PanopticNDT: Efficient and Robust Panoptic Mapping

Daniel Seichter, Benedict Stephan, Söhnke Benedikt Fischedick, Steffen Müller, Leonard Rabes, and Horst-Michael Gross

***Abstract*— As the application scenarios of mobile robots are getting more complex and challenging, scene understanding becomes increasingly crucial. A mobile robot that is supposed to operate autonomously in indoor environments must have precise knowledge about what objects are present, where they are, what their spatial extent is, and how they can be reached; i.e., information about free space is also crucial. Panoptic mapping is a powerful instrument providing such information. However, building 3D panoptic maps with high spatial resolution is challenging on mobile robots, given their limited computing capabilities. In this paper, we propose PanopticNDT – an efficient and robust panoptic mapping approach based on occupancy normal distribution transform (NDT) mapping. We evaluate our approach on the publicly available datasets Hypersim and ScanNetV2. The results reveal that our approach can represent panoptic information at a higher level of detail than other state-of-the-art approaches while enabling real-time panoptic mapping on mobile robots. Finally, we prove the real-world applicability of PanopticNDT with qualitative results in a domestic application.**

I. Introduction

Mobile robots that are supposed to operate autonomously in complex indoor environments must have a detailed scene understanding. The robot needs to have precise semantic knowledge about the scene as well as each object within. Panoptic mapping combines and integrates this information over time into a 3D representation, enabling the robot to have a broad understanding of its environment.

In our research projects CO-HUMANICS [1] and MORPHIA [2], our mobile robots act autonomously in domestic environments and allow relatives as well as caregivers to stay in contact with care-dependent people to let them participate in social life. We aim to enable inexperienced operators to remotely control our robots in such an environment. As shown in Fig 1, the operator should be able to send the robot to a specific target, e.g., the table with eight chairs. Moreover, it should find a suitable waiting position, which, for example, involves not blocking other chairs or taking a suitable position for further inspection via the camera. As domestic environments are typically dynamic, we rely on a robust long-term localization via RTABMap [3] and build short-term panoptic 3D representations of the current environment periodically. This enables gaining and representing panoptic knowledge while still being able to react to small but important changes, such as chair arrangements. In mobile robotics, voxel-based 3D representations are common due to their efficient processing [6 – 8]. While there are already approaches for semantic mapping [9 – 12] extending these classical approaches, research focusing on panoptic mapping is rare. Existing approaches typically do not focus on mobile applications and, thus, do not meet our requirements for efficient and robust panoptic mapping.

Fig. 1. Panoptic occupancy NDT (P-NDT) map built with predicted panoptic segmentation of EMSANet [4] and voxel size of 10cm for scene *ai_051_001* of the Hypersim [5] test split. Bottom: corresponding instance map. Best viewed in color at 300%. Black indicates *no_instance*, see Fig. 3 for semantic colors. Panoptic is visualized by small color differences.

Therefore, in this paper, we propose panoptic normal distribution transform (PanopticNDT) mapping – an approach that addresses the requirements of such a scenario. It builds upon the fast and robust occupancy NDT mapping [13, 14] and its semantic extension [15]. We examine how to further incorporate and track instance information in order to realize panoptic NDT maps. We show that our approach enables precise and efficient panoptic mapping at sub-voxel level at a framerate of ˜2.75Hz on our mobile robots.

For evaluating, we conduct experiments on two indoor benchmark datasets: the large-scale synthetic Hypersim [5] dataset and the real-world ScanNetV2 dataset [16]. Unlike other approaches, we first evaluate the capability of our approach to represent panoptic information independently of a preceding segmentation step, i.e., we use the ground-truth annotations for mapping. Subsequently, we switch to panoptic predictions of our EMSANet [4] – an efficient RGB-D panoptic segmentation approach – to evaluate mapping in a more realistic setting. For reporting results, we rely on the ScanNetV2 benchmark pipeline. We extend the existing pipeline with a complementary panoptic evaluation task in 2D and 3D and prepare Hypersim to be used in that pipeline. The quantitative results of our experiments demonstrate the performance and robustness of our approach. Finally, we present qualitative results in a real domestic environment,

Authors are with Neuroinformatics and Cognitive Robotics Lab, Technische Universität Ilmenau, 98693 Ilmenau, Germany. Contact: daniel.seichter@tu-ilmenau.de, ORCID: 0000-0002-3828-2926

This work has received funding from the German Federal Ministry of Education and Research (BMBF) to the project MORPHIA (16SV8426).

showing the real-world performance of our approach.

In order to enable other researchers to have a similar setup and to compare to our panoptic mapping approach, we share the pipeline for data preparation and evaluation as well as the training details and weights of all neural networks involved at GitHub: `https://github.com/TUI-NICR/panoptic-mapping`.

II. Related Work

Although many representations have been proposed [6, 17, 18], in mobile robotics, voxel-based representations are preferred due to their efficient processing and memory requirements. In the following, we summarize voxel-based representations and, subsequently, focus on integrating additional semantic and panoptic information.

A. Voxel-based Data Representations

In voxel-based representations, 3D space is divided into a regular grid of voxels with a predefined size. These voxels are typically managed using optimized data structures, such as octrees [6] or voxel hashing [19]. To enable downstream applications, each voxel usually stores an occupancy value, indicating its state either as occupied, free, or unknown.

However, the precision of such maps still highly depends on the used voxel size. Decreasing the voxel size heavily increases memory and computational requirements and, thus, is not feasible for mobile applications. Therefore, other approaches attempt to additionally model the surface of each voxel in order to enable higher precision at the same grid resolution. For example, truncated signed distance fields (TSDF) [20, 21] store the distance to the nearest surface, allowing a more accurate representation of its shape and position. However, for efficiency reasons, occupancy is only modeled and stored in a truncated area around the surface of objects. This can lead to missing information about unseen areas required for downstream applications in mobile robotics, such as obstacle avoidance or path planning. In contrast to that, surfel maps [22] use so-called surface elements characterized by the eigenvectors of the data distribution inside a voxel. However, this representation makes use of the flat surface assumption that becomes insufficient when modeling objects smaller than the grid resolution. A similar representation is the normal distribution transform (NDT) [13, 23, 24]. In contrast to surfel maps, NDT maintains the whole covariance matrix of the data distribution inside a voxel to model its spatial extents in terms of a normal distribution.

As the NDT representation is able to capture a lot of details [25] even with larger voxel sizes and can be efficiently updated [13], we follow this approach.

B. Semantic Mapping

In contrast to the aforementioned approaches, semantic mapping approaches do not only store an occupancy value in each voxel but also semantic information. The semantic information is retrieved by a preliminary semantic segmentation step using conditional random fields (CRF) [9] or approaches based on convolutional neural networks (CNN) [10, 15]. Most approaches have in common that a histogram is used to model the information. Approaches, such as [8, 11], create a multi-class problem (free + semantic classes) to track occupancy and semantic information in a single histogram. However, as this requires sampling data points for free space from range measurements, these approaches are susceptible to over-representing the free space class. As shown in [15], this effect gets further intensified when incorporating noisy segmentations. Therefore, [9, 10, 15] decouple both information and maintain occupancy and semantic independently.

In [15], we have shown that semantic information can be integrated in the occupancy normal distribution transform mapping of [13, 14], enabling efficient and precise semantic mapping at sub-voxel level. Therefore, our panoptic mapping approach builds on top of this work.

C. Panoptic Mapping

Panoptic mapping is challenging as it aims to create a representation of the environment including both, semantic and instance information, where each instance ID must be globally unique. Thus, panoptic mapping faces the additional challenge that observations are not independent anymore: instances already present in the map must be matched with the instances of the current observation before integrating. Similar to semantic mapping, panoptic mapping is usually done in multiple steps, featuring one or multiple approaches for panoptic segmentation, matching, and the mapping step. These steps are often further followed by a map refinement.

Most approaches in this context build upon Voxblox [21] – a method that creates voxel-based TSDF maps. Panoptic Fusion [26], for example, additionally stores a single panoptic ID and a corresponding weight in each voxel. For panoptic segmentation, the output of PSPNet [27] and Mask R-CNN [28] is combined. Instance inconsistency is resolved by projecting the current map back to the camera plane and an intersection over union (IoU)-based matching. The weight in a voxel is increased with matching observations and decreased when observations do not match. The panoptic label gets updated or replaced depending on that weight. Voxblox++ [29] also extends [21] but only focuses on instance mapping. An additional geometric depth segmentation divides each instance into segments. The map represents these segments while additionally keeping track of corresponding instance labels. The integration of new observations is similar to [26] but done in 3D. In [30], a further extension to panoptic mapping is proposed. However, it mainly focuses on fitting CAD models into the map. The recent Panoptic Multi-TSDFs [31] also builds on top of Voxblox but takes a different approach. It uses a collection of submaps to represent the geometry of individual entities such as object instances, the background class, or free space. By combining these submaps and explicitly modeling free space, a full volumetric map can be derived. Each submap has a different voxel resolution depending on the semantic class it is supposed to represent. To efficiently integrate panoptic information, a label tracking approach based on IoU, similar to [26], is used.

All approaches have in common that they do not model the underlying distribution of panoptic observations integrated into the map, making them less robust to misclassifications. Moreover, none of the proposed pipelines for panoptic mapping is real-time capable on a mobile robot due to computationally expensive segmentation approaches, such as Mask R-CNN or PSPNet, or additional map refinement.

III. Efficient and Robust Panoptic NDT-Mapping

Our panoptic mapping pipeline is depicted in Fig. 2 and is implemented using the middleware for robotic applications (MIRA) [32]. Given a precise localization in the environment, our approach comprises two steps. We first apply EMSANet [4] – an efficient RGB-D panoptic segmentation approach that extends ESANet [33] – to the current set of input images (color and depth). Unlike other approaches [9 – 12, 26, 29 – 31, 34], we decided in favor of an RGB-D approach, as depth provides complementary information that helps to segment cluttered indoor scenes [4, 33]. Afterward, the obtained panoptic segmentation, the depth image, and the current pose are passed to the mapping stage. In the following, we describe both parts of the pipeline in detail.

A. Panoptic Segmentation

Panoptic segmentation aims to assign a panoptic label $P(\boldsymbol{u}) \in \mathcal{P}$ to each input pixel, with $\boldsymbol{u} \in \mathbb{R}^2$ denoting the corresponding normalized image coordinates. To derive a pixel's panoptic label, both a semantic prediction $L(\boldsymbol{u}) \in \mathcal{L}$ and an instance prediction $Z(\boldsymbol{u}) \in \mathcal{Z}$ must be merged. Panoptic segmentation distinguishes two categories of semantic classes: *thing* classes $\mathcal{L}^{\text{Th}}$ – such as chair, table, or cabinet – for countable objects; and *stuff* classes $\mathcal{L}^{\text{St}}$ – such as wall, floor, or ceiling – for non-countable objects, such that $\mathcal{L}^{\text{Th}} \cup \mathcal{L}^{\text{St}} = \mathcal{L}$ and $\mathcal{L}^{\text{Th}} \cap \mathcal{L}^{\text{St}} = \emptyset$. An instance ID $z \in \mathbb{N}_{>0}$ is only assigned if a pixel belongs to one of the thing class, and $z = 0$ denotes *no_instance*.

For efficiency reasons, we use the lightweight EMSANet-R34-NBt1D (enhanced dual ResNet34-based encoder utilizing the Non-Bottleneck-1D block (NBt1D) [35]) for panoptic segmentation. EMSANet [4] predicts dense semantic and instance information in a bottom-up fashion (see Fig. 2).

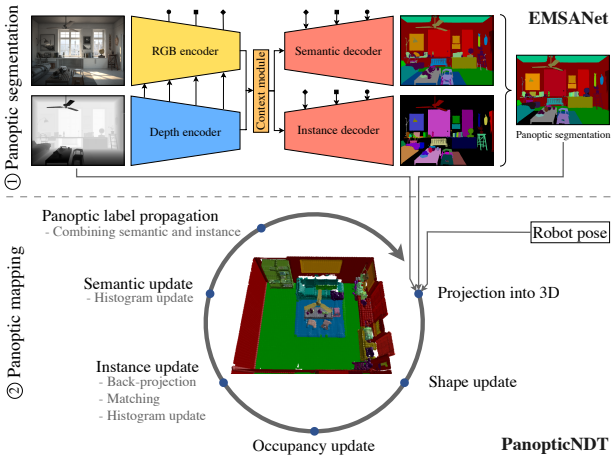

Fig. 2. Overview of our two-step approach for panoptic mapping.

The first task decoder directly assigns a semantic class to each pixel. The second task decoder predicts instance centers and instance center assignments. Both instance predictions are fused for pixels belonging to thing classes to obtain a class-agnostic instance segmentation. Note, unlike other top-down approaches [26, 29, 30] that utilize instance segmentation approaches, such as Mask-RCNN [28], this bottom-up approach leads to consistent instance IDs, i.e., there are no overlaps that must be resolved to determine a single instance ID for each pixel. The panoptic label $P(\boldsymbol{u})$ finally combines semantic and instance information in a tuple and assigns the most frequent semantic class for each instance:

$$P(\boldsymbol{u}) = \begin{cases} \langle \underset{\boldsymbol{u}' \in \{\boldsymbol{i} | Z(\boldsymbol{i}) = Z(\boldsymbol{u})\}}{\text{mode}}(L(\boldsymbol{u}')) \ , Z(\boldsymbol{u}) \rangle & L(\boldsymbol{u}) \in \mathcal{L}^{\text{Th}} \\ \langle L(\boldsymbol{u}), 0 \rangle & \text{otherwise.} \end{cases} \tag{1}$$

Note, Eq. 1 emphasizes the instance prediction, i.e., the semantic prediction might be overruled by the semantic mode for a predicted instance. In the following, we denote the elements in the tuple $P(\boldsymbol{u})$ as $\langle L^P(\boldsymbol{u}), Z^P(\boldsymbol{u}) \rangle$. Furthermore, we extend EMSANet to also provide dense confidence scores for the panoptic label and both elements to account for weak predictions in the subsequent mapping stage. We denote these scores as $S^{L^P}(\boldsymbol{u})$, $S^{Z^P}(\boldsymbol{u})$, and $S^P(\boldsymbol{u}) = S^{L^P}(\boldsymbol{u}) \cdot S^{Z^P}(\boldsymbol{u})$ for semantic, instance, and panoptic, respectively.

B. Panoptic Mapping: Overview

The panoptic mapping stage aims to integrate the predicted panoptic labels $P(\boldsymbol{u})$ over time into a consistent 3D map. Unfortunately, direct integration, as for semantic mapping, is not possible. The predicted instance IDs $Z^P(\boldsymbol{u})$ are inconsistent for the same object over multiple images processed in the first stage. Therefore, an instance matching and tracking step is required. This step can be done in 2D or 3D. We perform this step in 2D and project the 3D map back to the camera plane to match instances, similar to [26, 29 – 31]. Afterward, the instance and semantic information in each voxel of the map is updated according to the current observation. We introduce the underlying data representation in Sec. III-C and describe the instance and semantic update step in Sec. III-D and Sec. III-E, respectively. Note, unlike other approaches [26, 30, 31], we follow the class-agnostic bottom-up idea of EMSANet also for mapping, i.e., semantic and instance information are mapped independently. The only link between both is different processing depending on the stuff and thing class assignment. Especially for indoor environments, cluttered scenes may impede panoptic segmentation. Directly mapping panoptic information is susceptible to misclassifying related semantic classes, e.g., classifying an armchair as chair and sofa or a cabinet as counter and shelf in subsequent images. We observe a great performance and robustness boost when decoupling semantic and instance information. To derive the panoptic label for each voxel, we finally perform a panoptic-label-propagation step similar to Eq. 1. We introduce this step in Sec. III-F. Unlike [26, 31], we do not apply any subsequent map refinement, as this notably increases runtime.

C. Panoptic Mapping: Map and Data Representation

We build on top of the occupancy normal distribution transform (NDT) mapping approach presented in [13, 14] and its semantic extension [15]. Compared to traditional voxel maps [6], 3D occupancy NDT maps do not only store an occupancy value per voxel but also shape information, describing the underlying surface in terms of a normal distribution. These normal distributions can be updated efficiently in an incremental fashion [13]. To integrate panoptic information, we extend the tuple of data fields held in each voxel v of the map to:

$$v = \langle v^{\text{Shape}}, v^{\text{Occ}}, v^L, v^Z, v^P \rangle$$

The shape data field v^{Shape} stores a mean vector $\boldsymbol{\mu}(v) \in \mathbb{R}^3$ and a covariance matrix $\boldsymbol{\Sigma}(v) \in \mathbb{R}^{3\times3}$, describing the normal distribution of the surface. The occupancy data field v^{Occ} holds the occupancy $o(v)$ as log odd. As we focus on panoptic mapping, we refer to [13, 14] for the exact update process of these data fields. The additional data fields v^L, v^Z, store information for semantic and instances as histogram $\boldsymbol{h}^L(v) \in \mathbb{R}^{|\mathcal{L}|}$ and $\boldsymbol{h}^Z(v) \in \mathbb{R}^{|\mathcal{Z}^{\text{3D}}|}$, respectively. Moreover, we track the total number of increments for both histograms, denoted as $n^L(v) \in \mathbb{N}$ and $n^Z(v) \in \mathbb{N}$. The panoptic information $P(v) \in \mathcal{P}^{\text{3D}}$ is stored in v^P.

Note that we simplified the data representation for explanation purposes. In fact, for efficiency reasons, we also store the sum of both histograms and the proportion of stuff classes in the semantic histogram. Moreover, the instance histogram is sparse, with a maximum of 16 entries holding only entries for globally unique instance IDs observed for that voxel. We further keep this smaller histogram ordered and replace the smallest entry if the limit of 16 is reached.

The voxels are managed in an octree structure. In the following, we denote the mapping from $\boldsymbol{u}$ to a voxel v as:

$$v = \text{utov}(\boldsymbol{u}) := \text{findnode}(\boldsymbol{T}\boldsymbol{K}^{-1}\,[\boldsymbol{u}, 1]^T\, D(\boldsymbol{u})) \quad (2)$$

with $\boldsymbol{T}$, $\boldsymbol{K}$ being the current extrinsic and intrinsic camera parameters. $D(\boldsymbol{u})$ is the current depth value along the ray, and $\text{findnode}(\cdot)$ queries a voxel based on a point in 3D. Given this forward mapping, we can derive a corresponding mapping $\text{vtou}(v)$ that maps a voxel's information back to a set of coordinates $\mathcal{U}$ on the camera plane according to the NDT information of v. Note that the mappings are done once per frame.

D. Panoptic Mapping: Instance Update

We first perform the instance matching step to resolve the inconsistency in the instance IDs. Let $\mathcal{V}$ denote the set of voxels in the camera frustum affected by the forward mapping of the current observation (see Eq.2); then, we first project each relevant instance $\tilde{z} \in \mathcal{Z}^{\text{3D}}$ back to the camera plane and create a mask $\boldsymbol{M}^{\tilde{z}}$ for each instance. The elements $M^{\tilde{z}}(\boldsymbol{u})$ are updated as follows:

$$M^{\tilde{z}}(\boldsymbol{u}') = \begin{cases} 1 & \text{isthing}(v) \wedge \text{isintopz}(v, \tilde{z}) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$
$$\boldsymbol{u}' \in \text{vtou}(v), \quad v \in \mathcal{V}, \quad \tilde{z} \in \mathcal{Z}^{\text{3D}}.$$

Here, $\text{isthing}(v)$ ensures that voxels currently considered as stuff are ignored even if they contain any instance information. The function applies a threshold θ^{St} to the stuff proportion in the semantic histogram and is defined as:

$$\text{isthing}(v) := \frac{\sum_{l\in\mathcal{L}^{\text{St}}} h_l^L(v)}{\sum_{l\in\mathcal{L}} h_l^L(v)} < \theta^{\text{St}}. \quad (4)$$

Note that the instance information of previous observations is preserved and might be reactivated if the stuff/thing assignment of v changes. Tuning this threshold further helps to ignore false positives due to an imprecise segmentation at object borders that blended into the background (often stuff classes). The function $\text{isintopz}(v, z)$ ensures that only the most relevant instances (with the highest histogram scores) are back-projected. It is defined as:

$$\text{isintopz}(v, z) := \frac{\sum_{i=0}^{o_i=z} h_{o_i}^Z(v)}{\sum_{i\in\mathcal{Z}^{\text{3D}}} h_i^Z(v)} \geq 1 - \theta^{\text{B}} \quad \text{with} \quad \boldsymbol{o} = \text{argsort}^{\uparrow}(\boldsymbol{h}^Z(v)). \quad (5)$$

We choose $\theta^{\text{B}} = 0.8$ to project only ˜80% of the instance knowledge back for matching. This excludes only sporadically observed instances and, thus, speeds up matching.

Given the back-projected instance masks, we then compute the intersection over union (IoU) – also known as Jaccard index – between all back-projected instances and the currently observed instances:

$$J(z_a, z_b) = \text{IoU}(\{\boldsymbol{u}'|M^{z_a}(\boldsymbol{u}')=1\}, \{\boldsymbol{u}'|Z^P(\boldsymbol{u}')=z_b\}). \quad (6)$$

Finally, we derive the matched instance ID $\hat{Z}(\boldsymbol{u})$ based on:

$$\hat{z} = \begin{cases} \underset{\tilde{z}\in\mathcal{Z}^{\text{3D}}}{\text{argmax}}(J(\tilde{z}, z)) & \underset{\tilde{z}\in\mathcal{Z}^{\text{3D}}}{\max}(J(\tilde{z}, z)) > \theta^{\text{M}} \\ \text{newz}(z) & \underset{\tilde{z}\in\mathcal{Z}^{\text{3D}}}{\max}(J(\tilde{z}, z)) \leq \theta^{\text{N}} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

with $\hat{z} = \hat{Z}(\boldsymbol{u})$ and $z = Z^P(\boldsymbol{u})$. The parameters θ^{M} and θ^{N}, with $\theta^{\text{M}} \geq \theta^{\text{N}}$, define a minimum threshold for accepting matches and an upper limit for creating new instances, respectively, and $\text{newz}(z)$ defines a function that creates a new globally unique instance ID that is consistent for a given z within an update step. Note that choosing $\theta^{\text{N}} < \theta^{\text{M}}$ restricts instance creation and further allows increasing θ^{M} in order to prevent under-segmentation. Moreover, note that the instance matching is not exclusive, i.e., multiple predicted instances can match the same map instance. This compensates for temporary over-segmentation in the prediction.

Given the matched instances, the final instance update is straightforward. If the panoptic score of a matched instance exceeds a minimum threshold of θ^Z, we update the map and increment the corresponding histogram bin in each voxel by the panoptic score as well as update the observation counter:

$$h_{\hat{Z}(\boldsymbol{u}')}^Z(\text{utov}(\boldsymbol{u}')) \mathrel{+}= S^P(\boldsymbol{u}'), \quad n^Z(\text{utov}(\boldsymbol{u}')) \mathrel{+}= 1$$
$$\boldsymbol{u}' \in \left\{\boldsymbol{u}|\hat{Z}(\boldsymbol{u}) \neq 0 \wedge S^P(\boldsymbol{u}) > \theta^Z\right\}. \quad (8)$$

We use the panoptic score instead of the instance score to account for a possible inconsistent semantic in the first stage.

E. Panoptic Mapping: Semantic Update

The semantic update is performed subsequently to ensure consistent semantic information during back-projection. As the semantic predictions are independent, the update step is much simpler. If the semantic score of a panoptic label exceeds a minimum threshold of θ^L, we increment the corresponding histogram bin in each voxel by the semantic score and update the observation counter:

$$h^L_{L^P(\boldsymbol{u}')}(\mathrm{utov}(\boldsymbol{u}')) \mathrel{+}= S^{L^P}(\boldsymbol{u}'),\ n^L(\mathrm{utov}(\boldsymbol{u}')) \mathrel{+}= 1 \quad \boldsymbol{u}' \in \left\{ \boldsymbol{u} | S^{L^P}(\boldsymbol{u}) > \theta^L \right\}. \tag{9}$$

F. Panoptic Mapping: Panoptic Label Propagation

To derive the panoptic label $P(v) \in \mathcal{P}^{\mathrm{3D}}$ for each voxel, we follow a similar approach as for 2D:

$$P(v) = \begin{cases} \langle \underset{l \in \mathcal{L}^{\mathrm{Th}}}{\mathrm{argmax}}(\underset{v' \in \{i | \hat{Z}(i) = \hat{Z}(v)\}}{\mathrm{sum}}(h^L_l(v'))) \, , \hat{Z}(v) \rangle & \mathrm{pthing}(v) \\ \langle \hat{L}(v), 0 \rangle & \mathrm{otherw.} \end{cases} \tag{10}$$

with $\hat{L}(v) = \mathrm{argmax}_{l \in \mathcal{L}}(h^L_l(v))$ and $\hat{Z}(v) = \mathrm{argmax}_{i \in \mathcal{Z}^{\mathrm{3D}}}(h^Z_i(v))$. Here, $\mathrm{pthing}(v)$ determines whether to propagate thing information. It extends the previous $\mathrm{isthing}(v)$ and further incorporates the observation counters:

$$\mathrm{pthing}(v) := \mathrm{isthing}(v) \wedge \frac{n^Z(v)}{n^L(v)} \geq \theta^O \tag{11}$$

An observation ratio less than the threshold θ^O marks a voxel to currently contain only weak instance knowledge, i.e., even if it belongs to thing, only a fraction of θ^O of the observations entered the map as valid instance observations. Note, due to Eq. 10, this may introduce a garbage segment for thing classes. However, it prevents over-segmentation and is of great importance for real-world application, as it indicates thing areas that could not be separated into instances.

IV. Experiments

We evaluate the performance of our proposed panoptic mapping approach on the publicly available datasets Hypersim [5] and ScanNetV2 [16] in 3D as well as in 2D. In the following, we first introduce both datasets and describe the exact evaluation protocol. Subsequently, we provide details about the training of EMSANet and give further implementation details. Finally, we present quantitative and qualitative results demonstrating the performance of PanopticNDT.

A. Datasets

Both Hypersim and ScanNetV2 feature images for RGB and depth, dense panoptic annotations as well as the corresponding intrinsic and extrinsic camera parameters. Both datasets comprise a training, validation, and test split.

Hypersim [5]: Hypersim is a synthetic dataset with 77,400 photorealistic images of 461 indoor scenes rendered in 774 camera trajectories. Each trajectory is generated by a random walk and consists of 100 camera poses, ensuring reasonable visual coverage. The rendered images have a spatial resolution of 1024×768 pixels. However, some scenes were rendered using tilt-shift photography, which is incompatible with the simpler pinhole camera model used in robotic frameworks such as MIRA and ROS. This becomes a problem when projecting depth images into 3D space for mapping. Unfortunately, re-rendering the images is difficult as it requires buying all underlying assets. Therefore, we projected all annotations into 3D space using the provided camera parameters and then back to the simpler pinhole camera. This inevitably leads to an information loss of ˜5.3% as it is impossible to back-project all pixels without introducing ambiguities. However, this approach is still better than excluding scenes. We follow [15] to further filter invalid trajectories (void/single semantic class only, missing textures, invalid depth) from training and validation split.

ScanNetV2 [16]: ScanNetV2, on the other hand, is a real-world dataset for indoor scene understanding. It comprises 2.5M images of 807 indoor scenes captured in 1613 camera trajectories. Depth and RGB were captured at a spatial resolution of 640×480 and 1280×960 pixels, respectively. As the dataset is created from video sequences, it contains many similar images. We use a subsampling of 5 in our pipeline to reduce the number of samples.

Both datasets provide annotations for the 40 semantic classes of NYUv2 [36]. Following [4, 26], we treat wall, floor, and ceiling to be background (stuff classes). The ScanNetV2 benchmark further excludes 20 classes due to few annotations, which we do as well but only for ScanNetV2. We use the training split for network training and the validation split for tuning hyperparameters in our pipeline. The test split is only used for reporting results. For further details on data processing, we refer to our GitHub repository.

B. Evaluation Protocol

We build upon the publicly available ScanNetV2 benchmark pipeline [16] and evaluate our approach in 3D and 2D. Unfortunately, this pipeline does not feature a panoptic evaluation. Therefore, we extend the existing pipeline with a complementary panoptic evaluation task. The evaluation code is shared at GitHub. The mean intersection over union (mIoU) is reported for evaluating semantic information and the panoptic quality (PQ) [37] as well as the mIoU_P for evaluating panoptic information and the semantic of panoptic information, respectively. As it is part of ScanNetV2 instance evaluation on the hidden test split, we also report the average precision at 50% overlap (AP_{50}) for evaluating instances. Note, according to [37], PQ and AP track closely and an improvement in AP is expected to also improve PQ.

3D Evaluation: In the ScanNetV2 benchmark, evaluation in 3D is done by mapping created representations to annotated ground-truth point clouds. While ScanNetV2 provides annotated meshes as ground truth, such ground-truth representations are missing for Hypersim. Therefore, we generated them ourselves by applying a voxel-grid filter with a voxel size of $1\mathrm{cm}$ to the point cloud of each camera trajectory. The most frequent annotation is used to assign a ground-truth annotation to each voxel. For mapping points to our NDT map, the naïve way would be to assign a

ground-truth point by simply treating the NDT map as a voxel map, i.e., by querying the voxel into which the point would fall. However, as we have shape information in terms of an estimated normal distribution (see Sec. III-C), we can make a more informed matching. We compute the Mahalanobis distances between each ground-truth point and the normal distributions of the corresponding NDT voxel and its immediate neighborhood and assign the annotation based on the smallest distance. Every ground-truth point that cannot be matched is labeled as *unknown* (void).

2D Evaluation: Although it is natural to evaluate in 3D, there is a major drawback. As created representations are mapped to ground-truth representations, the actual spatial resolution of a created representation has less influence, i.e., modeling free space is less relevant. Therefore, we follow [8, 10, 15] and further evaluate back-projections of the map to the camera plane (see Sec. III-D). This evaluation also allows for a comparison to the raw network predictions and, thereby, assessing the temporal integration is possible.

C. Network Training & Implementation Details

We use the PyTorch implementation of EMSANet for network training [4]. The network input resolution is chosen based on the resolution of the RGB images. Network training was done for 500 epochs. For optimization, we used SGD with momentum and performed a grid search for determining a suitable learning rate. For each dataset, we derived the best configuration based on PQ on the corresponding validation split. We refer to the implementation at GitHub for the exact hyperparameters and the network weights. The results of EMSANet as well as the inference throughput is reported next to the mapping results in Tab. I and Tab. II.

Mapping is done at the resolution of provided depth images, i.e., at 1024×768 for Hypersim and at 640×480 for ScanNetV2. We further follow [10, 12, 15] and limit the maximum mapping distance for Hypersim to a reasonable value of $20\mathrm{m}$. Independent of the dataset, we choose the stuff proportion threshold θ^{St} (see Eq. 4) to 0.9; the thresholds for matching instances θ^{M} and creating new instances θ^{N} (see Eq. 7) to 0.2 and 0.1; and the instance to semantic observation ratio threshold θ^{O} (see Eq. 11) to 0.25. The score thresholds θ^{L} and θ^{Z} (see Eq. 8 and Eq. 9) are set to 0.7 and 0.4 for Hypersim; and to 0.7 and 0.1 for ScanNetV2.

D. Results on Hypersim

First, we evaluate our panoptic mapping approach on Hypersim in a setting assuming perfect data, i.e., using the provided ground-truth panoptic segmentation as well as perfect depth data for mapping. The goal is to determine how well panoptic information can be represented in the resulting maps. Tab. I (upper part) lists the results and further compares them to Panoptic Multi-TSDFs [31], a voxel-based version of our approach (ignoring NDT shape information), and the semantic NDT approach of [15] we build upon.

For both splits similar trends emerge. First, in 3D, PanopticNDT performs better with decreasing voxel size, while the step from $10\mathrm{cm}$ to $5\mathrm{cm}$ is already much smaller than the step from $20\mathrm{cm}$ to $10\mathrm{cm}$. A voxel size of $5\mathrm{cm}$ may already be close to the upper limit of our approach for representing panoptic information in 3D. All settings perform significantly better than Panoptic Multi-TSDFs. In 2D, i.e., when accounting for modeling free space accurately, the steps between the voxel sizes are larger for all metrics, however, again, the steps get smaller. Nevertheless, the results indicate that Hypersim contains a lot of small objects that require accurate modeling. The comparison to the simple voxel-based version of our approach (denoted as w/o NDT) shows that the NDT representation enables choosing the next coarser voxel size while still achieving at least the same precision.

Next, we use the outputs of EMSANet (see Tab. I (center) for results) as input for panoptic mapping. The results of the 3D evaluation in Tab. I (lower part) show that PanopticNDT performs similarly regardless of the used voxel size. This indicates that the performance of our approach is mainly limited by the quality of the preceding panoptic segmentation. However, the results of the 2D evaluation suggest that our mapping approach is able to improve segmentation by integrating observations over time. For the test split, merging

TABLE I: Results on Hypersim validation and test split when mapping with ground truth (top) and predicted segmentation (bottom) of EMSANet [4] (center). See Sec. IV-B for details on the reported metrics. Legend: gray: mapping with semantic only instead of panoptic information (black), *: re-application in our data pipeline, †: mapping with inputs at 640×480 instead of 1024×768 pixels, $x\mathrm{cm}$: voxel size, w/o NDT: PanopticNDT but evaluated without shape information, and FPS: average update rate in frames per second on the hardware of our mobile robot, i.e., for single-threaded mapping: Intel NUC11PHKi7C (Intel i7-1165G7) and for EMSANet: NVIDIA Jetson AGX Orin (Jetpack 5.0.2, TensorRT 8.4, Float16, 30W / 50W).

		Valid						Test						
		3D			2D			3D			2D			
		mIoU ↑	$\mathrm{mIoU_P}$ ↑	PQ ↑	mIoU ↑	$\mathrm{mIoU_P}$ ↑	PQ ↑	mIoU ↑	$\mathrm{mIoU_P}$ ↑	PQ ↑	mIoU ↑	$\mathrm{mIoU_P}$ ↑	PQ ↑	FPS ↑
Ground Truth	PanopticNDT (5cm)	91.84	91.36	76.04	84.46	84.18	65.78	87.22	86.74	73.30	84.68	84.34	67.67	1.10
	PanopticNDT (10cm)	90.31	89.04	73.24	76.84	75.87	53.40	87.01	86.24	69.39	77.54	77.02	57.03	2.73
	PanopticNDT (20cm)	83.30	81.03	61.52	64.71	63.37	37.57	82.07	80.29	59.86	64.46	63.35	40.77	3.35
	Panoptic Multi-TSDFs [31]*	—	70.82	51.00	—	—	—	—	69.69	54.69	—	—	—	2.95†
	PanopticNDT (5cm, w/o NDT)	91.33	90.94	75.55	70.37	70.60	48.47	86.84	86.40	72.80	71.85	71.87	52.17	—
	SemanticNDT [15]* (10cm)	90.31	—	—	76.84	—	—	87.01	—	—	77.54	—	—	4.32
	EMSANet	—	—	—	49.74	49.12	34.95	—	—	—	46.66	44.66	29.77	24.0 / 35.5
Prediction	PanopticNDT (5cm)	45.53	45.10	30.37	52.50	53.50	34.23	44.86	45.09	26.99	48.82	50.27	32.36	1.11
	PanopticNDT (10cm)	45.43	44.56	31.08	48.93	49.00	30.44	45.34	45.20	27.54	46.10	47.01	28.82	2.73
	PanopticNDT (20cm)	43.60	42.14	28.84	42.94	42.53	22.70	44.29	44.08	25.98	40.24	40.47	22.90	3.41
	Panoptic Multi-TSDFs [31]*	—	30.85	15.92	—	—	—	—	30.59	15.98	—	—	—	1.25†
	SemanticNDT [15]* (10cm)	44.31	—	—	48.45	—	—	44.80	—	—	45.56	—	—	4.32

semantic and instance predictions of EMSANet to derive panoptic segmentations (input for panoptic mappings) leads to a drop in mIoU from 46.66% to 44.66%. Integrating these merged panoptic observations over time in a panoptic NDT map with a voxel size of 10cm already almost closes this gap (mIoU: 46.10%) even if there is some unavoidable loss due to the much coarser map representation (see results with ground truth). Further combining the aggregated semantic and instance knowledge in the map by propagating panoptic labels additionally improves results (mIoU: 47.01%). For the smaller voxel size of 5cm, this effect is even stronger and also affects PQ. Fig. 3 depicts the per-class PQs and compares to the EMSANet result. It becomes obvious that the increase in PQ comes from improvements across most thing classes. Only for the stuff class floor is some larger decrease. Fig. 4 (top) further visualizes qualitative results.

The performance of Panoptic Multi-TSDFs [31] notably decreases when mapping with EMSANet outputs. We observed that this drop is mainly caused by mapping issues for smaller objects in thing classes. We assume that the submap approach is not able to account for such noisy observations.

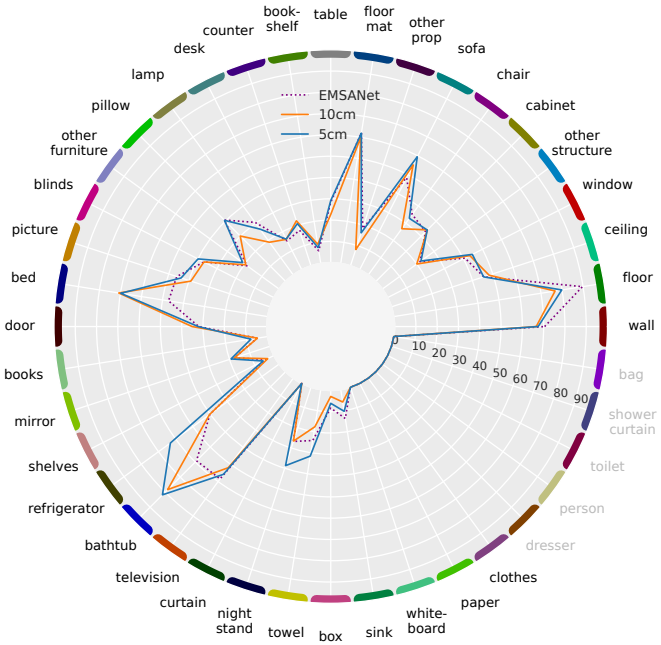

Fig. 3. Per-class 2D panoptic quality on the Hypersim test split for EMSANet and when mapping with its predictions and voxel sizes of 10cm and 5cm. Classes printed in gray do not appear in the test split.

E. Results on ScanNetV2

With the results of our experiments on Hypersim at hand, we evaluate our mapping pipeline on the real-world dataset ScanNetV2. The results are listed in Tab. II. Although depth data is not perfect anymore, the results are higher than for Hypersim. This is due to limiting the semantic classes to frequent classes and less small objects. However, PQ results also indicate that imperfect depth data affect back-projection and matching. Nevertheless, compared to the state-of-the-art approaches Panoptic Multi-TSDFs and Panoptic Fusion, our PanopticNDT approach is still much more stable. We assume that the bottom-up design of our approach as well as modeling the distributions for semantic and instance observations instead of storing a single label are the main reasons for better stability. This also holds true when evaluating on the hidden test split. Fig. 4 shows qualitative results proving the real-world applicability of our approach.

V. Application in Real-World Scenario

In addition to just evaluating on benchmark datasets, we further present qualitative results in a real-world application in the context of our MORPHIA project [2]. After an initial mapping phase using RTAB-Map, it is switched to localization mode to provide a reliable long-term localization. For efficient short-term panoptic mapping, we rely on our proposed PanopticNDT with a voxel size of 10cm. Panoptic segmentation is done using EMSANet-R34-NBt1D [4] trained on NYUv2 [36], Hypersim [5], SUNRGB-D [38], and ScanNetV2 [16]. Even though training data do not contain any images of our Kinect Azure sensor setup, the results prove the real-world applicability of our approach. For impressions, we refer to `https://youtu.be/xS9jCEKO-Uw`.

VI. Conclusion

In this paper, we have presented a novel approach for incorporating panoptic information into 3D representations. The approach has been integrated into efficient occupancy NDT maps, realizing panoptic occupancy NDT (PanopticNDT) maps. Our experiments on Hypersim and ScanNetV2 reveal that our approach represents panoptic information at a higher level of detail than other state-of-the-art approaches and enables real-time panoptic mapping on mobile robots. We have further demonstrated that the approach can be deployed to real-world applications, enabling mobile robots to gain a strong scene understanding of their environment.

TABLE II: Results on ScanNetV2 validation and hidden test split when mapping with predicted segmentation of EMSANet [4]. See Sec. IV-B for details on the reported metrics. Legend: gray: mapping with semantic only instead of panoptic information (black), *: re-application in our data pipeline, $x$cm: voxel size, †: expected to be much slower (see Sec. II-C), and FPS: average update rate in frames per second on our mobile robot, i.e., for single-threaded mapping: Intel NUC11PHKi7C (Intel i7-1165G7) and for EMSANet: NVIDIA Jetson AGX Orin (Jetpack 5.0.2, TensorRT 8.4, Float16, 30W / 50W).

	Valid								**Test**				
	3D				**2D**				**3D**		**2D**		
	mIoU $\uparrow$	**mIoU**$_\mathbf{P}^\uparrow$	**PQ** $\uparrow$	**AP**$_{50}^\uparrow$	**mIoU** $\uparrow$	**mIoU**$_\mathbf{P}^\uparrow$	**PQ** $\uparrow$	**AP**$_{50}^\uparrow$	**mIoU**$_\mathbf{P}^\uparrow$	**AP**$_{50}^\uparrow$	**mIoU**$_\mathbf{P}^\uparrow$	**AP**$_{50}^\uparrow$	**FPS** $\uparrow$
EMSANet	—	—	—	—	70.99	66.19	58.22	41.94	—	—	60.0	38.0	15.4 / 23.1
PanopticNDT (5cm)	70.07	69.37	57.38	52.79	68.31	67.71	52.48	39.18	—	—	—	—	1.25
PanopticNDT (10cm)	69.47	68.39	59.19	52.65	67.70	66.17	50.48	37.76	68.1	50.9	64.8	39.8	7.77
Panoptic Multi-TSDFs [31]*	—	47.29	34.75	—	—	—	—	—	—	—	—	—	5.85
Panoptic Fusion [26]	—	—	33.5	—	—	—	—	—	52.9	47.8	—	—	†
SemanticNDT [15]* (10cm)	69.59	—	—	—	67.71	—	—	—	—	—	—	—	8.73

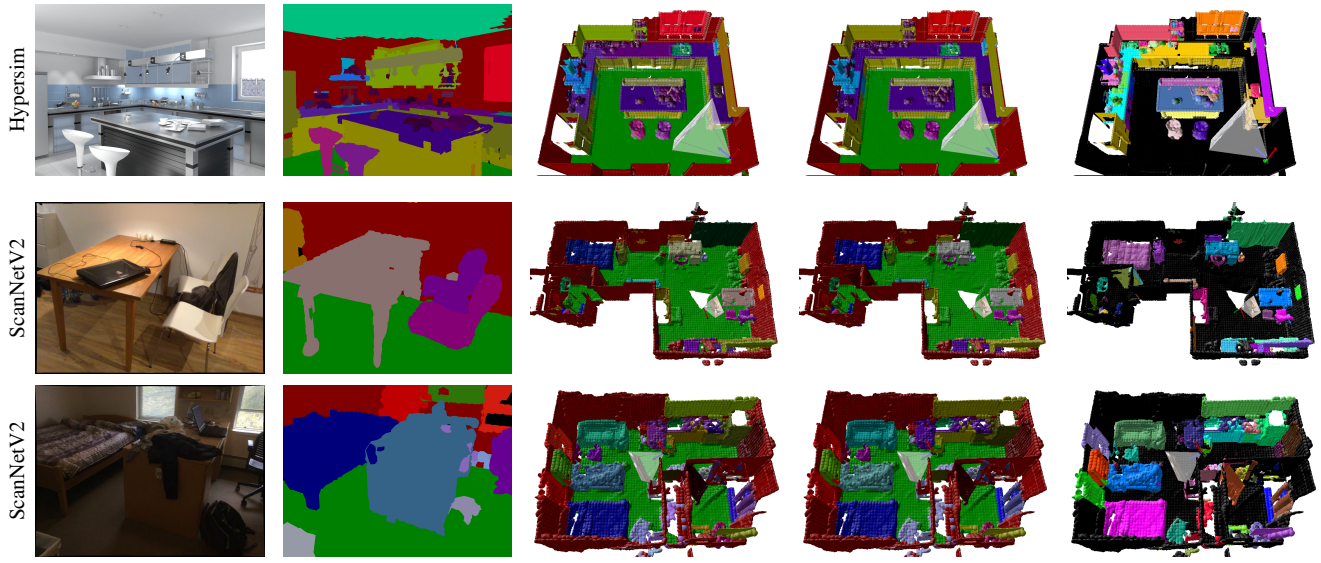

Fig. 4. Qualitative results for scene *ai_001_10* of the Hypersim test split (top) and for *scene_0757_00* and *scene_0761_00* of the hidden ScanNetV2 test split (bottom) when mapping with EMSANet predictions and voxel size 10cm. Left to right: color image and panoptic back-projection for the given camera pose (see 3D scenes), panoptic, panoptic semantic, and panoptic instance NDT map. Best viewed in color at 200%. Black indicates *void/no_instance*, for the semantic colors, we refer to Fig. 3. Panoptic labels are visualized by small color differences based on the semantic color.

REFERENCES

[1] S. B. Fischedick, K. Richter, T. Wengefeld, *et al.*, "Bridging Distance with a Collaborative Telepresence Robot for Older Adults – Report on Progress in the CO-HUMANICS Project," in *Proc. of ISR*, 2023.

[2] T. Wengefeld, B. Schuetz, G. Girdziunaite, *et al.*, "The MORPHIA Project: First Results of a Long-Term User Study in an Elderly Care Scenario from Robotic Point of View," in *Proc. of ISR*, 2022.

[3] M. Labbé and F. Michaud, "RTAB-Map as an Open-Source Lidar and Visual Simultaneous Localization and Mapping Library for large-scale and long-term Online Operation," *JFR*, 2019.

[4] D. Seichter, S. Fischedick, M. Köhler, and H.-M. Gross, "Efficient Multi-Task RGB-D Scene Analysis for Indoor Environments," in *Proc. of IJCNN*, 2022.

[5] M. Roberts, J. Ramapuram, A. Ranjan, *et al.*, "Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding," in *Proc. of ICCV*, 2021.

[6] A. Hornung, K. M. Wurm, M. Bennewitz, *et al.*, "OctoMap: An Efficient Probabilistic 3D Mapping Framework based on Octrees," *AR*, 2013.

[7] J. Wang and B. Englot, "Fast, Accurate Gaussian Process Occupancy Maps via Test-Data Octrees and Nested Bayesian Fusion," in *Proc. of ICRA*, 2016.

[8] K. Doherty, J. Wang, and B. Englot, "Bayesian Generalized Kernel Inference for Occupancy Map Prediction," in *Proc. of ICRA*, 2017.

[9] S. Sengupta and P. Sturgess, "Semantic Octree: Unifying Recognition, Reconstruction and Representation via an Octree Constrained Higher Order MRF," in *Proc. of ICRA*, 2015.

[10] S. Yang, Y. Huang, and S. Scherer, "Semantic 3D Occupancy Mapping through Efficient High Order CRFs," in *Proc. of IROS*, 2017.

[11] M. Jadidi, L. Gan, S. Parkison, *et al.*, "Gaussian Processes Semantic Map Representation," *ArXiv preprint arXiv:1707.01532*, 2017.

[12] L. Gan, R. Zhang, J. W. Grizzle, *et al.*, "Bayesian Spatial Kernel Smoothing for Scalable Dense Semantic Mapping," *RAL*, 2020.

[13] E. Einhorn and H.-M. Gross, "Generic 2D/3D SLAM with NDT Maps for Lifelong Application," in *Proc. of ECMR*, 2013.

[14] E. Einhorn, "Visuelle Umgebungswahrnehmung und Kartierung zur Navigation mobiler Serviceroboter in realen Einsatzumgebungen," Ph.D. dissertation, Ilmenau University of Technology, 2019.

[15] D. Seichter, P. Langer, T. Wengefeld, *et al.*, "Efficient and Robust Semantic Mapping for Indoor Environments," in *Proc. of ICRA*, 2022.

[16] A. Dai, A. X. Chang, M. Savva, *et al.*, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes," in *Proc. of CVPR*, 2017.

[17] Y. Zhou, W. Wang, W. Guan, *et al.*, "Visual Robotic Object Grasping Through Combining RGB-D Data and 3D Meshes," in *Proc. of MMM*, 2017.

[18] D. Shin, C. C. Fowlkes, and D. Hoiem, "Pixels, Voxels, and Views: A Study of Shape Representations for Single View 3D Object Shape Prediction," in *Proc. of CVPR*, 2018.

[19] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale using Voxel Hashing," *ACMTG*, 2013.

[20] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Proc. of PACMCGIT*, 1996.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *Proc. of IROS*, 2017.

[22] J. Kläß, J. Stückler, and S. Behnke, "Efficient Mobile Robot Navigation using 3D Surfel Grid Maps," in *Proc. in ROBOTIK*, 2012.

[23] P. Biber and W. Straßer, "The Normal Distributions Transform: A new Approach to Laser Scan Matching," in *Proc. of IROS*, 2003.

[24] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan Registration for Autonomous Mining Vehicles using 3D-NDT," *JFR*, 2007.

[25] T. Stoyanov, M. Magnusson, H. Almqvist, and A. J. Lilienthal, "On the Accuracy of the 3D Normal Distributions Transform as a Tool for Spatial Representation," in *Proc. of ICRA*, 2011.

[26] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "PanopticFusion: Online volumetric semantic mapping at the level of stuff and things," in *Proc. of IROS*, 2019.

[27] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. of CVPR*, 2017.

[28] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *Proc. of ICCV*, 2017.

[29] M. Grinvald, F. Furrer, T. Novkovic, *et al.*, "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery," *RAL*, 2019.

[30] M. Han, Z. Zhang, Z. Jiao, *et al.*, "Reconstructing interactive 3d scenes by panoptic mapping and cad model alignments," in *Proc. of ICRA*, 2021.

[31] L. Schmid, J. Delmerico, J. Schönberger, *et al.*, "Panoptic Multi-TSDFs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency," in *Proc. of ICRA*, 2022.

[32] E. Einhorn, T. Langner, R. Stricker, *et al.*, "MIRA – Middleware for Robotic Applications," in *Proc. of IROS*, 2012.

[33] D. Seichter, M. Köhler, B. Lewandowski, *et al.*, "Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis," in *Proc. of ICRA*, 2021.

[34] Q.-H. Pham, B.-S. Hua, T. Nguyen, *et al.*, "Real-time Progressive 3D Semantic Segmentation for Indoor Scenes," in *Proc. of WACV*, 2019.

[35] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *ITS*, 2018.

[36] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Proc. of ECCV*, 2012.

[37] A. Kirillov, K. He, R. Girshick, *et al.*, "Panoptic Segmentation," in *Proc. of CVPR*, 2019.

[38] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite," in *Proc. of CVPR*, 2015.

[VII Appendix]
PanopticNDT: Efficient and Robust Panoptic Mapping

Daniel Seichter, Benedict Stephan, Söhnke Benedikt Fischedick, Steffen Müller, Leonard Rabes, and Horst-Michael Gross

Due to the space restrictions, some details had to be omitted from the main part of the paper. We present them here. In Sec. VII-A and VII-B, we give further details on the training of EMSANet [4] and on our adapted implementation of Panoptic Multi-TSDFs [31] that serves for comparing PanopticNDT to Panoptic Multi-TSDFs. Afterward, in Sec. VII-C and VII-D, we present omitted quantitative and qualitative results. Finally, in Sec. VII-E, we give more details on the network and the pipeline used for real-world application.

A. Further Network Details

Network training was done with a batch size of 8 and without any further pre-training. Only the encoder was pre-trained on the ImageNet dataset [4]. As the number of samples in both Hypersim and ScanNetV2 is large, we used a random subsampling in each epoch of 30% and 25% for Hypersim and ScanNetV2, respectively. For ScanNetV2, we further applied a fixed subsampling of 50 to both the training and the validation split. For both networks, the full training command including all hyperparameters is shared in our GitHub repository next to the network weights. As we focus on fast inference for real-world applications, we did not apply any test-time tricks, such as horizontal flipping or multiscale inputs, when extracting panoptic segmentations for mapping.

B. Further Details on Panoptic Multi-TSDFs

To apply Panoptic Multi-TSDFs [31] in our data pipeline, we adapted the implementation available at GitHub[1]. We created additional data loaders for both Hypersim and ScanNetV2. Independently of the dataset, mapping was done with inputs at the default resolution of 640×480 pixels. Moreover, as both datasets follow the same semantic classes, we set the voxel sizes only once and used them for both datasets likewise. We followed the already predefined sizes for assigning the voxel size for each class. For *void* and the stuff classes wall, floor, and ceiling, the preset large (5cm) was used. For thing classes, the preset small (2cm) or medium (3cm) was used depending the object appearance. Note that this means that Panoptic Multi-TSDFs features much finer voxel sizes than our proposed PanopticNDT. For the remaining hyperparameters, we performed a grid search. While most hyperparameters remained at their default value, we identified three hyperparameters with larger impact. Tab. III lists these hyperparameters and assigns the values used for reporting results in our experiments. It is important to note that we achieved better results when disabling the submap deactivation (last parameter in Tab. III) completely. However, this would also mean that the change detection – one of the core features of Panoptic Multi-TSDFs – gets deactivated. Therefore, we decided for a compromise and set the deactivation parameter to 20 frames.

TABLE III: Most relevant hyperparameters for Panoptic Multi-TSDFs [31]. * indicates that even higher values may improve panoptic results. However, this would also mean that the change detection – one of the core features of Panoptic Multi-TSDFs – gets deactivated. Therefore, we decided for a compromise and set this parameter to 20 frames.

Parameter	Default	Hypersim	ScanNetV2
id_tracker/ match_acceptance_threshold	0.1	0.1	0.15
tsdf_integrator/ use_instance_classification	false	true	true
map_management/activity_manager/ deactivate_after_missed_detections	5	20*	20*

As Panoptic Multi-TSDFs itself does not provide a dedicated pipeline for panoptic evaluation, we implemented such a pipeline ourselves in order to be able to compare our PanopticNDT to Panoptic Multi-TSDFs. For evaluation in 3D, i.e., mapping the created representations to ground-truth point clouds, each point of the ground-truth point cloud is queried inside the map according to the defined order described in [31]. Conflicts due to overlapping submaps are resolved using the signed distance and the belonging probability. The latter describes the likelihood of a voxel being part of a submap. This procedure was also suggested by the authors of [31] in a related GitHub issue[2]. After deriving the 3D representations, we follow the evaluation protocol described in Sec. IV-B.

C. Further Quantitative Results

Due to the restricted space in the main part, we decided to omit the results for mapping with ground-truth annotations on ScanNetV2. Tab. IV complements Tab. II and shows the full results for ScanNetV2.

When mapping with PanopticNDT on ScanNetV2 in the ground-truth setting, similar trends as for Hypersim (see results in Tab. I) emerge. PanopticNDT performs better with decreasing voxel size, while the step from 10cm to 5cm is already much smaller than the step from 20cm to 10cm. A voxel size of 5cm may already be close to the upper limit of

[1]GitHub Panoptic Multi-TSDFs: https://github.com/ethz-asl/panoptic_mapping/ (online: 28.07.2023)

[2]Related GitHub issue: https://github.com/ethz-asl/panoptic_mapping/issues/67 (online: 28.07.2023)

TABLE IV: Results on ScanNetV2 validation and hidden test split when mapping with ground truth (top) and predicted segmentation (bottom) of EMSANet [4] (center). The results listed in this table complement the results shown in Tab. II. See Sec. IV-B for details on the reported metrics. Legend: gray: mapping with semantic only instead of panoptic information (black), *: re-application in our data pipeline, $x\mathrm{cm}$: voxel size, †: expected to be much slower (see Sec. II-C), and FPS: average update rate in frames per second on our mobile robot, i.e., for single-threaded mapping: Intel NUC11PHKi7C (Intel i7-1165G7) and for EMSANet: NVIDIA Jetson AGX Orin (Jetpack 5.0.2, TensorRT 8.4, Float16, 30W / 50W).

		Valid								Test				
		3D				2D				3D		2D		
		mIoU ↑	$\mathrm{mIoU_P}$ ↑	PQ ↑	$\mathrm{AP_{50}}$ ↑	mIoU ↑	$\mathrm{mIoU_P}$ ↑	PQ ↑	$\mathrm{AP_{50}}$ ↑	$\mathrm{mIoU_P}$ ↑	$\mathrm{AP_{50}}$ ↑	$\mathrm{mIoU_P}$ ↑	$\mathrm{AP_{50}}$ ↑	FPS ↑
GT	PanopticNDT (5cm)	89.91	90.22	80.30	94.14	85.91	85.73	68.20	77.15	—	—	—	—	1.23
GT	PanopticNDT (10cm)	87.23	87.77	78.44	91.13	82.06	81.30	61.22	62.82	—	—	—	—	7.51
GT	PanopticNDT (20cm)	81.36	82.21	72.92	82.27	72.09	71.24	48.02	44.03	—	—	—	—	11.47
GT	Panoptic Multi-TSDFs [31]*	—	61.46	47.11	54.30	—	—	—	—	—	—	—	—	6.45
GT	SemanticNDT [15]* (10cm)	87.05	—	—	—	82.11	—	—	—	—	—	—	—	8.73
	EMSANet	—	—	—	—	70.99	66.19	58.22	41.94	—	—	60.0	38.0	15.4 / 23.1
Prediction	PanopticNDT (5cm)	70.07	69.37	57.38	52.79	68.31	67.71	52.48	39.18	—	—	—	—	1.25
Prediction	PanopticNDT (10cm)	69.47	68.39	59.19	52.65	67.70	66.17	50.48	37.76	68.1	50.9	64.8	39.8	7.77
Prediction	PanopticNDT (20cm)	66.44	64.74	56.49	48.21	60.85	58.85	41.19	28.66	—	—	—	—	10.96
Prediction	Panoptic Multi-TSDFs [31]*	—	47.29	34.75	27.24	—	—	—	—	—	—	—	—	5.85
Prediction	Panoptic Fusion [26]	—	—	33.5	—	—	—	—	—	52.9	47.8	—	—	†
Prediction	SemanticNDT [15]* (10cm)	69.59	—	—	—	67.71	—	—	—	—	—	—	—	8.73

our approach for representing panoptic information, i.e., even smaller voxel sizes might not necessarily improve results. However, in contrast to Hypersim, the results for PQ are consistently higher. Moreover, especially in 2D, the results for 10cm and 5cm are much closer. Both observations can be attributed to fewer small instances in ScanNetV2.

D. Further Qualitative Results

Fig. 5 and Fig. 6 complement the qualitative results shown in Fig. 4 in the main part. For creating 3D visualizations for Panoptic Multi-TSDFs, we take the surface points[3] of all submaps and concatenate them in a single point cloud. For creating 2D visualizations, we follow the approximate back-projection that is already part of the system's internal instance tracker. To create the back-projection for a given camera pose, the surface points of all visible submaps – that are either active or persistent – are projected onto the 2D camera plane. To fill this sparse representation, each point receives an individually-sized rectangle at its center according to its distance to the camera and the voxel size of the related submap. For back-projecting panoptic information, we use the panoptic label of the submap as fill value for all related rectangles.

The results in Fig. 5 for mapping with ground-truth annotations show that both PanopticNDT and Panoptic Multi-TSDFs are capable of creating panoptic maps. However, PanopticNDT can handle instances much better than Panoptic Multi-TSDFs. Even with the larger voxel size of 10cm, a lot of small instances are still present in the resulting map. When switching to mapping with panoptic predictions of EMSANet, a lot of noise is introduced in the mapping process as the predictions of EMSANet are not perfect, especially for small instances far away from the camera (as shown in this example). However, PanopticNDT is still able to integrate such noisy predictions over time in a robust way, creating maps well suited for application. By contrast, Panoptic Multi-TSDFs almost fails completely in this setting. The bad performance most likely stems from the random camera trajectories together with the noisy panoptic predictions of EMSANet. Panoptic Multi-TSDFs expects temporal consistency between incoming frames to properly allocate, track, and deactivate submaps. Mapping with perfect ground-truth annotations can still be handled well since map regulation can fuse submaps and eliminates erroneous data by running comparisons between old and new information. This ability is likely impaired by additional errors and inaccuracies introduced by mapping with predictions of EMSANet.

The results in Fig. 6 for ScanNetV2 show a similar picture. In general, ScanNetV2 is less challenging as it has less semantic classes and less small objects. However, compared to Hypersim, there is additional noise in the depth measurements, making this dataset a good benchmark for real-world applications. PanopticNDT performs well in this setting. As also indicated by the results in Tab. II and Tab. IV, there is no reason to switch to the smallest voxel size in such a setting. By contrast, Panoptic Multi-TSDFs is struggling again. There are a lot of submaps overlapping each other. Moreover, the upper example shows that resolving overlaps using the change detection may also lead to missing geometry in the resulting map (the table is not present).

E. Further Details on Real-World Application

As already described in the main part and shown in the video, we also use EMSANet-R34-NBt1D [4] for predicting panoptic segmentations in our real-world applications. The application network processes inputs at the Hypersim input resolution (1024×768 pixels) but was trained with samples from the four datasets NYUv2 [36], Hypersim, SUNRGB-D [38], and ScanNetV2. We decided in favor of combining these datasets, as each dataset contributes additional information to the training process. Hypersim contains a lot of (small) instances that greatly boost instance segmentation. We observed that pre-training on Hypersim already helps a lot, but adding Hypersim examples to the target training has an even larger impact on the generalization capability of the final network for real-world application.

[3]During mapping, Panoptic Multi-TSDFs incrementally computes an iso-surface for each submap, describing the submap's surface by a set of points.

However, as Hypersim itself is large and provides only perfect depth data, we limited the number of samples and used a random subsampling of 10% in each epoch. The remaining three datasets are captured in real environments and, thus, provide valuable information for application. We used the raw depth in all datasets for training to force the network to cope with incomplete depth data. To prevent ScanNetV2 from prevailing the real-world part of the training data, we used a random subsampling of 25% in each epoch. NYUv2 and SUNRGB-D are used along with the additional annotations provided in [4]. Note that we used both datasets even though SUNRGB-D already contains all samples of NYUv2. However, in the SUNRGB-D version, the last three semantic classes are omitted and mapped to *void*. Using both datasets at the same time increases the number of real-world examples for these three classes. For SUNRGB-D, we further refined the pipeline of [4] for extracting instance annotations from 3D bounding boxes. We refer to the version of SUNRGB-D with refined instance annotations presented in this paper as PanopticNDT version. Tab. V shows that the total number of available instance annotations could be greatly increased. The refined annotations are shared in our GitHub repository. The best checkpoint was chosen based on the performance on the SUNRGB-D test split, as it has the biggest impact on our application due its variability in scenes and cameras.

Tab. VI shows the results obtained for our final application network. Note that EMSANet-R34-NBt1D [4] is a multi-task approach that also predicts instance orientations as well as an overall scene label for a given input. We also integrate both information over time in our NDT representation, enabling our mobile robots to gain an even stronger scene understanding. However, this is not in the scope of this publication. The results in the upper part of Tab. VI indicate that the application network benefits from being trained on multiple datasets. For the real-world datasets NYUv2 and SUNRGB-D, the results already exceed those from [4]. The results for Hypersim and ScanNetV2 reflect that combined training puts less focus on these datasets. However, when comparing the results for ScanNetV2 to those in Tab II, the results are still good, considering that the model was trained on all 40 classes instead of only 20[4]. The lower part of Tab. VI further presents results when fine-tuning the application network on the individual datasets. Again, especially the results for the real-world datasets NYUv2 and SUNRGB-D indicate that the fine-tuned network has much stronger generalization capabilities. For Hypersim and ScanNetV2, the results are close to those listed in Tab. I and Tab. II in the main part. We share the weights for all networks in our GitHub repository.

TABLE V: Comparison between annotations provided for the SUNRGB-D dataset [38] in the EMSANet publication [4] (EMSANet version) and the refined annotations proposed in this publication (PanopticNDT version).

	Split	# Images	# Instances	# Orientations
SUNRGB-D (EMSANet version [4])	train	5,285	18,171	13,076
	test	5,050	16,961	12,440
SUNRGB-D (PanopticNDT version)	train	5,285	24,184	19,292
	test	5,050	23,769	19,409

[4]For the ScanNetV2 benchmark, networks are typically trained directly on 20 classes, as distinguishing 20 classes is easier than distinguishing 40 classes. Our application network instead was trained on 40 classes to enable combining all four datasets. For evaluating the 20-classes setting, we simply mapped predictions for ignored classes to void.

TABLE VI: Upper part: Results obtained for EMSANet-R34-NBt1D in the full multi-task settings of [4] for real-world applications (our application network). While the network was trained on NYUv2 (40 semantic classes), Hypersim (40 semantic classes, with camera model correction presented here), SUNRGB-D (37(+3) semantic classes, instance annotations from here – PanopticNDT version), and ScanNetV2 (40 semantic classes), the training process was solely monitored based on the performance on the SUNRGB-D test split (37 semantic classes). Thus, the results additionally reported for the NYUv2 test split, the Hypersim validation/test split and the ScanNet validation split (subsample 100, 40 classes and 20 classes (=benchmark mode)) are all within the epoch the best performance on SUNRGB-D was reached. The best individual result may be higher. Lower part: We further present results when fine-tuning this application network on the individual datasets. We refer to Sec. V-B in [4] for details on the reported metrics. Legend: *italic*: metric used for determining the best checkpoint; gray: best result within the same run; Sem: semantic segmentation; Sce: scene classification (unified classes of [4]); Ins: instance segmentation; Or: instance orientation estimation; and LR: learning rate.

	Tasks	Task weights	LR	Dataset	Semantic decoder	Instance decoder		Scene head	Panoptic results (after merging)				
					mIoU ↑	PQ ↑	MAAE ↓	bAcc ↑	mIoU ↑	PQ ↑	RQ ↑	SQ ↑	MAAE ↓
Application network	**Sem + Sce + Ins + Or**	1 : 0.25 : 3 : 0.5	0.0005	SUNRGB-D	49.30	61.33	18.31	55.45	47.32	*50.91*	58.40	86.06	14.28
					49.31	62.12	18.24	58.56	47.32	*50.91*	58.40	86.22	14.19
				NYUv2	56.55	63.54	16.70	70.29	55.42	48.47	56.77	84.27	14.58
				Hypersim (valid)	28.60	47.25	—	30.02	27.06	17.23	21.30	74.09	—
				Hypersim (test)	29.03	42.40	—	46.17	26.75	17.95	22.05	69.74	—
				ScanNetV2 (40 classes)	50.84	58.68	—	49.01	49.32	40.49	49.13	81.20	—
				ScanNetV2 (20 classes)	66.96	58.60	—	49.01	65.27	53.92	63.71	84.15	—
Application network + Fine-tuning	**Sem + Sce + Ins + Or**	1 : 0.25 : 3 : 0.5	0.002	SUNRGB-D	50.86	63.40	16.95	58.55	47.88	*52.48*	60.03	86.30	13.02
					50.91	63.61	16.87	60.17	48.30	*52.48*	60.03	86.39	13.00
			0.004	NYUv2	59.02	64.79	16.12	73.33	57.83	*51.15*	59.59	84.80	14.31
					59.13	65.17	15.76	76.82	57.90	*51.15*	60.04	84.99	13.86
			0.001	Hypersim (valid)	49.06	53.90	—	32.89	47.80	*34.22*	39.98	82.88	—
					49.06	54.22	—	35.48	47.80	*34.22*	82.98	39.98	—
				Hypersim (test)	50.23	49.99	—	54.62	47.58	31.62	37.17	73.06	—
			0.002	ScanNetV2 (40 classes)	52.08	59.67	—	49.65	50.19	*41.72*	50.13	79.42	—
					52.26	60.28	—	50.32	50.25	*41.72*	50.53	80.51	—
			0.0005	ScanNetV2 (20 classes)	70.52	70.60	—	50.46	67.65	*58.47*	67.65	85.94	—
					70.81	71.13	—	51.33	67.69	*58.47*	67.65	86.07	—

The hyperparameters for PanopticNDT in real-world applications follow the values described in Sec. IV-C for the ScanNetV2 experiments. The only exception are the IoU thresholds for instance matching (see Eq. 7). For application, we use slightly higher values for θ^{M} and θ^{N} and set them to 0.3 and 0.2, respectively. For the environment shown in the video, mapping runs at 6.89Hz on average (the actual map update rate slightly varies depending on the captured part of a scene) on the hardware of our mobile robot (Intel NUC11PHKi7C (Intel i7-1165G7)). Incorporating panoptic information almost halves the throughput compared to SemanticNDT [15] (12.29Hz) and almost cuts the throughput into thirds compared to plain NDT of [13, 14] (18.08Hz). Due to the voxel size of 10cm, the resulting panoptic NDT map for the entire flat shown in the video is only 13.5MB. Storing panoptic information increases the map size by ˜53% and ˜255% compared to semantic NDT maps of [15] (8.8MB) and plain NDT maps of [13, 14] (3.8MB).

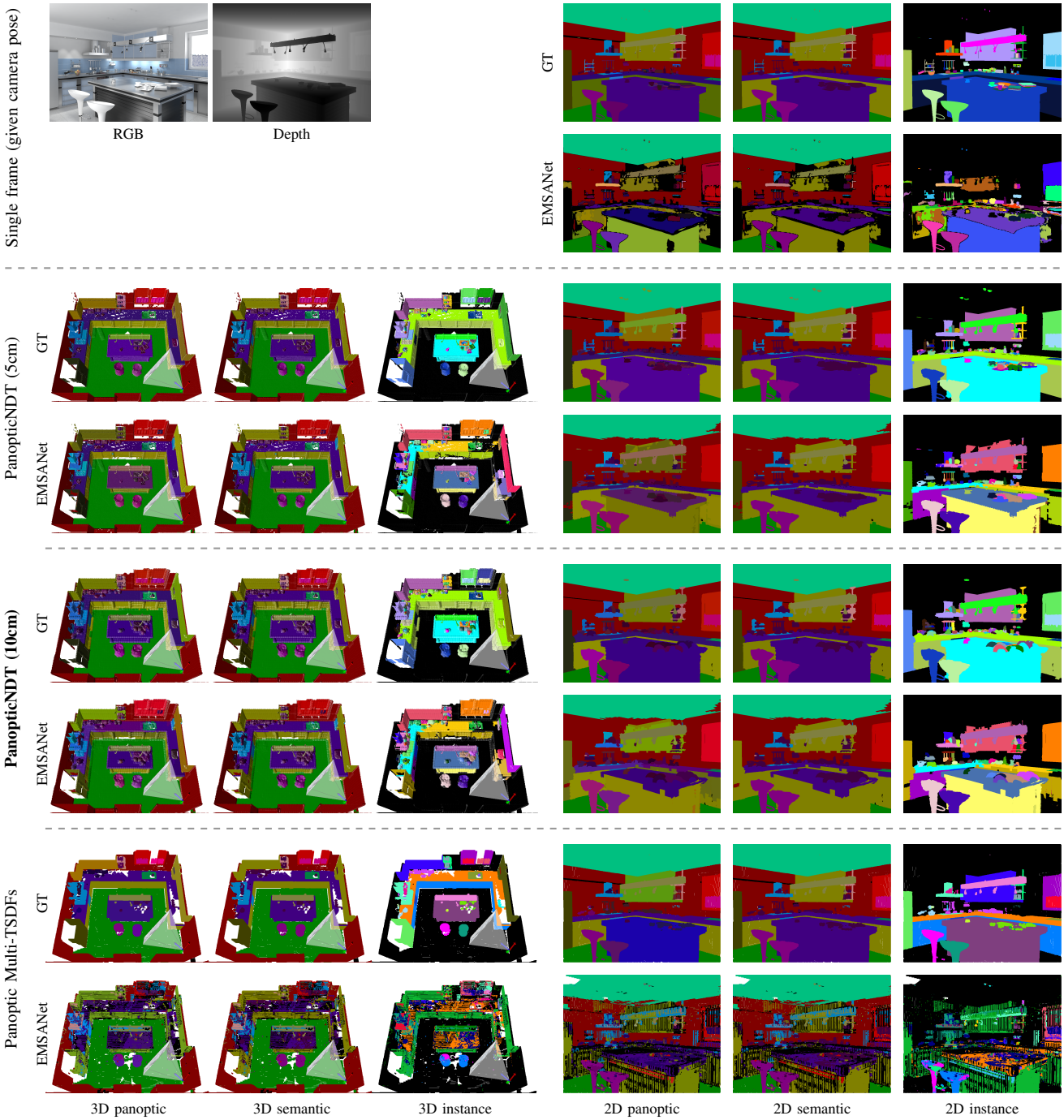

Fig. 5. Qualitative results for scene *ai_001_10* of the Hypersim test split. The upper part shows the dataset's ground truth as well as the thresholded predictions of EMSANet [4] (see Sec. IV-C). The lower part compares our proposed PanopticNDT with voxel sizes 5cm and 10cm to Panoptic Multi-TSDFs [31]. For each mapping approach, results are visualized for both when mapping with ground truth (top) and when mapping with predicted segmentation of EMSANet (bottom). Best viewed in color at 300%. Black indicates *void/no_instance*, for the semantic colors, we refer to Fig. 3. Panoptic labels are visualized by small color differences based on the semantic color.

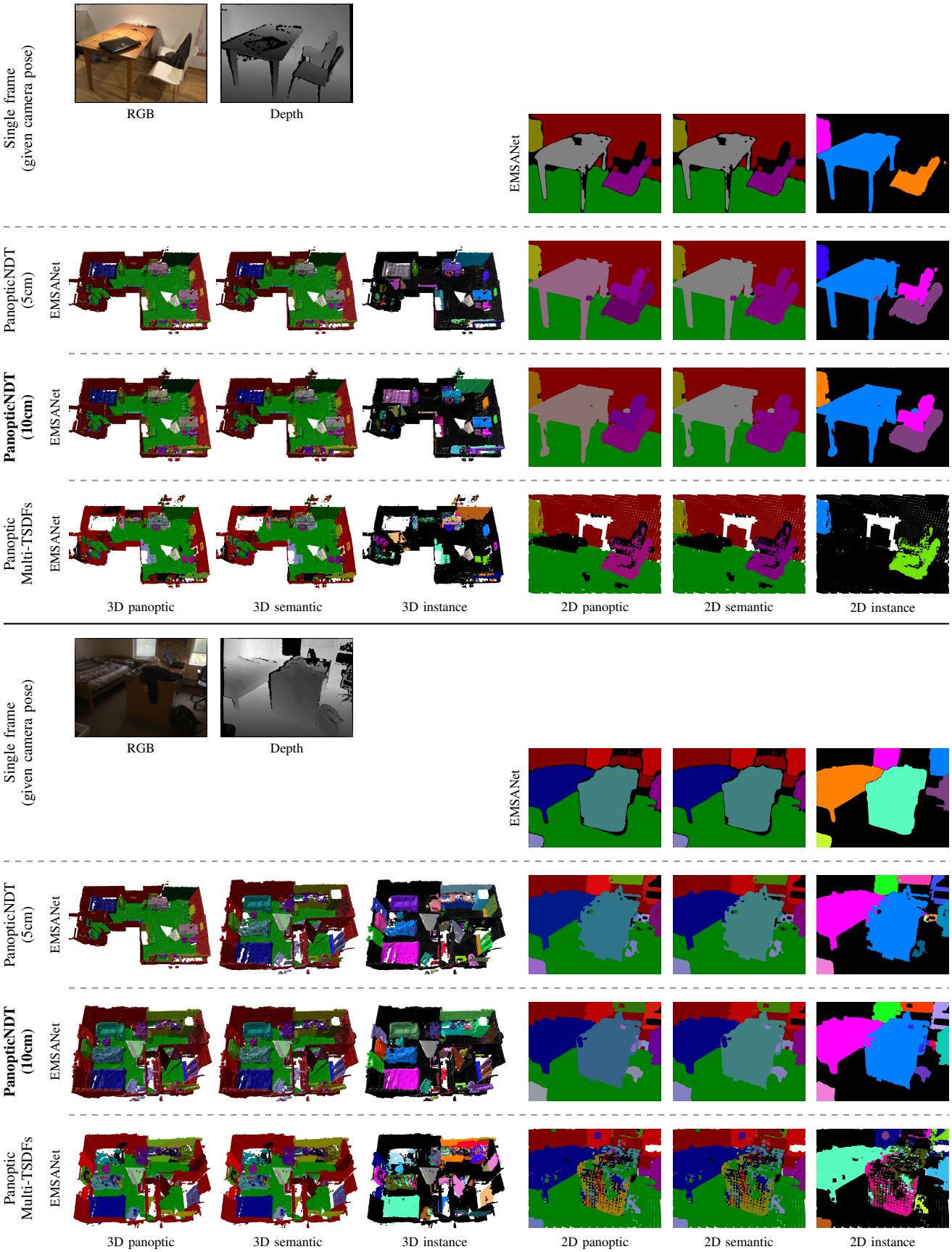

Fig. 6. Qualitative results for *scene_0757_00* (top) and *scene_0761_00* (bottom) of the hidden ScanNetV2 test split. The upper part for each scene shows the thresholded predictions of EMSANet [4] (see Sec. IV-C). The lower part for each scene compares our proposed PanopticNDT with voxel sizes 5cm and 10cm to Panoptic Multi-TSDFs [31]. Best viewed in color at 300%. Black indicates *void/no_instance*, for the semantic colors, we refer to Fig. 3. Panoptic labels are visualized by small color differences based on the semantic color.