ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision and Geometry Lab

OD3DU: Object Detection based Dynamic 3D Scene Understanding

Bachelor's Thesis

Elena Koller
Department of Computer Science D-INFK

Advisors: Dr. Zuria Bauer and Dr. Dániel Béla Baráth
Supervisor: Prof. Dr. Marc Pollefeys
Computer Vision and Geometry Group
Department of Computer Science D-INFK

November 1, 2024

Abstract

Effective navigation in complex environments is essential for autonomous robotics. A rich and adaptable way to represent these surroundings is 3D dynamic scene graphs. By capturing spatial, semantic, and relational data in a hierarchical, multi-modal structure, they enable better understanding and interaction within diverse environments. Recent advancements in RGB-D sensing technologies have enhanced accessibility and fueled research in 3D scene understanding. While early works assumed static environments, nowadays, research has also shifted toward dynamic environments.

While several existing approaches address navigation in dynamic environments, they often rely on custom-trained models, 3D features and operate at the class level, which limits their adaptability. To overcome these limitations, we introduce a novel method for 3D scene understanding at the instance level in slow-moving indoor settings using 2D object detection and matching based on RGB-D data. The goal is to identify reference objects in a rescan of the scene. By first matching predicted object segments to their corresponding object instances and then projecting those into 3D space using an IoU-based voting mechanism, we predict 3D object centers that can be used to update scene graphs and reflect changes. Our method - OD3DU - leverages only pre-trained models and 2D features, simplifying implementation and enhancing adaptability in real-world conditions. We demonstrate the method's effectiveness through qualitative and quantitative evaluations in various dynamic scenarios.

Acknowledgements

I want to express my gratitude to my supervisor, Mark Pollefeys, for the opportunity to pursue this research and for creating a great lab environment that encourages collaboration and innovation.

I'm also incredibly thankful to my advisors Zuria Bauer and Dániel Baráth for their valuable input, trust, and encouragement. Their insights and ongoing support have deepened my understanding of this fascinating field and helped me refine my research approach. As this was my first practical research project, I am excited for the future opportunities that lie ahead.

I want to give special thanks to my colleagues in the CVG laboratory for fostering such a collaborative and inspiring environment. I'm particularly grateful to Yang Miao for the work on his codebase, knowledge of the used dataset and generously dedicating his time to assist me. I also want to acknowledge Rupal Saxena for her practical advice and insights, which were invaluable. This experience not only enhanced my knowledge, coding and planing skills but also allowed me to forge wonderful new friendships along the way.

Lastly, I'd like to extend my heartfelt gratitude to my friends and family, particularly Sven Wey and Jan Hochstrasser, who were always there to listen and discuss my ideas in a "rubber duck" fashion. Thank you!

Contents

Abbreviations

GT Ground Truth

3D Three-Dimensional

2D Two-Dimensional

RGB Red-Green-Blue

RGB-D Red-Green-Blue-Depth

SLAM Simultaneous Localization and Mapping

IoU Intersection of Union

K-NN K Nearest Neighbors

ViT Visual Transformer

List of Figures

List of Tables

Chapter 1

Introduction

As we move further into the era of autonomous robotics, effective navigation in complex environments increasingly depends on detailed and accurate representations of those surroundings. For robots to interact seamlessly with their environments, these representations must capture multiple levels of detail, account for dynamic changes, and remain lightweight enough for efficient use. A promising data structure that meets these requirements is the Three-Dimensional (3D) (dynamic) scene graph, which has gained significant attention in recent years [2, 31].

3D scene graphs provide a hierarchical and multi-modal representation of environments, making them particularly well-suited for autonomous navigation, scene understanding, and robotic manipulation tasks. They capture information ranging from raw spatial geometry to higher-level semantic and relational data, making them an adequate tool for systems that must interpret and interact with complex environments. The rise of this data structure has spurred significant research across fields such as Simultaneous Localization and Mapping (SLAM) [32, 16], scene graph generation [39, 14], and 3D scene understanding [34, 15]. Advances in algorithms for real-time scene graph construction [16] and localization systems based on scene graphs [27] exemplify the momentum in this area, showing the broad applicability of scene graphs in robotics, computer vision, and even augmented reality.

Recent technological developments, especially in Red-Green-Blue-Depth (RGB-D) sensing technologies such as the Intel RealSense [20] and Microsoft Kinect [28] have further opened new avenues for scene understanding and environment reconstruction. RGB-D sensors provide rich color and depth information, enabling detailed 3D models of indoor environments to be constructed at relatively low costs. These advancements have not only made 3D scene graph generation more accessible but have also inspired new research directions, particularly in the construction and understanding of dynamic, real-world scenes.

While early research primarily assumed static environments [9, 4], recent efforts have shifted toward dynamic environments where changes occur in two distinct ways. In some cases, objects can be tracked in real-time as they move, allowing for immediate updates to the environment representation. In other cases, environments experience changes over time without being directly observed, requiring the system to adapt once these changes become apparent. This setting is particularly relevant for long-term SLAM, where continuously updating and refining environment representations as unseen changes accumulate presents a critical challenge. The proposed work focuses on the second type of dynamic environment.

Recent works have developed approaches to address challenges in low dynamic environments, such as instance relocalization in dynamic environments [18, 42] and techniques for adjusting scene graphs in response to environmental changes [13]. Instance relocalization detects and re-identifies objects once they are moved,

enabling updates to scene representations without the need to remap the entire environment. Similarly, scene graph adjustment methods modify the graph's structure to reflect changes in spatial and semantic relationships between objects. These approaches mark essential steps toward making 3D scene representations more adaptive and suitable for dynamic settings. These methods, however, often rely on custom-trained models and 3D features or operate only at the class level.

This OD3DU builds on recent advancements by exploring a novel approach to scene change detection by predicting potentially moved 3D object centers of reference objects in indoor environments using Red-Green-Blue (RGB) detections in RGB-D data. The initial process for segmenting and matching objects to their reference object instance is primarily based on Two-Dimensional (2D) RGB and then projected into world coordinates, allowing for detailed analysis and 3D representation of the environment. An Intersection of Union (IoU)-based voting mechanism is utilized to extract the predicted object centers. By leveraging pre-trained models and focusing on instance-level understanding, this work aims to bridge the gap in current research, providing robust techniques for comprehending dynamic environments without relying on extensive re-training or complete scene observation. This approach predicts the new 3D object centers, which helps to update dynamic scenes on a high level. It extends the utility of 3D scene graphs and offers a method for enhancing autonomous robotic systems operating in dynamic, real-world settings.

To summarize, our contributions are:

- A method for predicting 3D object instance centers of previously seen objects in a rescan of a scene based on 2D RGB predicted segmentation of the novel RGB-D sequence. The 2D predictions are coupled with back projection and a voting mechanism, enhancing scene change detection in low dynamic environments.
- An approach exclusively utilizing pre-trained models simplifies the implementation and reduces the need for extensive re-training, while making no assumptions about the indoor environment, camera path, and scene coverage.
- An evaluation of the proposed method providing both qualitative and quantitative analyses to demonstrate its effectiveness and robustness across various dynamic scenarios.

The released code can be found here: https://github.com/elikoller/OD3DU

Chapter 2

Related Work

2.1 (3D Dynamic) Scene Graphs

Scene graphs have historically been utilized in computer graphics, particularly within game engines, to represent and manage 3D environments efficiently [37]. The concept of employing graph-based structures was later extended to computer vision, initially focusing on the representation of 2D images [22]. Early applications of scene graphs in this domain included tasks such as image retrieval [19], high-level image understanding [8], and image captioning [1]. [2] first proposed the application of scene graphs to 3D environments, but the focus remained on static scenes.

Expanding on this foundation, [31] introduced the concept of 3D dynamic scene graphs. These structures encapsulate dense 3D models by abstracting higher-level spatial and semantic information. The proposed framework consists of five hierarchical layers, ranging from a semantic mesh at the lowest level, followed by objects and agents, places and structures, rooms, and finally, entire buildings. Both inter- and intra-layer edge connections represent relationships and associations, such as ”the pillow is on the sofa” or ”the living room is inside building A.” This multi-layered graph structure is particularly advantageous due to its lightweight nature relative to the rich information it encodes. Furthermore, the graph’s queryable and actionable properties make it highly suitable for applications such as autonomous navigation. Recent research has further explored the generation and application of 3D scene graphs, focusing on their incremental construction and real-time use cases[39, 16, 14].

2.2 Low Dynamic Environment Graph Representation and Updates

The assumption of a static environment poses significant limitations for real-world applications in robotics, as most environments exhibit dynamic characteristics over time - even if not directly observed. Researchers have developed various approaches integrating dynamics into scene graph representations to overcome this limitation. A fundamental commonality across these approaches is the preservation of map structures that focus on objects persistent over time. For instance, in [12], the authors augment grid maps by incorporating only features that persist over time, thus filtering out transient elements. Similarly, [35] presents a method that retains only scans corresponding to static objects, discarding those that do not align with the stable environment.

Other methods focus on continuously updating the map representation. In [38], the map is refreshed by

removing outdated objects and integrating new ones, ensuring the map remains up to date. Change detection strategies are also employed, as demonstrated in [26], where environmental representations are compared over time, and overlaps in objects are analyzed to identify changes.

Additionally, approaches such as [13] aim to estimate the movability of objects in the environment by aggregating information from multiple mapping sessions, ultimately producing a movability score for each object. This score indicates how likely the object is to move over time.

2.3 3D Instance Segmentation and Object Detection

3D scene understanding is fundamental for constructing scene graphs that integrate semantic and relational information. Recent methods for 3D instance segmentation and object detection aim to enhance scene comprehension in such environments.

In [11], instances are predicted by assigning votes from each point to its potential object center. These votes are then grouped into proposals and clustered into object instances based on higher-level features. Another approach, introduced in [24], employs a tree structure built upon learned superpoint features to extract final object instances.

Scene graphs constructed in [39] focus on learning relationships between geometrically segmented regions of RGB-D data to generate object instances incrementally. Additionally, some methods integrate 2D information to support 3D instance recognition. For instance, [15] learns 2D features, which are back-projected into a voxel grid, where both 2D features and 3D data are fused to infer object instances. Building further on 2D information and segmentation, [40] combines grouped 3D primitives with 2D object masks generated by Segment Anything (SAM)[21]. These 2D masks are then integrated with the over-segmented 3D data, allowing for the refinement of object instance predictions. While this method is promising, it similarly falls short in adapting to object movements over time, as it does not establish robust correspondence between 2D masks and their 3D counterparts. The work in [14] leverages a similar concept to our proposed approach by first applying 2D segmentation using a generic segmentation model and extracting features out of these segments, which then enable to predict 3D object instances. Nonetheless, the 2D information used in the above works focuses on making the detection and extraction of object instances more stable, while temporal changes and re-identification of such objects are not further explored.

Another notable approach, presented in [23], relies on a 2D object detection model to place bounding boxes around objects in 2D and then creates 3D frustums encapsulating the objects. The 3D bounding boxes are subsequently predicted by estimating object orientation and performing bounding box regression within those frustums.

While these approaches have demonstrated effectiveness in instance segmentation and object detection, their assumption of static environments constrains them largely as they are not designed to handle scene dynamics or changes at the instance level over time.

2.4 Scene Change Detection

Several proposed works predict scene changes, addressing the limitations of static environment assumptions. In the 2D domain, [7] leverages tracking models, but instead of using a temporal sequence as input, it compares the previous state image with the current state image to detect changes. In contrast, [33] extends scene change detection into 3D by utilizing two RGB inputs. It estimates depth and extracts visual features

at multiple resolutions. These features are compared to detect differences, which are then processed to predict bounding boxes around the regions of change. However, this method is limited to areas that are visible and match among both input images, restricting its ability to detect changes in regions that are not directly comparable between the two views.

In the 3D domain, [30] detects scene changes using two-point clouds. The method first computes the difference between the point clouds and then uses a fixed threshold to identify potential change regions. Point features are extracted from these regions, followed by using a region feature extractor to process the data. Although this approach provides descriptions of the scene changes, it operates at the class level only, which can be limiting in applications that require more granular object-level detection.

2.5 3D Instance Relocalization

Several works have focused on instance-level analysis, aiming to predict object instance correspondences and poses following unobserved movements in low dynamic environments. These methods enable more fine-grained scene change detection and facilitate scene graph updates.

In [18], a fully convolutional 3D correspondence network is proposed, which operates on RGB-D input data to identify matching features of corresponding key-points. This approach is completed by a 6DoF pose estimation mechanism to predict relocalizations. In contrast, [43] employs segmented point clouds from two different timestamps. This method uses Hungarian matching to align instance features and an algorithm to estimate the transformation between the matched point clouds. Hungarian matching can be used to solve the assignment problem in graph theory. In other words, Hungarian matching can find maximum-weight matchings in bipartite graphs - in this case, finding the instance point clouds in the different timestamps that correspond to each other based on the extracted 3D features.

Chapter 3

Method

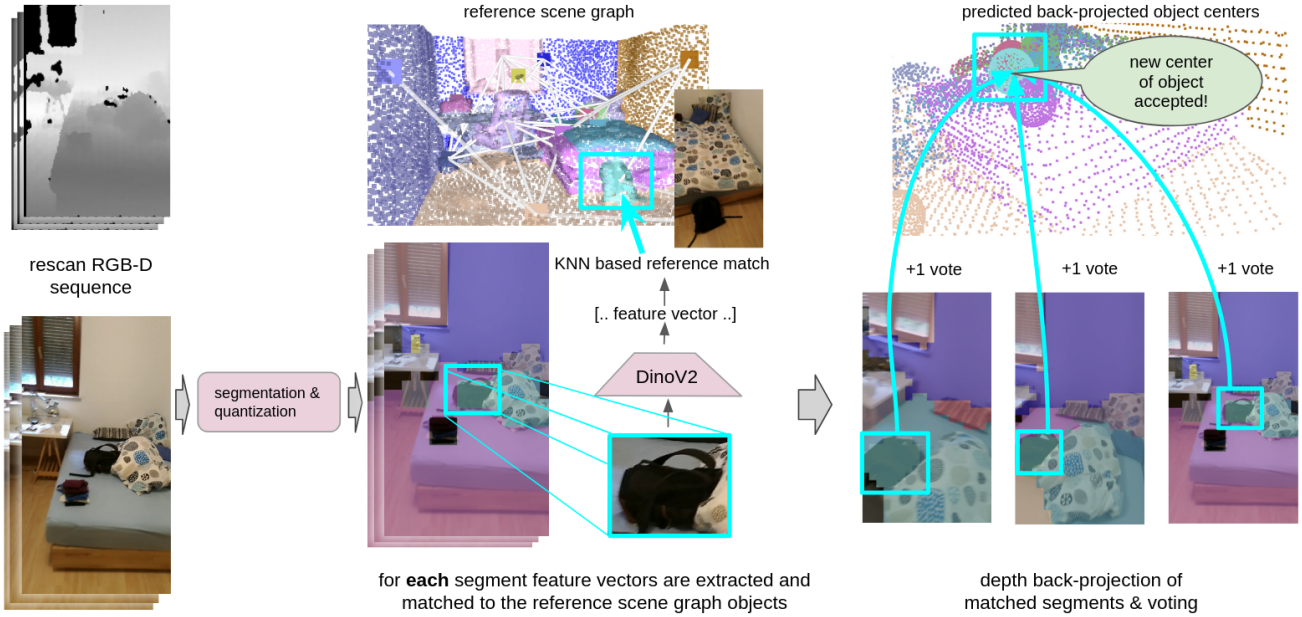


Figure 3.1: **OD3DU Pipeline:** OD3DU predicts object centers of reference objects from a reference 3D dynamic scene graph at t_0 in a rescan of the scene at t_i. It takes an unseen RGB-D sequence, camera in-, extrinsics, and the coordinate transformation between the input sequence and the reference scene graph as input. OD3DU extracts segmentation masks for every frame using a pre-trained segmentation model and quantizes them. Using the bounding boxes of the predicted segments, we compute feature vectors using DinoV2. For each segment, the feature vector gets compared to the feature vectors of the objects in the reference scene graph. Cosine similarity-based K-NN is applied to obtain an accurate match. Each segment finally gets back-projected using the depth map: A voxel IoU-based voting mechanism is used to cluster the points to obtain the predicted object instance center.

The high-level OD3DU pipeline, as shown in fig. 3.1, takes as input a reference scene graph at time t_0, which consists of the minimal requirements of annotated images and an object center & point cloud per object, and an unseen RGB-D sequence captured at a different time t_i. This second sequence - the ”rescan” - can follow an arbitrary camera path and may only partially cover the original indoor scene. The camera intrinsics and extrinsics - defining the internal parameters and external positioning of the camera - along with

the transformation from rescan to reference coordinates are also provided for accurate alignment. Despite relying solely on pre-trained models, OD3DU operates at an instance level. It predicts the 3D object centers of objects known from the reference scan within a changed environment, allowing the update of the scene graph. Changes include moved, newly added, and removed objects.

The method first generates feature vectors using a generic, pre-trained model (Dinov2[29]) for the objects in the reference scene graph. After these initial preparations (section 3.1), the pipeline performs semantic segmentation on the rescan sequence, isolating 2D objects at the instance level (section 3.2). Next, features for the detected instances are extracted (section 3.3) using the same pre-trained model as in the reference feature generation (section 3.1). Using the same model ensures consistent feature representation across both timestamps. These segment features are then matched with object instances in the reference scene graph using K Nearest Neighbors (K-NN) based on cosine similarity, establishing accurate correspondence between the two scans (section 3.4). To determine the 3D object centers, depth map points for each predicted object are projected into world coordinates (section 3.5.1). A voting and clustering mechanism is applied to aggregate these points, extracting the final object point cloud and center predictions (section 3.5.2).

3.1 Extracting 2D Features for Reference Objects

We construct a well-defined feature space for the reference scene graph to identify and distinguish between various objects. Our method follows this procedure: Using the bounding boxes extracted by the provided 2D instance annotation of the scene graph (section 4.1.1), the corresponding image regions of the objects are cropped and resized to a standardized 224x224 resolution. This resizing ensures compatibility with the DinoV2 Visual Transformer (ViT)-L/14[29] (ViT) architecture and provides uniformity across different object instances and object sizes.

Next, each 224x224 image patch is processed using the DinoV2 backbone ViT-L/14 [29], presented in section 3.1.1, generating a high-dimensional feature vector that captures the semantic properties of the object. Since DinoV2 generates feature vectors for patches within the input image, the dimensionality of the output is high (eq. (3.1)). We apply average pooling to reduce the feature vectors' dimensionality, resulting in a more compact and efficient feature representation (eq. (3.2)). This feature space is populated by the resulting feature vectors, each associated with a specific object and its instance identifier (ID).

Since objects may appear across multiple frames from different viewpoints, we store a list of feature vectors for each reference object. This allows for a more comprehensive object representation and enhances robustness during matching in later stages of the pipeline.

The transformation of a 224x224 input image into a feature vector in the DinoV2 ViT-L/14 architecture:

$$F = \text{DinoV2}(I) \in \mathbb{R}^{256 \times 1563} \tag{3.1}$$

Where:

- $I \in \mathbb{R}^{224 \times 224 \times 3}$ is the input RGB image.
- F is the output feature vector with 256 patches and a descriptor length of 1563.

The average pooling operation applied to the extracted feature vectors across the patches of a 224x224 input image can be expressed as follows:

$$\bar{F} = \frac{1}{256} \sum_{i=1}^{256} F_i \tag{3.2}$$

Where:

- $\bar{F} \in \mathbb{R}^{1563}$ is the final feature vector after averaging across the 256 patches.

3.1.1 DinoV2 for Feature Extraction

DinoV2[29] is a self-supervised, task-agnostic framework designed to learn high-quality image representations without the need for labeled data. Instead of relying on annotations, DinoV2 employs a teacher-student learning paradigm, where a student network is trained to match the output of a pre-trained teacher network [3]. This approach enables the model to learn rich and generalizable image features that can be applied across various computer vision tasks.

The core architecture of Dinov2 is based on Vision Transformers (ViTs). In ViTs, the input image is partitioned into smaller patches, treating the image as a sequence of patches comparable to tokens in natural language processing. Each patch is embedded into a feature vector, representing the local information contained in that patch [10]. A key advantage of this patch-based approach is its ability to capture global context across the entire image, allowing the model to understand relationships between distant regions of the image and infer information about occluded objects.

3.2 Input RGB Segmentation

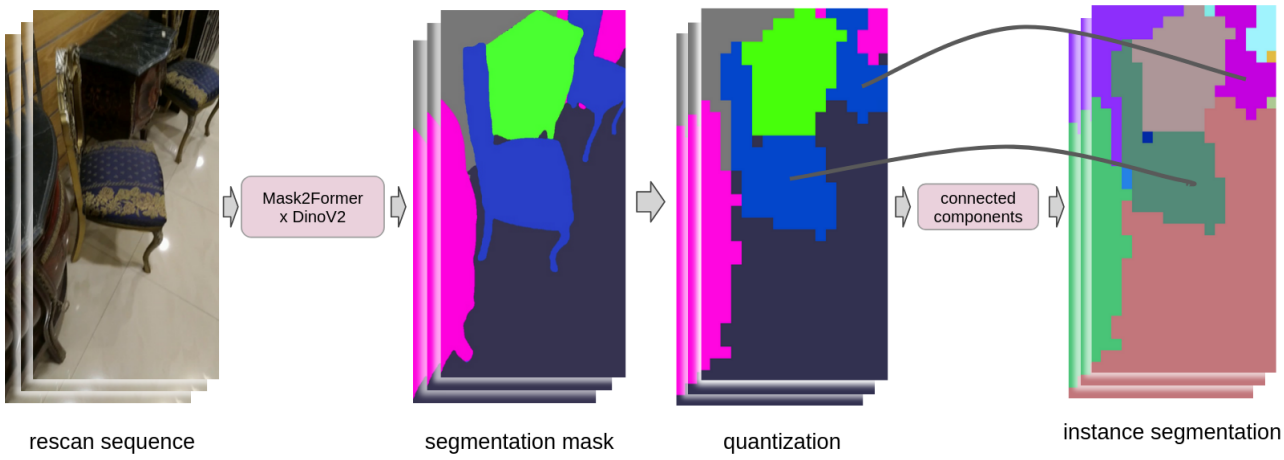


Figure 3.2: **Predicted Segmentation Masks** In order to retrieve object instances, segmentation masks get predicted for the raw input image using a pre-trained model consisting of the Mask2Former pipeline with a ViT-Adapter on the pre-trained DinoV2 ViT-G/14 backbone. To counter noise introduced by the segmentation, we quantize into 32x18 patches (consistent with the GT in section 4.1.1). Since the obtained masks are at a class level, an algorithm extracting connected components is used to obtain separate instance masks.

In the OD3DU pipeline, input segmentation is a critical pre-processing step, establishing a foundation for instance-level segmentation and comprehensive scene understanding. We use a model, which is pre-trained on the ADE20K dataset[41] and presented in section 3.2.1, to perform this segmentation, which, based on our experimental evaluation (detailed in section 4.2), achieves a granularity comparable to the ground truth annotations. This level of precision is crucial for accurate downstream tasks such as object matching and scene graph updates. The segmentation is applied to frames from the unseen RGB-D sequence. Since DinoV2's output is not a panoptic segmentation, additional steps are required to achieve instance-level predictions. We implement a two-step post-processing pipeline to refine the segmentation results and isolate distinct object instances, as shown in section 3.2

First, we quantize the segmentation mask into 32x18 patches — a resolution consistent with the ground truth/ reference scene graph pre-processing outlined in section 4.1.1 and section 3.1. This quantization reduces potential noise and artifacts generated during the prediction of the masks. Quantization ensures uniformity in comparison with the ground truth and enables more efficient downstream evaluation. Second, we apply a region-growing algorithm to further delineate distinct object instances from the quantized mask, ensuring that each object is uniquely identifiable. Each frame stores the original and quantized predicted segmentation masks and bounding boxes associated with unique frame object IDs, enabling efficient indexing and retrieval during object matching.

3.2.1 Mask2Former based Semantic Segmentation using DinoV2 Backbone

To perform semantic segmentation, we leveraged the on the ADE20K[41] pre-trained model provided by the authors of DinoV2 (section 3.1.1). While DinoV2 can effectively tackle semantic segmentation tasks using linear and multi-scale heads, integrating the DinoV2 backbone into the Mask2Former pipeline yields significantly improved performance [29]. The Mask2Former architecture [6] is adapted by employing the frozen weights of the ViT-G/14 Dinov2 backbone combined by a Visual Transformer adapter [5]. Mask2Former features multi-scale decoders that capture both local and global features, along with a masked attention mechanism that focuses the decoder's attention on the foreground regions of each object, thereby facilitating the extraction of relevant features while minimizing background noise [6].

3.3 Predicted 2D Segment Feature Extraction

To enable the comparison between segmented objects in the rescan and reference objects (represented by feature vectors), we follow the same feature extraction process as detailed in section 3.1. This consistent approach ensures that the feature vectors of both reference and rescan objects share the same feature space, allowing for accurate matching. For the rescan, the predicted object 2D bounding boxes are used to crop and resize the corresponding image regions into standardized 224x224 patches. These patches are processed by the DinoV2 backbone (ViT-L/14) [29], ensuring that the extracted features are directly comparable to those generated during the reference scan preparation (section 3.1).

For each instance segment identified in a frame, we compute a feature vector corresponding to the object's frame object ID (eq. (3.1)). Afterward, average pooling is applied to the output of the DinoV2 model (eq. (3.2)), reducing the dimensionality of the feature vector while preserving critical object characteristics. Finally, the feature vector for each instance is stored alongside its frame object ID, creating a structured representation of the rescan's segmented objects.

3.4 K-NN Feature Matching between Predicted Segments and Reference Objects

The proposed matching process operates within a feature space defined by the reference scene's objects and the features extracted from predicted object masks in the rescan. The objective is to correctly associate each segment in every frame of the unseen sequence with the corresponding reference object instance in the reference scene graph. If a segment belongs to an unseen object, it should not get matched to a reference object.

To achieve this, the method employs a K-Nearest Neighbors (K-NN) approach, with k=5. After evaluating various values of k in section 4.3.2, we found that k=5 provided the best performance. K-NN works the following way: For each input feature vector representing a predicted object segment, the method compares it against all the reference feature vectors using cosine similarity as the distance metric (eq. (3.3)). Based on this metric, it extracts the k closest or most similar reference feature vectors. We determine the object instance associated with the predicted segment using the majority vote of the five nearest neighbors. The choice of k=5 provides a balance between capturing enough neighboring context for robust predictions without imposing overly strict assumptions on the reference scene and unseen rescan, which may contain variability due to changes in viewpoint and occlusions.

In contrast to Euclidean distance, which measures the absolute distance between vectors in the feature space, cosine similarity evaluates the orientation or angle between the vectors. Specifically, cosine similarity quantifies how similar the two vectors are in terms of their direction, regardless of their magnitude. The cosine similarity between two vectors $\mathbf{A}, \mathbf{B} \in \mathbb{R}^n$ is defined as follows:

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.3}$$

Where:

- $\mathbf{A} \cdot \mathbf{B}$ is the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$.
- $\|\mathbf{A}\|$ is the norm (magnitude) of vector $\mathbf{A}$.
- $\|\mathbf{B}\|$ is the norm (magnitude) of vector $\mathbf{B}$.
- n is the number of dimensions of the vectors.

A similarity threshold is implemented to enhance prediction reliability further. Specifically, the average cosine similarity of the majority group from K-NN must exceed a predefined threshold of 0.6. This ensures that predictions are based on consistently high similarity, which is crucial given that feature vectors may exhibit variability. If the mean similarity meets or exceeds this threshold, the predicted object region is assigned the object instance ID of the majority-voted neighbors. This thresholding mechanism ensures that the matching process focuses solely on previously seen objects, thereby minimizing the risk of inaccurate associations with unseen objects in the evolving scene. We evaluated this in section 4.3.2.

Previous attempts to additionally identify and integrate new objects through simple thresholding, discussed in section 4.3.4, were unsuccessful due to high variability in feature vectors.

3.5 Prediction of 3D Centers

3.5.1 Back-Projection of Matched Object Segments

The first stage of 3D center prediction involves back-projecting (eq. (3.4)) the depth map into the world coordinate system. To facilitate comparison with the reference scene graph objects, we transform the coordinate system to align with that of the reference scene (eq. (3.5)). Since object segment correspondences have already been predicted, we perform selective back-projection, treating the depth map as a segmented structure. As outlined in section 3.2, the input RGB segmentation includes a stored quantized mask for each object segment. In this stage, we utilize the non-quantized segmentation of the entire image to refine the instance segmentation. By accessing the quantized mask, we isolate the relevant portions of the segmented image, extracting a finer-grained mask that captures more details of the underlying object. This refined mask is then used for back-projection, ensuring that only the depth information corresponding to the object is transformed into 3D space. This process is repeated for each segment across every frame, resulting in distinct point clouds for each instance segmentation. Given the predicted matches, we associate the corresponding object instance IDs with the point clouds.

To back-project points from a depth map into 3D world coordinates, we use the pixel coordinates and depth values, along with the camera's intrinsic parameters like focal length and principal point, to accurately compute the 3D position of each pixel in space. The focal length scales the pixel coordinates based on how far the point is from the camera, ensuring that the projection matches the camera's field of view. This process reconstructs the scene geometry by transforming 2D image data into spatially accurate 3D world coordinates.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{(u-c_x)\cdot z}{f_x} \\ \frac{(v-c_y)\cdot z}{f_y} \\ z \end{bmatrix} \tag{3.4}$$

Where:

- (u, v): The pixel coordinates in the image plane.
- z: The depth value from the depth map at pixel (u, v), representing the distance from the camera to the 3D point.
- (X, Y, Z): The corresponding 3D world coordinates for the pixel (u, v).
- f_x and f_y: The focal lengths of the camera in the x and y directions, respectively. These are intrinsic camera parameters.
- c_x and c_y: The coordinates of the camera's principal point (optical center) in the image, often located near the center of the image plane.

To facilitate the comparison of back-projected coordinates with those in the reference scan, we perform a transformation of the coordinate system from the rescan to the reference frame:

$$\begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \\ z_{\text{ref}} \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{rescan}} \\ y_{\text{rescan}} \\ z_{\text{rescan}} \\ 1 \end{bmatrix} \tag{3.5}$$

Where:

- $(x_{\text{ref}}, y_{\text{ref}}, z_{\text{ref}})$: The coordinates of the 3D point in the reference coordinate system.
- $(x_{\text{rescan}}, y_{\text{rescan}}, z_{\text{rescan}})$: The coordinates of the 3D point in the rescan coordinate system.
- $\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$: The 3x3 rotation matrix that rotates points from the rescan frame to the reference frame.
- $\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$: The translation vector that shifts points from the rescan frame to the reference frame.
- The last row $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ ensures a homogeneous transformation, preserving the correct scale and orientation.

3.5.2 Voting and IoU-based Clustering

Once the separate point clouds and their corresponding object instance IDs have been predicted, we apply a voting mechanism to cluster these predictions into a list of point cloud clusters for each object. For each object instance ID, the associated back-projected point clouds are obtained. To identify relevant clusters, we apply a voxel grid with a fixed voxel size (0.2) to discretize the space, allowing for precise comparisons between point clouds. We compute the overlap of each point cloud with existing clusters that share the same object ID using IoU, defined as the shared volume between the current point cloud and a cluster divided by their combined volume:

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.6}$$

Where:

- A and B are sets of voxels (3D grid points) corresponding to two 3D objects or predictions.
- $|A \cap B|$ is the number of voxels in the intersection of A and B (the voxels they both share).
- $|A \cup B|$ is the number of voxels in the union of A and B (all voxels present in either A or B).

A point cloud is assigned to any cluster where the IoU exceeds a predefined threshold (0.1). If multiple clusters exceed this threshold, the point cloud is assigned to the one with the highest IoU. Conversely, a new cluster is initialized if no existing cluster meets the threshold. Each cluster represents overlapping point clouds that satisfy the IoU criterion.

By focusing on the cluster with the highest IoU, we allow for the emergence of multiple clusters for the same object instance, each reflecting different confidence levels. Each successful assignment contributes a vote to the corresponding cluster. As the process progresses, clusters accumulate votes, enabling some to become dominant representations of the object. This approach enhances the accuracy of object instance identification within the point cloud data and provides valuable insights into the characteristics and confidence levels associated with each object instance.

3.5.3 Refinement of Final Predictions

In the final step, we extract the most representative object clusters from each object's predicted list of clusters. We do so by iterating through its associated clusters and selecting the one with the highest vote count. To ensure reliability, we retain only clusters with a total of at least three votes. In cases where two clusters have the same number of votes, we choose the cluster with the larger size in terms of the number of points.

After identifying the representative clusters, we downsample them using a voxel size of 0.07, which aligns their density more closely with that of the point clouds in the reference scene graph. To further mitigate noise, we apply statistical outlier removal with fixed parameters: number of neighbors=20 and a standard deviation ratio of 1.75.

From the cleaned point cloud, we compute the object's center by calculating the median of all points, effectively reducing the influence of any remaining noise. Finally, we filter the downsampled clusters to retain only those containing more than 45 points, ensuring we focus on sufficiently large and reliable clusters. The specified parameters were chosen based on our evaluation in section 3.5.

Chapter 4

Experiments

The OD3DU approach employs two main components: pre-trained machine learning models and custom algorithms, including feature matching using K-NN and a voting-based clustering algorithm. We conducted qualitative experiments to evaluate the efficacy of the machine learning models used for segmentation.

For the custom algorithms, the R3Scan dataset (section 4.1) was divided into training (270) and testing sets (90). The randomly chosen scenes all belong to the original train set since the dataset also provides corresponding scene graphs. Various parameter configurations were explored using the test set to identify optimal settings. The generalization capability of the algorithms was assessed by applying the best-performing parameters to an independent test set.

4.1 R3Scan Dataset

The 3RScan Dataset [18] is a large-scale, real-world dataset allowing for benchmarking tasks that involve slow-changing environments. One example of such a task would be long-term SLAM or object instance relocalization. 3RScan consists of a total of 478 different reference scenes and 1004 corresponding rescans providing natural changes of the given reference scenes. Changes are in the form of moved, removed, and newly added objects. Each sequence provides the following information:

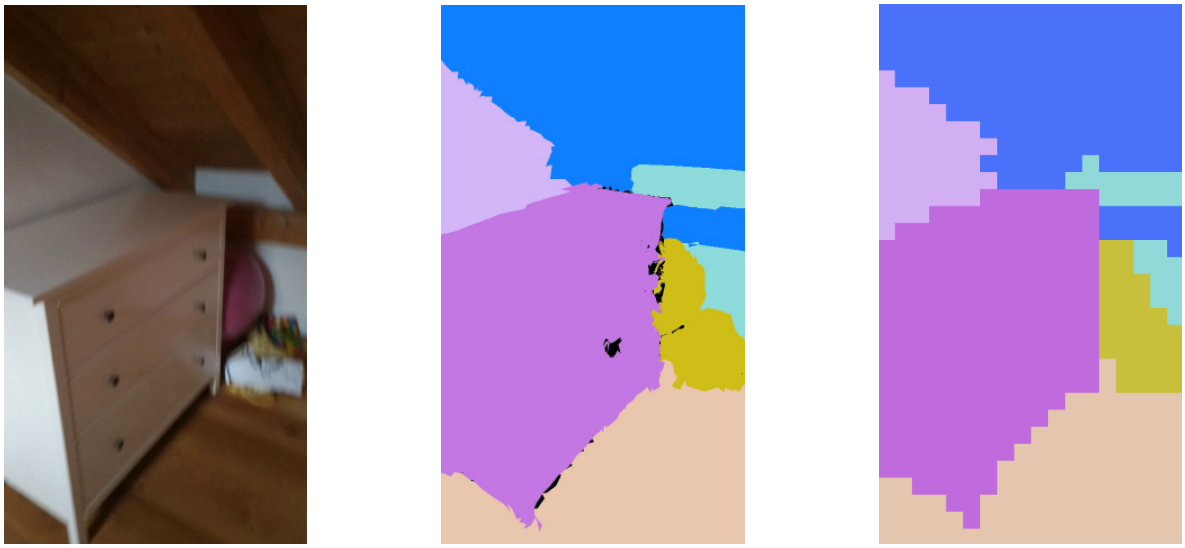
- Calibrated RGB-D sequence with coordinate transformations.
- Camera poses and calibration parameters.
- Dense instance-level semantic annotation aligned to the reference scan. For one scene the instance id is kept consistent across reference- and rescans.

Along with the scenes, the authors also provide the corresponding dataset as a 3D semantic scene graph[36]. This captures the scene data in the form of a rich yet lightweight graph structure, allowing us to evaluate our proposed task of updating a 3D dynamic scene graph based on RGB-D input.

4.1.1 2D Annotation Generation for Ground Truth and Reference Scene Graph

The 3RScan dataset provides a semantically annotated mesh. As OD3DU relies on 2D annotation input data, we generate the 2D Ground Truth (GT) annotation for each frame of a sequence based on the given

mesh. To obtain the GT we use the pinhole camera model and simulate the ray casting of the camera. As shown in the mesh projection in fig. 4.1 the projection does include some noise. To minimize the introduced error yet still to be able to later compare our results on a meaningful level, we use a quantized version of the projected GT: We divide the image into 32x18 patches of size 30x30 pixels. Since it is a projection of a 3D mesh, it does occur that some pixels are labeled with no object (black areas in fig. 4.1. The size of such regions can strongly vary and can take up more than half of the image.



Original RGB | Mesh projection | Quantized GT

Figure 4.1: **2D GT generation** The R3Scan only provides a 3D GT. To generate a 2D GT we simulate ray casting the annotated mesh using the pinhole model. This projection can introduce regions that are not annotated (black). To mitigate this noise, we quantize the GT into 32x18 patches.

4.1.2 Considerations and Limitations for the Evaluation and Method Development

Given that the 3RScan data is not synthetically generated, one can make only little assumptions regarding its quality and consistency. Motion blur, along with over- and underexposure, can occur and complicate an accurate segmentation. The camera paths can drastically vary between reference- and rescan, providing unseen perspectives of objects within the scene. While these factors introduce considerable challenges, they also allow for a more realistic evaluation of the proposed approach under real-world conditions.

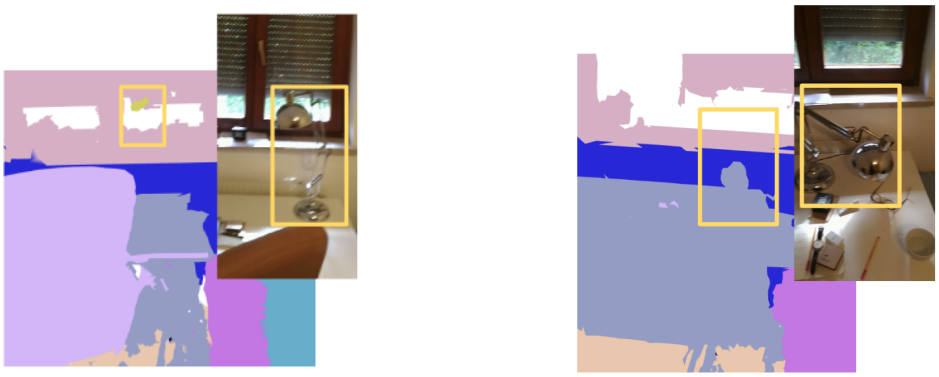
Object Granularity and Annotations

Another crucial challenge arises from the predefined granularity of the semantically annotated mesh. The level of detail can vary significantly, which may lead to inconsistencies in object recognition and classification. We show such an example for illustration purposes. While the small cup is annotated as an object, the larger, clearly visible lamp is not (teal and pink box in fig. 4.2). Such disparities in labeling can negatively impact the accuracy of downstream tasks, such as object detection and scene understanding. While the provided segmentation is clearly valuable, it can still be challenging to apply it in practice - the expected segmentation is often task-specific and highly dependent on context. Lastly, discrepancies can arise in the annotations between the reference and rescan images, leading to instances where objects appear to vanish

despite being visible in the accompanying image sequences. For example, as illustrated in fig. 4.3, the lamp is only sparsely annotated in the reference scene but is completely omitted in the rescan.

Real-life Objects Annotated Objects

Figure 4.2: **Object Granularity in the Dataset** The 3RScan dataset provides object annotations. Comparing them with the RGB sequences, the observation is made that the granularity of these annotations can vary strongly. While the cup (pink box) is annotated in the GT mesh, the comparably larger lamp (teal box) is not. This can lead to inconsistencies in the evaluation of object detection.



Annotated Objects Reference Scan Annotated Objects Rescan

Figure 4.3: **Missing Object Annotations** The 3RScan dataset provides object annotations. In some cases, inconsistencies are visible between the reference and rescan. One example is the lamp (yellow box), which is barely annotated in the reference scan and omitted entirely in the rescan.

4.2 Qualitative Experiment

4.2.1 Input Segmentation Model

As discussed in Section section 4.1.2, the granularity of the dataset and the employed scene graph are dictated by the characteristics of the dataset. Therefore, selecting an appropriate model is crucial for effective object detection. OD3DU segments all input images into distinct object instances. To identify the model that

best aligns with the dataset's granularity, we evaluate several pre-trained models, including both panoptic and semantic segmentation models, the latter of which can be adapted for instance segmentation.

- **DinoV2xMask2Former** [29] (Mask2Former using DinoV2 ViT-G/14 backbone, pre-trained on ADE20K [41]), specializing in semantic segmentation at class level.
- **SAM**[21], (ViT-H) which excels in image segmentation.
- **OneFormer** [17] (DiNAT_L, trained on the COCO dataset [25]), chosen for its panoptic segmentation capabilities that facilitate a more comprehensive segmentation of image components compared to traditional instance segmentation.

Figure 4.4 presents representative segmentation results obtained from the evaluated models. The results reveal notable variations in segmentation granularity, with DinoV2xMask2Former and OneFormer demonstrating comparable levels, while SAM yields a more fine-grained segmentation.

For our specific application, the segmentation granularity achieved by DinoV2xMask2Former and OneFormer is deemed more suitable. However, a critical distinction arises between the predicted masks of OneFormer and DinoV2xMask2Former: while DinoV2xMask2Former effectively covers the entire image with its predicted masks, OneFormer exhibits areas where no mask is applied (see fig. 4.4).

Further qualitative analysis (appendix A.1) indicates that the segmentation performance of the DinoV2xMask2Former model is best suited for our pipeline, given its achieved granularity and comprehensive mask coverage.

4.3 Quantitative Experiments

OD3DU's 2D feature matching and 3D object center prediction depend highly on various parameters. We conducted a series of experiments to identify the optimal configurations utilizing a training set of 180 scenes. Following this, we assessed the generalizability of the identified parameters on a separate test set comprising 90 previously unseen scenes.

To evaluate the performance of the parameter configurations, we employed several metrics: precision, recall, and F1 score. These metrics provide a comprehensive assessment of the effectiveness of our model in accurately predicting 3D object centers and matching features between 2D data. The results from the training set are visualized in a heat map format, where each entry corresponds to the precision, recall, or F1 score achieved for different parameter combinations. Together, precision, recall, and F1 score provide a multi-faceted evaluation, capturing various dimensions of model performance — precision focuses on prediction accuracy, recall on completeness, and the F1 score balances both to offer a more comprehensive assessment (section 4.3.1). By systematically exploring the parameter space and evaluating the outcomes through these metrics, we aim to establish a robust set of configurations that enhance the performance of our approach in both ”training” and ”testing” scenarios.

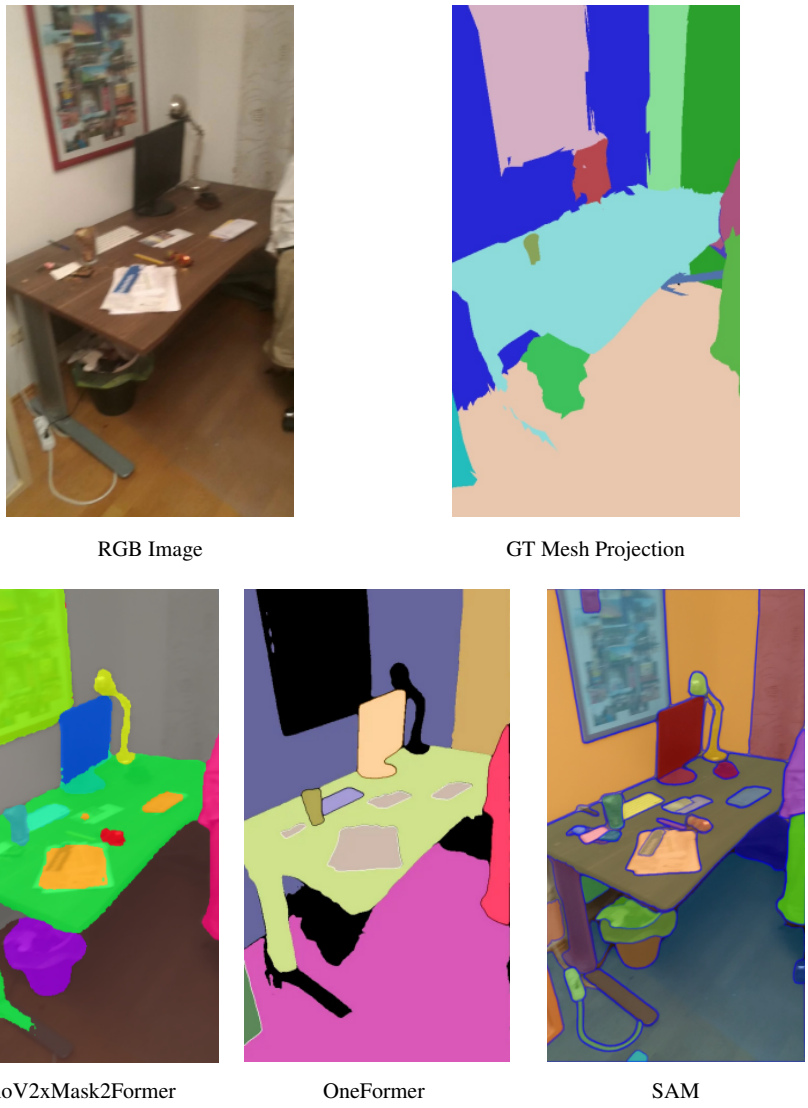


Figure 4.4: **Qualitative Segmentation Results:** This figure shows the segmentation results of different pre-trained segmentation models compared to the original RGB input image and the GT. Notably, the SAM segmentation over-segments the image compared to the GT, while DinoV2 and OneFormer have a segmentation granularity comparable to the GT. Given that the DinoV2 mask encompasses the entire image - unlike OneFormer, which does not segment the black areas - we chose DinoV2 as our segmentation model.

4.3.1 Evaluation Metrics

Precision measures the accuracy of the correct predictions, indicating the proportion of true positives among all predicted positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.1}$$

Where:

- **TP**: True Positive.

- **FP**: False Positive.
- **FN**: False Negative.

Recall evaluates OD3DU's ability to identify all relevant instances, reflecting the proportion of true positives among all actual instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

The F1 score combines precision and recall into a single metric, providing a balanced measure of a OD3DU's overall performance.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

To assess the final 3D center deviations, we calculate the average center distance in meters, offering a more intuitive and easily interpretable measure of accuracy

$$\text{Average 3D Center Distance} = \frac{1}{N} \sum_{i=1}^{N} \left\| C_{\text{pred},i} - C_{\text{GT},i} \right\| \tag{4.4}$$

Where:

- $C_{\mathbf{pred},i}$: Predicted center of object i.
- $C_{\mathbf{GT},i}$: Ground truth center of object i.
- **N**: Total number of predicted objects (including both TP and FP).

4.3.2 2D Feature Matching between Predicted Segments and Reference Objects

Section 3.4 outlines how our approach employs a matching algorithm that aligns each segmented region within the RGB frames with corresponding objects in the reference scene graph. This alignment is achieved by comparing pre-computed features of the reference scene graph and rescan using K-nearest neighbors (K-NN) and calculating the average cosine similarity among the majority reference object ID of the k most similar object features.

- **K-NN:** The choice of k represents the number of nearest neighbors considered in the matching process.
- **Cosine Similarity Threshold:** The threshold for the average cosine similarity of the selected majority.

We conduct the evaluations on quantized masks, which consist of 32×18 patches (refer to section 4.1.1 for ground truth and section 3.3 for predicted segmentation). This quantization helps capture only the relevant information necessary for effective matching.

To accurately evaluate the quality of the features and the shape properties of the predicted segmented objects, we avoid straightforward patch-wise comparisons. Instead of relying on the Intersection over Union (IoU) metric, we aim to identify the underlying ground truth object IDs in the patch regions indicated by the predicted quantized masks. This approach mitigates the impact of over-segmentation and under-segmentation

on the evaluation scores, as it does not penalize the score for over-segmented areas, provided that they correspond to the correct object ID. Our analysis focuses exclusively on those objects present in the reference scene graph. In the following experiments, TP, FP, and FN used for the metrics (section 4.3.1) are defined in the following way:

True Positive (TP): The predicted object ID of the segment corresponds to the underlying most frequent GT ID present at the location of the mask.

$$\hat{o}(s) = \arg\max_{o \in O} \left(\sum_{p \in P(s)} \mathbb{I}(m_{\text{GT}}(p) = o) \right) \tag{4.5}$$

False Positive (FP): The predicted object ID does not correspond to the underlying most frequent GT ID present at the location of the mask.

$$\hat{o}(s) \neq \arg\max_{o \in O} \left(\sum_{p \in P(s)} \mathbb{I}(m_{\text{GT}}(p) = o) \right) \tag{4.6}$$

False Negative (FN): A GT object ID present in the frame was not predicted at all.

$$m_{\text{GT}}(p) \notin \hat{O} \tag{4.7}$$

Where:

- $\hat{o}(s)$ is the predicted object ID for segment s.
- $m_{\text{GT}}(p)$ is the ground truth object mask value at patch location p.
- O is the set of all possible object IDs.
- $P(s)$ represents all the patches belonging to segment s in the predicted mask.
- $\mathbb{I}$ is the indicator function defined as:

$$\mathbb{I}(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{if } A \text{ is false} \end{cases} \tag{4.8}$$

Results Train Set

In the training set, we evaluated the impact of various parameters on performance by testing a range of cosine similarity thresholds [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8] and K-NN values [1, 3, 5, 7, 9]. The F1 score, illustrated in table 4.1, demonstrates a generally balanced performance, with a notable peak at K-NN = 5 and a cosine similarity threshold of 0.6.

However, significant differences were observed in the precision (table 4.2) and recall (table 4.3) metrics. Specifically, stricter criteria tended to enhance precision while adversely affecting recall. Given that exploring all possible combinations would exponentially increase the complexity of the test space in the subsequent experiments for 3D center prediction, we opted for fixing the parameters highlighted in green in table 4.1: K-NN = 5 and threshold = 0.6.

These values were selected based on their ability to yield the highest F1 score. Moreover, we chose K-NN = 5 to avoid imposing overly strict assumptions on the test set: Using a higher K-NN value, such as 9, would imply that each object must have at least five occurrences and corresponding representative feature vectors in the reference scene graph in order to be considered for prediction.

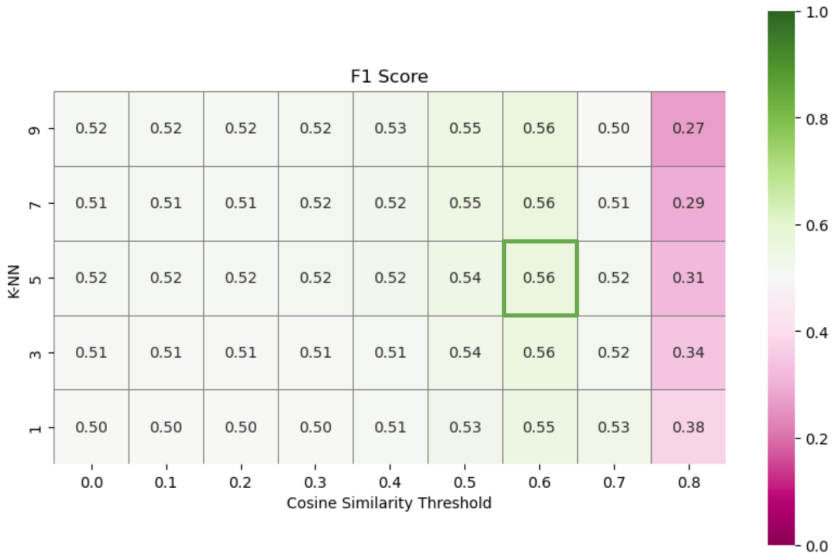


Table 4.1: **F1 Score 2D Feature Matching between Predicted Segments and Reference Objects - Train Set:** Using average-pooled DinoV2 features, the 2D segments get matched to reference objects in the scene graph based on K-NN majority voting and average cosine similarity thresholding of the chosen majority. The heatmap reflects the F1 score when using this approach for different configurations on the training set.

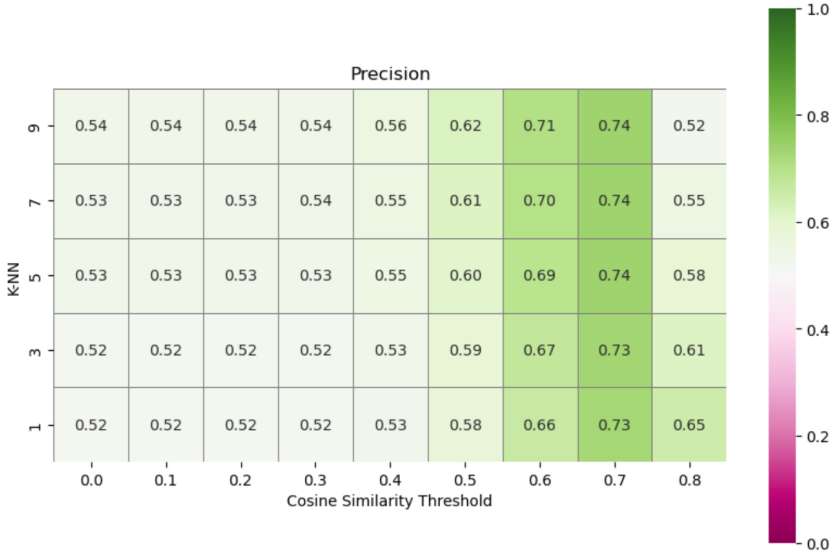


Table 4.2: **Precision 2D Feature Matching between Predicted Segments and Reference Objects - Train Set:** Using average-pooled DinoV2 features, the 2D segments get matched to reference objects in the scene graph based on K-NN majority voting and average cosine similarity thresholding of the chosen majority. The heatmap reflects the precision when using this approach for different configurations on the training set.

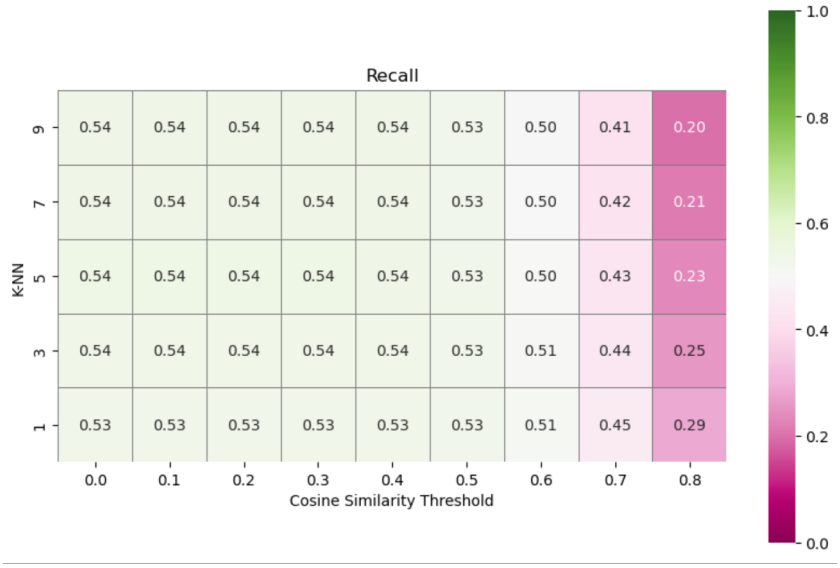


Table 4.3: **Recall 2D Feature Matching between Predicted Segments and Reference Objects - Train Set:** Using average-pooled DinoV2 features, the 2D segments get matched to reference objects in the scene graph based on K-NN majority voting and average cosine similarity thresholding of the chosen majority. The heatmap reflects the recall when using this approach for different configurations on the training set.

Results Test Set

Based on the parameters selected from the training set - K-NN = 5 and a cosine similarity threshold of 0.6, as discussed in Section 4.3.2 - we computed the precision, recall, and F1 score on the unseen test set presented in table 4.4. This evaluation revealed that the chosen parameters generalize effectively to unseen sequences, with scores varying by only 0.01 compared to the results from the train set.

Metric	Score
Precision	0.68
Recall	0.51
F1 Score	0.56

Table 4.4: **Results 2D Feature Matching between Predicted Segments and Reference Objects using K-NN = 5 and Threshold = 0.6 - Test Set:** This evaluation is based on average-pooled features from the DinoV2 model, where the 2D segments are matched to reference objects in the scene graph using K-NN majority voting and the average cosine similarity of the selected majority. k =5 and a threshold of 0.6 were chosen based on their performance on the train set

4.3.3 3D Object Predictions

For 3D object prediction, OD3DU leverages the 2D segmentation-to-object matches described in section 3.4, utilizing the selected parameters of K-NN = 5 and a cosine similarity threshold of 0.6 (section 3.5). These computed matches facilitate the back-projection of the depth map into world coordinates, resulting in a list of potential clusters for each predicted object.

To integrate a new point cloud into an existing cluster, OD3DU selects the cluster with the highest voxel IoU overlap and applies an additional IoU threshold. If this voxel IoU overlap threshold is exceeded, the point cloud is incorporated into the cluster, contributing an additional vote to its tally.

In the next step, OD3DU identifies the most promising point cloud for each predicted object by selecting the one with the highest vote count. After downsampling the resulting point cloud, it filters the objects to retain only those that meet the minimum thresholds for both the number of points and votes. The following parameters are evaluated for F1 score, precision, recall and average center distance (section 4.3.1):

- **Voxel Overlap IoU** The voxel IoU overlap with a fixed voxel size of 0.2 during the clustering of different point clouds.
- **Votes** The minimum number of votes required to determine the final object point clouds.
- **Points** The minimum number of points retained in the final point clouds after downsampling with a fixed voxel size of 0.07, combined with statistical outlier removal (using fixed parameters: nb_neighbors = 20 and std_ratio = 1.75).

To enhance computational efficiency, this evaluation is conducted only for the most promising parameters identified in the feature matching evaluation (section 4.3.2), specifically for feature matches based on k = 5 and cosine similarity threshold equal to 0.6.

To assess whether an object is predicted at the correct location, we utilize the ground truth 3D bounding boxes and define TP, FP, and FN in the following way:

True Positive (TP): The predicted object center falls within the corresponding 3D bounding box.

False Positive (FP): An object center is predicted but located outside the bounding box.

False Negative (FN): An object is not predicted at all despite being present in the GT.

Results Train Set

For the train set evaluation, we explored a variety of parameters, specifically minimum voxel overlap IoU values [0.1, 0.3, 0.5],minimum point thresholds [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, and 60], and minimum votes [1,2,3,4,5,6]. We present the results for a minimum overlap of 0.1 here, as these yielded the most promising results. The remaining results can be found in the appendix A.2.2. Heatmaps illustrate the precision (table 4.6), Recall (table 4.7), and F1 Score (table 4.5), along with the average center distance in meters (table 4.8) according to the definitions provided in section 4.3.1.

The minimum precision across all configurations is 0.64, while the minimum recall is 0.51. Similar to the results discussed in section 4.3.2, we observe the same trend of increasing precision and decreasing recall as stricter thresholds are applied. For the test set evaluation, we selected the parameters with a minimum vote count of 3 and a minimum cluster size (points) of 45. Although other parameter combinations achieved a higher F1 score, these values provided the best balance between precision (0.71) and recall (0.70). The strong performance of this configuration is further supported by an average center distance of 0.74 meters.

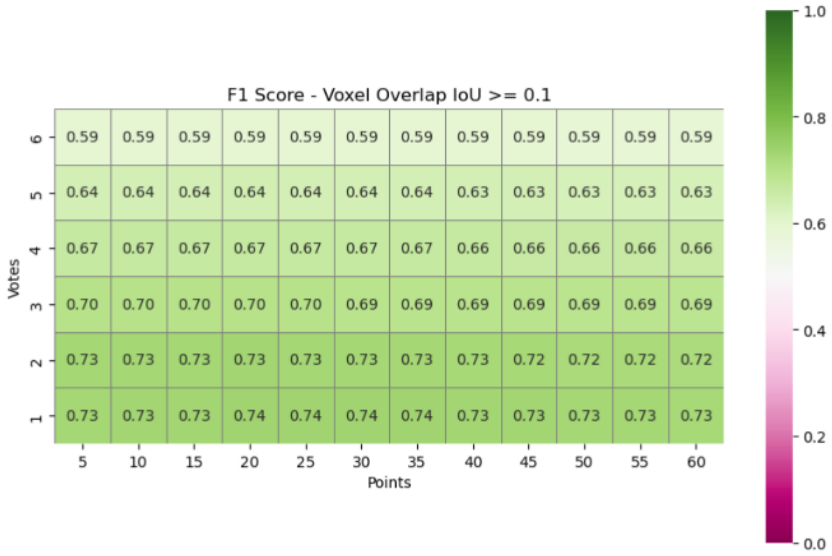


Table 4.5: **F1 Score 3D Object Center Prediction - Train Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the F1 score with a fixed parameter of a minimal overlap threshold of 0.1 for different thresholds for minimum votes and cluster sizes (points).

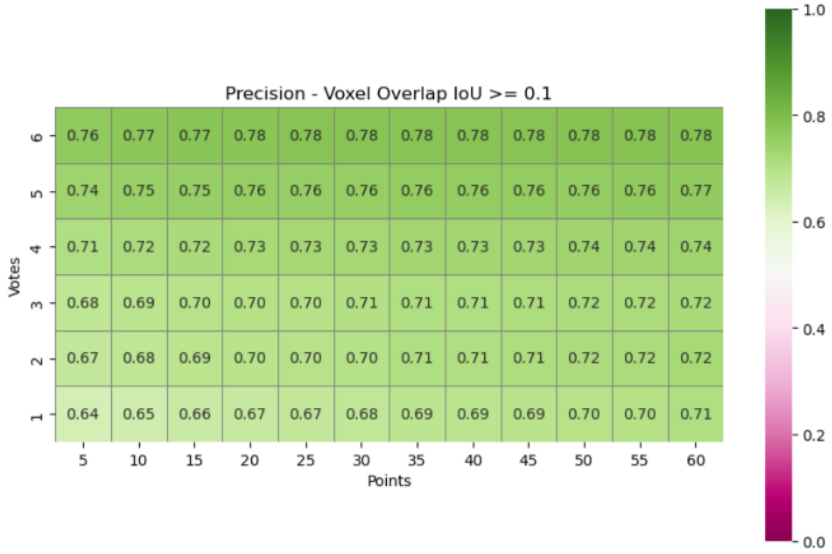


Table 4.6: **Precision 3D Object Center Prediction - Train set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the precision with a fixed parameter of a minimal overlap threshold of 0.1 for different thresholds for minimum votes and cluster sizes (points).

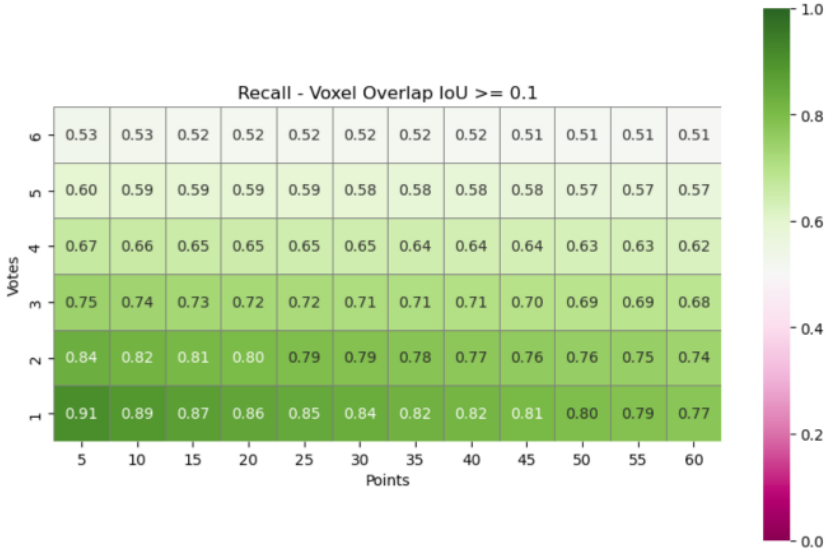


Table 4.7: **Recall 3D Object Center Prediction - Train Set:** D3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the recall with a fixed parameter of a minimal overlap threshold of 0.1 for different thresholds for minimum votes and cluster sizes (points).

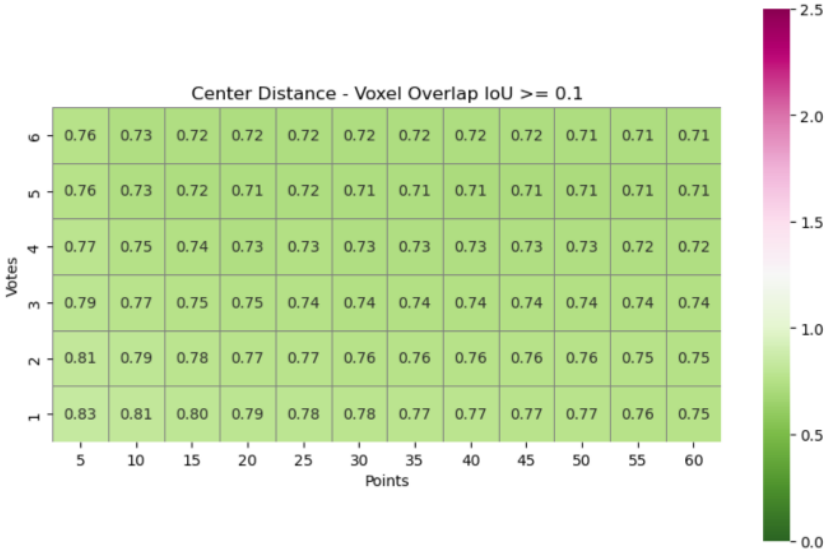


Table 4.8: **Average Center Distance 3D Object Center Prediction - Train Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the average center distance with a fixed parameter of a minimal overlap threshold of 0.1 for different thresholds for minimum votes and cluster sizes (points).

Test Set

For the test set, we used fixed parameters: a voxel IoU threshold of 0.1, a minimum cluster size of 45, and a minimum vote count of 3 (see evaluation of the training set section 4.3.3). We computed precision, recall,

F1 score, and average center distance. The results, presented in table 4.9, demonstrate that the selected parameters generalize well to unseen sequences, with scores even surpassing training results (precision = 0.74, recall = 0.74, F1 score = 0.73). This consistency reflects a balanced trade-off between precision and recall, as observed in the training set.

Metric	Score
Precision	0.74
Recall	0.74
F1 Score	0.73
Average Center Distance	0.71m

Table 4.9: **Results 3D Object Center Prediction Using Voxel IoU Threshold = 0.1, Minimum Votes = 3, Minimum Cluster Size = 45 - Test Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The table shows the result for the precision, recall, f1 score, and average center distance with a fixed parameter of minimal overlap threshold of 0.1, minimal votes of 3, and minimal cluster size (points) of 45.

4.3.4 Integration of unseen Objects in 2D Feature Matching between Predicted Segments and Reference Objects

We modified the pipeline to enable the prediction of unseen objects by implementing a straightforward thresholding mechanism. The logic for the K-NN remains unchanged. Instead of discarding feature matches with a similarity score below the cosine similarity threshold (section 3.4), we classify these matches as potentially unseen objects. To evaluate this approach, we assign negative object IDs to these predicted unseen objects and adapt the definition of true positives (TP) accordingly. Specifically, for a predicted object labeled with a negative ID, the corresponding ground truth ID must also be associated with an unseen object, while the specific instance can be disregarded. The definitions for false positives (FP) and false negatives (FN) are adjusted following the same pattern of adjusting the ID of unseen objects to a negative number (comparable to appendix A.2.1). For illustration purposes, we present only the results for the precision here; the rest can be found in the appendix A.2.1.

The results presented in table 4.10 reveal a notable pattern: as the cosine similarity threshold begins to assign unseen objects, the precision of predictions declines. From our experiments in section 4.3.2, we understand how thresholding affects performance when only considering objects from the reference scene. Even when limited to reference objects, thresholding is essential for distinguishing between valid and invalid detections (table 4.2). This suggests that seen versus unseen objects cannot be effectively differentiated using a simple cosine similarity thresholding approach. This conclusion is further supported by data inspection: no new objects are predicted at the optimal threshold parameters, only those from the reference set.

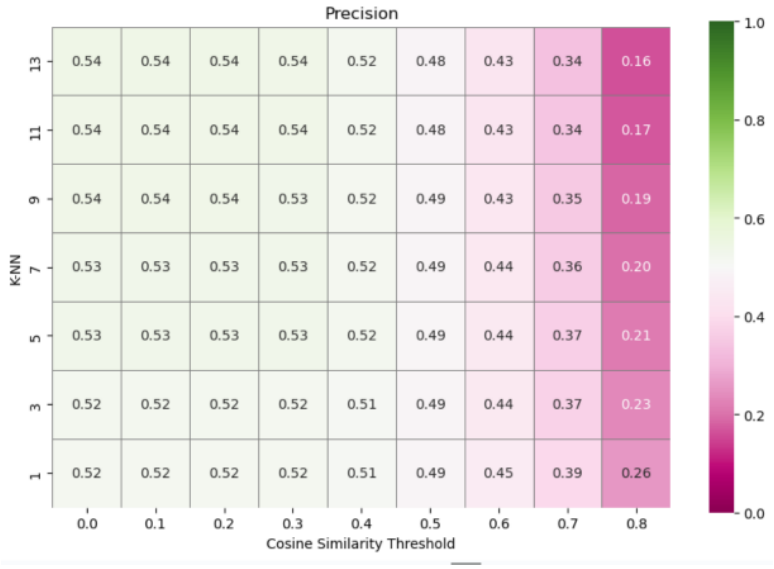


Table 4.10: **Precision 2D Feature Matching between Predicted Segments including New Objects - Train Set:** Using average-pooled DinoV2 features, the 2D segments get matched to reference objects in the scene graph based on K-NN majority voting and average cosine similarity of the chosen majority. A cosine similarity threshold is employed to distinguish between seen and unseen objects. The threshold is not effective in predicting new objects. The heatmap reflects the precision when using this approach for different configurations on the training set.

Chapter 5

Discussion

The objective of OD3DU is to accurately predict 3D reference object instance centers in a low dynamic, real-world environment, using a reference scene graph and a previously unseen RGB-D sequence as input - along with associated extrinsics and intrinsics. Our evaluation demonstrates that leveraging 2D features and predicted semantic masks, combined with a novel voxel IoU- and voting-based clustering process, is promising.

On average, OD3DU yields decent or even good results on the test scenes, as demonstrated in fig. 5.2 and fig. 5.3, where we show successfully processed scenes with accurate object center predictions. For some scenes, our approach can even yield excellent center predictions (fig. 5.1).

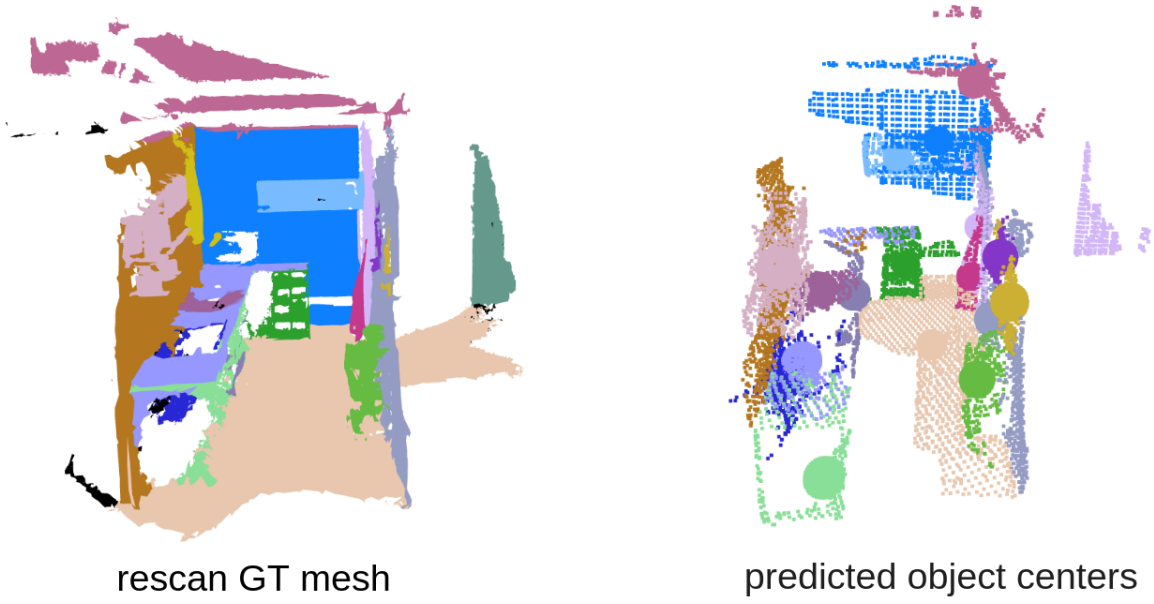


Figure 5.1: **Example for Excellent Performance:** This figure shows the resulting predicted object centers along with the extracted point clouds (right) compared to the GT (left). The performance of OD3DU in this scene is excellent. Parameters: minimal votes = 3, minimal points = 45. F1 score: 0.94, precision: 0.94, recall: 0.94.

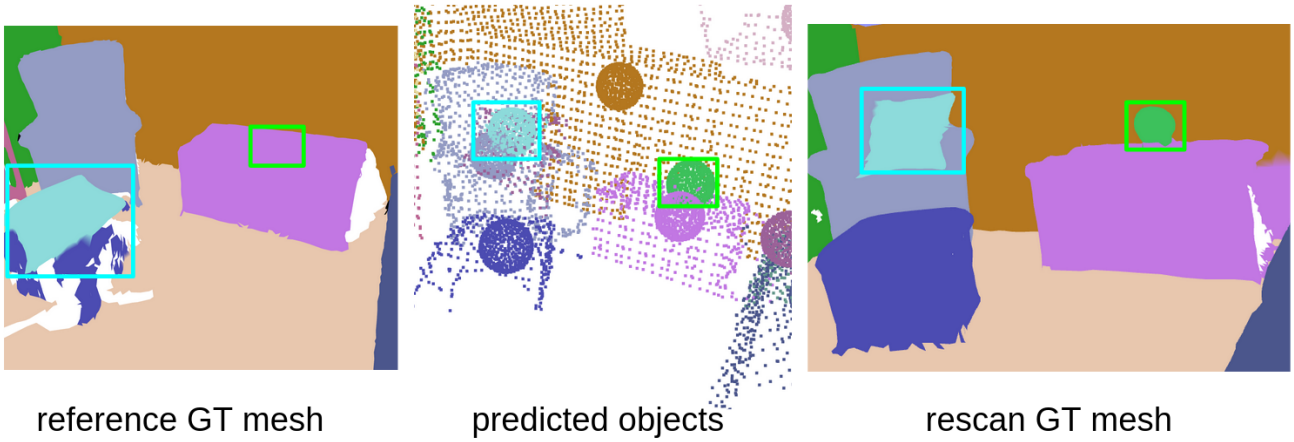


Figure 5.3: **Object Center Prediction Dynamic:** The boxes in the figure show objects that moved between the capture times of the reference scene and the rescan scene. All objects in the frame were correctly predicted. The prediction for the entire scene used minimal votes = 3 and minimal points = 45. F1 score: 0.89, precision: 0.93, recall: 0.82.

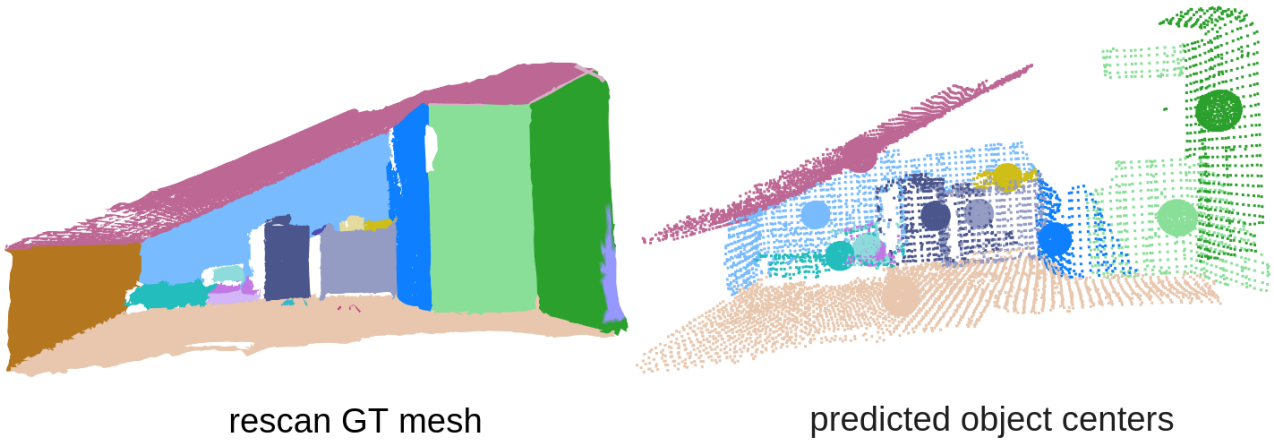


Figure 5.2: **Object Center Prediction Static Scene:** This figure shows the resulting predicted object centers along with the extracted point clouds (right) compared to the GT (left). Parameters: minimal votes = 3, minimal points = 45. F1 score: 0.75, precision: 1.0, recall: 0.6.

Nonetheless, our experiments also reveal limitations: particularly in the step of predicting object segments, stable features that remain invariant to varying viewpoints and motion blur in the input sequence are crucial. Without good quality features, the matching process can not be successful. These limitations become evident when analyzing the precision of 2D segment-to-reference object matching (section 4.3.2), where the best performing parameters in terms of F1 Score achieve a precision of only 0.69 and a recall of 0.50 on the train set. Although our 3D clustering approach can push the final precision to 0.82 (table A.4) for the 3D center prediction, strict thresholds are required to achieve this - hence minimizing recall on the 3D predictions drastically to 0.32 (table A.4). Balancing precision with recall demands more moderate threshold choices for the 3D predictions. This trade-off allows for an optimal balance, achieving a precision of 0.74 and recall of 0.74 on 90 previously unseen test scenes (table 4.9). This way, however, the precision achieved in the 2D matching stage is directly propagated to the 3D center prediction, underlining the significant bottleneck in

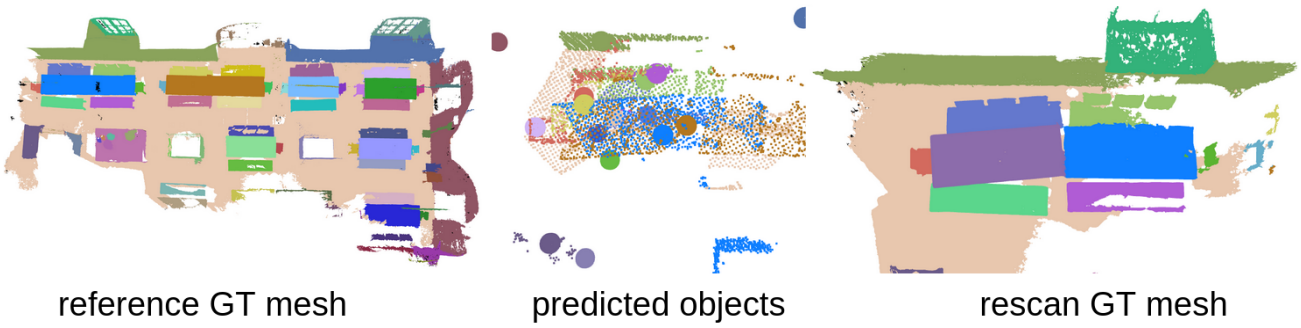


Figure 5.4: **Limitation in Repetitive/ Symmetric Environment:** Distinguishing between almost identical objects in a repetitive environment is very difficult since the features are hardly distinguishable: for parameters minimal vote = 3 and minimal points = 45, we get F1 score:0.37, precision: 0.31, and recall: 0.45.

performance. Typically, when a scene performs poorly, it tends to have multiple confusions, often due to feature matching struggling in highly symmetrical and repetitive environments with nearly identical objects (fig. 5.4) or even in more typical scenes (fig. 5.5). This could also be attributed to the moderate thresholds used, which might exacerbate these confusions in difficult scenes.

Our method effectively harnesses the information derived from the 2D steps, resulting in a substantially higher recall in the final 3D center prediction compared to the one in the 2D segment-to-reference matching stage. This is due to the design of the voting and clustering mechanism: even if an object is infrequently detected and matched in 2D, it can still contribute valuable votes to the clustering process, thereby enabling the inclusion of object instances that might otherwise be overlooked using 2D data alone. Further, compared to a simple point cloud clustering of the candidate object points - which relies on purely geometric information - the incorporation of the vote numbers allows for a confidence-based center prediction. It is also important to highlight that our 3D evaluation for precision and recall is particularly strict, relying on ground truth 3D bounding boxes with very narrow error margins. This is a sharper constraint than more straightforward distance-based metrics, where smaller objects may benefit from disproportionately large thresholds. Nonetheless, our method performs well with respect to the average center distance (for both correctly and incorrectly predicted object locations), achieving an average error of 0.72m, indicating a relatively high accuracy level.

Unlike other approaches introducing new models or architectures, OD3DU builds on pre-trained models and existing feature extraction techniques. The growing trend of incorporating 2D data and pre-trained models in 3D tasks, such as 3D instance segmentation, change detection, and scene graph generation, aligns with our findings. However, instead of relying only on 3D data or using 2D features solely to detect object instances, we use them to match reference object instances, establishing intra-temporal relations to the reference scene graph rather than intra-frame or intra-segmentation relations within the input sequence. We achieve this without making assumptions about the indoor environment, camera path, or rescan scene coverage. Our results suggest that further integration of 2D data could enhance performance in 3D settings.

Figure 5.5: **Example for Poor Performance:** In some scenes, many objects get mixed up and inaccurately predicted. This can be attributed to the lenient thresholds in the predictions. To show the difference, we plot the predicted object centers on top of the GT mesh of the rescan. For minimal votes = 3 and minimal points = 45, we get F1 score: 0.6, precision: 0.56, and recall: 0.64.

Chapter 6

Conclusion

In this work we propose OD3DU, a method to accurately predict 3D reference object centers within a RGB-D rescan of a low dynamic, real-life indoor environment by leveraging pre-trained models and combining 2D features with a voxel IoU- and voting-based clustering process. The achieved final precision, recall, and F1 score of 0.74,0.74,0.73 and an average distance between ground truth and predicted centers of 0.71m indicate that our approach successfully balances reliability and accuracy.

While our results demonstrate significant potential, challenges remain regarding viewpoint changes and motion blur that could significantly affect performance - especially in the 2D segment to object matching. Our findings underline that the current trend in research of utilizing 2D data and 2D features for 3D scene understanding is not limited to predicting intra-frame relations but can and should also be used to enhance high-level, intra-temporal 3D scene (graph) changes and updates.

Chapter 7

Future Work and Limitations

While OD3DU demonstrates promising results underlining the importance and effectiveness of incorporating 2D information for 3D scene updates, several limitations were identified, particularly in the 2D segmentation and feature matching phase. One notable issue is that the masks need to match the object granularity of the reference scene graph: An over- and under-segmentation will negatively impact the creation of representative object features and matching them accurately to the reference objects. Another challenge arises when the model encounters similar-looking objects, such as two identical chairs (fig. 5.4), which can confuse object recognition.

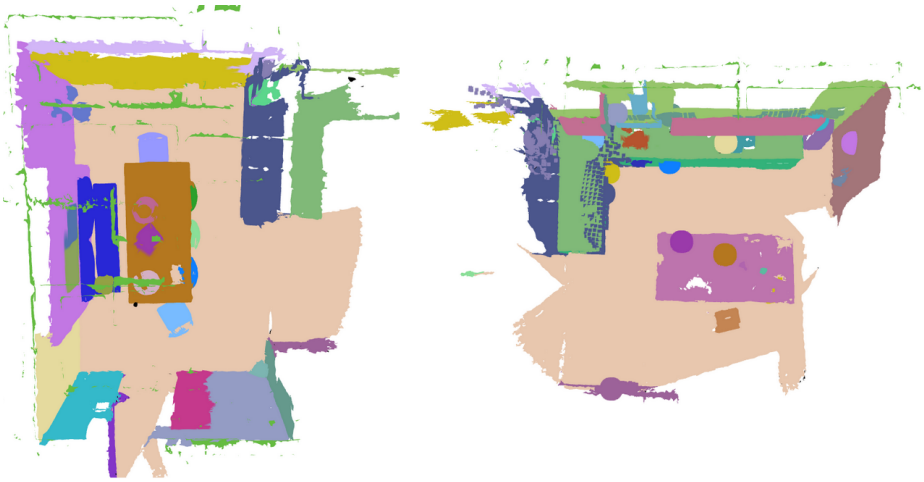


Figure 7.1: **Example New Region Limitation - Unseen Kitchen:** When inspecting the reference scene corresponding to the rescan, it is easily visible that most of the objects in the rescan have not been seen before. This scene shows a significant limitation: unseen and seen objects are not always easily distinguishable, leading to false detections. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.32, precision: 0.25, recall: 0.44.

Although our system effectively detects previously seen reference objects, significant challenges were encountered when extending 2D segmentation to predict unseen objects using simple thresholds. Unfortunately, this approach did not yield reliable results for incorporating new items into the scene graph, emphasizing that segmentation remains a bottleneck in our method and adversely affects the system's adaptability

to evolving environments, especially when exploring previously unseen parts of the scene (fig. 7.1).

There are numerous paths for further research in the future: OD3DU can currently be used to predict changes in a scene graph based solely on its center predictions. Enhancements could be made by applying additional change detection methods to extract regions of interest based on the depth data and then focusing our approach solely on those regions. This could also make computations significantly more efficient and potentially more accurate since we do not rely on predicting the entire scene. Also, investigating advanced methods for additionally stably extracting more accurate point clouds and estimating the bounding boxes and poses of the predicted objects based on our proposed approach could enrich the spatial understanding of the scene. Nonetheless, our results show that applying 2D information for intra-temporal changes in low dynamic environments is a promising avenue for future research.

Appendix A

The First Appendix

A.1 Additional Results Qualitative Experiments

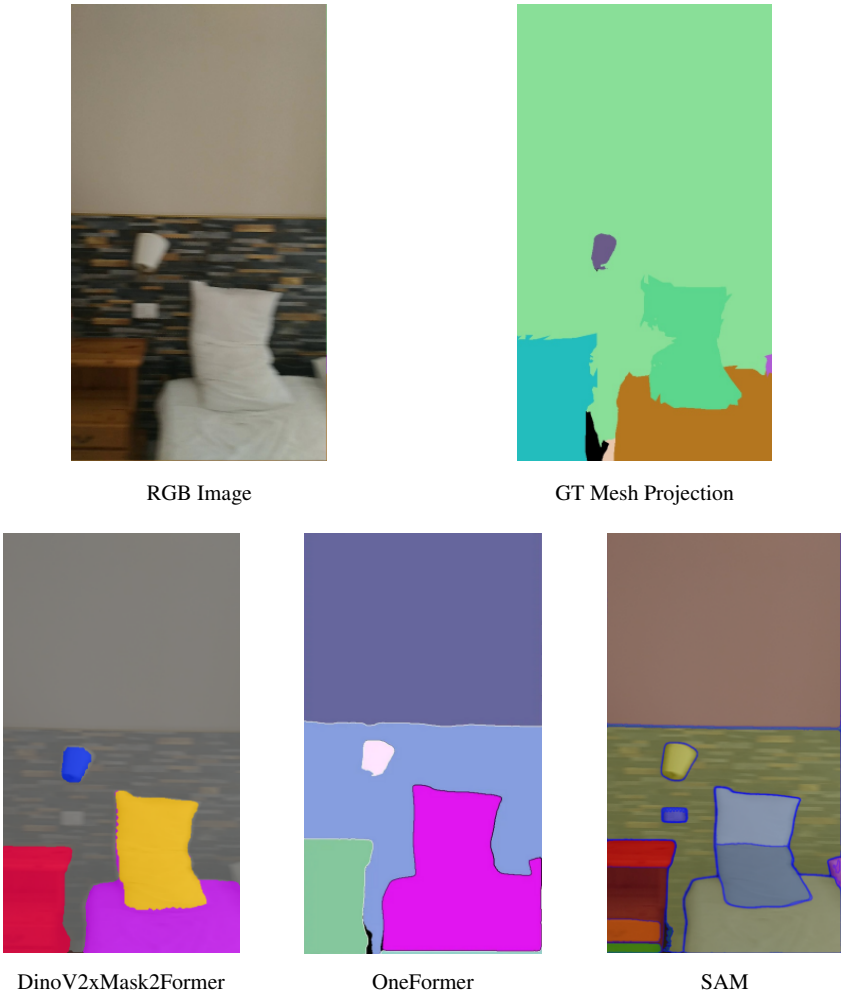


Figure A.1: **Qualitative Segmentation Results Bed Scene:** This figure shows the segmentation results of different pre-trained segmentation models compared to the original RGB input image and the GT. The models used for segmentation are Mask2Former with a DinoV2 backbone, OneFormer panoptic segmentation and SAM.

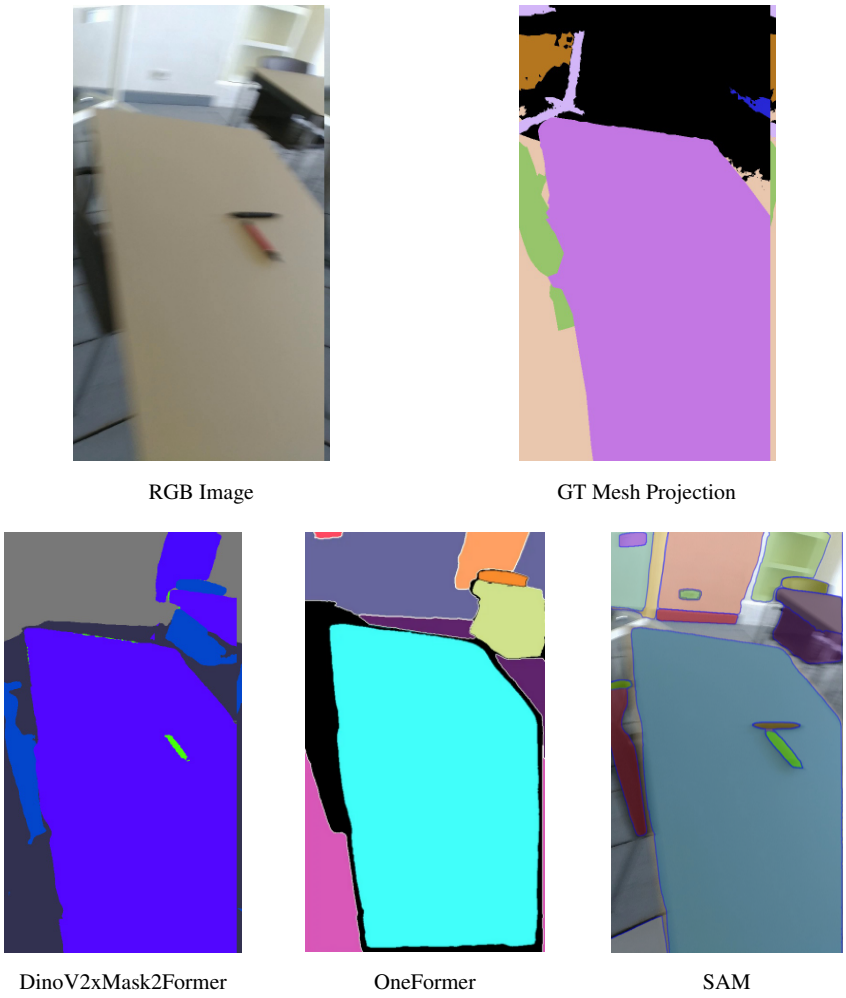


Figure A.2: **Qualitative Segmentation Results Table:** This figure shows the segmentation results of different pre-trained segmentation models compared to the original RGB input image and the GT. The models used for segmentation are Mask2Former with a DinoV2 backbone, OneFormer panoptic segmentation and SAM.

A.2 Additional Results Quantitative Experiments

A.2.1 Introduction of New Objects based on Thresholding Results

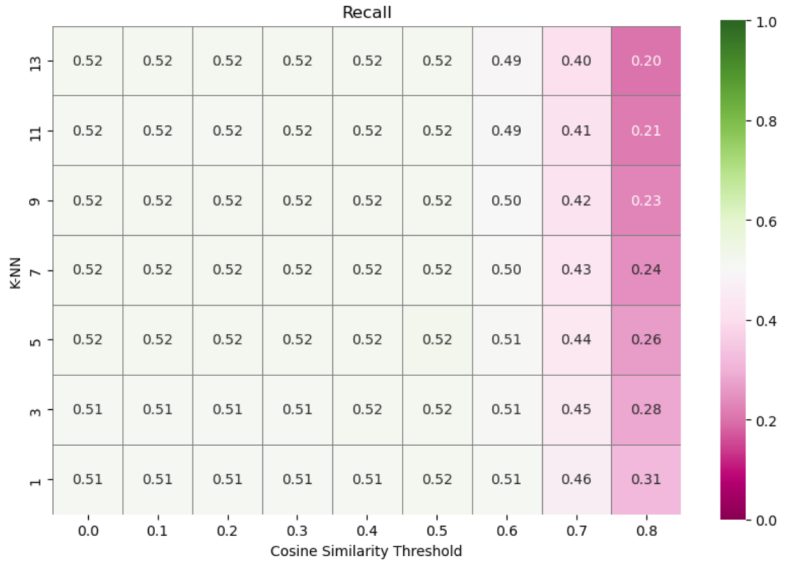


Table A.1: **Recall 2D Feature Matching between Predicted Segments including New Objects - Train Set:** Using average-pooled DinoV2 features, the 2D segments get matched to reference objects in the scene graph based on K-NN majority voting and average cosine similarity of the chosen majority. A cosine similarity threshold is employed to distinguish between seen and unseen objects. The heatmap reflects the recall when using this approach for different configurations on the training set.

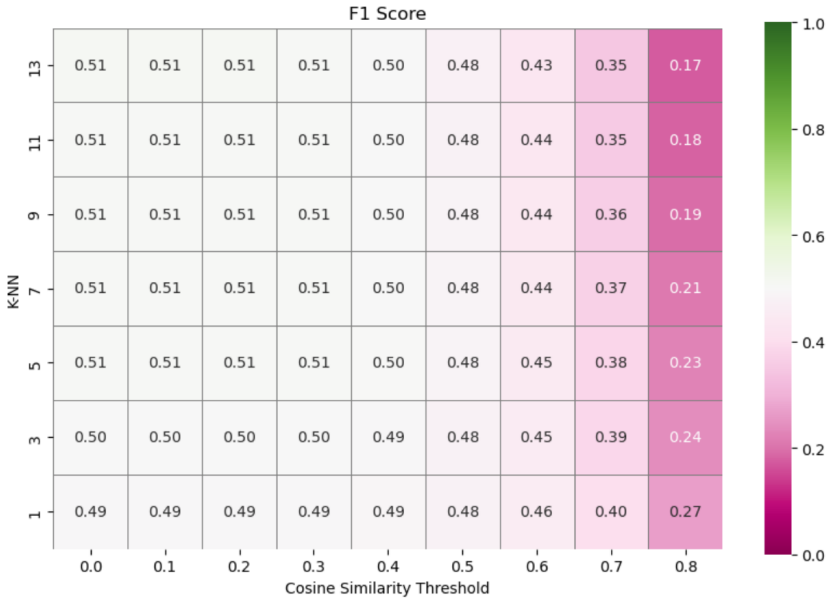


Table A.2: **F1 Score 2D Feature Matching between Predicted Segments including New Objects - Train Set:** Using average-pooled DinoV2 features, the 2D segments get matched to reference objects in the scene graph based on K-NN majority voting and average cosine similarity of the chosen majority. A cosine similarity threshold is employed to distinguish between seen and unseen objects. The threshold is not effective in predicting new objects. The heatmap reflects the F1 score when using this approach for different configurations on the training set.

A.2.2 IoU based Clustering Results for Different IoU Thresholds

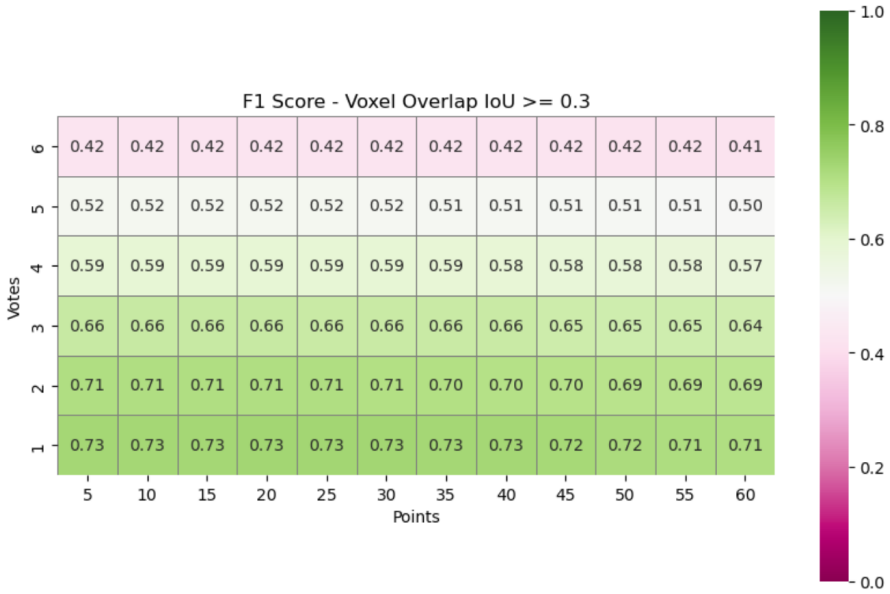


Table A.3: **F1 Score 3D Object Center Prediction - Train Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the F1 score with a fixed parameter of a minimal overlap threshold of 0.3 for different thresholds for minimum votes and cluster sizes (points).

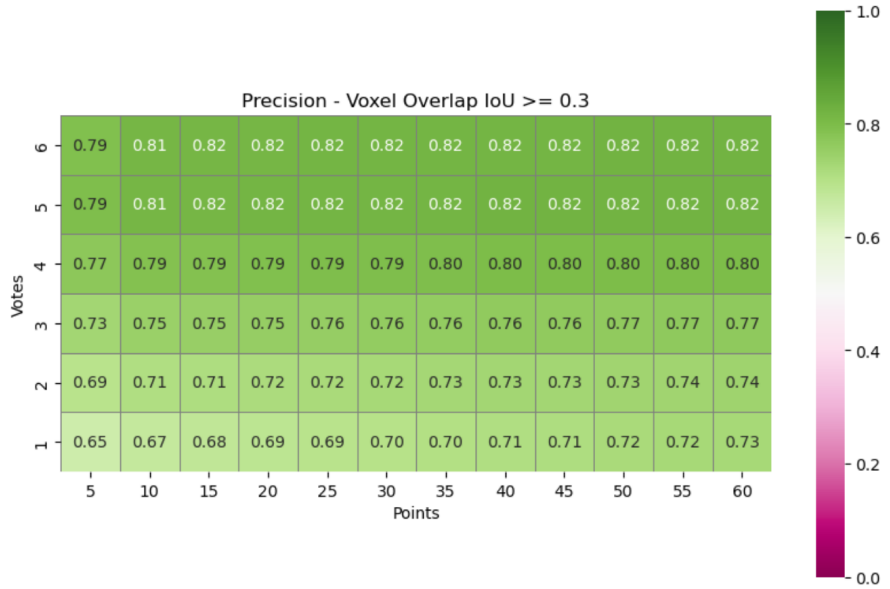


Table A.4: **Precision 3D Object Center Prediction - Train set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the precision with a fixed parameter of a minimal overlap threshold of 0.3 for different thresholds for minimum votes and cluster sizes (points).

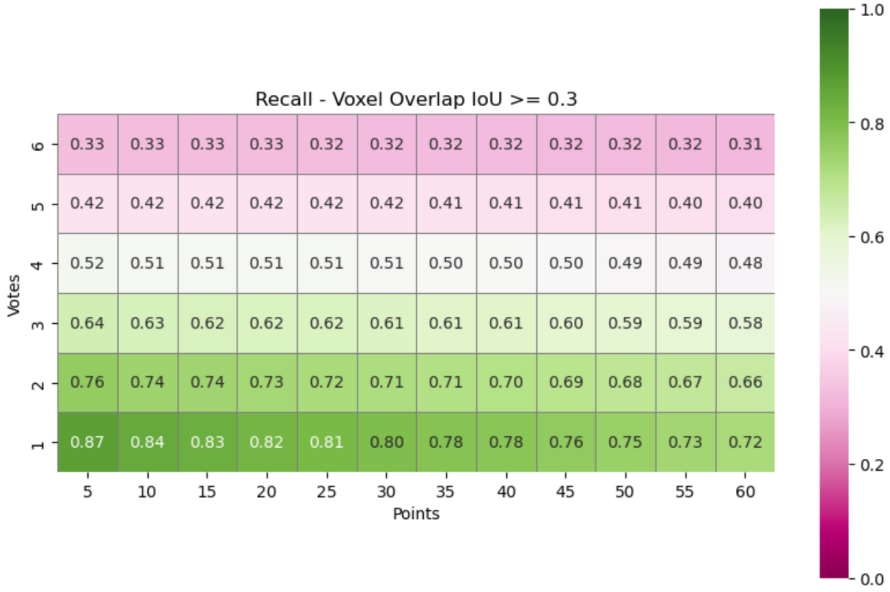


Table A.5: **Recall 3D Object Center Prediction - Train Set:** D3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the recall with a fixed parameter of a minimal overlap threshold of 0.3 for different thresholds for minimum votes and cluster sizes (points).

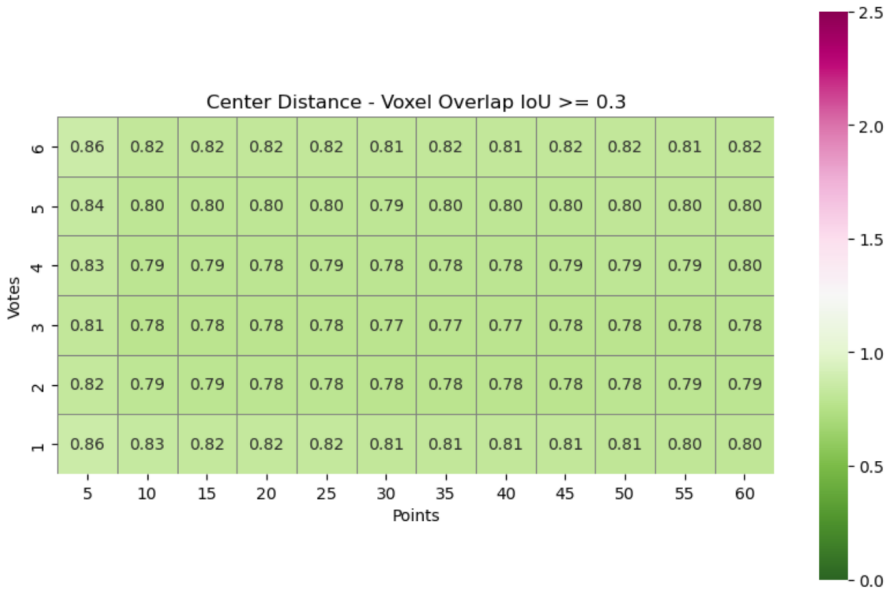


Table A.6: **Average Center Distance 3D Object Center Prediction - Train Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the average center distance with a fixed parameter of minimal overlap threshold of 0.3 for different thresholds for minimum votes and cluster sizes (points).

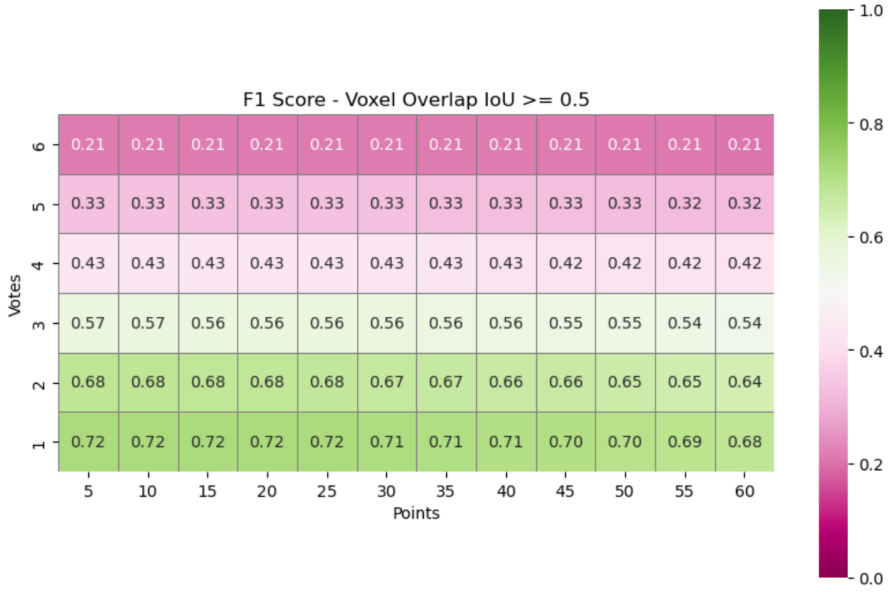


Table A.7: **F1 Score 3D Object Center Prediction - Train Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the F1 score with a fixed parameter of a minimal overlap threshold of 0.5 for different thresholds for minimum votes and cluster sizes (points).

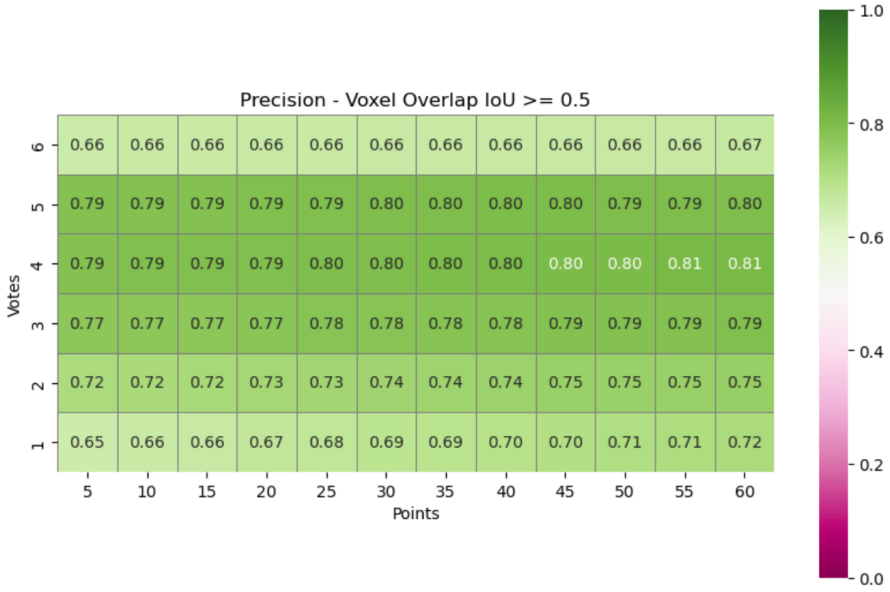


Table A.8: **Precision 3D Object Center Prediction - Train set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the precision with a fixed parameter of a minimal overlap threshold of 0.5 for different thresholds for minimum votes and cluster sizes (points).

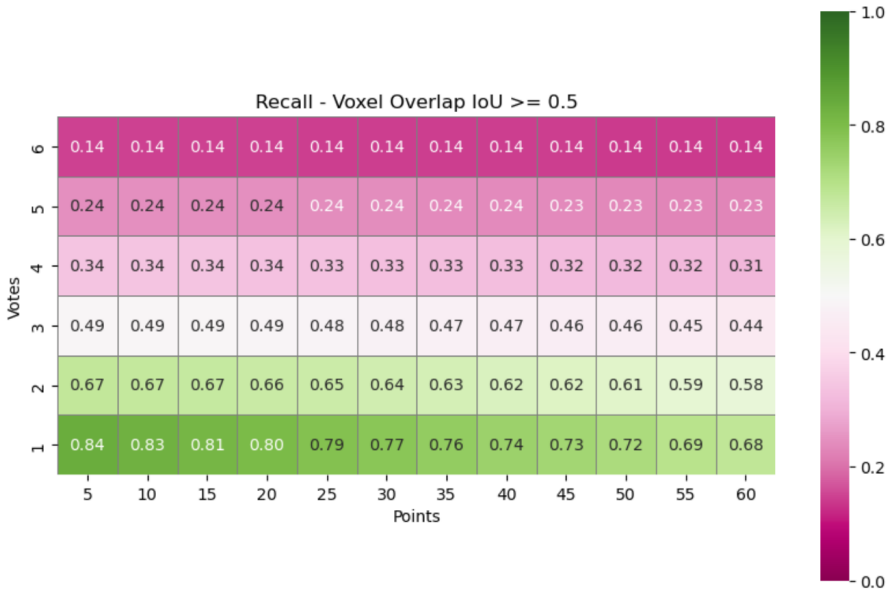


Table A.9: **Recall 3D Object Center Prediction - Train Set:** D3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the recall with a fixed parameter of a minimal overlap threshold of 0.5 for different thresholds for minimum votes and cluster sizes (points).

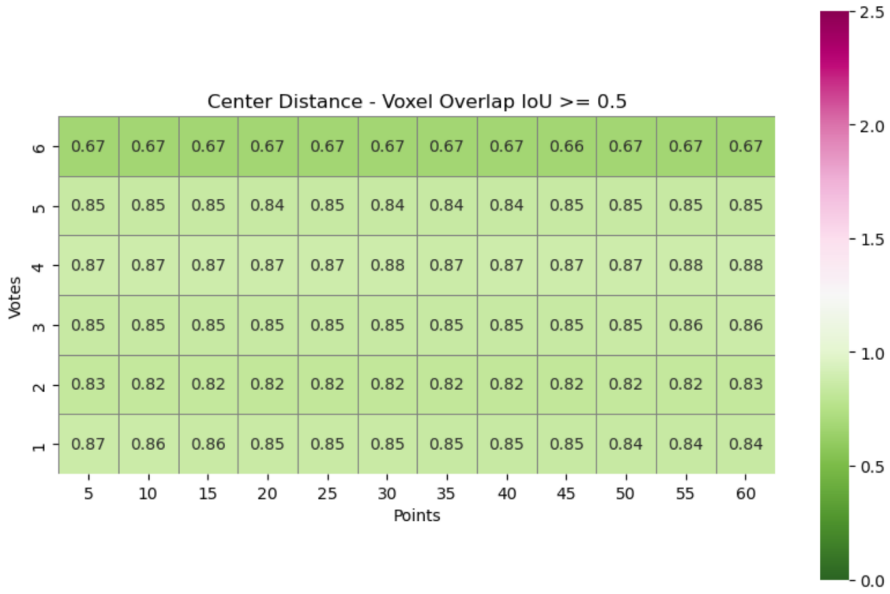


Table A.10: **Average Center Distance 3D Object Center Prediction - Train Set:** OD3DU uses a voxel IoU based voting mechanism to predict 3D object centers. The heatmap shows the result for the average center distance with a fixed parameter of minimal overlap threshold of 0.5 for different thresholds for minimum votes and cluster sizes (points).

A.3 Qualitative Results of 3D Predictions

A.3.1 Focus on Moved Object Instances

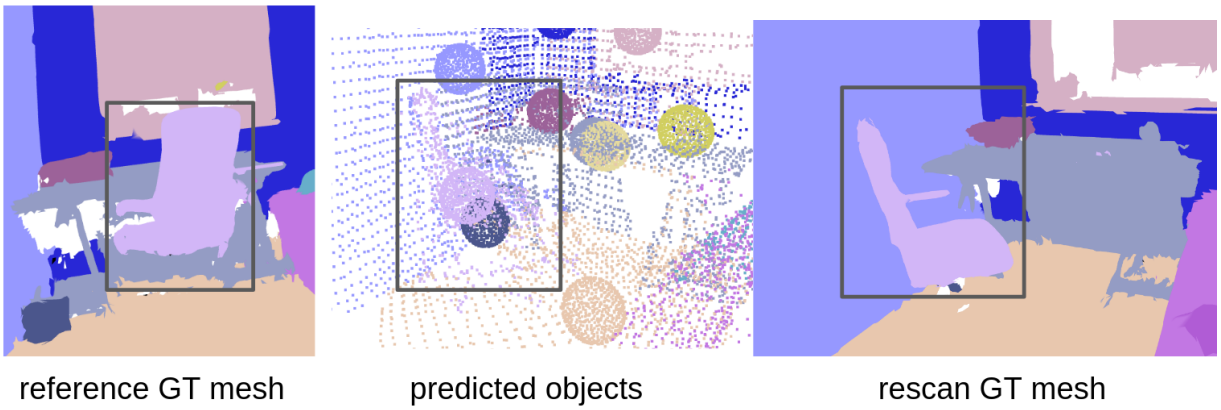


Figure A.3: **Object Center Prediction Dynamic Chair:** The boxes in the figure show the chair that moved between the capture times of the reference scene and the rescan scene. The chair in the frame were correctly predicted. The prediction for the entire scene used minimal votes = 3 and minimal points = 45. F1 score: 0.87, precision: 0.82, recall: 0.94.

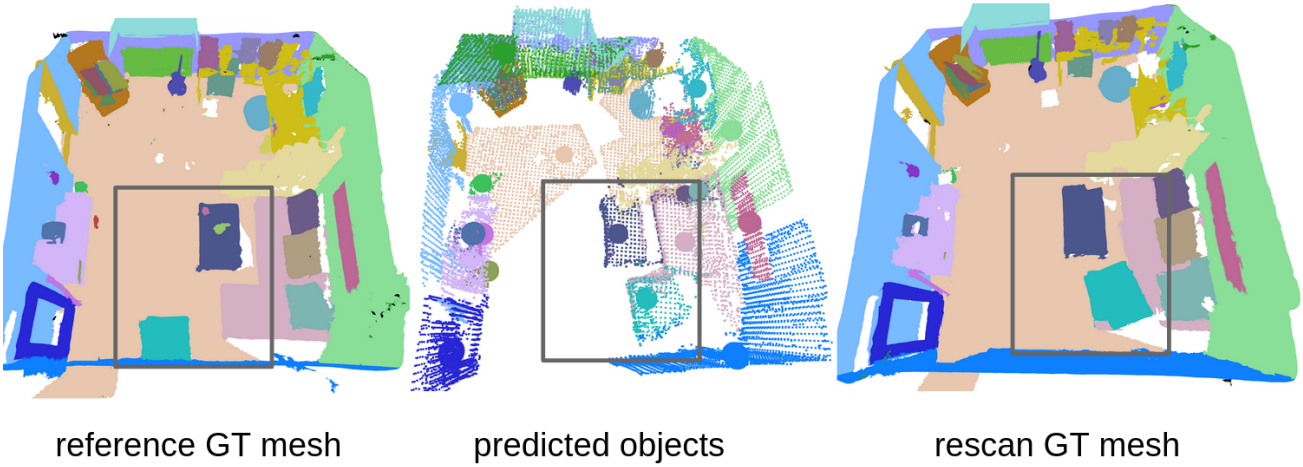


Figure A.4: **Object Center Prediction Dynamic Area:** The boxes in the figure show the movement area between the capture times of the reference scene and the rescan scene. The turquoise furniture moved to the right while the small elements on the table disappeared. The prediction for the entire scene used minimal votes = 3 and minimal points = 45. F1 score: 0.67, precision: 0.53, recall: 0.89.

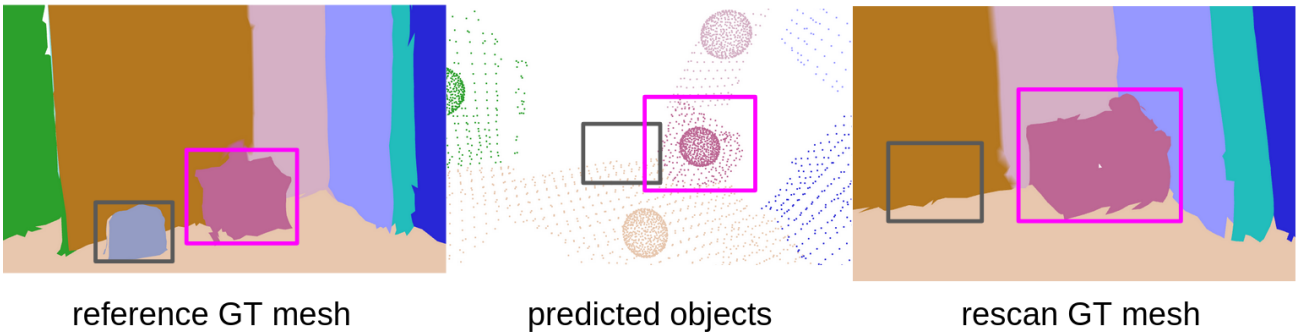


Figure A.5: **Object Center Prediction Dynamic Area Small Movement:** The boxes in the figure show the objects that moved between the capture times of the reference scene and the rescan scene. The gray object disappeared while the pink one moved slightly to the right. The prediction for the entire scene used minimal votes = 3 and minimal points = 45. F1 score: 0.92, precision: 0.92, recall: 0.92.

A.3.2 Complex Scenes

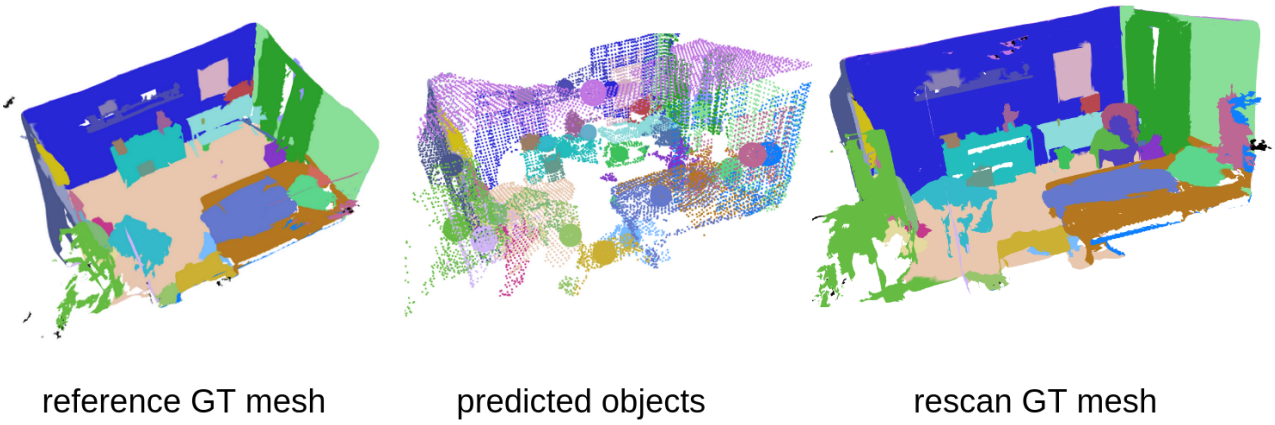


Figure A.6: **Object Center Prediction Complex Scene 1:** This scene contains many objects and also introduces new objects compared to the reference scene. They successfully do not get mixed up with reference objects. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.91, precision: 0.86, recall: 0.97.

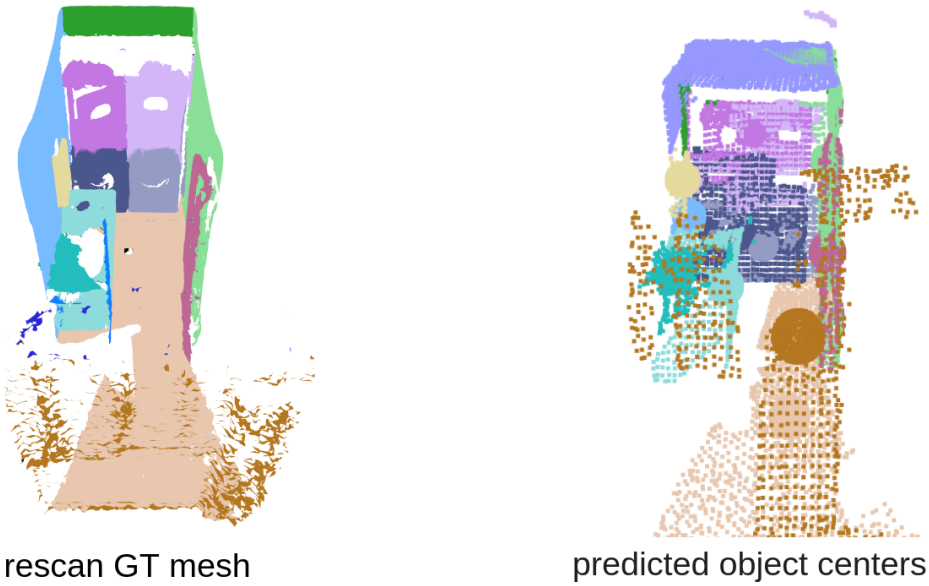


Figure A.7: **Object Center Prediction Complex Scene 2:** This scene is difficult because it contains 4 similar objects right next to each other. Even with noisy point cloud predictions the center get matched correctly. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.93, precision: 0.1, recall: 0.86.

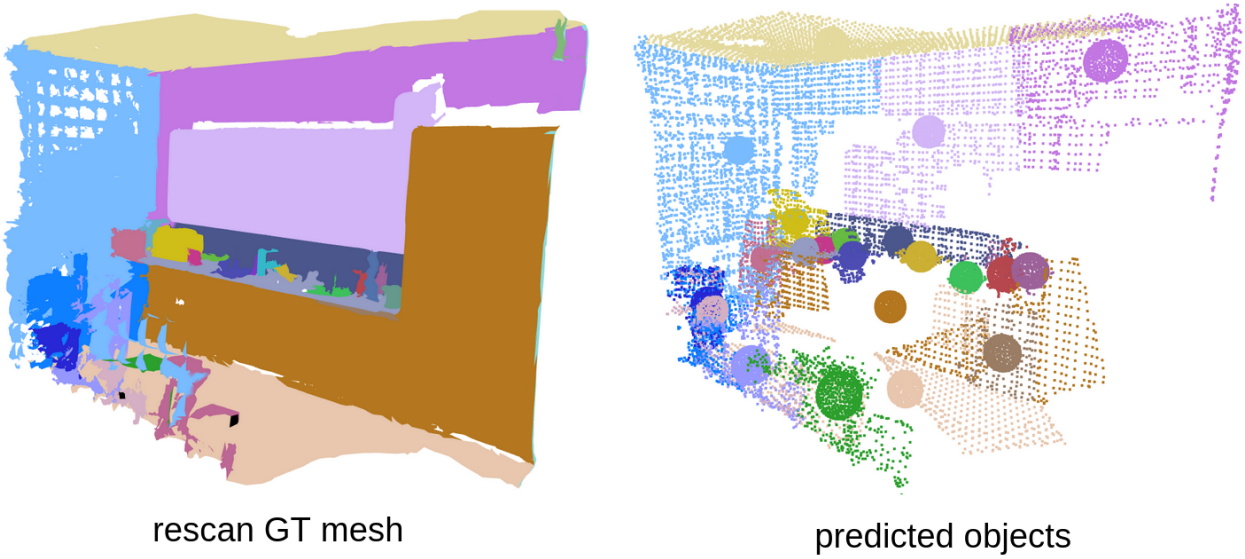


Figure A.8: **Object Center Prediction Complex Scene 3:** This scene shows a kitchen setup with many small items. Compared to the reference scan, no movement is visible. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.71, precision: 0.74, recall: 0.68.

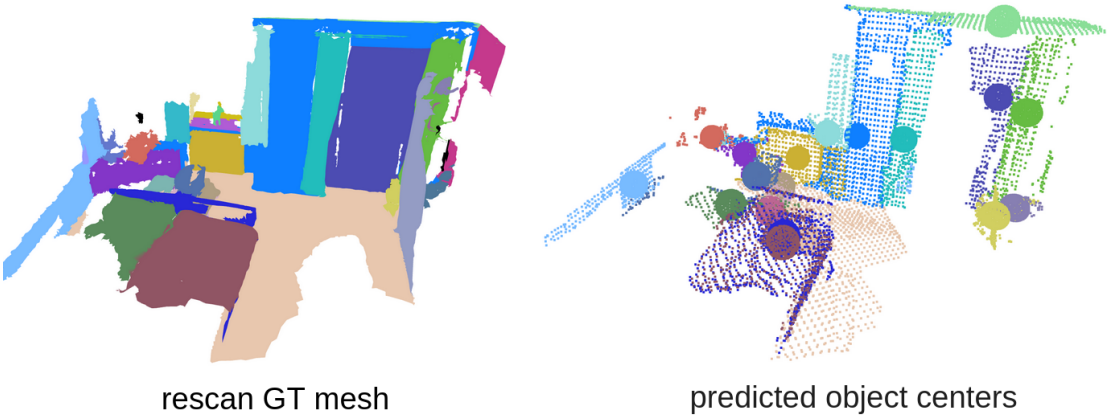


Figure A.9: **Object Center Prediction Complex Scene 4:** In this scene the predictions are very precise but some objects get omitted. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.73, precision: 0.89, recall: 0.62.

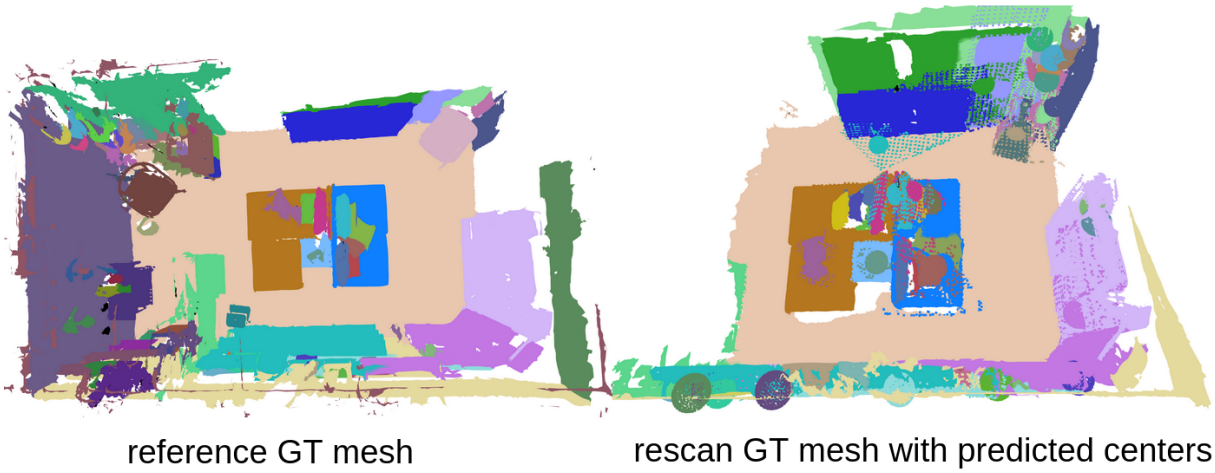


Figure A.10: **Object Center Prediction Complex Scene 5:** This scene has symmetries and many objects of the same instance, e.g., pillows. The distinction between these objects is difficult. There are also difficulties with the scene in general e.g. the walls. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.58, precision: 0.46, recall: 0.8.

A.3.3 Limitations

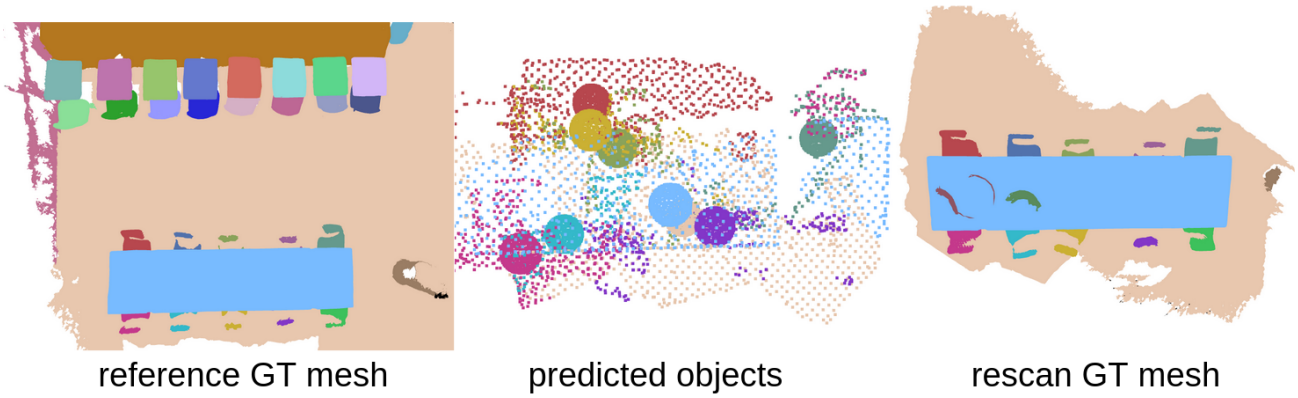


Figure A.11: **Example for Duplicate Limitation - Table and Chairs:** The reference scene consists of many tables and chairs, all populating the feature space with their respective features. In the rescan, different instances can not be distinguished. The prediction used minimal votes = 3 and minimal points = 45. F1 score: 0.37, precision: 0.31, recall: 0.45.

Bibliography

[1] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. *ArXiv*, abs/1607.08822, 2016.

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera, 2019.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021.

[4] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[5] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions, 2023.

[6] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation, 2022.

[7] Kyusik Cho, Dong Yeop Kim, and Euntai Kim. Zero-shot scene change detection, 2024.

[8] Wongun Choi, Yu-Wei Chao, Caroline Pantofaru, and Silvio Savarese. Understanding indoor scenes using 3d geometric phrases. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 33–40, 2013.

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

[11] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi proposal aggregation for 3d semantic instance segmentation, 2020.

[12] Jonathan Ginés, Francisco Martín, Vicente Matellán, Francisco J. Lera, and Jesús Balsa. Dynamics maps for long-term autonomy. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 85–90, 2017.

[13] Clara Gomez, Alejandra C. Hernandez, Erik Derner, Ramon Barber, and Robert Babuška. Object-based pose graph for dynamic indoor environments. *IEEE Robotics and Automation Letters*, 5(4):5401–5408, 2020.

[14] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023.

[15] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans, 2019.

[16] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization, 2022.

[17] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. 2023.

[18] Nassir Navab Federico Tombari Matthias Niessner Johanna Wald, Armen Avetisyan. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings IEEE International Conference on Computer Vision (ICCV)*, 2019.

[19] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015.

[20] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras, 2017.

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[22] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32 – 73, 2016.

[23] Jean Lahoud and Bernard Ghanem. 2d-driven 3d object detection in rgb-d images. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4632–4640, 2017.

[24] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks, 2021.

[25] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[26] Julian Mason and Bhaskara Marthi. An object-based semantic world model for long-term change detection and semantic querying. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3851–3858, 2012.

[27] Yang Miao, Francis Engelmann, Olga Vysotska, Federico Tombari, Marc Pollefeys, and Dániel Béla Baráth. Scenegraphloc: Cross-modal coarse visual localization on 3d scene graphs, 2024.

[28] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[29] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[30] Yue Qiu, Shintaro Yamamoto, Ryosuke Yamada, Ryota Suzuki, Hirokatsu Kataoka, Kenji Iwata, and Yutaka Satoh. 3d change localization and captioning from dynamic scans of indoor scenes. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1176–1185, 2023.

[31] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans, 2020.

[32] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: from slam to spatial perception with 3d dynamic scene graphs, 2021.

[33] Ragav Sachdeva and Andrew Zisserman. The change you want to see (now in 3d), 2023.

[34] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image, 2016.

[35] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J.J. Leonard. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS*, pages 1871–1878, Vilamoura, Portugal, October 2012.

[36] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] Rui Wang and Xuelei Qian. *OpenSceneGraph 3.0: Beginner's Guide*. Packt Publishing, 2010.

[38] Daniel Wilbers, Lars Rumberg, and Cyrill Stachniss. Approximating marginalization with sparse global priors for sliding window slam-graphs. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 25–31, 2019.

[39] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences, 2021.

[40] Yingda Yin, Yuzheng Liu, Yang Xiao, Daniel Cohen-Or, Jingwei Huang, and Baoquan Chen. Sai3d: Segment any instance in 3d scenes, 2024.

[41] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019.

[42] Liyuan Zhu, Shengyu Huang, and Iro Armeni Konrad Schindler. Living scenes: Multi-object relocalization and reconstruction in changing 3d environments. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[43] Liyuan Zhu, Shengyu Huang, Konrad Schindler, and Iro Armeni. Living scenes: Multi-object relocalization and reconstruction in changing 3d environments, 2024.