OpenMaskXR: Open-Vocabulary Scene Understanding in Extended Reality

Alexander Zank
ETH Zürich
alzank@ethz.ch

Michael Siebenmann
ETH Zürich
msiebenmann@ethz.ch

Hanqiu Li Cai
ETH Zürich
hlicai@ethz.ch

Omar Majzoub
ETH Zürich
omajzoub@ethz.ch

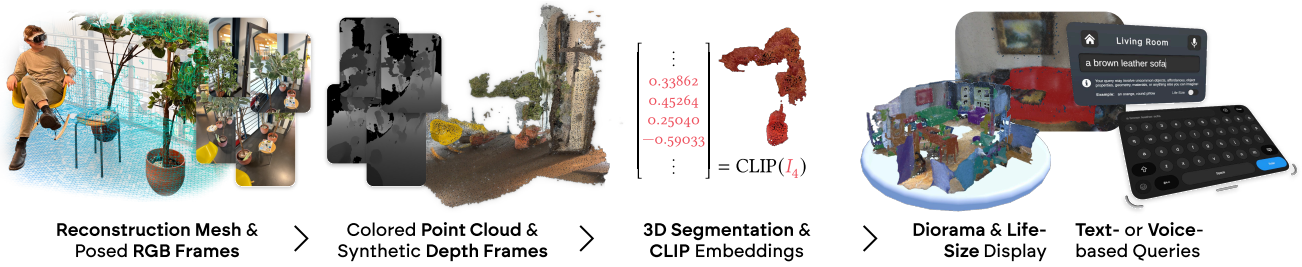


Figure 1. With OpenMaskXR, we demonstrate an end-to-end workflow for advanced scene understanding in XR. We implement various software components whose tasks range from scanning the environment using commodity hardware to processing and displaying it for open-vocabulary object querying.

Abstract

We present OpenMaskXR, a novel adaptation of the open-vocabulary 3D instance segmentation framework OpenMask3D, tailored for Extended Reality (XR) applications. OpenMaskXR enables users to query and highlight objects in their environment through text or voice commands in natural language, leveraging segmentation masks and CLIP embeddings derived from egocentric RGB-D images and 3D reconstructions. Our end-to-end workflow spans the entire process, from scene scanning using an iOS/visionOS application to real-time object querying in an OpenXR client whose usability we validate in two small-scale user studies. OpenMaskXR showcases how to embed neural scene understanding into commodity XR systems, suggesting new possibilities for intelligent immersive applications. Our source code is available at https://github.com/alexlike/openmaskxr.

1. Introduction

Scene understanding in Extended Reality (XR) remains in its early stages, with current consumer systems typically limited to approximate geometry reconstruction, plane detection, and rudimentary surface classification (e.g., doors, walls, tables) [29–31, 33]. However, deepening XR systems' understanding of the real world, for instance, by detecting 3D object instances and capturing their semantics, holds significant potential to enhance immersion, usability, and accessibility—whether by grounding interactions in users' real-world environments or selectively altering their perception of their surroundings.

We propose OpenMaskXR, an experimental adaptation of the open-vocabulary 3D instance segmentation method OpenMask3D [65] designed to evaluate its suitability for interactive XR applications. From a point cloud—precomputed or interpolated from a user-scanned mesh—and egocentric RGB-D images, OpenMask3D generates segmentation masks and CLIP embeddings for all objects in the scene. Leveraging these, we enable users to query objects by text or voice inputs and visually highlight their intended target(s).

Our contributions are threefold:

- We propose OpenMaskXR, an interactive XR application allowing users to highlight instances in a pre-scanned scene or their environment based on an arbitrary, open-vocabulary query.
- We outline the architecture and software components supporting this XR application.
- We evaluate the usability of the OpenMaskXR client application by conducting two small-scale user studies.

2. Related Work

This section aims to provide an overview of related work in the fields of closed and open-vocabulary scene segmentation, as well as multimodal AI assistants for human-computer interaction that drove the development of OpenMaskXR.

Closed-Vocabulary 3D Segmentation 3D semantic segmentation is a long-researched task that aims to assign a specific semantic class to each point in a 3D scene, enabling detailed scene understanding for applications in robotics, augmented reality, and beyond [1–3, 8, 14, 15, 19, 25, 26, 28, 38, 41, 46, 49, 50, 56, 57, 67, 68, 71, 73, 74, 77]. 3D instance segmentation extends this task by identifying distinct objects within the same semantic category, assigning unique instance masks for each object [13, 16, 22, 24, 35, 40, 61, 66, 70, 72, 78]. ScanNet200 is the standard benchmark used to evaluate 3D segmentation tasks in indoor environments [9, 60]. Mask3D [61] achieves state-of-the-art performance on this benchmark, leveraging a transformer architecture to generate 3D mask proposals along with their semantic labels. However, similar to other supervised methods, Mask3D relies on large sets of annotated 3D data such as the one provided by ScanNet200. Furthermore, the number of semantic categories is dictated by the training set. Both of these limitations restrict the generalization of these methods to real-world scenarios containing novel objects of unseen categories.

Open-Vocabulary 2D Segmentation With the rise in popularity of large vision-language models for image recognition, numerous works have targeted the task of open-vocabulary image semantic segmentation. [10, 18, 20, 23, 34, 39, 44, 47, 51, 59, 75, 76, 79, 80]. A number of these approaches rely on CLIP [58] features for image-level embeddings. More recently, works such as OpenSeg [18] and OV-Seg [47] extend the capabilities of foundation image-language models to semantic image segmentation by learning a pixel-level embedding. Nonetheless, their outputs heavily depend on the accuracy of 2D segmentation masks and thus require fine-tuning and further training.

Open-Vocabulary 3D Segmentation Building on the success of 2D open-vocabulary segmentation models, the research interests in recent years have shifted to extending the foundation models for 3D scene understanding. Aside from OpenMask3D, there exists a number works that aim to address this task [6, 11, 17, 21, 27, 34, 36, 37, 52, 55, 62, 63]. OpenScene [55] proposes using CLIP features extracted from the posed images of a given scene to achieve open-vocabulary 3D semantic segmentation. Some other approaches, such as LERF [36] and DFF [37] integrate language with NeRFs [53] by optimizing an extra feature field, capturing the semantics of the scene. It is important to note that all of these approaches achieve a limited understanding of object *instances* and face challenges when dealing with instance-related tasks. OpenMask3D addresses these issues by computing *class agnostic masks* that identify the object *instances* and then computes the CLIP features for each instance by leveraging 2D segmentation models and the posed RGB-D images.

Multimodal AI for Human-Computer Interaction With the rise of multimodal AI models, there has been a growing interest in the Human-Computer Interaction field, which aims to leverage these models to enhance user capabilities or provide assistance to people with disabilities. [5, 7, 12, 43, 45, 69]. Some works, such as WorldScribe [5] and XR-Objects [12], leverage Vision-Language Models (VLM) and Multimodal Large-Language Models (MLLM) to improve the Operating System's understanding of user surroundings with the goal of supporting new interactions. OpenMaskXR envisions a similar future, using open-source models in the design to allow users not only to query object instances but also to ask arbitrary questions about their surroundings.

3. Method

We now provide an overview of the end-to-end pipeline that forms the backbone of OpenMaskXR.

3.1. Architecture

The architecture of OpenMaskXR (see Figure 2) consists of four main components: the UI frontend, the main processor, the OpenMask3D service, and the sensing backend. A standalone XR headset is connected to our GPU-powered server via a RESTful API, as explained in Section 3.3.

The UI frontend implemented in Unity is part of the client application on the headset (Section 3.4) and includes scene selection from preprocessed examples, scaling and placement of the scene, and query capabilities through keyboard or voice input, with matching instances being highlighted.

The main processor and the OpenMask3D service run on the server (Section 3.3). The main processor, written in Python, performs point cloud sampling, depth image synthesis, triangle-to-instance mapping, and point cloud coloring, and returns CLIP embeddings from text. The OpenMask3D service is implemented in Python and runs in a Docker container, performing instance segmentation and providing per-instance CLIP embeddings.

Finally, the sensing backend (Section 3.2) operates both on an XR headset and an iOS device. The iOS application performs marker localization, environment mesh reconstruction, and camera frame delivery, whereas the headset is responsible for microphone streaming to the server.

3.2. Sensing App

Modern commodity XR headsets typically restrict direct access to onboard sensors for application developers, often requiring a strict enterprise entitlement to protect user privacy.

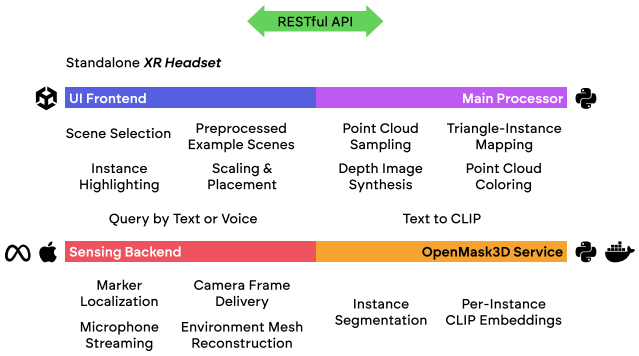


Figure 2. Overview of OpenMaskXR's architecture.

However, access to the main camera is essential for processing objects in users' environments. To address this limitation, we developed the OpenMaskXRSensingEmulator, an iOS application that emulates this functionality.

Built using Apple ARKit, RealityKit, Model I/O, and SwiftUI from the iOS 18.0 and visionOS 2.0 SDKs, the application allows users to scan their surroundings while capturing *reconstruction meshes* (stitched together into one Wavefront OBJ), *posed RGB frames from the main camera* at 5 FPS, and *camera intrinsics*. See Figure 1 for a visualization of these outputs. Additionally, it supports world-space *localization of a printed marker* for manual alignment at display time in the OpenMaskXR client.

The sensing app is also compatible with Apple Vision Pro. However, deploying it on this platform requires an enterprise entitlement, which we could not secure during development. As a result, its functionality on Vision Pro remains untested.

3.3. Server

We developed our backend API using the Flask Python library, with several routes the headset might call. Some of these routes were originally meant to be used for real-time scene processing, but such processing was found to be far from real-time. Our RTX 4090-powered server required more than 5 minutes for an average ScanNet scene and more than 10 minutes for users' scans. Therefore, we decided to perform scene processing offline on the server, not as part of the API (Sections 3.3.1 and 3.3.2). The server provides a route for *uploading mesh files and camera intrinsics*, a route for *receiving posed RGB frames*, and a route originally intended for *recalling reconstructed mesh files* from the server, returning them in a zip folder, together with the corresponding triangle id to object id and object id to CLIP embedding mappings as JSON files.

Furthermore, the API includes a route for *obtaining a CLIP embedding from a textual user query*, and two more routes to incorporate a *virtual AI assistant* into the API as explained in Section 3.5.

3.3.1. Pre-processing

As laid out previously in Section 3.2, the user-scanned data is output as a set of posed RGB frames, the uncolored triangle mesh of the scanned environment, as well as the camera intrinsics $\mathbf{K}$. At this stage, the data is not ready to be fed into OpenMask3D, as it requires depth images and the environment in point cloud format. The preprocessing pipeline has two main components: *depth-image rendering* and *point cloud sampling*. This pipeline was fully implemented in Python using the Open3D library [81].

Depth image rendering We simulate the depth camera using a pinhole model with the same camera intrinsics as the RGB camera. Given the poses, we project rays within the camera frustum onto the mesh. Specifically:

- The camera's extrinsic matrices $\mathbf{E}_i$ for the i-th frame are computed as the inverse of the respective 4×4 homogeneous pose matrix. This extrinsic matrix $\mathbf{E}_i$ transforms world coordinates to the camera's coordinate system at the respective frames.
- For each pixel in the camera view, a ray is traced onto the mesh surface to determine the intersection point.
- The depth of the intersection point is extracted as the z coordinate of the point expressed in the camera frame (z_c), resulting in a depth map for the entire image plane.

To ensure compatibility with OpenMask3D, the depth values are scaled to millimeter precision and stored as 16-bit integers. Depth values corresponding to pixels where no intersection occurs are set to zero. The resulting depth images are then saved in the required format for OpenMask3D.

Point cloud sampling OpenMask3D takes as input a point cloud $\mathbf{P} \in \mathbb{R}^{P \times 3}$, with P being the number of points in the point cloud. To obtain this, we begin by uniformly sampling the mesh with $P = 200000$. The result of this is then refined using Poisson disk sampling at a downsampling ratio $r_d = 0.75$, leading to an uncolored point cloud $\mathbf{P}$ of 150000 points evenly distributed in the scanned environment.

We then proceed to paint this point cloud by first projecting the points onto the image plane for each frame using the intrinsic matrix $\mathbf{K}$ and the extrinsic matrix $\mathbf{E}_i$, obtaining the 2D homogeneous coordinates $\mathbf{p_{2D}}$:

$$\mathbf{p_{2D}} = (u, v, 1)^\top = \mathbf{K} \cdot \mathbf{E}_i \cdot \mathbf{x_w}$$

Where $\mathbf{x_w}$ denotes the coordinates of a given point expressed in the world frame. Note that not all of the points are within the camera frustum, thus we filter out the ones with their 2D homogeneous coordinates falling out of the image height H and width W.

For each point in the camera frustum, we compare the corresponding depth (z_c) value with the synthetic depth image rendered in the previous step. If the difference in depth

is within a small threshold ϵ, the point is considered to be visible. Using the pixel coordinates, we retrieve the RGB value at the corresponding pixel and mark the corresponding point as painted.

With this approach, points that do not project onto valid pixels or have inconsistent depth values retain their original uncolored state. Furthermore, the already painted points will not be painted a second time. This was considered sufficient for our purposes, but it might require careful handling and further improvement. In the final step, the points with outlier colors are determined by the Euclidean distance between their normalized RGB value and the average normalized RGB value of their k-nearest neighbors in 3D space. If the difference is larger than a threshold $\delta = 0.5$, the color of the point is updated using the average value. We use a value of $k = 100$ points to ensure smooth color transitions across the point cloud.

3.3.2. Post-processing

OpenMask3D outputs a binary mask of dimension $M \times P$, with M being the number of instance masks proposed by the module and P the number of points in the point cloud. This binary mask indicates whether the j-th point in the cloud belongs to the i-th instance. Furthermore, it returns the CLIP feature vectors associated with each instance. As the XR client displays meshes instead of point clouds, we need to convert these point-based mask outputs to mesh-based formats so the client can use this information to correctly highlight object instances. The postprocessing pipeline consists of two steps: *triangle-to-instance mapping* and *feature export*.

Triangle-to-Instance Mapping To associate mesh triangles with object instances, we use a nearest-neighbor approach based on the point cloud and the output binary instance mask. Let the triangular mesh $\mathbf{M}$ be defined by its set of vertices $\mathbf{V}$ and set of triangles $\mathbf{F}$. Given the 3D coordinates $\mathbf{x}$ of the vertices of triangle F_i, we compute its centroid $\mathbf{C}_i$:

$$\mathbf{C}_i = \frac{\mathbf{x}_{i,0} + \mathbf{x}_{i,1} + \mathbf{x}_{i,2}}{3}$$

Given each triangle centroid, we use a k-nearest neighbor search on the input point cloud $\mathbf{P}$. The value of $k = 10$ was set for the nearest neighbor search, which returned satisfactory results for our purposes. The triangle is assigned to the instance m with the highest count of neighbors belonging to it. To avoid duplicate assignments, we maintain a set $\mathcal{A}$ containing the already assigned triangles.

Feature export We export the obtained features from the previous step (Triangle-Instance mask), as well as the CLIP features returned from OpenMask3D, into JSON format, which is then stored by the headset, enabling the querying and instance highlighting directly on-device (apart from a required server API call to embed the user query into the CLIP space).

We also slightly modify the source code of OpenMask3D to keep, per instance, the top k views in which the instance is most visible. We keep these most visible images to facilitate VLM queries, with the implementation details to be further explained in Section 3.5.

3.4. XR Client

We developed a cross-platform XR application using Unity 6000.0.23f1 with OpenXR as an XR plug-in provider. While we developed and tested our app mainly on a Meta Quest 3 headset, only relying on OpenXR means we can easily build for other XR headsets implementing the OpenXR specification, for instance, the HTC Vive Focus 3, Pico 4, or Magic Leap 2. On Apple Vision Pro, we let Unity directly render a fully immersive space via Compositor Services. To save time, we ignored the primary input modality of eye gaze-and-gesture and instead reused our hand interaction system. The lack of OpenXR compliance in visionOS means that interactions relying on OS-level gesture recognition remain unavailable.

Our application is based on Unity's Mixed Reality project template[1]. We allow XR interaction both through controllers and hand-tracking, with an automatic switch between these options when the user picks up or puts down controllers. All UI elements and 3D objects can be manipulated directly through poking/grabbing or indirectly through a ray interaction. Subsequently, we will briefly describe the functionality of our home and query menus and their implementation.

Home Menu The main purpose of the home menu (see Fig. 3) is for users to choose a scene for open-vocabulary querying. We selected six diverse 3D scans of the ScanNet200 dataset, namely an apartment, a bathroom, a living room, a laboratory, a kitchen, and a gym. Further, we also support interaction with a custom scan of the user's actual environment through our sensing app, see Section 3.2. As soon as the user selects a scene, the home menu UI elements slowly fade away, and the scene model ascends in front of the user, with the query UI panel appearing. We implemented these transitions by utilizing coroutines that smoothly interpolate between desired transparency states, object positions, and rotations, relying on different easing functions[2] for visually pleasing results. Users can freely toggle whether they want to interact with our app in

[1] https://docs.unity3d.com/Packages/com.unity.template.mixed-reality@2.0/manual/index.html, accessed on 01.01.2025

[2] https://easings.net/, accessed on 01.01.2025

Figure 3. Overview of UI elements in OpenMaskXR. The left panel illustrates the home menu UI, featuring elements for scene and functionality selection. The right panel displays the query menu, including a panel for querying objects through voice or text (top) and a panel for filtering highlighted instances (bottom).

passthrough mode, where they can see their real environment, or hide it in VR mode. Further, it is possible to switch between slider behaviors (elaborated upon in the next section) and show/hide debug logs.

Query Menu In the query menu (see Fig. 3), users can enter open-vocabulary queries through either voice, where we rely on Meta's cross-platform Voice SDK[3], or through a keyboard, where we adapted Unity's Spatial Keyboard, part of the XR Interaction Toolkit. We support two scene visualization modes, between which the user can freely switch:

- **Diorama Mode:** The 3D model is scaled to 10% of its original size and placed on a round table. It can be freely moved around and scaled. This mode allows the user to get an overview of the scene and inspect it from a birds-eye perspective.
- **Life-Size Mode:** The 3D model is visualized with the original size, aligned to the floor, and cannot be moved. In case a custom scan is used, and passthrough mode is enabled, we hide the scanned mesh (since it is identical to the real environment) and align the virtual model with the real world through a custom marker. This mode allows users to fully immerse themselves in the scene.

After an open-vocabulary query is entered, matching instances of the current scene are visually highlighted with distinct colors. How closely an instance matches a user query is determined through the cosine similarity of their normalized CLIP vectors. When the *non-linear slider* option is turned off, we allow users to directly specify a threshold for cosine similarity to be considered a match through a slider. A histogram of cosine similarities of all instances in the scene is integrated into the slider, making the process of specifying a fitting threshold more intuitive. In case the *non-linear slider* is enabled, the slider is instead used to specify the fraction of closest matching objects that should be shown. For example, if we set the slider to 50%, 50% of all objects are highlighted, namely those where the cosine similarity is above the median cosine similarity.

[3] https://developers.meta.com/horizon/documentation/unity/voice-sdk-overview/, accessed on 01.01.2025

3.5. Multimodal AI Models

To process more complex queries, we used a large language model (LLM) and a VLM provided by Ollama [54]. A route was added to the Flask API that uses the Llama 3.2 LLM [32] with 3 billion parameters to extract a physical object from a query. For instance, if the query is "Hmm, could you help me to find the monstera plant please?" the LLM is expected to extract the object *monstera plant*. This physical object can then be fed into OpenMask3D as a query. As mentioned in Section 3.3.2, we modified the OpenMask3D source code to output the top k views where the instance is most visible. The implementation outputs a JSON file with the IDs of instances as the keys and the IDs of the images corresponding to the top k views of those instances as the values. As a result, these images can be fed into a VLM along with the original complex query. A route using the Llava VLM [48] with 7 billion parameters was integrated into the API to process queries on such images. In our example, this corresponds to passing the instance id of the queried object and the name of the folder containing the JSON file that specifies the object's top k views into the VLM along with the original user query. The API returns the answer provided by the VLM. This allows for more complex queries not only concerned with a single object but also relationships between objects, such as "What is above the brown table?". However, as a single VLM query required a processing time of around 10 minutes on a PC, we refrained from integrating VLM querying into the client application.

4. Evaluation

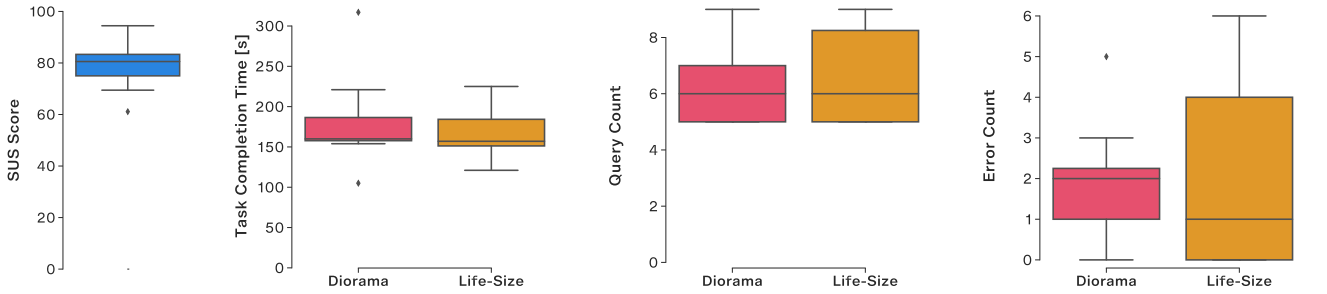
To evaluate the usability of the OpenMaskXR client and compare its features, we ran two small-scale user studies ($N \in \{13, 8\}$, classified Type 2 by Ledo et al. [42]).

In unstructured *usability sessions*, participants explored OpenMaskXR at their own pace, providing mostly qualitative feedback. In a controlled *user experiment*, we evaluated participants' performance finding objects, comparing the two display modes Diorama and Life-Size.

4.1. Experimental Setup

4.1.1. Usability Session Design

The usability sessions followed a semi-structured approach, allowing participants to freely interact with OpenMaskXR. While participants provided verbalized *qualitative feedback* throughout their experience, we also gathered quantified experience in a structured questionnaire, namely *intuitiveness*



(a) SUS scores indicating overall usability ratings.

(b) Total time spent locating five room-object pairs in two display modes.

(c) Total queries issued locating five room-object pairs in two display modes.

(d) Total verbalized errors and unexpected behaviors locating five room-object pairs in two display modes.

Figure 4. Box plots illustrating our quantitative results. Note that no significant differences were observed.

of scene partitioning, *system usability* (on the System Usability Scale [4]), and *preference for display modes, slider types, and input methods*.

4.1.2. User Experiment Design

For the user experiment, we chose a fully counterbalanced within-subjects design to minimize order effects. The independent variable was the display mode with two levels: *Diorama* and *Life-Size*. The dependent variables included *task completion time [s]*, *query count* (total, including erroneous ones), and *error count* (as verbalized).

4.1.3. Participants

For both studies, we recruited 8 participants (4 men, 4 women), ages 22 to 68 ($M = 39$, $SD = 21$) via convenience sampling. They reported their alertness on the Stanford Sleepiness Scale [64], which ranges from 1 (wide awake) to 7 (fully asleep), and their previous experience with XR. Participants' mean sleepiness score was 2.5, and all had previously experienced head-worn XR at least once.

Extending the participant pool for the unstructured usability sessions are 5 people who visited our booth at ETH Zürich's Mixed Reality Demo Day.

4.1.4. Apparatus

Participants used the OpenMaskXR client with hand tracking on Meta Quest 3 and Apple Vision Pro (50/50 split). Besides the precomputed ScanNet [60] example scenes, we hard-coded a custom scan of a mock environment setup for Demo Day. The headsets connected to our server via WiFi for text-to-CLIP conversion. We observed no latency or outages.

4.1.5. Procedure

Upon arrival, we introduced participants to OpenMaskXR using our poster, answered their immediate questions, and recorded their age, gender, alertness, and XR experience. Then, participants could freely explore the app, opening scenes, switching between display modes and slider types, and querying objects using the keyboard and their voice. We took notes of their comments and followed up with a questionnaire capturing their quantified experience.

Those also participating in our user experiment were then reimmersed and asked to "open the [specified] room, query for the [specified] object and adjust the slider such that the object is barely visible" for five room-object pairs (short visual descriptions) twice—once in Diorama and once in Life-Size mode. We asked users to do so "as quickly and accurately as possible [and to] speak out loud when something doesn't behave as [they]'d like." We recorded their verbal comments, number of queries, uttered errors, and completion times.

4.2. Results

4.2.1. Quantified Experience

All 13 participants agreed that the way OpenMask3D partitioned the scene "made sense intuitively." Following Brooke [4], OpenMaskXR yields System Usability Scores from 61 to 94 ($M = 80$, $SD = 9$, Figure 4a).

Comparing display modes, 77% of participants preferred *Diorama* over *Life-Size* when it comes to feeling "more oriented" and "faster to use." However, 69% of participants rate *Life-Size* to be "more fun to use." All but one participant found the *non-linear* slider "more intuitive" than the *linear* one and expressed a preference for *voice* over *text* input.

4.2.2. Performance

With the normality assumption intact (Shapiro-Wilk's $p \geq 0.7$) on all quantitative measures, we conducted three paired-sample t-tests:

We found no significant difference in *task completion time [s]* between *Diorama* ($M = 181$, $SD = 63$) and *Life-Size* ($M = 168$, $SD = 34$); $t(7) = 0.695$, $p = 0.509$.

We found no significant difference in *query count* between *Diorama* ($M = 6$, $SD = 1$) and *Life-Size* ($M = 7$, $SD = 2$); $t(7) = -0.482$, $p = 0.644$.

We found no significant difference in *error count* between *Diorama* ($M = 2$, $SD = 2$) and *Life-Size* ($M = 2$, $SD = 2$); $t(7) = 0.000$, $p = 1.000$.

See Figure 4 for an illustration of our quantitative results.

4.2.3. Qualitative Feedback

Using a deductive approach, two coders identified six codes in participants' verbalized feedback (Cohen's $\kappa = 0.71$):

Partition Quality All participants praised OpenMask3D's segmentation, which our prototype visualizes with colored overlays. For instance, P7 notes "it recognizes chairs, pictures on the wall, small objects placed on a table, instruments, individual cupboards," P4 liked the "distinct colors," and P6 remarked that, as expected, "floors and walls [were] not counted as partitions."

Results Precision Feedback on the goodness of results for user queries was mixed. For instance, P1 remarked how "sometimes [their] queries did not match what [they] had expected. When [they] said plant, it highlighted a painting." However, instead of blaming the system, two looked for excuses: P1 then said, "That makes sense since a plant is in the painting." P8 remarked they "had the feeling that the objects were better identified in [the user experiment]." Crucially, P8 had often queried for affordances during the exploratory phase.

UI Design Participants commented on OpenMaskXR's user interface: P2 "like[d] the overall design" and found OpenMaskXR's "computation [..] fast, almost instant." P10 "could easily choose the different scenes" because "the main menu was very intuitive." When transitioning between Diorama and Life-Size display modes, many participants gasped. P4 "enjoyed the UI transitions."

Criticizing OpenMaskXR, P2 said "the color contrast could be better" and suggested the distance between the user and the main menu was "a bit close, especially with the limited POV."

Unexpected Behavior Participants reported unexpected behavior for two UI subsystems:

When querying using their voice, P7 said they "don't understand when it is listening and how [they] can end a recording." P5 blamed the "button [which] remain[ed] grey when aiming at it, although recording had already started." Occasionally, the speech-to-text backend would "incorrectly transcrib[e] 'chair' as 'share'," frustrating P3 and other participants.

When adjusting the results threshold, P7 (and P1) noted, "the slider is a bit hard to operate, especially if I look at the diorama at the same time and my hand is outside of view [and] the tracking stops."

Suggestions Two participants verbalized feature requests. According to P10, "it would also have been nice to be able to choose objects by clicking on them, and, for example, see similar objects highlighted." P7 would "find it useful to be able to see through walls facing [them] as if opening a dollhouse."

Comparison Commenting on the display variants, P1 remarked "it's easier to see things quickly in Diorama. In Life-Size, [they] need to orient [themselves] better."

5. Discussion

Users' Experience The evaluation of OpenMaskXR revealed promising usability and user engagement outcomes, evidenced by high System Usability Scores (see Figure 4a) and largely positive verbal feedback, especially regarding UI design. Our design decision to incorporate both a Diorama and Life-Size mode has been quantitatively justified, as users report feeling more oriented in Diorama mode but being more engaged in Life-size mode. Interestingly, no significant difference in task completion time, query count, or error count was observed between these two modes. This is unsurprising for the error count, as the majority of verbalized errors were attributed to confusions related to the voice input interface (refer to the *Unexpected Behavior* subsection in Section 4.2.3), which is independent of the current display mode. The confusion arose because the voice input button turned red to indicate activation but turned to a lighter shade of grey when hovered over, with the latter behavior taking higher priority. As a result, the button remained grey even after being clicked, only turning red when the XR ray was no longer hovering over it. We suspect significant differences in task completion time between modes could occur for more challenging environments than the rather small and familiar scenes from the ScanNet200 dataset.

The fact that all but one participant described the non-linear slider as "more intuitive" aligns with our expectations. Still, we believe that visualizing the precise distribution of similarities between objects and having the abil-

ity to adjust the absolute threshold value provides valuable insights for researchers to gain a deeper understanding of the performance of the open-vocabulary model. Therefore, we retain the non-linear slider mode as the default setting while providing the option to switch between modes in the app settings.

Limitations Existing consumer XR devices such as Meta Quest, Microsoft HoloLens and Apple Vision Pro provide limited access to the sensors that capture the required data to run OpenMask3D. From the available options, the only feasible XR device on which to develop the sensing backend was the Magic Leap 2. Due to the limited availability of this device during the project, we opted for an alternative option requiring the user to scan their surroundings using a custom iOS application. We use a marker to align the scanned 3D data with the environment in the XR application, effectively achieving a spatial anchor effect. This is not ideal as the initial vision for the project was to achieve an end-to-end app that allows user to both scan, view and interact with their environment within the XR device.

Furthermore, while querying pre-processed scenes in OpenMaskXR runs in real-time, we inherit OpenMask3D's limitation of requiring long processing times in the order of minutes to process a new scene. Together with the lack of sensor access, this stops our approach from becoming generalised into everyday scenarios.

Despite these limitations, OpenMaskXR remains a successful initial exploration of integrating the latest research in 3D scene understanding into a fully functional XR application. Looking beyond the scope of our work, XR operating systems could potentially leverage ad-hoc open vocabulary scene understanding to place smart widgets around the user's environment.

Future Work Future work includes porting the full sensing backend to the Magic Leap, removing the need for users to rely on an external device to scan their environment and allowing for a more seamless experience. Other future features include the integration of the Large Language Model and the Visual Language Model mentioned in Section 3.5 into the client application, which would leverage the top-k images in terms of visibility selected by OpenMask3D for each instance mask. This would enable users to ask more sophisticated questions about the scene, extending the capabilities of the application.

6. Conclusion

In this work, we introduced OpenMaskXR, an adaptation of OpenMask3D designed to bring open-vocabulary 3D instance segmentation to consumer-grade, head-worn Extended Reality. By enabling intuitive object querying through natural language inputs, OpenMaskXR highlights the potential of neural scene understanding for enhancing user interaction in mixed-immersion settings. Through an end-to-end workflow and small-scale user studies, we demonstrated its practicality and usability, despite current limitations related to sensor access and scene processing time.

Our results suggest that XR applications can feasibly embed intelligent scene understanding, paving the way for richer, more adaptive experiences. Future enhancements, including full backend integration with devices like Magic Leap 2 and the incorporation of advanced multimodal models, could refine the system and broaden its use cases. OpenMaskXR serves as a stepping stone towards more sophisticated, everyday intelligent XR systems.

References

[1] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for 3d point clouds. *International Journal on Robotics Research (IJRR)*, 2011. 2

[2] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Transactions On Graphics (TOG)*, 2018. 2

[4] J Brooke. Sus: A quick and dirty usability scale. *Usability Evaluation in Industry*, 1996. 6

[5] Ruei-Che Chang, Yuxuan Liu, and Anhong Guo. Worldscribe: Towards context-aware live visual descriptions. In *The 37th Annual ACM Symposium on User Interface Software and Technology (UIST 2024)*, 2024. 2

[6] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S. Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. *arXiv preprint arXiv:2209.09874*, 2022. 2

[7] Junlong Chen, Jens Grubert, and Per Ola Kristensson. Analyzing multimodal interaction strategies for llm-assisted manipulation of 3d scenes. In *Proceedings of the 32nd IEEE Conference on Virtual Reality and 3D User Interfaces (VR 2025)*, 2025. 2

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[10] Jian Ding, Nan Xue, Gui-Song Xia, and Dengxin Dai. Decoupling zero-shot semantic segmentation. 2022. 2

[11] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[12] Mustafa Doga Dogan, Eric J. Gonzalez, Karan Ahuja, Ruofei Du, Andrea Colaço, Johnny Lee, Mar Gonzalez-Franco, and David Kim. Augmented object intelligence with xr-objects. In *The 37th Annual ACM Symposium on User Interface Software and Technology (UIST 2024)*, 2024. 2

[13] Cathrin Elich, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. 3d-bevis: Birds-eye-view instance segmentation. In *German Conference on Pattern Recognition (GCPR)*, 2019. 2

[14] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. In *International Conference on Computer Vision (ICCV) Workshops*, 2017. 2

[15] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. Know what your neighbors do: 3d semantic segmentation of point clouds. In *European Conference on Computer Vision (ECCV) Workshops*, 2018. 2

[16] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi proposal aggregation for 3d semantic instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[17] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Clip on wheels: Zero-shot object navigation as object localization and exploration. *arXiv preprint arXiv:2207.04429*, 2022. 2

[18] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Scaling open-vocabulary image segmentation with image-level labels. In *European Conference on Computer Vision (ECCV)*, 2021. 2

[19] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[20] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[21] Huy Ha and Shuran Song. Semantic abstraction: Open-world 3d scene understanding from 2d vision-language models. In *Conference on Robot Learning (CoRL)*, 2022. 2

[22] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[23] Sunan He, Taian Guo, Tao Dai, Ruizhi Qiao, Bo Ren, and Shu-Tao Xia. Open-vocabulary multi-label classification via multi-modal knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023. 2

[24] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[25] Zeyu Hu, Xuyang Bai, Jiaxiang Shang, Runze Zhang, Jiayu Dong, Xin Wang, Guangyuan Sun, Hongbo Fu, and Chiew Lan Tai. Vmnet: Voxel-mesh network for geodesic-aware 3d semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[26] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[27] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 2

[28] Jing Huang and Suya You. Point cloud labeling using 3d convolutional neural network. In *International Conference on Pattern Recognition (ICPR)*, 2016. 2

[29] Apple Inc. ARKit in visionOS. https://developer.apple.com/documentation/arkit/arkit_in_visionos, . 1

[30] Microsoft Inc. Scene understanding. https://learn.microsoft.com/en-us/windows/mixed-reality/design/scene-understanding, .

[31] Magic Leap Inc. Spatial mapping. https://developer-docs.magicleap.cloud/docs/guides/features/spatial-mapping, 2024. 1

[32] Meta Platforms Inc. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, . 5

[33] Meta Platforms Inc. Scene understanding. https://developers.meta.com/horizon/design/mr-design-scene, . 1

[34] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. Conceptfusion: Open-set multimodal 3d mapping. In *Robotics: Science and Systems (RSS)*, 2023. 2

[35] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[36] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023. 2

[37] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[38] Hema Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Neural Information Processing Systems (NeurIPS)*, 2011. 2

[39] Weicheng Kuo, Yin Cui, Xiuye Gu, AJ Piergiovanni, and Anelia Angelova. Open-vocabulary object detection upon frozen vision and language models. In *International Conference on Learning Representations (ICLR)*, 2023. 2

[40] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R. Oswald. 3d instance segmentation via multi-task metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[41] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[42] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. Evaluation strategies for hci toolkit research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 174, New York, NY, USA, 2018. ACM. 5

[43] Jaewook Lee, Jun Wang, Elizabeth Brown, Liam Chu, Sebastian S. Rodriguez, and Jon E. Froehlich. Gazepointar: A context-aware multimodal voice assistant for pronoun disambiguation in wearable augmented reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024. 2

[44] Boyi Li, Kilian Q. Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[45] Jiahao Nick Li, Yan Xu, Tovi Grossman, Stephanie Santosa, and Michelle Li. Omniactions: Predicting digital actions in response to real-world multimodal sensory inputs with llms. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024. 2

[46] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Neural Information Processing Systems (NeurIPS)*, 2018. 2

[47] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[48] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024. 5

[49] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[50] Yan Lu and Christopher Rasmussen. Simplified markov random fields for efficient semantic labeling of 3d point clouds. In *International Conference on Intelligent Robots and Systems (ICIRS)*, 2012. 2

[51] Chao Ma, Yu-Hao Yang, Yanfeng Wang, Ya Zhang, and Weidi Xie. Open-vocabulary semantic segmentation with frozen vision-language models. In *British Machine Vision Conference (BMVC)*, 2022. 2

[52] Kirill Mazur, Edgar Sucar, and Andrew Davison. Feature-realistic neural fusion for real-time, open set scene understanding. In *International Conference on Robotics and Automation (ICRA)*, 2023. 2

[53] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[54] Ollama. Ollama. https://ollama.com/. 5

[55] Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser. Openscene: 3d scene understanding with open vocabularies. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[56] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[57] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Neural Information Processing Systems (NeurIPS)*, 2017. 2

[58] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. 2

[59] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[60] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 6

[61] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *International Conference on Robotics and Automation (ICRA)*, 2023. 2

[62] Nur Muhammad Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur D. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2304.01456*, 2023. 2

[63] Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint arXiv:2210.03080*, 2022. 2

[64] Azmeh Shahid, Kate Wilkinson, Shai Marcu, and Colin M. Shapiro. Stanford sleepiness scale (sss). In *STOP, THAT and One Hundred Other Sleep Scales*, pages 369–370. Springer New York, New York, NY, 2011. 6

[65] Ayca Takmaz, Elisabetta Fedele, Robert Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. OpenMask3D: Open-vocabulary 3D instance segmentation. In *Neural Information Processing Systems (NeurIPS)*, 2023. 1

[66] Ayça Takmaz, Jonas Schult, Irem Kaftan, Mertcan Akçay, Bastian Leibe, Robert Sumner, Francis Engelmann, and Siyu Tang. 3d segmentation of humans in point clouds with synthetic data. In *International Conference on Computer Vision (ICCV)*, 2023. 2

[67] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *International Conference on 3D Vision (3DV)*, 2017. 2

[68] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[69] Minh Duc Vu, Han Wang, Zhuang Li, Jieshan Chen, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. Gptvoicetasker: Advancing multi-step mobile task efficiency through dynamic interface exploration and learning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST 2024)*, 2024. 2

[70] Thang Vu, Kookhoi Kim, Tung M. Luu, Xuan Thanh Nguyen, and Chang D. Yoo. Softgroup for 3d instance segmentation on 3d point clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[71] Tianyi Wang, Jian Li, and Xiangjing An. An efficient scene semantic labeling approach for 3d point cloud. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2015. 2

[72] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[73] Silvan Weder, Hermann Blum, Francis Engelmann, and Marc Pollefeys. Labelmaker: Automatic semantic label generation from rgb-d trajectories. In *International Conference on 3D Vision (3DV)*, 2024. 2

[74] Daniel Wolf, Johann Prankl, and Markus Vincze. Fast semantic segmentation of 3d point clouds using a dense crf with learned parameters. In *International Conference on Robotics and Automation (ICRA)*, 2015. 2

[75] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[76] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Odise: Open-vocabulary panoptic segmentation with text-to-image diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[77] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[78] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. In *Neural Information Processing Systems (NeurIPS)*, 2019. 2

[79] Nir Zabari and Yedid Hoshen. Semantic segmentation in-the-wild without seeing any segmentation examples. *arXiv preprint arXiv:2112.03185*, 2021. 2

[80] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[81] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 3