HoloLabel: Mixed Reality Scene Annotation Tool on the HoloLens

Véronique Kaufmann
vkaufman@student.ethz.ch

Yi Fei (Jessica) Bo
yiboyi@student.ethz.ch

Dhruv Agrawal
dagrawal@student.ethz.ch

Janik Lobsiger
ljanik@student.ethz.ch

Abstract

As the popularity of supervised and semi-supervised learning tasks in embodied perception rises, so does the demand for large-scale annotated 3D scene datasets. Devices equipped with RGB cameras and depth sensors being readily available simplifies the task of capturing and reconstructing an environment as a 3D mesh. However, the task of manually labelling this 3D data is still non-trivial. To obtain semantic information about the scene, experts often have to work for many hours using non-intuitive and error-prone tools. To address this problem, we propose a HoloLens application to allow a user to directly annotate the scene with semantic information while simultaneously capturing the spatial environment mesh. Our tool focuses on 3D mesh semantic segmentation, i.e. each face of the 3D mesh should be annotated with a predefined set of semantic classes, such that the output is a segmented 3D mesh. We leverage the HoloLens' Spatial Mapping feature to generate a 3D mesh of the scene and apply an automatic segmentation algorithm to generate segmentation proposals. The user can use a virtual paintbrush to refine the segments or create new ones. Finally, we allow the option to add richer semantic descriptions to segments using voice-to-text technology. We aim to lay the groundwork to leverage upcoming mixed reality devices for intuitive 3D scene annotation directly in the real world.

1. Introduction

Large-scale 3D scene datasets with good semantic annotation are increasingly needed for training models that solve tasks in embodied perception, robotics, and scene understanding. Examples of semantically-annotated scene datasets include [27], [1], [6], where the meshes and sometimes RGB images of rooms are captured alongside the semantic label for prominent structures in the room, such as *wall*, *bed*, *lamp*, and so forth.

Traditional 3D scene annotation is a laborious process performed primarily by trained experts using a 2D screen and can take several hours for a single room [19], which is non-trivial in terms of labour time and costs. The labelling process is also disjoint and offline from the data capturing step, introducing inefficiencies in the dataset generation pipeline. A more intuitive approach is to localize the annotator in the room itself, allowing them to simultaneously capture, segment, and label the environment in a natural and interactive method. The data, once exported from the live annotation session, would require only minimal additional processing.

Augmented reality technology, such as the Microsoft HoloLens, provides a technical platform to implement a live 3D scene annotation tool. The HoloLens comes with the built-in functionality for capturing and reconstructing spatial environment meshes. Additional libraries, such as the Mixed Reality Toolkit (MRTK), offer capabilities to develop interactive augmented user interfaces. In this work, we introduce the HoloLabel, an AR scene annotation tool with which a non-expert user can directly annotate the semantic information of a scene while simultaneously capturing its spatial information. The mesh of the room is geometrically segmented in the background on a remote Python server. Once complete, the user can modify the segmentation proposals by virtual painting using hand gestures and assign unlabelled segments with semantic labels. Richer semantic descriptions (such as color and texture) 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 mesh are exported for downstream tasks.

We identify our main contributions as follows:

- We create a gesture based HoloLens app for online human-in-the-loop 3D semantic scene annotation.
- We introduce the ability to easily customize semantic descriptions using the HoloLen's voice-to-text feature and place description pins into the environment.
- We evaluate the efficiency and efficacy of the tool in a small-scale user study.

In this report, Section 2 starts with an overview of the

related works. Section 3 describes the HoloLabel tool's features and gives some implementation details. Finally Sections 4 and 5 assess the functionalities of the tool and provide results and limitations.

2. Related Work

Annotation Tools We reviewed several existing works aimed to improve the ease of annotation of 3D environments. The thread of commonality between these works is that the low-level computation is done in the background, thus presenting an intuitive user interface that streamlines the annotation process. Different works also choose different approaches for the interactivity element. Wong et al. [26] developed the SmartAnnotator, a semi-automated system for labelling RGBD images with human annotators in the loop to provide feedback on the proposed semantic labels. The system can also improve incrementally through learning from previously labelled scenes as priors. Similarly, Nguyen et al.'s [19] improved RGBD segmentation model allows annotators to interactively refine and annotate the scene. Repeating objects can be template-matched through searching the scene, which speeds up annotation. Miksik et al. [18] uses a laser pointer as an AR "paintbrush" to allow user-defined scene segmentation. Saran et al. [24] created a simultaneous scanning + bounding box annotation system, but using depth sensors in an iPad. Ramirez et al. [23] take a creative approach by gamifying scene labelling as a first-person shooting game. Spiekermann et al. [25] put the user in a VR headset and allow them to perform annotation through simple gestures and textual descriptions.

We identify that none of the existing works use augmented reality tools such as the Microsoft HoloLens, which is increasingly used by developers and professionals alike. The advantage of an AR headset is that the annotator can interact with the 3D environment around them as normally as possible. The HoloLens also has built-in gesture recognition, which allows us to implement a feature like the virtual paintbrush, but without requiring additional hardware.

Segmentation Methods Segmentation can be performed in two ways: semantic segmentation, which segments a scene into semantic classes; or geometric segmentation, which segments the scene into geometric objects without semantic meaning. Many algorithms perform scene segmentation using RGD-D images [17], [14], [15]. Other approaches directly segment three-dimensional representations of the scene, most of which focus on the segmentation of point clouds [21], [22]. Bassier et al. [2] show that performing segmentation on mesh-structure and point clouds achieves comparable results. In this work we choose to perform geometric segmentation automatically on point clouds using a RANSAC-based algorithm [20] and allow the user to adjust the segmentation proposals and assign semantic labels.

3. Method

Our tool leverages the advantages of AR devices to perform 3D scene annotation in the form of semantic segmentation. In this section we describe the features of the tool, as well as the user workflow. In Section 3.1 we describe the user interface. Then, we go through the process of scene annotation from the perspective of a user (see Figure 1). The first step consists of the user entering the room and the mesh being scanned (Section 3.2). Then, the user calls the segmentation algorithm, which automatically proposes scene segments (Section 3.3.1). Afterwards the user has to process and label these segments. They have the option to give whole segments or individual faces a semantic category through virtual mesh painting (Section 3.3.2). Further, the user can also add descriptions to segments (Section 3.4).

3.1. User Interface

Our main user interface is implemented using the HoloLens hand menu UX pattern [9]. We chose this pattern since it gives the user the ability to quickly pull up and hide the menu from anywhere in the room 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 in this section, subsequent sections explain each feature in more detail.

Central Menu The central menu is the core controller of the whole application. We describe its functions as follows:

- Update Scene: request a scene mesh update from the HoloLens (see 3.2)
- Auto Segment: send the scene mesh to a Python server for auto segmentation (3.3.1)
- Undo: reverts the recent history of the segmented mesh, including segment painting and description pins
- Editing Modes:
  - Labelling Mode: create and place description pins (3.4)
  - Face Drawing Mode: face-level painting (3.3.2)
  - Segment Drawing Mode: segment-level painting (3.3.2)
- Label (*option*): create description pins using one of the the three control options (3.4)
- Toggle Label Menu: hide or show the label menu and dictation panel
- Toggle Mesh: hide or show the HoloLens' wireframe mesh

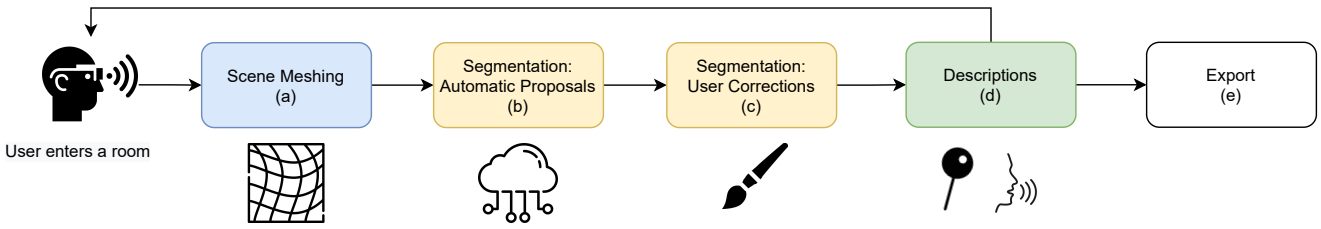


Figure 1. User Workflow: When the user enters a new room, we first reconstruct the 3D scene (a). Next, the scene mesh is sent to a Python server for automatic segmentation (b). The user then processes these segmentation proposals using a virtual brush (c). Finally, the user can add richer semantic descriptions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

✓ Export Labels: export the final annotated mesh to the Python server (3.4.1)
⚙ IP: edit the IP address of the Python server

Label Menu This menu contains a list of semantic classes supported by our tool. We focus on room features like 'wall', 'floor', and 'ceiling', as well as major furniture categories like 'table', 'chair', 'couch', and so forth. There is an 'erase' option to get rid of existing labels on segments and an 'other' option to cover semantic classes not included in the label menu. When the user is in segment- or face-drawing mode, they can use this menu to change the drawing color, which corresponds uniquely to a semantic class.

Dictation Panel The dictation panel is used in labelling mode and allows the user to record and attach rich descriptions to pins, described in section 3.4. This can also be used to attach a semantic label not included in predefined the label menu to a segment.

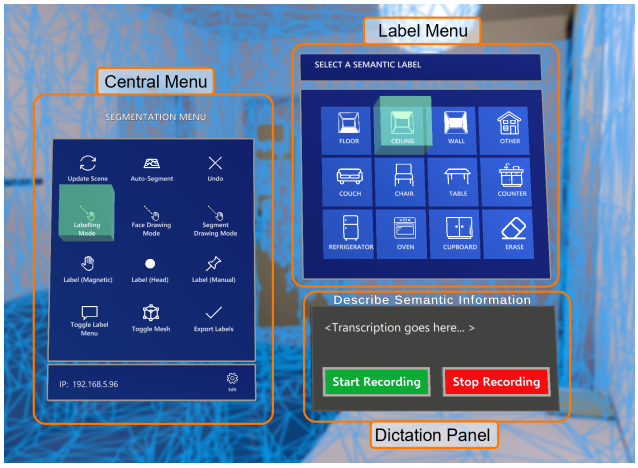


Figure 2. GUI: The user interface is split into three main components: the *central menu* to control the whole application, the *label menu* to change classes while drawing, and the *dictation panel* used to record rich descriptions of segments

3.2. Scene Meshing

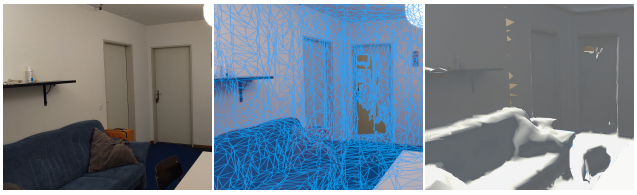
The HoloLens automatically reconstructs a surface mesh of the scene through *Spatial Mapping* [10], which we call the *HoloLens mesh*. This surface mesh is updated dynamically, i.e., not only are new parts added, but existing parts of the mesh can change over time. For our setting this is highly impractical, since annotated parts of the mesh should not change anymore. For this reason we add our own scene meshing layer, which builds on top of Spatial Mapping. We maintain our own Unity mesh of the surface reconstruction, which we call the *scene mesh*. When scanning a room, we display the HoloLens mesh using a wireframe shader to give the user some visual feedback on the scanning progress (see Figure 3 b)). Once a user has scanned a room, they can select the *Update Scene* button, upon which the vertices and faces from the HoloLens mesh are added to our scene mesh. This mesh is rendered densely and will not change anymore, so that the user can safely annotate it (see Figure 3 c)). The above process is simple for the first scene update since our 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 the already existing parts of the scene mesh. This is handled by Algorithm 1. Simply put, for every face in the HoloLens mesh, we check if a face is already present in the scene mesh using ray casting.

3.3. Segmentation

Once the user requests segmentation proposals, the scene mesh is sent to an external Python server, where automatic geometric segmentation is performed.

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



(a) (b) (c)

Figure 3. (a) Real world scene (b) HoloLens mesh (c) Scene mesh

3.3.1 Segmentation and Automatic 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 do so, we use *trimesh* [8], which samples a pointcloud from a mesh by sampling multiple points per triangle, and stores the triangle-point mapping. To perform geometric segmentation, we use multi-order RANSAC and unsupervised DBSCAN, where we closely follow [20] and adapt the algorithm for our use case. In particular, we add functionality 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 according to Equation 1, where the denominator constant is chosen through manual tuning. We err on the side of over-segmentation rather than under-segmentation, as it is easier for users to join together small segments than to divide larger segments.

$$N = number\ of\ vertices/800 \tag{1}$$

After segmenting the point cloud, we project the segmentation back to our input mesh using 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 of the majority of those points to the face.

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

Finally we return the segmented mesh to the HoloLens in the form of a vector containing the corresponding segment ID for each vertex. On the HoloLens, the user gets the proposals and can either accept or adapt them. The next section explains how this is done.

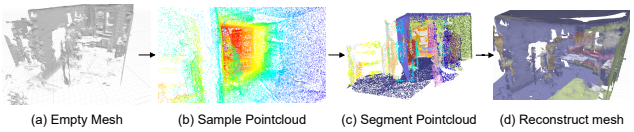


Figure 4. Segmentation Pipeline. We first sample a point cloud (b) from the input mesh (a). This point cloud is then segmented (c) as described in Section 3.3 and finally we map the segmentation back to the mesh structure to return to the HoloLens (d).

3.3.2 User-in-the-Loop Corrections

Once the user gets the segmentation proposals, they have the option to accept and/or adapt them using our virtual painting features. More concretely, the user can assign each segment with a semantic category by drawing over them with our semantic label colors, using the segment-level painting function. Alternatively if they wish to redefine the boundary of segments or create new ones, the user can use our face-level painting function to manually create new segments. The user can switch between these modes from the central menu (see Figure 2).

Face-level painting This feature consists of painting individual faces with color (semantic categories). This can be used for manual segmentation due to the automatic segments being either insufficient or unavailable and manual adjustment is required. When the user enters *face 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 draw at the current position. We then intersect the hand ray with the scene mesh, which gives us the mesh face that the user is pointing at. We do coloring on a vertex-level, i.e. all three vertices of the triangle are colored with the currently selected class. This enables us to use MRTK's vertex color shader to display the segmentation. Additionally, we store a segment ID for each vertex. This allows us to let the user use segment-level painting (see below) also on these manually created segments. Segments can also be created by a

single stroke, which is begins where the user starts pinching and ends when the pinch is released.

Segment-level painting The automatic segmentation proposals are meant to facilitate and speed-up the drawing process. To do so, we auto-segment the scene such that each scene object can be composed of multiple segments. (3.3.1). When the user is in *segment drawing mode*, we follow a similar approach as described in face-level painting, however when we intersect a face with the hand ray, we first look up the segment the face belongs to, and then color all vertices of that segment with the selected class. The automatic segmentation algorithm over-segments the scene by construction, i.e. each object is divided into multiple segments. This means that the user may have to color 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 less face-level painting is needed.

3.4. Descriptions

Since choosing from a list of labels might be too restrictive, our tool provides the option to place pins into the scene, to which the user can attach rich semantic descriptions, or a label unavailable in the label menu, using voice-to-text technology. An example of such a pin is shown in Figure 5. The user can choose from three methods to place these pins: *magnetic*, *head* and *manual* (see third row of the central menu in Figure 2). We implemented three options, such that the user can choose the method most convenient to them. Magnetic and head options use the MRTK *Surface Magnetism Solver* [13], 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* [12] and allows the pin to be selected, rotated, and displaced with pinching motions of the hand.

The user can attach a description to each placed pin. This is achieved using a dictation panel (shown on the bottom right of Figure 2). With a pin selected, the user can use the record buttons to record a verbal description. The spoken words are translated to text using MRTK's off-the-shelf dictation handler [11]. The text is stored together with the pin. Upon export, each pin and its corresponding description are mapped to its closest segment.

3.4.1 Export

Export is handled by sending the annotation data to our Python server and saving files there. To be more precise we export the following files:

- An .obj file of the scene mesh with per-vertex colors for easy visualization in e.g., *MeshLab* [7]

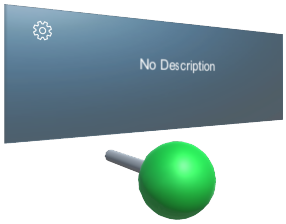


Figure 5. Description Pin as described in Section 3.4

- A list of of created pins, containing their 3D position and assigned description
- 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 description

For 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 looking up the segment this vertex belongs to.

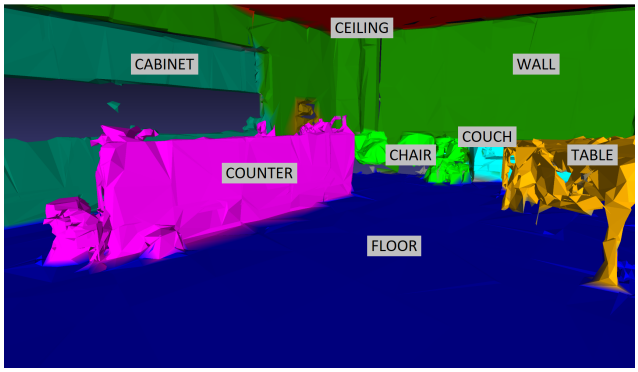
4. Usability Evaluation

For our interactive tool, we need to evaluate the efficiency of the interface design and the accuracy of the segmentation outputs with real users. We recruited 4 non-expert users with limited or no experience with the HoloLens to try our tool in a standardized indoor room. The environment is an open kitchen and living room, featuring landmarks 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 noisy and needed offline post-processing. This was done using vertex level drawing in MeshLab. The post-processing took 1.5 hours compared to the initial collection which took 12 minutes. The processed ground truth is shown in Figure 6.

The users are given an overview of the annotation 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 the accuracy of the semantic classes.

We record the total annotation time per trial, shown in 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. Participants #2 and #3 who self-reported

(a) RGB image of the room.



(b) Ground truth mesh annotation with labels.

Figure 6. RGB image and ground truth mesh of the testing environment.

having previous experience using the HoloLens were able to complete the task fastest. 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 several problems that hindered the progress of most or all participants:

- The sensitivity of the pinch detection is too high, so many people unintentionally painted the wrong segments while in the *Segment Drawing Mode* and took extra time to correct them.
- The coloured mesh segments sometimes did not render well through the HoloLens display, so participants could not see what colour the segments are (this may be a HoloLens issue rather than an app issue).
- Participants mistook the underlying blue wireframe HoloLens mesh as the mesh generated by our *Auto-Segmentation* algorithm, and thus wasted time trying to annotate it.

The exported mesh, segments, and labels were then analyzed offline using standard 3D segmentation metrics: Overall Accuracy (OAcc) and Intersection over Union (IoU) [16]. 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

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	-

the ratio of true positive samples over the union of the predicted positives and all positive samples (true positive + false positives + false negatives). In Equations 3 and 4, for a sample p_{xy}, x denotes the predicted class and y denotes the true class, such that p_{ii} is a true positive, p_{ij} is a false positive, and p_{ji} is a false negative. We first consider that a sample should be a vertex in the mesh, but soon discovered that this disproportionally 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 belongs to. Vertices that compose larger faces will have more weight in the computation, so the metrics would be approximately based on area.

$$OAcc = \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij}} \tag{3}$$

$$IoU = \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij} + \sum_{j=0}^{K} p_{ji} - p_{ii}} \tag{4}$$

Since the HoloLens coordinate system can drift significantly between trials, we first process the participant annotated meshes by manually aligning them and applying iterative closest point (ICP) to the ground truth mesh in *MeshLab* [7]. Due to the dynamic nature of the HoloLens' mesh, each user's scanned mesh can be slightly different from ground truth. We attempt to eliminate most of the discrepancy by deleting any extra structures in the user meshes that were not also captured in the ground truth. Then for each vertex in the user meshes, we search from the closest corresponding labelled ground truth vertex using a k-d tree. Based on the vertex-to-vertex mapping, we apply Equations 3 and 4 to each semantic class, as well as compute the mean by summing up the per-class metrics and dividing by the number of classes. For OAcc, we also calculate the percentage of correctly labelled vertices overall, as this more reasonable for imbalanced classes. The results of the participants, as well as an expert user (who represents the optimal results), are shown in Table 2.

	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*	-	-	-	-	-

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

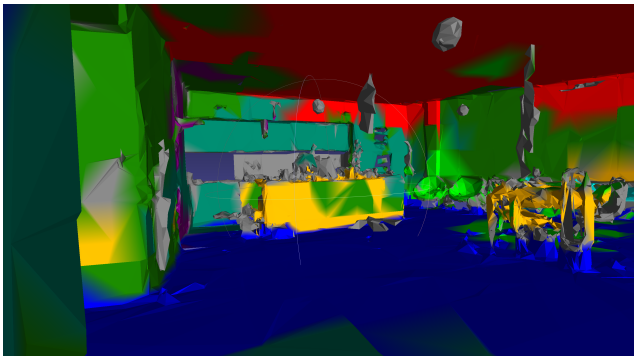
The participant OAcc and IoU scores are on the low side, but we identified several causes of this which are not related to the annotation skill of the participants: HoloLens color rendering issues and inconsistently defined segment boundaries. The latter occurs if the auto segmentation algorithm performs 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 eye, it has a huge effect on accuracy metrics. Otherwise, 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.

We can see by visual inspection of 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 the HoloLens and our app, as well as more time and encouragement to make a detailed and accurate segmentation.

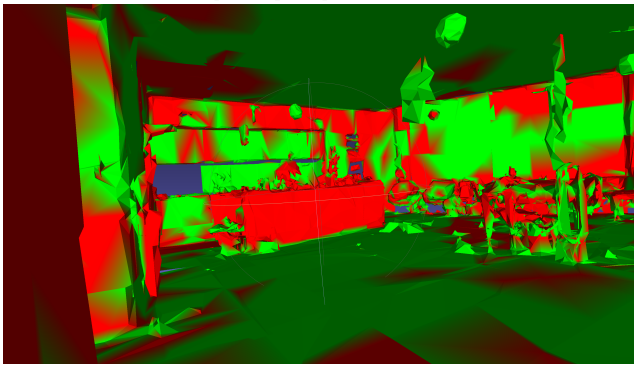
As a comparison, we also include the results from an expert user to show what a realistic upperbound on accuracy might be. It is noticeable that while the results are significantly better, the overall OAcc over all vertices is only 73.59%. More testing would be needed to full verify the reason for the discrepancy, but it is likely that the main contributor is just the natural variations between trials. It is not possible to yield identical results from two different 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 our tool.

After concluding the annotation, the users are asked to fill out a 10-question survey from the System Usability Scale (SUS) [3] 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 maximum achievable score is 100 and the typical average score is 68. Other VR systems (of purposes unrelated to scene annotation) scored 72.5 [4], 71.6 [28], and 70 [5]. Our SUS scores are in Table 1, showing an average of 61.11. This is definitely 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 the HoloLens platform.

As a bonus experiment, we also asked participant #2, who demonstrated the 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,



(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.

followed by head magnetism, then hand magnetism.

5. Discussion and Future Work

While our tool greatly simplifies the process of performing scene annotations, it still has room for improvement. First, we note that for the user study we focused on segmenting kitchens. Therefore the supported semantic categories also focus on kitchen objects. In a later stage, the labelling GUI could support multiple types of rooms which contain different objects (E.g. there could be a 'shower' label for a bathroom). Additionally, the tool is currently missing the option of adding pre-set custom label categories that are not in the proposed label set. This functionality should also be added in the future.

Considering that the whole segmentation is done based on colors, the usefulness of the physical pins has to be reassessed. At the very least, the three versions of the pin should be re-evaluated and only the best performing one should be kept to minimize our user menu. However, getting rid of the pins altogether would also be an option. It might be more intuitive to simply add the additional information upon segment creation, instead of physically attaching them afterwards.

Further, we note that the tool focuses on semantic segmentation. In a later state, the focus could be changed to instance segmentation. The choice of segmentation algorithm is based on performance and simplicity concerns. The result could be improved by investigating different segmentation algorithms, and look into tools such as PointNet [22]. Testing on our data did not show improved performance. To leverage the advantages of such a network would probably require retraining on a dataset that is specific to our case.

During the user evaluation, 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 our tool. Generally, we also found that the tool does need some experience with the HoloLens, and we therefore suggest to have the user complete a HoloLens training session 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.

There is an option within Unity to control the triangle density at which the surface reconstruction happens in the HoloLens. Adding this as an option for the user at runtime could be a useful feature to let the user decide at which granularity they would like to annotate a scene.

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.

6. Conclusion

In this paper, we introduce as semi-automatic tool to allow 3D scene annotations be conducted 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 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. This shows that the tool can also help non-expert users to perform the task of scene annotation. While the results look promising, we also present possible improvements mostly on the user interface and general user experience, but also on implementation details and efficiency.

References

[1] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017. 1

[2] Maarten Bassier, Maarten Vergauwen, and Florent Poux. Point cloud vs. mesh features for building interior classification. *Remote Sensing*, 12(14), 2020. 2

[3] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996. 7

[4] Ann L Butt, Suzan Kardong-Edgren, and Anthony Ellertson. Using game-based virtual reality with haptics for skill acquisition. *Clinical Simulation in Nursing*, 16:25–32, 2018. 7

[5] Ann L Butt, Suzan Kardong-Edgren, and Anthony Ellertson. Using game-based virtual reality with haptics for skill acquisition. *Clinical Simulation in Nursing*, 16:25–32, 2018. 7

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 1

[7] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 5, 6

[8] Dawson-Haggerty et al. trimesh. 4

[9] Microsoft Docs. Mixed-Reality: Hand Menu. 2

[10] Microsoft Docs. Mixed-Reality: Spatial Mapping. 3

[11] Microsoft Docs. MRTK: Dictation. 5

[12] Microsoft Docs. MRTK: Object manipulator. 5

[13] Microsoft Docs. MRTK: Solver overview, Surface Magnetism. 5

[14] Ruiqi Guo and Derek Hoiem. 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, pages 2144–2151, United States, Jan. 2013. Institute of Electrical and Electronics Engineers Inc. 2013 14th IEEE International Conference on Computer Vision, ICCV 2013 ; Conference date: 01-12-2013 Through 08-12-2013. 2

[15] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, June 2013. 2

[16] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. Deep learning based 3d segmentation: A survey. *CoRR*, abs/2103.05423, 2021. 6

[17] Umberto Michieli, Maria Camporese, Andrea Agiollo, Giampaolo Pagnutti, and Pietro Zanuttigh. 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, New York, NY, USA, 2019. Association for Computing Machinery. 2

[18] Ondrej Miksik, Vibhav Vineet, Morten Lidegaard, Ram Prasaath, Matthias Niessner, Stuart Golodetz, Stephen L. Hicks, Patrick Pérez, Shahram Izadi, and Philip H.S. Torr. 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, page 3317–3326, New York, NY, USA, 2015. Association for Computing Machinery. 2

[19] Duc Thanh Nguyen, Binh-Son Hua, Lap-Fai Yu, and Sai-Kit Yeung. A robust 3d-2d interactive tool for scene segmentation and annotation, 2016. 1, 2

[20] Florent Poux. How to automate 3D point cloud segmentation and clustering with Python. 2, 4

[21] Florent Poux and Roland Billen. Voxel-based 3d point cloud semantic segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, 8(5), 2019. 2

[22] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2, 8

[23] Pierluigi Zama Ramirez, Claudio Paternesi, Daniele De Gregorio, and Luigi Di Stefano. Shooting labels by virtual reality. 2019. 2

[24] Vedant Saran, James Lin, and Avideh Zakhor. Augmented annotations: Indoor dataset generation with augmented reality. 2018. 2

[25] Christian Spiekermann, Giuseppe Abrami, and Alexander Mehler. Vannotator: a gesture-driven annotation framework for linguistic and multimodal annotation. 2019. 2

[26] Yu-Shiang Wong, Hung-Kuo Chu, and Niloy Mitra. Smartannotator: An interactive tool for annotating rgbd indoor images. *Computer Graphics Forum*, 34, 03 2014. 2

[27] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018. 1

[28] Jonathan Yépez, Luis Guevara, and Graciela Guerrero. Aulavr: Virtual reality, a telepresence technique applied to distance education. In *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–5, 2020. 7