HoloLabel: Augmented Reality User-In-The-Loop Online Annotation Tool for As-Is Building Information

Dhruv Agrawal*, Janik Lobsiger*, Jessica Yi Bo, Véronique Kaufmann, and Iro Armeni
ETH Zurich, Switzerland

Abstract

As-is building models are becoming increasingly common in the Architecture, Engineering, and Construction industry, with many stakeholders requesting this information throughout the lifecycle of a building. Devices equipped with RGB cameras and depth sensors being readily available simplifies the task of capturing and reconstructing an environment (scene) as a spatial 3D mesh or point cloud. However, the task of converting this purely geometric information to a semantically meaningful as-is building model is non-trivial. State-of-the-art practice follows a first step of acquiring the spatial 3D mesh on site and subsequently resorts in manual or assisted semantic labeling in the office, where experts often have to work for many hours using non-intuitive and error-prone tools. To address this inefficiency, we develop *HoloLabel*, an Augmented Reality application on HoloLens that allows users to directly and on-site annotate a scene in 3D with rich semantic information while simultaneously capturing its spatial 3D mesh. Our tool follows a user-in-the-loop protocol to perform the task of 3D semantic segmentation, i.e., each face of the 3D mesh should be annotated with a semantic label. We leverage the HoloLens's Spatial Mapping feature and build a 3D mesh of the scene while the user is walking around; at intervals, we apply an automatic geometry-based segmentation algorithm to generate segmentation proposals. The user then assigns predefined semantic labels to the proposals and – if necessary – uses a virtual paintbrush to refine the proposed segments or create new ones. Finally, the user has the option to add rich semantic descriptions (e.g., material, shape, or relationship to another object) to segments using voice-to-text technology. We aim to lay the groundwork to leverage upcoming mixed reality devices for intuitive synchronous as-is semantic building model generation directly in the real world.

Introduction

In recent years, the Architecture, Engineering, and Construction (AEC) industry is exploring ways to acquire faster and more accurately as-is information about the built environment. In several countries, as-is building models are becoming a requirement (e.g., USA (Administration 2022) and United Kingdom (Blackwell 2012)) and stakeholders are requesting this information throughout the lifecycle of a building to perform several downstream tasks (e.g., facility management or for renovation purposes).

State-of-the-art practice for creating an as-is building model consists of two steps: (i) *3D reconstruction*, where the environment's 3D geometry is captured on site as a 3D spatial mesh or point cloud with a sensing device, such as a laser scanner or an RGB-D camera; and (ii) *semantic modeling*, where - given the previous output - the user provides semantic information about the 3D geometry. Recent advancements in capturing systems allow for a faster and increasingly accurate 3D geometric representation of the environment (*scene*) with the use of easily portable devices, such as those that are handheld (e.g., Kaarta and Leika BLK2GO) or in backpacks (e.g., NavVis VLX and Leica Pegasus). Such devices can provide a 3D reconstruction of a large-scale, highly cluttered, and highly partitioned space in 1/10 of the time when compared to traditional tripod-based systems[1].

However, the step of semantic modeling has not seen similar progress. After capturing the environment, the user is required to return to the office and use specialized 3D software in order to create the as-is building model. This process is performed primarily by trained experts using a 2D screen and can take several hours for a single room (Nguyen et al. 2016), which is non-trivial in terms of labor time and costs (Brilakis et al. 2010, Woo et al. 2010, Jung et al. 2014). Extensive literature exists on automating this step that explores fully automatic semantic segmentation of 3D spatial data (e.g., (Tchapmi et al. 2017, Choy et al. 2019, Qi et al. 2017)). However, these approaches are neither accurate nor flexible enough for the AEC industry requirements. To meet these requirements, a user-in-the-loop approach is more appropriate (e.g., (Dai et al. 2017, Armeni et al. 2019, Wong et al. 2014)), since it allows to leverage the most out of automation while addressing accuracy concerns with user participation. However, despite addressing issues of accuracy and flexibility, the labeling process is still disjoint and offline from the data capturing step, introducing inefficiencies in the building model generation pipeline (e.g., when on-site one can make faster decisions on semantic labels). A more intuitive approach is to perform the capturing and labeling steps synchronously while on-site.

Augmented Reality (AR) technology, such as the Microsoft HoloLens (Microsoft 2022*g*), provides a technical platform to implement such an online capturing and labeling tool where the user is localized in the scene and can simultaneously capture, segment, and label the 3D spatial data in a natural and interactive method. The final semantically augmented data, once exported from the live annotation session, would require only minimal additional processing. The HoloLens comes with the built-in functionality for capturing and reconstructing spatial 3D meshes. Additional libraries, such as the Mixed Reality Toolkit (MRTK) (Microsoft 2022*f*), offer capabilities to

* Both authors contributed equally

[1]This is an empirical finding of the authors after performing several experiments with a variety of devices.

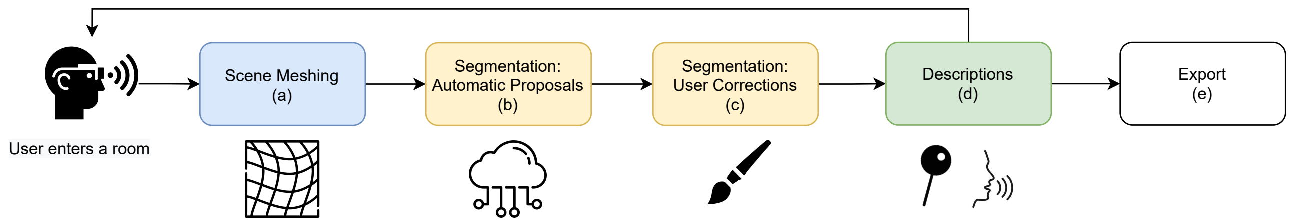

User enters a room
Scene Meshing (a)
Segmentation: Automatic Proposals (b)
Segmentation: User Corrections (c)
Descriptions (d)
Export (e)

Figure 1: User Workflow: When the user enters a new room, (a) we first reconstruct the 3D scene. Next, (b) the scene mesh is sent to a Python server for automatic segmentation. The user then (c) processes these segmentation proposals using a virtual brush. Finally, the user can (d) add richer semantic descriptions into the scene using a voice-to-text tool. The user can then proceed with the next room or export the annotations.

develop interactive augmented user interfaces.

In this work, we introduce the **HoloLabel**, an AR 3D scene semantic labeling tool with which *non-expert* users can directly annotate the semantic information of a scene on-site while simultaneously capturing its spatial information. Initially, the continuously captured 3D mesh of the room is automatically segmented into smaller geometrically meaningful segments (*segmentation proposals*) in the background, using a remote Python server. Once complete, users can assign semantic labels to these segments and – if necessary – they can edit the proposals with intuitive hand gestures using a virtual painting tool. Richer semantic descriptions (e.g., material, shape, and color) can be optionally added via a voice-to-text tool and assigned to segments using a virtual *pin*. After capturing the scene, the semantic annotations and the 3D mesh are exported for downstream tasks.

The contributions of this paper can be summarized as:

- We develop a gesture-based HoloLens application for on-site and online user-in-the-loop 3D semantic scene reconstruction and labeling.
- We allow to easily customize semantic descriptions using the HoloLens voice-to-text feature and place description *pins* on the 3D mesh of the scene.
- We evaluate the efficiency and efficacy of the tool in a small-scale user study.

Related Work

Semantic Annotation Tools for 3D environments

There exists a variety of semantic annotation tools for 2D and 3D data, however a detailed review is outside the scope of this paper. Here, we focus on the following two aspects of 3D data annotation systems.

Automation & User-in-the-loop

Several works perform a first step of high-level segmentation to streamline the annotation process. Wong et al. (2014) developed the *SmartAnnotator*, a semi-automated pipeline for labeling RGBD images. A learning system proposes semantic labels on which human annotators in the loop provide feedback. The system can also improve its abilities incrementally through learning from previously labeled scenes as priors. Armeni et al. (2019) employ a pretrained 2D instance segmentation network to provide initial proposals on RGB images that are automatically aggregated on the underlying 3D mesh. Users are then asked to verify and edit the aggregation results on 2D images. Nguyen et al. (2021) begin with automatically detecting 3D bounding boxes in LiDAR point clouds and include a user-in-the-loop component to identify missing annotations with a single click. Others employ lower-level automation to accelerate annotations. Dai et al. (2017) propose such a system, where a web-interface is used for the manual annotation of 3D mesh models of indoor spaces. Specifically, it begins with an over-segmentation of the scene using a graph-cut based approach and users are then prompted to label these segments with the goal of 3D object instance segmentation. Nguyen et al. (2016) has a similar starting point; the resulting over-segments are further grouped into larger regions based on geometry and appearance cues. These regions are edited by users to get the final object semantic annotations. Russell & Torralba (2009) employ object segmentation masks and labels from 2D annotations to automatically recover the 3D scene geometry. We follow a similar setup of an automated low-level initial segmentation, followed by a user-in-the-loop approach for refinement and semantic annotation.

Virtual & Augmented Reality Systems

The above methods are restricted on a traditional 2D interaction setting. However, semantic annotation tools have also been developed as applications in virtual and augmented reality. Miksik et al. (2015) use a laser pointer as an AR paintbrush to allow user-defined outdoor scene segmentation. Saran et al. (2018) create an iOS application for simultaneous scanning and user-defined bounding box annotation. Ramirez et al. (2019) take a creative approach by gamifying scene labeling as a first-person shooting game. Spiekermann et al. (2019) focus on semantic annotation of text in 3D and place users in a virtual reality headset to perform annotation through simple gestures and textual descriptions. Zingsheim et al. (2021) present a collaborative labeling system in virtual reality, where remote users provide sparse annotations on the 3D mesh.

There is a gap concerning annotation tools that use market-available AR headsets such as the Microsoft HoloLens, which is increasingly used by developers and professionals. The advantage of an AR headset is that the annotator can interact with the 3D scene in a more intuitive and safe manner than through a tablet or phone. Also the collaboration of Trimble and HoloLens has created an AR hard-hat that can be used in construction sites (Trimble 2022), hence increasing the applicability of such a tool in AEC industry for existing and under construction buildings.

Automatic 3D Segmentation Methods

Automatic 3D segmentation can be categorized in *semantic* segmentation, which groups a scene into segments that belong to the same semantic class (e.g., chair or window), and *geometric* segmentation, which segments the scene into segments with similar geometric properties. Many algorithms perform 3D scene segmentation by processing RGB-D images (Michieli et al. 2019, Guo & Hoiem 2013, Gupta et al. 2013). Other approaches directly segment 3D visual data, the majority of which operate on 3D point clouds (Poux & Billen 2019, Qi et al. 2017). Bassier et al. (2020) show that performing segmentation on mesh-structure and point clouds achieves comparable results. Despite great progress in this field, the results are not accurate enough for the AEC industry and the methods are inherently rigid since they learn to identify a predefined set of semantic classes. In this work we choose to perform geometric segmentation, due to the flexibility that offers to users to adjust the segmentation proposals and assign their own semantic labels.

HoloLabel

Overview

HoloLabel leverages the advantages of AR devices, and specifically of HoloLens, to perform 3D scene annotation in the form of semantic segmentation. An overview of the annotation pipeline is shown in Figure 1: (a) the user enters the scene and begins reconstructing the 3D mesh (see *Scene Meshing*); at intervals, the user performs the semantic labeling process and begins by (b) triggering the segmentation algorithm, which automatically proposes geometrically-meaningful scene segments (see *Semantic Segmentation*); (c) the user has to then edit - if needed - and label these segments. There is the option to assign a semantic category to entire segments or individual mesh faces through virtual mesh painting (see *User-in-the-Loop Semantic Labeling*); (d) further, the user can also add rich semantic descriptions to segments using voice-to-text technology (see *User-based Descriptions*).

User Interface

The user interface is implemented using the HoloLens hand menu UX pattern (Microsoft 2022*a*). We chose this pattern since it gives the user the ability to quickly pull up and hide the menu from anywhere in the scene by raising and lowering their palm. The menu, shown in Figure 2, is split into three main components: *central menu*, *label menu*, and *dictation panel*. We list their features below and provide more details in subsequent sections.

Central Menu

The central menu is the core controller of the *HoloLabel* application and has the following functions:

Update Scene: Requests a scene mesh reconstruction update from the HoloLens (*Scene Meshing*).

Auto Segment: Sends the reconstructed scene mesh to a remote Python server for automatic segmentation into proposals (*Semantic Segmentation*).

Undo: Reverts recent history of the segmented mesh, including any virtual segment painting and *pins*.

Editing Modes:

Labeling Mode: Allows the creation and placement of description pins (*User-based Descriptions*).

Face Drawing Mode: Allows face-level painting (*User-in-the-Loop Semantic Labeling*).

Segment Drawing Mode: Allows segment-level painting (*User-in-the-Loop Semantic Labeling*).

Label *(in labeling mode)*: Creates description pins using one of three options (*User-based Descriptions*).

Toggle Label Menu: Hides/Shows the label menu.

Toggle Mesh: Hides/Shows the wireframe mesh.

✓ **Export Labels:** Exports the final annotated 3D mesh to the remote Python server (*Export*).

IP: Edits the IP address of the remote Python server.

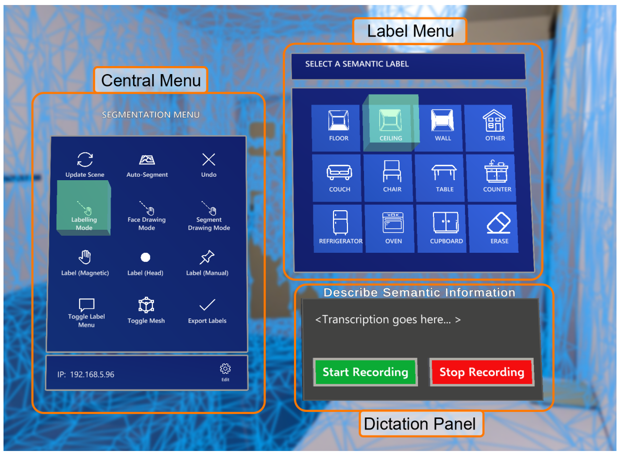


Figure 2: The user interface is split into three main components: the central menu that controls the HoloLabel application, the label menu that allows to select and change classes while in editing mode, and the dictation panel that can assign rich descriptions.

Label Menu

This menu contains a list of predefined semantic object categories supported by *HoloLabel*. Although the user has the option to define on the spot new categories, an existing library can facilitate and speed-up the process. We focus on categories like *wall*, *floor*, and *ceiling*, as well as major furniture categories (e.g., *table*, *chair*, and *couch*). When the annotation process starts, the user enters *editing* mode and selects an object category from the menu. Depending on whether the *segment-* or *face-* drawing mode is enabled, the specified object category will be assigned to the user-pinpointed segment or face. Each object category is associated with a unique RGB color which is visualized on the mesh when a segment or face receives this label. In this way the user can keep track of which parts of the scene have been annotated and which require further processing.

Dictation Panel

The dictation panel is used in *labeling* mode and allows the user to record and attach rich descriptions to *pins*. It can also be used to attach a semantic label not included in the predefined label menu.

Scene Meshing

The HoloLens automatically reconstructs a 3D surface mesh of a scene through *Spatial Mapping* (Microsoft 2022*b*), hereby referred to as *HoloLens mesh*. This surface mesh is updated dynamically, i.e., not only are new parts added but existing parts of the mesh can change over time. For our setting this is highly impractical, since annotated parts of the mesh should not change anymore. For this reason, we store on the device an independent 3D mesh representation of the reconstruction (*scene mesh*) that builds on top of the *Spatial Mapping* output. When reconstructing a scene, we actively display the *HoloLens mesh* using a wireframe shader to provide the user with visual feedback on the reconstruction progress (Figure 3b). Once a user has reconstructed a room or a portion of it and is ready to annotate it, the *scene mesh* is updated based on the current *HoloLens mesh*. This is triggered by the *Update Scene* button in the menu. The *scene mesh* is rendered densely and will not change anymore, so that the user can safely annotate it (Figure 3c). The above process is simple for the first scene update since the *scene mesh* is still empty. However, after the second update, we need to decide which parts of the *HoloLens mesh* are scene additions, which we have to add to our scene mesh, or scene changes, which we have to ignore in favor of keeping the existing parts in the scene mesh unaltered. We perform this by checking for every face in the *HoloLens mesh* if it is already present in the *scene mesh* using ray casting. Details on this are shown in Algorithm 1.

Algorithm 1 Scene Mesh Update

```
input scene mesh: the current state of the scene
input HoloLens mesh: new surface scan
output updated scene mesh
for all faces $f \in$ HoloLens mesh do
    $n \leftarrow$ normal of $f$
    $v_1, v_2, v_3 \leftarrow$ vertices of $f$
    $m \leftarrow \frac{1}{3}(v_1 + v_2 + v_3)$
    $r \leftarrow Ray\{origin : m + \tau \cdot n, \quad direction : -n\}$
    if not ($r$ intersects scene mesh) then
        add $f, v_1, v_2, v_3$ to scene mesh
    end if
end for
```

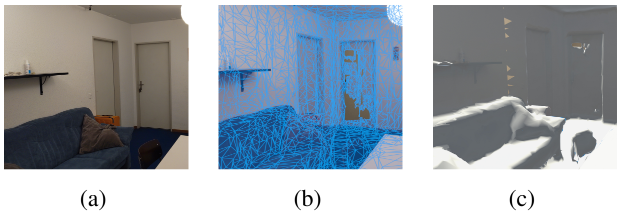


Figure 3: (a) Real world scene; (b) HoloLens mesh projected on the display; (c) Scene mesh stored independently.

Semantic Segmentation

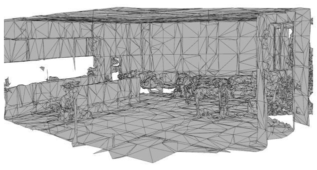
Following the scene update, the user can proceed with the annotation process. To speed up this stage, the application contains an automatic geometric segmentation option that will provide the user with segmentation proposals. Once the user requests segmentation proposals via the central menu, the *scene mesh* is sent to a remote Python server, where automatic geometric segmentation is performed. The user can also proceed without this step and directly start annotating the mesh by selecting with intuitive hand gestures (drawing stroke) mesh faces that belong to the selected semantic class. In the following, a *segment* refers to either one of the proposals generated by the automatic segmentation algorithm or to a single drawing stroke made by the user. Both kinds of segments are considered equivalent for all other computations.

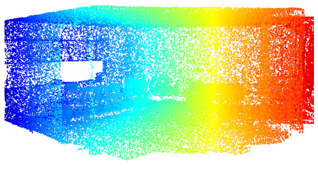
Automatic Generation of Segmentation Proposals

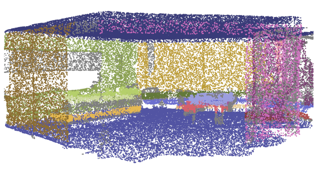
The automatic geometric segmentation pipeline is described in Figure 4. We first sample the mesh to obtain a point cloud representation, which yields more flexibility in the choice of segmentation algorithm. To perform geometric segmentation, we use a combination of RANSAC (Fischler & Bolles 1981) to identify planar segments and the euclidean clustering Density-Based Spatial Clustering (DBSCAN, (Kriegel et al. 2011)) to identify remaining segments of spatially contiguous groups of points. We make sure to track which faces of the mesh belong to which segment, to later convert the point cloud segmentation back to a mesh representation. The maximum number of segments is chosen dynamically depending on the mesh size following Equation (1), where the denominator constant is chosen through manual tuning. We err on the side of an over-segmentation rather than an under-segmentation regime, as it is easier for users to join together small segments than to divide larger ones. The computational complexity is $O(N \times O(RANSAC))$, where:

$$N = number\ of\ mesh\ vertices/800 \qquad (1)$$

After segmenting the sampled point cloud, we project the segmentation results back to the mesh. To identify the final segment to which a face belongs we use majority voting according to Equation (2), where s_i is the segment of face f_i, $p_j \in f_i$ means that point p_j has been sampled from face f_i and $s(p_j)$ is the segment assigned to point p_j. In other words, for each face we consider all the points sampled from it and assign the segment to which the majority of


(a) Input Mesh


(b) Sampled Point Cloud


(c) Automatic Segmentation Results

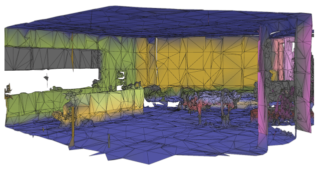

(d) Final Segmented Mesh

Figure 4: Segmentation Pipeline: Given an input mesh (a), we first sample a point cloud (b). This point cloud is segmented (c) as described in Automatic Generation of Segmentation Proposals and the results, after being consolidated back on the mesh, are returned on the HoloLens (d). Note that different colors in (c) and (d) denote different segments.

those points belong.

$$s_i = \arg\max_k \left(\sum_{j:p_j \in f_i} \mathbb{I}[s(p_j) = k] \right) \tag{2}$$

The segmented mesh is then returned to the HoloLens.

User-in-the-Loop Semantic Labeling

Once the segmentation proposals are received, the user can accept and/or adapt them using the *HoloLabel* virtual painting features. More concretely, if the proposal consists of mesh faces that belong to one semantic category, the user can directly assign that category using the segment-level painting function. If, however, the proposal erroneously contains faces from multiple categories, the user can use the face-level painting function to manually override and create new segments. The user can switch between the modes from the central menu (Figure 2).

Segment-level painting The automatic segmentation proposals are meant to facilitate and speed-up the drawing process. To do so, we automatically segment the scene such that each scene object can be composed of multiple segments. This means that the user may have to paint multiple segment proposals per object until the whole object is segmented. This however ensures that automatically generated segments are less likely to bleed into neighboring objects, thus minimizing the need for face-level painting. When the user enters the *segment drawing mode*, we use the MRTK hand ray pointer as a virtual paintbrush. In every frame we detect whether the user is currently pinching (i.e., the index finger and thumb are touching), which indicates that the user would like to paint at the current position. We then intersect the hand ray with the scene mesh, which gives us the mesh face at which the user is pointing. If this face belongs to a segment proposal, we assign the selected semantic category to all vertices of the segment. Otherwise, the pinch for this face gets ignored.

Face-level painting This feature consists of assigning semantic categories to individual faces. It can be used for manual segmentation due to the automatic segmentation proposals being either incorrect or not available. When the user enters *face drawing mode*, we follow a similar approach to above. However, instead of searching for the segment ID, we are overriding it (or setting it from scratch if non-existent). Segments are created with a single stroke, which begins when the user starts pinching and ends when the pinch is released.

User-based Descriptions

Since choosing from a list of labels might end up being restrictive to the multitude of semantic categories that exist, our tool provides the option to place virtual *pins* (Figure 5) on the 3D mesh. Using voice-to-text technology the user can provide rich semantic descriptions or a semantic label that is not available in the label menu. The user can choose from three methods to place the pin: *magnetic*, *head*, and *manual* (see third row of the central menu in Figure 2)[2]. Magnetic and head options use the MRTK's *Surface Magnetism Solver* (Microsoft 2022*e*), which performs a ray cast along the hand-ray or head-direction respectively, and places the pin at the intersection point with the *scene mesh*. The third, manual, option uses the *Object Manipulator Script* (Microsoft 2022*d*) and allows the pin to be selected, rotated, and moved with pinching motions. The user can attach a description to each placed pin. This is achieved using the dictation panel (shown on the bottom right of Figure 2). With a pin selected, the user can record a verbal description. The spoken words are translated to text using MRTK's off-the-shelf *dictation handler* (Microsoft 2022*c*). The text is stored along with the pin. Upon export, each pin and its corresponding description are mapped to the closest segment.

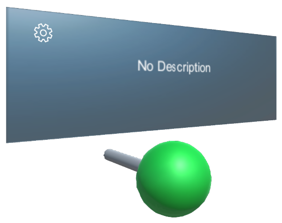


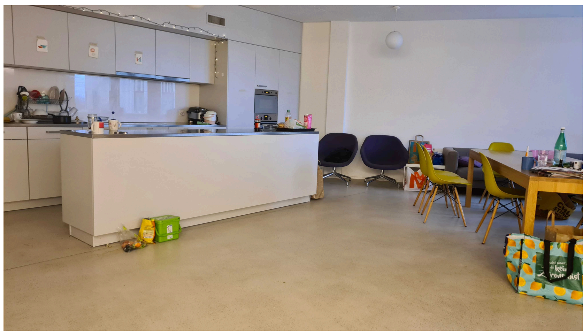
Figure 5: Example of an empty description pin

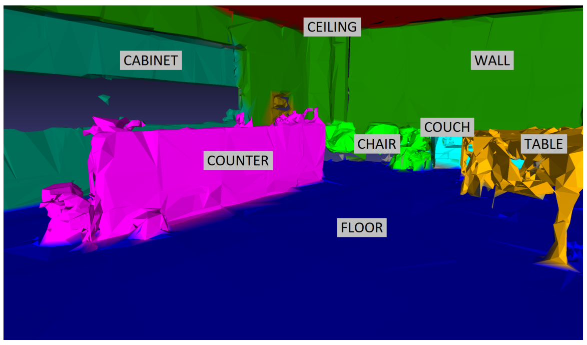
Export

The following files can be exported by sending the *scene mesh* and semantic annotations to the Python server:

- A wavefront file (.obj) of the *scene mesh* with per-vertex colors for easy visualization.
- A list of created pins, containing their 3D position and assigned description.

[2]We perform an evaluation on pin options in the user study.


(a) RGB image of the room.



(b) Ground truth mesh annotation with labels.

Figure 6: A view in the scene and the corresponding ground truth as obtained from the 3D mesh.

- A list of all *scene mesh* vertices, containing their 3D position, semantic class, and segment ID.
- A list of segments, containing segment ID, class, and assigned description.

To generate the last one, we need to map each description pin to a segment. This is done by mapping the 3D location of the pin to its closest vertex and subsequently identifying the segment to which this vertex belongs.

Experiments

In this section, we evaluate the efficiency of *HoloLabel's* interface design and the accuracy of the as-is semantic information in a limited user study. We recruited 4 non-expert users with limited or no experience with the HoloLens to try our tool in a standardized indoor scene. The environment is an open kitchen and living room, featuring objects such as a table, chairs, a couch, counters, cabinets, a refrigerator, and an oven. Prior to the study, we used our tool to generate an initial ground truth segmentation of the room. This initial ground truth was at-places noisy and needed offline post-processing. The post-processing took 15 minutes compared to the initial collection which took 12 minutes. Annotating the same scene in a fully manual way took 50 minutes (it does not include the time required for reconstruction) vs 27 minutes with the *HoloLabel* and refinement step (this includes the reconstruction time). The processed ground truth is shown in Figure 6.

The users are given an overview of the task and introduced to our tool via a video tutorial, which shows the key features of the interface and demonstrates an example of a simple room segmentation. The environment shown in the tutorial is different from the test environment, so the users are encouraged to generalize what they learned to different room geometries and furnishings. Once familiarized with the list of semantic categories, the users are instructed to segment the semantically distinct structures of the room to the best of their abilities, keeping in mind to be precise with segmentation boundaries and semantic classes.

We record the total annotation time per trial (Table 1). The average time is nearly 14 minutes, which is slightly above the time it took our team to create the initial ground truth.

Table 1: Annotation time, SUS score, and level of experience with the HoloLens.

Participant	Time	SUS Score	Experience
# 1	19:36	50	None
# 2	7:22	63.89	Limited
# 3	12:05	44.44	Limited
# 4	16:34	86.11	None
Average	13:54	61.11	-

Participants #2 and #3 – who self-reported having previous experience using the HoloLens – were able to complete the task faster. This is corroborated with our qualitative observation that they were able to adapt to the gesture controls faster and operated more independently compared to participants #1 and #4. We note the following limitations that hindered the progress of most or all participants:

- Sensitivity of the pinch detection was too high. Segments were unintentionally painted wrongly while in *Segment Drawing Mode* which took extra time to fix.
- The colored mesh segments sometimes did not render well through the HoloLens display, so participants could not see the color/label of the segments.
- Participants mistook the underlying blue wireframe *HoloLens mesh* as the mesh generated by the automatic segmentation algorithm, and thus wasted time trying to annotate it.

The exported mesh, segments, and labels were analyzed offline using standard 3D segmentation metrics: Overall Accuracy (OAcc) and Intersection over Union (IoU) (He et al. 2021). OAcc represents the number of correctly classified samples, computed at both the class level and scene level. IoU is a semantic segmentation metric that computes the ratio of true positive samples over the union of the predicted positives and all positive samples. We first consider that a sample should be a vertex in the mesh, but soon discovered that this disproportionately weighs the areas with higher detail more, as smooth flat surfaces are comprised of fewer vertices than edges and corners. Thus for each vertex, we compute the *area-weighted* vertex value, which is the average area of the surrounding faces that the vertex

Table 2: OAcc and IoU metrics for study participants and an expert user.

	OAcc					***IoU***				
Participant	**#1**	**#2**	**#3**	**#4**	***Expert***	**#1**	**#2**	**#3**	**#4**	***Expert***
FLOOR	15.61	84.33	88.12	86.78	*94.16*	13.76	73.16	73.42	61.97	*86.67*
CEILING	0.00	97.55	87.56	88.21	*98.17*	0.00	76.58	72.44	78.95	*95.38*
WALL	15.13	25.67	31.01	18.55	*49.00*	13.81	24.43	26.97	16.74	*47.83*
COUCH	44.31	1.12	30.57	76.25	*83.19*	32.96	1.10	29.67	35.89	*52.12*
CHAIR	0.00	0.00	27.02	0.00	*24.12*	0.00	0.00	8.12	0.00	*23.57*
TABLE	51.40	13.66	46.12	26.91	*62.00*	3.59	11.74	24.10	25.11	*45.79*
COUNTER	28.60	0.00	0.00	0.04	*78.15*	12.28	0.00	0.00	0.04	*64.44*
REFRIGERATOR	0.00	61.72	40.21	98.35	*91.26*	0.00	36.49	21.13	52.86	*85.63*
OVEN	0.00	20.08	0.00	1.06	*50.50*	0.00	10.12	0.00	0.32	*29.96*
CABINET	0.00	0.00	33.44	16.86	*50.91*	0.00	0.00	25.04	14.79	*44.93*
UNLABLLED	46.14	68.60	34.07	45.73	*98.88*	6.06	5.72	5.07	5.78	*10.79*
Avg over # classes	16.77	31.06	34.84	38.23	*65.03*	6.87	19.94	23.83	24.37	*48.93*
Avg over # vertices	13.50	51.54	55.94	49.83	*73.59*	-	-	-	-	-

belongs to. As a result, vertices that compose larger faces will have more weight in the computation.

Since the HoloLens coordinate system can drift significantly between trials in the world coordinate system, we first register the participant annotated meshes to the ground truth by manually aligning them to the ground truth mesh and applying Iterative Closest Point (ICP, (Besl & McKay 1992)) for finetuning the pose. Due to the dynamic nature of the mesh reconstruction in HoloLens, each user's scanned mesh can be slightly different from ground truth. We attempt to eliminate most of the discrepancy by deleting any structures in the user meshes that were not captured in the ground truth. Then for each vertex in the user meshes, we search for the closest corresponding labeled ground truth vertex using a k-d tree. Based on the vertex-to-vertex mapping, we compute the evaluation metrics per semantic class, as well as compute the mean value. For OAcc, we also calculate the percentage of correctly labeled vertices overall, to account for imbalanced classes. The results of the participants, as well as of an expert user (*oracle*), are shown in Table 2.

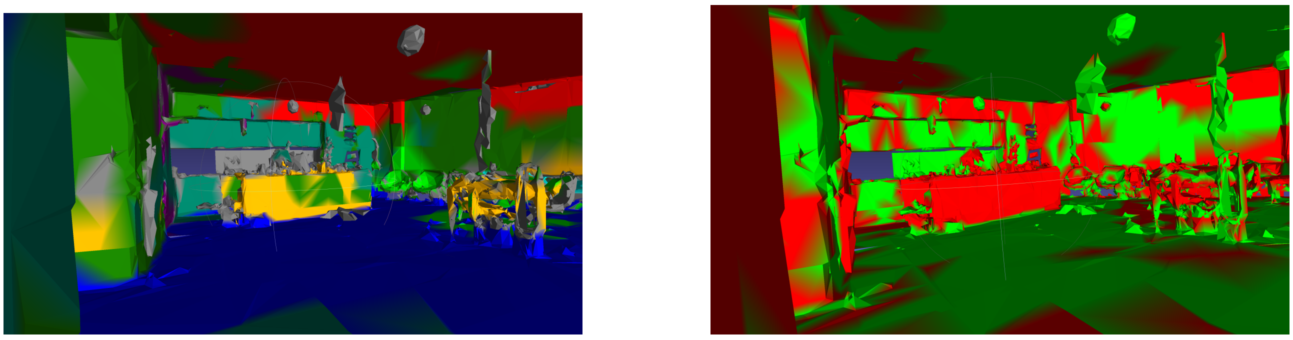
The participant OAcc and IoU scores present low values, but we identified the following two causes that are not related to their annotation skill: HoloLens color rendering issues and inconsistently defined segment boundaries. The latter occurs if the auto segmentation algorithm performs inconsistent between trials, or if the user has a tendency to over-segment or under-segment objects. One example of this occurrence is the boundary between the floor and the wall; in some trials it is slightly off the ground instead of being at the exact edge. While this mistake is not obvious to the naked eye, it has a significant effect on accuracy metrics. The remaining painting mistakes can be attributed to fine-motor control difficulties due to lack of training and practice, confusion between semantic classes (i.e. mistaking the counter for a table), time limitations, and general carelessness where the user was not motivated to segment everything in the scene.

We can see in Figure 7, showing the best overall user segmentation from participant #3, that the larger segments of *floor* and *ceiling* are mostly correct, corresponding to OAcc of 88.12% and 87.56% and IoU of 73.16% and 72.42%, respectively. Some classes like *counter* and *oven* were not attempted to be labelled at all, which leads to a drop in the class-averaged OAcc. Overall, the proportion of areas labelled correctly range from 13.50% to 55.94% across all four participants. We strongly believe that this can be improved if the participants are given more training with *HoloLabel*, as well as more time and encouragement to make a detailed and accurate segmentation.

As a comparison, we also include the results from an expert user to show what a realistic upperbound on accuracy might be. It is noticeable that while the results are significantly better, the overall OAcc over all vertices is only 73.59%. More testing would be needed to fully verify the reason for the discrepancy, but it is likely that the main contributor is the natural variation between trials. It is not possible to yield identical results from two trials due to the stochasticity of the HoloLens generated mesh and the segmentation algorithm, as well as natural variations in the manual segmentation. For this reason, the numerical metrics can be regarded as over-sensitive to *HoloLabel*.

After concluding the annotation, the users are asked to fill out a 10-question survey [3] from the System Usability Scale (SUS, (Brooke et al. 1996)) to gauge the usability of the interface. The SUS score is an accepted and reliable tool for evaluating usability, including factors like system integration, ease of usage, and learning curve. The

[3]https://forms.gle/4HawBhuXQdRssSVh8


(a) Sample of a participant-annotated mesh.

(b) Correct (*green*) and incorrect (*red*) labels from a sample participant's mesh.

Figure 7: Sample of participant segmented mesh and its errors with respect to the ground truth.

maximum achievable score is 100 and the typical average score is 68. Other VR systems (of purposes unrelated to scene annotation) scored 72.5 (Butt et al. 2018*a*), 71.6 (Yépez et al. 2020), and 70 (Butt et al. 2018*b*). Our SUS scores are in Table 1, showing an average of 61.11. This is lower than average, but it can be largely attributed to the participant's inexperience with the HoloLens. Optimistically, we received high scores in the areas of *confidence while using tool*, *system integration*, and *system consistency*. The lowest scores came from *ease of usage*, *ability to independently operate*, and *quickly learnable by others*, which indicates that the problem is partially shared by non-familiarity with the HoloLens platform.

As a bonus experiment, we also asked participant #2, who demonstrated good fluency with the system, to evaluate the three physical labelling methods — hand magnetism, head magnetism, and manual placement. They were asked to place each of the labels to a close-distance target (within 0.5 meter) and a far-distance target (over 2 meters away), then rate the preferred method for each task. For both tasks, the participant ranked manual placement first, followed by head magnetism and then hand magnetism.

Discussion

Limitations: While *HoloLabel* greatly simplifies the process of performing scene annotations, there is still room for improvement. Currently the tool focuses on semantic segmentation, however, an instance segmentation focus could prove more practical for several AEC tasks. Furthermore, the choice of the automatic segmentation algorithm was based on performance and simplicity concerns. The result could be improved, e.g., with the use of learning-based algorithms such as PointNet (Qi et al. 2017). However, direct testing on our data did not show improved performance, which can be attributed to the different data statistics. To leverage the advantages of such algorithms would probably require retraining on a dataset that is specific to our case or employing domain adaptation techniques. In addition, the HoloLens surface reconstruction suffers from artifacts such as holes, floating hallucinations, or rough surfaces. At the scene mesh update step, standard mesh processing could be applied to reduce those artifacts before the mesh is displayed to the user. This could lead to an improved user experience and a better final result.

With respect to the user experience, we found that the pinching motion was often triggered by accident. It requires further investigation to determine if more robust pinching detection would be possible or if another motion would be more suitable for *HoloLabel*. Generally, we also found that the tool does require some experience with the HoloLens, and we therefore suggest to have the user complete a HoloLens training before using the tool. In terms of user friendliness, investigating adjustments of the paintbrush to perform the face/segment painting could decrease the time needed to perform the painting tasks. Further, some users reported that the segment colors were not clearly visible from certain angles. This also requires further investigation and stability improvements.

Future Impact: Given the rapid advancement of Mixed Reality technologies, applications such as *HoloLabel* will be part of the daily activities of AEC professionals. *HoloLabel* can be used by architects and surveyors to acquire the as-is status of an existing space, as well as by construction project managers to capture the state of a construction site. It has the capability to decrease communication time between the site and the office – e.g., the pin functionality can be also used to attach notes and other information on the mesh, for documentation or communication purposes.

Conclusion

In this paper, we introduce as semi-automatic tool to allow generating 3D as-is semantic information of a scene in an immersive and intuitive manner using the Microsoft HoloLens. In particular, we combine the steps of scanning a room and segmenting it, which are usually separated. This allows users to annotate a room in a single pass, which eliminates the need for the offline annotation step required by traditional methods. We demonstrate that the tool is convenient to use by non-expert on the task users with little-to-no HoloLens experience. However, our user study suggests that at least an introductory training should be completed before starting to use the tool. While the results look promising, there is room for improvement with possible improvements present in the *Discussion* section.

References

Administration, U. G. S. (2022), ‘https://www.gsa.gov/bim3D-4D Building Information Modeling’. Accessed: 2022-01-26.

Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J. & Savarese, S. (2019), 3d scene graph: A structure for unified semantics, 3d space, and camera, *in* ‘Proceedings of the IEEE/CVF International Conference on Computer Vision’, pp. 5664–5673.

Bassier, M., Vergauwen, M. & Poux, F. (2020), ‘Point cloud vs. mesh features for building interior classification’, *Remote Sensing* **12**(14).
URL: *https://www.mdpi.com/2072-4292/12/14/2224*

Besl, P. J. & McKay, N. D. (1992), Method for registration of 3-d shapes, *in* ‘Sensor fusion IV: control paradigms and data structures’, Vol. 1611, International Society for Optics and Photonics, pp. 586–606.

Blackwell, B. (2012), ‘https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/34710/12-1327 − *building* − *information* − *modelling.pdfBuildingInformationModelling′.Accessed* : 2022 − 01 − 26.

Brilakis, I., Lourakis, M., Sacks, R., Savarese, S., Christodoulou, S., Teizer, J. & Makhmalbaf, A. (2010), ‘Toward automated generation of parametric bims based on hybrid video and laser scanning data’, *Advanced Engineering Informatics* **24**(4), 456–465.

Brooke, J. et al. (1996), ‘Sus-a quick and dirty usability scale’, *Usability evaluation in industry* **189**(194), 4–7.

Butt, A. L., Kardong-Edgren, S. & Ellertson, A. (2018*a*), ‘Using game-based virtual reality with haptics for skill acquisition’, *Clinical Simulation in Nursing* **16**, 25–32.

Butt, A. L., Kardong-Edgren, S. & Ellertson, A. (2018*b*), ‘Using game-based virtual reality with haptics for skill acquisition’, *Clinical Simulation in Nursing* **16**, 25–32.

Choy, C., Gwak, J. & Savarese, S. (2019), 4d spatio-temporal convnets: Minkowski convolutional neural networks, *in* ‘Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition’, pp. 3075–3084.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. & Nießner, M. (2017), Scannet: Richly-annotated 3d reconstructions of indoor scenes, *in* ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 5828–5839.

Fischler, M. A. & Bolles, R. C. (1981), ‘Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography’, *Communications of the ACM* **24**(6), 381–395.

Guo, R. & Hoiem, D. (2013), Support surface prediction in indoor scenes, *in* ‘Proceedings - 2013 IEEE International Conference on Computer Vision, ICCV 2013’, Proceedings of the IEEE International Conference on Computer Vision, Institute of Electrical and Electronics Engineers Inc., United States, pp. 2144–2151. 2013 14th IEEE International Conference on Computer Vision, ICCV 2013 ; Conference date: 01-12-2013 Through 08-12-2013.

Gupta, S., Arbelaez, P. & Malik, J. (2013), Perceptual organization and recognition of indoor scenes from rgb-d images, *in* ‘Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on’.

He, Y., Yu, H., Liu, X., Yang, Z., Sun, W., Wang, Y., Fu, Q., Zou, Y. & Mian, A. (2021), ‘Deep learning based 3d segmentation: A survey’, *CoRR* **abs/2103.05423**.
URL: *https://arxiv.org/abs/2103.05423*

Jung, J., Hong, S., Jeong, S., Kim, S., Cho, H., Hong, S. & Heo, J. (2014), ‘Productive modeling for development of as-built models of existing indoor structures’, *Automation in Construction* **42**, 68–77.

Kriegel, H.-P., Kröger, P., Sander, J. & Zimek, A. (2011), ‘Density-based clustering’, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1**(3), 231–240.

Michieli, U., Camporese, M., Agiollo, A., Pagnutti, G. & Zanuttigh, P. (2019), Region merging driven by deep learning for rgb-d segmentation and labeling, *in* ‘Proceedings of the 13th International Conference on Distributed Smart Cameras’, ICDSC 2019, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/3349801.3349810*

Microsoft (2022*a*), ‘https://docs.microsoft.com/en-us/windows/mixed-reality/design/hand-menuMixed-Reality: Hand Menu’. Accessed: 2022-01-26.

Microsoft (2022*b*), ‘https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mappingMixed-Reality: Spatial Mapping’. Accessed: 2022-01-26.

Microsoft (2022*c*), ‘https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/input/dictation?view=mrtkunity-2021-05MRTK: Dictation’. Accessed: 2022-01-26.

Microsoft (2022*d*), ‘https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/ux-building-blocks/object-manipulator?view=mrtkunity-2021-05MRTK: Object manipulator’. Accessed: 2022-01-26.

Microsoft (2022*e*), 'https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/ux-building-blocks/solvers/solver?view=mrtkunity-2021-05surfacemagnetismMRTK: Solver overview, Surface Magnetism'. Accessed: 2022-01-26.

Microsoft (2022*f*), 'https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05Mixed Reality Toolkit'. Accessed: 2022-01-26.

Microsoft (2022*g*), 'https://www.microsoft.com/en-us/hololensMicrosoft HoloLens 2'. Accessed: 2022-01-26.

Miksik, O., Vineet, V., Lidegaard, M., Prasaath, R., Niessner, M., Golodetz, S., Hicks, S. L., Pérez, P., Izadi, S. & Torr, P. H. (2015), The semantic paintbrush: Interactive 3d mapping and recognition in large outdoor spaces, *in* 'Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems', CHI '15, Association for Computing Machinery, New York, NY, USA, p. 33173326.
URL: *https://doi.org/10.1145/2702123.2702222*

Nguyen, D. T., Hua, B.-S., Yu, L.-F. & Yeung, S.-K. (2016), 'A robust 3d-2d interactive tool for scene segmentation and annotation'.

Nguyen, T., Hua, B.-S., Nguyen, D. T. & Phung, D. (2021), 'Single-click 3d object annotation on lidar point clouds'.

Poux, F. & Billen, R. (2019), 'Voxel-based 3d point cloud semantic segmentation: Unsupervised geometric and relationship featuring vs deep learning methods', *ISPRS International Journal of Geo-Information* **8**(5).
URL: *https://www.mdpi.com/2220-9964/8/5/213*

Qi, C. R., Su, H., Mo, K. & Guibas, L. J. (2017), 'Pointnet: Deep learning on point sets for 3d classification and segmentation'.

Ramirez, P. Z., Paternesi, C., De Gregorio, D. & Di Stefano, L. (2019), Shooting labels by virtual reality.

Russell, B. C. & Torralba, A. (2009), Building a database of 3d scenes from user annotations, *in* '2009 IEEE Conference on Computer Vision and Pattern Recognition', IEEE, pp. 2711–2718.

Saran, V., Lin, J. & Zakhor, A. (2018), Augmented annotations: Indoor dataset generation with augmented reality.

Spiekermann, C., Abrami, G. & Mehler, A. (2019), Vannotator: a gesture-driven annotation framework for linguistic and multimodal annotation.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J. & Savarese, S. (2017), Segcloud: Semantic segmentation of 3d point clouds, *in* '2017 international conference on 3D vision (3DV)', IEEE, pp. 537–547.

Trimble (2022), 'https://fieldtech.trimble.com/en/products/mixed-reality/trimble-xr10-with-hololens-2Trimble XR10 with HoloLens 2'. Accessed: 2022-01-26.

Wong, Y.-S., Chu, H.-K. & Mitra, N. (2014), 'Smartannotator: An interactive tool for annotating rgbd indoor images', *Computer Graphics Forum* **34**.

Woo, J., Wilsmann, J. & Kang, D. (2010), Use of as-built building information modeling, *in* 'Construction Research Congress 2010: Innovation for Reshaping Construction Practice', pp. 538–548.

Yépez, J., Guevara, L. & Guerrero, G. (2020), Aulavr: Virtual reality, a telepresence technique applied to distance education., *in* '2020 15th Iberian Conference on Information Systems and Technologies (CISTI)', pp. 1–5.

Zingsheim, D., Stotko, P., Krumpen, S., Weinmann, M. & Klein, R. (2021), 'Collaborative vr-based 3d labeling of live-captured scenes by remote users', *IEEE Computer Graphics and Applications* .