unsurpassable upper bound on performance.

In future work, we wish to examine whether these results can be extended to real world binpicking scenarios. While training on synthetic data often does not generalize well to real settings, we are more confident in this case because we expect much of the complexity of the real world to be masked when using semantic segmentation instead of RGB images, for example. Perhaps, such steps depend mainly on the realism of the depth augmentation, which hasn't been examined in detail because we did not commit this framework to replicating the defects of a specific sensor. Also, the utility of the refined semantic maps in downstream tasks such as pose estimation could be examined. In the synthetic setting, our cluttered bin generator in its current state could already generate the required training data.

Ultimately, we are excited about the future progress in this field and, of course, are hopeful to participate in it.

List of Figures

1.1 An overview of the core idea. *Left*: Fuse RGB-D measurements in a voxel grid. *Center*: Refine the grid with a 3D CNN. *Right*: Render 2D views from it and backprop rendering loss to 3D CNN. 2

3.1 An example of hierarchical sampling of points along a ray. *Blue*: Stratified samples. *Green*: Weights calculated according to blue samples along ray cast through down-sampled density grid $\tilde{\mathcal{V}}_\mu$. *Red*: Points resampled using green weights as distribution. *Black*: Weights calculated using blue and red samples along ray in $\mathcal{V}_\mu$. 13

3.2 The semantic map refinement architecture consist of several modified stacked hourglass blocks (Chang et al. [16]) which are drop-in replacements for 3D CNN residual blocks. The layout of an individual block is depicted above. Our complete model consists of 4 stacked hourglass blocks. Diagram generated using [17]. 15

4.1 Examples from an annotated simulated scene created by our cluttered bin generator. **(a)**: RGB, **(b)**: depth, **(c)**: semantic segmentation, **(d)**: object poses, **(e)**: bounding boxes and **(f)**: bounding boxes shrunk to object visibility. 18

4.2 Measurements of a generic scene using an Intel RealSense L515 (from [24]). Missing depth measurements are visible on the tabletop and along edges. The background displays measurement noise and quantization due to limited sensor resolution. 20

4.3 *Left*: A binpicking scene captured using an Intel RealSense L515. *Right*: Simulated measurements of a similar scene. 21

4.4 *Right*: Compact representation of the surface normal map relative to sensor viewing direction. *Left*: Distributions of measurement dropout probability at different viewing angles given the angle of total reflectance θ_{Total}. 22

4.5 *Right*: Ground truth simulated depth measurement. *Left*: Depth map after applying data augmentation. White pixels represent missing measurements. 22

4.6 Example predictions of our semantic segmentation model $\mathcal{M}_y$ for validation data. 24

6.1 The completeness score as plotted against the number of fusion steps. Each box plot summarizes the score of the 100 test scenes at a specific number of fusion steps. A quick progression towards map saturation is evident. 38

6.2 Backprojection mIoU plotted against fusion steps for refined and unrefined renders. In the top row, convergence of the scores for unrefined data towards the pseudo ground truth baseline is visible. For the other datasets, performance for unrefined data also decreases with additional fusion steps. The refined data does not rely heavily on additional fusion steps for this problem. *Datasets from top to bottom*: *apc_original*, *apc_moderate*, *apc_heavy* 39

6.3 Novel-view mIoU plotted against fusion steps for refined and unrefined renders. In this case additional fusion steps are clearly beneficial when the data is not noisy. The map refinement exploits additional map entries for all degrees of data corruption. *Datasets from top to bottom*: *apc_original*, *apc_moderate*, *apc_heavy* 40

6.4 Rendering a pseudo ground truth grid reveals that it suffers from voxel grid resolution artefacts which make objects appear larger than they are. Refined semantic maps do not suffer from this issue and produce renders which are visually more similar to the ground truth. 41

6.5 The progression of semantic rendering of one novel view into the same scene in *apc_original* over multiple fusion steps. The topmost box shows the input data which contributed to each fusion step. The rows depict the ground truth, the unrefined and refined render for the given view. Error maps highlight deviation from ground truth semantic labels in the renders. The columns show the progression of this data over multiple fusion steps. Note how the model is able to complete the tote when receiving a very sparse initial semantic map. The spatula in the lower right box corner is initally incomplete, as its orientation cannot be determined until the next fusion step. The lowermost row of error maps show how the refinement model adds objects as they appear in the input data, even when the entity is partially occluded. This highlights that our proposed method has learned reasonable priors. 42

6.6 The progression of semantic rendering of one novel view into the same scene in *apc_moderate* over multiple fusion steps. The topmost box shows the input data which contributed to each fusion step. The rows depict the ground truth, the unrefined and refined render for the given view. Error maps highlight deviation from ground truth semantic labels in the renders. The columns show the progression of this data over multiple fusion steps. In the initial step, the semantic map is extremely incomplete, and an object which is not present is added to the tote, potentially to explain the floating blue object which actually exists. At the next fusion step, the prediction improves as new data indicates which objects are present in the far side of the box. At the 4th fusion step, an unobstructed view into the box is fused to the map. Immediately, various objects are completed in a plausible manner even though this region of the tote is cluttered. 43

6.7 The progression of semantic rendering of one novel view into the same scene in *apc_heavy* over multiple fusion steps. The topmost box shows the input data which contributed to each fusion step. The rows depict the ground truth, the unrefined and refined render for the given view. Error maps highlight deviation from ground truth semantic labels in the renders. The columns show the progression of this data over multiple fusion steps. This scene was specifically chosen to be the same as the one depicted in Figure 6.6. Note that while the refined semantic map is still rendered to servicable novel views, the objects are noticeably more blurry than in the previous example. This shows how overwhelming sensor noise can impede learning of sharp object shapes, which is also not specifically targeted by our pixel-wise training loss function. . 44

6.8 *Left*: The input semantic segmentation features an entire mislabeled object and several wrong labels where the scene is cluttered. *Right*: When rendered to the same view from a refined semantic map, these defects are corrected. 45

6.9 Backprojection to cumulatively fused input sensor frames performed for a scene in *apc_heavy*. The unrefined semantic maps are barely comprehensible, but refinement nonetheless brings forth plausible object arrangements and denoises the input semantic segmentation labels. These are depicted alongside their own error maps in the rows marked by the yellow box. 46

List of Tables

5.1 Overview of data augmentation configuration settings for our datasets. 27

6.1 Test metrics for novel-view synthesis on *apc_original*. 33
6.2 Test metrics for novel-view synthesis on *apc_moderate*. 34
6.3 Test metrics for novel-view synthesis on *apc_heavy*. 35
6.4 Test metrics for backprojection on *apc_original*. 36
6.5 Test metrics for backprojection on *apc_moderate*. 36
6.6 Test metrics for backprojection on *apc_heavy*. 37
6.7 Performance. 47

Bibliography

[1] A. Nüchter and J. Hertzberg. "Towards semantic maps for mobile robots." In: *Robotics auton. syst.* 56 (2008), pp. 915–926.

[2] J. Stückler, N. Biresev, and S. Behnke. "Semantic mapping using object-class segmentation of RGB-D images." In: *2012 ieee/rsj international conference on intelligent robots and systems*. 2012, pp. 3005–3010.

[3] S. Bultmann, J. Quenzel, and S. Behnke. "Real-time multi-modal semantic fusion on unmanned aerial vehicles with label propagation for cross-domain adaptation." In: *Robotics and autonomous systems* 159 (2023), p. 104286. ISSN: 0921-8890. URL: https://www.sciencedirect.com/science/article/pii/S0921889022001750.

[4] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. "Plenoxels: Radiance Fields without Neural Networks." In: *Corr* abs/2112.05131 (2021). arXiv: 2112.05131. URL: https://arxiv.org/abs/2112.05131.

[5] C. Sun, M. Sun, and H. Chen. "Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction." In: *Corr* abs/2111.11215 (2021). arXiv: 2111.11215. URL: https://arxiv.org/abs/2111.11215.

[6] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett. "Recurrent-OctoMap: Learning State-based Map Refinement for Long-Term Semantic Mapping with 3D-Lidar Data." In: *Corr* abs/1807.00925 (2018). arXiv: 1807.00925. URL: http://arxiv.org/abs/1807.00925.

[7] A. Dai and M. Nießner. "3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation." In: *Corr* abs/1803.10409 (2018). arXiv: 1803.10409. URL: http://arxiv.org/abs/1803.10409.

[8] W. Hu, H. Zhao, L. Jiang, J. Jia, and T. Wong. "Bidirectional Projection Network for Cross Dimension Scene Understanding." In: *Corr* abs/2103.14326 (2021). arXiv: 2103.14326. URL: https://arxiv.org/abs/2103.14326.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." In: *Corr* abs/2003.08934 (2020). arXiv: 2003.08934. URL: https://arxiv.org/abs/2003.08934.

[10] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison. "In-Place Scene Labelling and Understanding with Implicit Scene Representation." In: *Corr* abs/2103.15875 (2021). arXiv: 2103.15875. URL: https://arxiv.org/abs/2103.15875.

[11] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand. "Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering." In: *Corr* abs/2106.02634 (2021). arXiv: 2106.02634. URL: https://arxiv.org/abs/2106.02634.

[12] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. M. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth. "NeSF: Neural Semantic Fields for Generalizable Semantic Segmentation of 3D Scenes." In: *Corr* abs/2111.13260 (2021). arXiv: 2111.13260. URL: https://arxiv.org/abs/2111.13260.

[13] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger. "SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks." In: *Corr* abs/1609.05130 (2016). arXiv: 1609.05130. URL: http://arxiv.org/abs/1609.05130.

[14] J. T. Kajiya and B. P. Von Herzen. "Ray Tracing Volume Densities." In: *Proceedings of the 11th annual conference on computer graphics and interactive techniques*. SIGGRAPH '84. New York, NY, USA: Association for Computing Machinery, 1984, pp. 165–174. ISBN: 0897911385. URL: https://doi.org/10.1145/800031.808594.

[15] N. Max. "Optical models for direct volume rendering." In: *Ieee transactions on visualization and computer graphics* 1.2 (1995), pp. 99–108.

[16] J. Chang and Y. Chen. "Pyramid Stereo Matching Network." In: *Corr* abs/1803.08669 (2018). arXiv: 1803.08669. URL: http://arxiv.org/abs/1803.08669.

[17] H. Iqbal. *PlotNeuralNet*.

[18] A. Odena, V. Dumoulin, and C. Olah. "Deconvolution and Checkerboard Artifacts." In: *Distill* (2016). URL: http://distill.pub/2016/deconv-checkerboard.

[19] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields." In: *Corr* abs/2103.13415 (2021). arXiv: 2103.13415. URL: https://arxiv.org/abs/2103.13415.

[20] N. Müller, Y. Siddiqui, L. Porzi, S. R. Bulò, P. Kontschieder, and M. Nießner. *DiffRF: Rendering-Guided 3D Radiance Field Diffusion*. 2022. URL: https://arxiv.org/abs/2212.01206.

[21] A. S. Periyasamy, M. Schwarz, and S. Behnke. "SynPick: A Dataset for Dynamic Bin Picking Scene Understanding." In: *Corr* abs/2107.04852 (2021). arXiv: 2107.04852. URL: https://arxiv.org/abs/2107.04852.

[22] M. Schwarz and S. Behnke. "Stillleben: Realistic Scene Synthesis for Deep Learning in Robotics." In: *Corr* abs/2005.05659 (2020). arXiv: 2005.05659. URL: https://arxiv.org/abs/2005.05659.

[23] A. Boltres, A. Villar-Corrales, J. Nogga, and P. Schütt. *sl-cutscenes*.

[24] *Intel Realsense LiDAR Camera L515*. https://www.intelrealsense.com/lidar-camera-l515/. Accessed: 21-02-2023.

[25] T. Mallick, P. P. Das, and A. K. Majumdar. "Characterizations of Noise in Kinect Depth Images: A Review." In: *Ieee sensors journal* 14.6 (2014), pp. 1731–1740.

[26] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Corr* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[27] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers." In: *Corr* abs/2105.15203 (2021). arXiv: 2105.15203. URL: https://arxiv.org/abs/2105.15203.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database." In: *2009 ieee conference on computer vision and pattern recognition*. 2009, pp. 248–255.

[29] M. Berman and M. B. Blaschko. "Optimization of the Jaccard index for image segmentation with the Lovász hinge." In: *Corr* abs/1705.08790 (2017). arXiv: 1705.08790. URL: http://arxiv.org/abs/1705.08790.