Supplementary Materials for *ImaginateAR: AI-Assisted In-Situ Authoring in Augmented Reality*

Summary of Supplementary Materials

1 Depth Map Enhancement Using Monocular Depth Estimator

Figure 1 shows an example. To overcome sensor inaccuracies (especially outdoors for objects farther than 6-10m), we scale a monocular estimator's output [13] using valid sensor depth points, yielding dense metric maps.

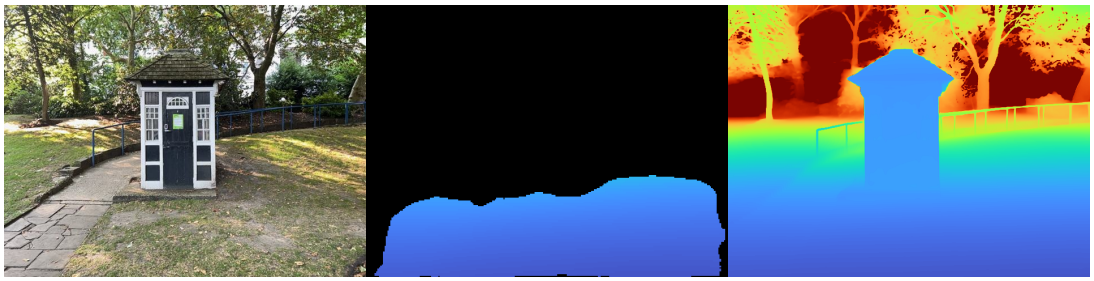


Fig. 1. Comparing inputs and output for monocular depth enhancement: The figure shows the source image (left), the sensor's original depth map (center), and the improved metric monocular depth map after enhancement (right). In the depth maps, red colors signify points farther from the viewer. The depth maps use a colormap where red represents farther points.

2 Scene Understanding Pipeline: Limitations and Future Works

The proposed scene understanding pipeline is effective in generating compact scene graph with semantic labels. However, we have identified three main issues that can be addressed in future work. First, results generated by our method are conditioned by the initial set of masks predicted by OpenMask3D [11]. This method, trained indoors [2],

Author's Contact Information:

detects masks in outdoor scenarios, but suffers from a domain gap. Nevertheless, this space is moving rapidly, and more robust methods were released after our user study. For instance, the method in [4], which has an example of scene understanding in an outdoor scenario, may be able to improve our results. A better initial set of masks is crucial to remove the preliminary filtering step: in fact, this filter acts without prior scene information and potentially removes or merges valid masks. Second, our clustering refinement is directly affected by the quality and the consistency of the labels predicted by the vision-language model (VLM). When these labels are incorrect, the results of the clustering step on top of the semantic point cloud can cause errors. For example, this may happen when the VLM ignores the prompt and tries to classify what has been covered by the mask, as shown in Figure 2. In this case, the misclassified bounding box—labeled as drinking fountain instead of bush—increases the chances of it being merged with the bounding box of the actual fountain, thereby generating a noisy bounding box for the fountain. In our experiments, we have noticed that the majority of the predicted labels are stable across different runs; however, a few of them might vary. This effect explains the slightly higher variation observed in the last row of the benchmark (see Table 2 of the main document). Validation checks could be enforced in future work to mitigate the problem. Third, our scene graphs are snapshots of the world at the time of scanning. This is generally not a problem because they mainly represent static objects (dynamic objects—such as parked cars—could potentially be removed using semantic labels), however we do not include real-time scene understanding components during the user experience. Future work can address this limitation by incorporating live components to further enhance the scene graphs.

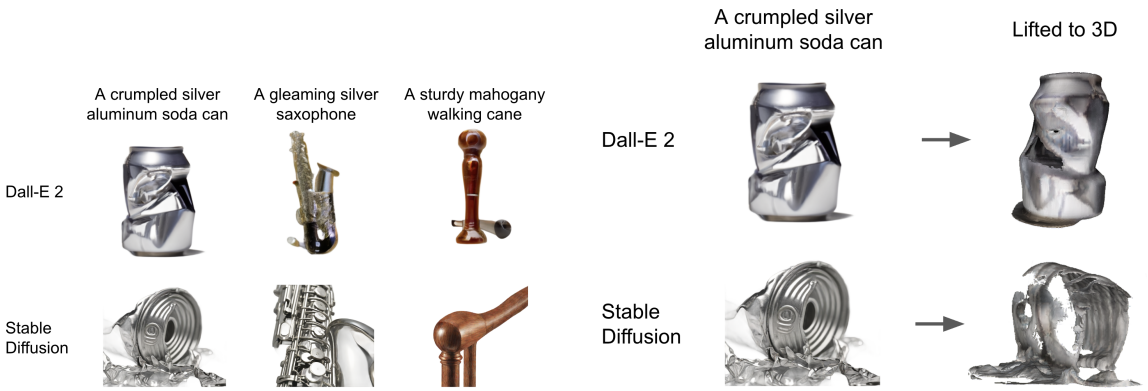
Fig. 2. Object and Context crops for a failure case. In this case, the VLM tried to *look through* the white mask and predicted drinking fountain instead of bush or vegetation.

3 AI generated 3D Assets

As reported in the main paper, the *Action Plan* Agent can invoke the generation of a virtual 3D asset on-the-fly.

We chose DALL-E2 [8] for its ability to generate images conditioned on input masks. Although its visual quality is inferior to other generative models, such as Stable Diffusion [9], the inpainting constraint often yields more complete objects on a plain background, which directly facilitates background removal [7] and image-to-3D reconstruction [12]. Figure 3 shows examples of assets generated by StableDiffusion Turbo [10] that are partially visible in the image. These assets can lead to ambiguous 3D models when lifted by InstantMesh [12].

Figure 4 compares the images generated by our pipeline for the same prompts with and without using prompt boosting. Using prompt boosting results in assets that are better suited for 3D reconstruction.



(a) Generated images. (b) Generated 3D assets

Fig. 3. Comparison between DALL-E2 [8] and StableDiffusion Turbo [10]. In these examples the inpainting strategy helps in generating fully visible objects. In contrast, StableDiffusion Turbo generates high-quality images of partially visible objects. When lifted in 3D, partially visible assets can lead to ambiguous or poorly defined 3D models. We used prompt-boosting for both the strategies.

4 Prompts

In this section, we report the prompts used by our AI agents.

Figure 5 presents an example of a prompt used by the *Object Classifier* agent. Instructions are given to the agent in GPT's *System mode* [6], while additional inputs—the object and context crops, as well as the list of previously predicted labels—are passed in GPT's *User mode* [6]. Finally, the agent replies with the semantic label of the object. It is worth noting that all inputs are generated by the system itself during execution, with no human intervention required.

Figure 6 shows an example of the prompt used by the *Prompt Boosting* agent. Again, we use *System mode* to instruct the agent about the task, and in *User mode*, we provide two visual examples (these examples are not request-specific; we use the same example in every request) and the initial user prompt to boost. The agent replies with the boosted prompt, which is then used to generate an image suited for image-to-3D methods.

Figure 7 illustrates the prompt used by the *Brainstorming* agent. This agent generates a short story-based description of an AR experience given a list of real and virtual objects in the scene, as well as conversations with the user.

Figure 8 reports the prompt used by the *Action Plan* agent. Based on the current state of the AR experience and a list of already-generated assets, the agent defines the action to apply to each existing and new virtual object to better align it with the user-described experience.

Finally, Figure 9 shows the prompt adopted by the *Assembly* agent. This agent receives the actions planned by the *Action Plan* agent (e.g., modifying an object's position) and applies them to virtual objects.

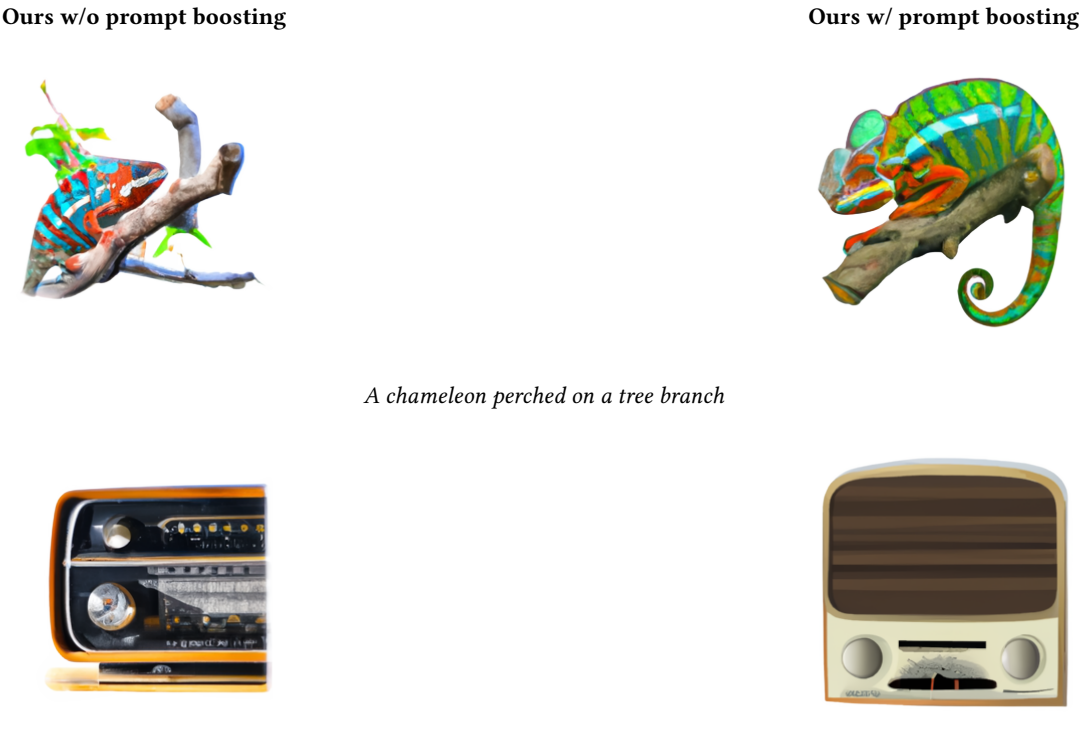


Fig. 4. A qualitative comparison of generated images: without (first column) and with (second column) prompt boosting. Prompt boosting results in objects that are more fully visible.

Role: System

We need your help to classify the object depicted in the image. The image contains two parts, divided by a black vertical section:
- on the left you can see the object of our interest. The object could be either things (such as a `vase`) or stuff (such as `vegetation`). You ****MUST IGNORE**** what is masked out with a white mask, because we don't care about it.
- on the right you can see the object plus some context. You can use this image to understand the object better, but the object of interest is in the left image.

We also provide the list of previously detected objects in the scene (comma separated). This list might be empty, meaning that no object has been detected yet. You can use this list to avoid repeating the same label, but you can also provide a new label if you think it's more appropriate to describe the object.

Your task is to provide the semantic label that better describes the visible object in the left image, ignoring the white mask. You have to consider the context of the object to provide a better answer: for example, if the object is not particularly relevant (e.g., the object is a ventilation grille hanged to the wall), you might want to provide a more general label (e.g., `wall`). However, remember to ignore the white mask.

Role: User

Role: User

Here the list of already detected objects: `path`, `rock`

Role: User

****DON'T LABEL**** what has been covered by the white mask in the left image.

Response

class_name: `grass`

Fig. 5. Example of the prompt used by the Object Classifier agent

Role: System

We want to generate a 3D mesh from an initial image. We use DALL-E 2 to generate the initial image starting from a textual prompt given by the user. Then we use a model that generates a 3D mesh starting from the image generated by DALL-E 2. For this reason, it is important that the generated image by DALL-E 2 is easy to segment and that the object of interest is clearly visible, complete and resembles a 3D asset.

We provide you two images, called **BAD** and **GOOD**, both generated by DALL-E 2 using the following prompt: *'a dragon that is fully visible and that looks at the camera.'*

Here why BAD and GOOD are different:
- **BAD** is a bad example because it does not show the object entirely and it looks flat, so it is not good for meshing.
- **GOOD** is a good example because it shows the object in a clear way, it looks a 3D asset and it is good for meshing.

We also provide you the prompt that contains the object to generate. We call this prompt **PROMPT**, and it can contain characteristics of the object. An example of possible **PROMPT** is: *`a cute puppy dragon`*

Your goal is to expand **PROMPT** returning the final prompt that we can use to generate a good image with DALL-E 2. Please only provide the prompt in your response.

Please consider these ****IMPORTANT**** points when you expand **PROMPT**:
- the object must be easy to segment. Difficult backgrounds or complex details might make the object hard to segment. Reflection or difficult lighting conditions are not good for meshing.
- the object must be easy to mesh. Flat, complex and intricate objects are not good for meshing. The object must be clearly visible.
- the object must be fully visible. Partially visible objects are not good for meshing.
- we only care about the object. The background is not important, as well as the context. For instance, if the object is a dragon, we don't care if the dragon is flying or standing on a rock. We only care about the dragon. Don't ask to generate additional objects or elements that are not part of the object.

Role: User

Here the **GOOD** example generated by DALL-E 2:

Role: User

Here the **BAD** example generated by DALL-E 2:

Role: User

Here **PROMPT**: a parrot

Response

a vibrant parrot sitting upright, fully visible with its colorful feathers clear against a plain white background, showcasing its distinct beak and eye features, with no distractions in the background.

Fig. 6. Example of the prompt used for *Prompt Boosting* when generating new assets.

Role: System

The user wants to create an AR experience. Your job is to help the user with brainstorming this AR experience.
Here is what the user said: **$userinput**
You will be given scene graphs in JSON format representing a 3D AR scene with real and virtual objects. Each object is defined by the 3D bounding box that encloses it (i.e., a Unity BoxCollider). The coordinate system is left-handed, Y-up coordinate system.
Here is scene graph JSON structure in more detail:

```
{“objects” :
  [     {
        "guid": "string. A unique ID of this object. Do not change this.",
        "class_name": "string. Name of the object.",
        "pos": {
           "x": "float. X position.",
           "y": "float. Y position.",
           "z": "float. Z position."
        },
        “rot”: {
           "x": "float. X rotation. Object appears right-side up with this rotation, but rotation still needs to be adjusted.“,
           "y": "float. Y rotation. Object appears right-side up with this rotation, but rotation still needs to be adjusted.”,
           "z": "float. Z rotation. Object appears right-side up with this rotation, but rotation still needs to be adjusted.”
        },
        "scale": {
           "x": "float. X scale.",
           "y": "float. Y scale.”,
           "z": "float. Z scale.”
        },
        "minBound": {
           "x": "float. Minimum x-coordinate when fitting a BoxCollider around the object.",
           "y": "float. Minimum y-coordinate when fitting a BoxCollider around the object.",
           "z": "float. Minimum z-coordinate when fitting a BoxCollider around the object."
        },
        "maxBound": {
           "x": "float. Maximum x-coordinate when fitting a BoxCollider around the object.",
           "y": "float. Maximum y-coordinate when fitting a BoxCollider around the object.",
           "z": "float. Maximum z-coordinate when fitting a BoxCollider around the object."
        },
        "onScreen": "boolean. Represents whether this object is currently visible on the user’s phone screen.“,
        "action": “string. No actions need to be done just yet, so this is likely set to ‘none’.”
     }
  ]
}
```

You will receive two JSONs:
Real World Scene Graph represents real-world objects such as trees in a park. This is to help you understand what is around the user so that you can tailor your brainstorming around the user’s context.
Here is the real world scene graph: **$realobjects**
Virtual Objects Scene Graph represents existing virtual objects. This may be empty, meaning there are currently no virtual objects in the AR scene. If not empty, then you should also consider virtual objects around the user when brainstorming an idea.
And here is the virtual objects scene graph: **$virtualobjects**
Finally, here is the result from a previous discussion between the user and a different AI agent. This may also be empty: **$context**.
Come up with a short, simple, and creative one sentence description of an AR experience based on the user’s request and contextual information. Use simple language. Make sure to not output anything else.

Fig. 7. Prompt used by the *Brainstorming* agent.

Role: System

The user wants to create an AR experience using two AI agents. You are the first AI agent. Your job is to create an action plan by assigning actions to virtual objects in the AR environment for the next two agents to execute.
Here is the user requested AR experience: **$userinput**.
You will receive three JSONs:
1. Real World Scene Graph: Represents real-world objects (e.g., trees in a park). Action should always be 'none'. No action should be performed on real-world objects. Use this JSON as a reference to understand the user's surroundings.
Real World Scene Graph Structure:

```
{
  "objects":[
    {
      "guid":"string. A unique ID of this object. Do not change this.",
      "class_name":"string. Name of the object.",
      "onScreen":"boolean. Represents whether this object is currently visible on the user's phone screen."
    }
  ]
}
```

2. Virtual Objects Scene Graph: Represents existing virtual objects. May be empty. Assign actions to each virtual object.
Virtual Objects Scene Graph Structure: Shares the same structure as the Real World Scene Graph.
3. Existing Assets JSON: Represents assets in the user's application database. Based on the user's request, you may pick objects to be created by the system.
Existing Assets JSON Structure:

```
{
  "objects":[
    {
      "class_name":"string. Name of the object.",
      "prefabLocation":"string. Should be one of two values: 'resources' or 'persistent'. 'resources' means the object is in a folder called 'resources' in the database. 'persistent' means the object is in a folder called 'persistent'."
    }
  ]
}
```

Here is your input:
Real World Scene Graph: **$realobjects**
Virtual Objects Scene Graph: **$virtualobjects**
Existing Assets JSON: **$existingassets**
Your output: Generate a modified Virtual Objects Scene Graph by assigning an "action" to each virtual object. You should determine which objects in the user's request are virtual, as we can only perform actions on virtual objects. The possible actions are:
- none: No change needed.
- remove: Remove from the AR scene.
- update: Modify an existing object's properties (e.g., position, rotation, scale).
- create_resources: Create an object. This object must have a "prefabLocation" of "resources" according to the Existing Assets JSON.
- create_persistent: Create an object. This object must have a "prefabLocation" of "persistent" according to the Existing Assets JSON.
- create_new: Generate a new object. This is if an asset does not exist in the user's database according to the Existing Assets JSON. Only at most one object in the output scene graph JSON can be marked as "create_new."
Output Scene Graph Structure:

```
{
  "objects":[
    {
      "guid":"string",
      "class_name":"string",
      "action":"string"
    }
  ]
}
```

For new objects, leave the guid field empty.
Make sure to not output anything else besides this scene graph JSON. Do not include any miscellaneous or trailing characters.

Fig. 8. Prompt used by the *Action Plan* agent.

Role: System

The user wants to create an AR experience using two AI agents. You are the second agent. The first AI agent created the necessary virtual objects and assigned action tasks for each virtual object. Your job is to execute these actions, which will involve changing the position, rotation, and/or scale of each virtual object so that it is placed naturally in the real world.
You will be given a scene graph in json format representing a 3D AR scene with real and virtual objects, where each object is defined by the 3D bounding box that encloses it. The coordinate system of the 3D world is a left-handed, Y-up coordinate system. I will help you understand the json format. Each object's size and position is given as an axis-aligned bounding box in xyz-coordinates. So for example 'min_x' represents the minimal value of the bounding box along the x-axis. Your goal is to put arrange the virtual objects in the scene, so that they are placed in sensible locations. Rotation of the object is given as a forward vector, which is the direction the object is facing. In the end, virtual objects should be positioned and rotated in a natural way and be sized appropriately in relation to other objects in the AR scene.
Here is the user requested AR experience: **$userinput**.

You will receive three JSONs:
1. Real World Scene Graph: Represents real-world objects (e.g., trees in a park). Action should always be 'none'. No action should be performed on real-world objects. Use this JSON as a reference to understand the user's surroundings.
Real World Scene Graph Structure:{
  "objects":[
    {
      "guid":"string. A unique ID of this object. Do not change this.",
      "class_name":"string. Name of the object.",
      "min_x":"float. minimal value of the bounding box along the x-axis",
      "max_x":"float. maximum value of the bounding box along the x-axis",
      "min_y":"float. minimal value of the bounding box along the y-axis",
      "max_y":"float. maximum value of the bounding box along the y-axis",
      "min_z":"float. minimal value of the bounding box along the z-axis",
      "max_z":"float. maximum value of the bounding box along the z-axis",
      "forward_x":"float. direction the object is facing along the x-axis",
      "forward_y":"float. direction the object is facing along the y-axis",
      "forward_z":"float. direction the object is facing along the z-axis",
      "onScreen":"boolean. represents whether this object is currently visible on the user's phone screen.",
      "action":"string. should be one of five values: 'none', 'remove', 'update', 'create_resources', or 'create_persistent'."
    }
  ]
}
2. Virtual Objects Scene Graph: Represents existing virtual objects. May be empty. Assign actions to each virtual object. Adjust bounds as necessary.
Virtual Objects Scene Graph Structure: Shares the same structure as the Real World Scene Graph.
Here is an explanation of each 'action':
- none: No change needed.
- remove: Remove from the AR scene. You should treat this object as not in the AR scene.
- update: Modify an existing object's properties (e.g., position, rotation, scale).
- create_resources: Create an object. This object has already been made by the previous AI agent. Your job is to change its position, rotation, and/or scale so that this object is placed in a sensible location.
- create_persistent: Create an object. This object has already been made by the previous AI agent. Your job is to change its position, rotation, and/or scale so that this object is placed in a sensible location.
- create_new: Generate a new object. This object has already been generated by the previous AI agent. Your job is to change its position, rotation, and/or scale so that this object is placed in a sensible location.
3. Player Scene Graph: Represents the player's current position and rotation. You should use this information to arrange virtual objects in a way that is visible to the player, if possible.
Here are your inputs:
Real World Scene Graph: **$realobjects**
Virtual Objects Scene Graph: **$virtualobjects**
Player Scene Graph: **$player**

Your output: Generate a modified Virtual Objects Scene Graph by rearranging the necessary virtual objects. The final AR scene should be realistic, sensible, and follow natural laws. Objects should never overlap. If virtual objects need to be placed close to other real or virtual objects, ensure that they maintain a reasonable minimum distance to avoid visual clutter or unnatural proximity. Objects should not be inside of other objects or grounds. Grounds, or objects at ground level, can be described in many ways, including dirt and grass. As an example, for a virtual object A to be on top of grass ground B, you should compare the 'min_y' of object A with the 'max_y' of object B and make sure 'min_y' of object A is at least greater than 'max_y' of object B. This way, the virtual object A is on top of grass ground B. Ensure that virtual objects respect real-world physics. For instance, objects should rest on flat surfaces (like the ground) instead of floating in mid-air unless intentional. Objects should also face natural directions. For example, when placing a virtual object next to a real world object, they should face similar directions so that the front of both objects are visible from the same player angle.
Do not change any virtual objects with action of 'none'. Also do not change any virtual objects that are irrelevant to the user's request.
If the user's request is ambiguous, you should not always just place virtual objects near objects with onScreen of true. Try to use the onScreen parameter only if it is necessary to fulfill the user's request.

Objects also should not be too big or too small relative to other objects in the AR scene. Ensure that virtual objects are proportionate to other real and virtual objects in the scene. For example, a virtual chair should be similar in size to any real chairs nearby. If there are no similar real objects, adjust the scale based on common sense.
Lastly, in cases where a virtual object cannot be placed naturally or you are unsure whether to place a virtual object, place it close to the player so that it is at least visible.
Ensure the guid, class_name, onScreen, and action parameters remain unchanged.
Make sure to not output anything else besides this modified Virtual Objects Scene Graph JSON. Do not include any miscellaneous or trailing characters.

Fig. 9. Prompt used by the *Assembly* agent.

5 Study Materials

This section contains references to the validated questionnaires we used, as well as custom questions and semi-structured interview questions we developed for the study.

5.1 Pre-Study Questionnaire

The pre-study questionnaire asked participants various demographics and experience questions, as shown in Listing 1.

Listing 1. The pre-study questionnaire.

```
1. What is your age?
[Number]

2. What gender do you self-identify with? (E.g. woman, non-binary, man, etc.)
[Short answer]

~ Prior Experience with Technology ~
3. How familiar are you with augmented reality (AR), such as Pokemon GO, Apple Vision Pro, Meta Quest,
      etc.?
[Options: Not at all familiar to Very familiar on a 5-point scale]

4. List any AR technologies and applications you have used before and what you used them for (or write
      "None").
[Long answer]

5. How often do you use augmented reality (AR) technologies or applications?
[Options: Multiple times each day to Never on a 7-point scale]

6. How familiar are you with artificial intelligence (AI) chat systems, such as chatbots, ChatGPT, etc
     .?
[Options: Not at all familiar to Very familiar on a 5-point scale]

7. List any AI chat systems you have used before and what you used them for (or write "None").
[Long answer]

8. How often do you use AI chat system(s)?
[Options: Multiple times each day to Never on a 7-point scale]

9. How familiar are you with 2D creativity tools, such as the Adobe suite (e.g. Photoshop, Premiere
     Pro), Figma, etc.?
[Options: Not at all familiar to Very familiar on a 5-point scale]

10. List any 2D creativity tools you have used before and what you used them for (or write "None").
[Long answer]

11. How often do you use 2D creativity tool(s)?
[Options: Multiple times each day to Never on a 7-point scale]

12. How familiar are you with 3D creativity tools, such as Maya, Blender, Unity, Unreal, etc.?
[Options: Not at all familiar to Very familiar on a 5-point scale]

13. List any 3D creativity tools you have used before and what you used them for (or write "None").
[Long answer]

14. How often do you use 3D creativity tool(s)?
```

```
[Options: Multiple times each day to Never on a 7-point scale]
```

5.2 Part 1: Comparison Task Questionnaires

There were three comparison task questionnaires in the first part of the study. We provided the first one to participants after every trial of the five features (e.g., brainstorming, modifying objects, etc.) for one of the three systems (A, B or C). This meant each participant completed this questionnaire 15 times (5 features by 3 systems). It is shown in Listing 2. This survey included custom questions about creativity, the UMUX-LITE [5] questionnaire to assess usability, and questions from the NASA-TLX [3] to assess task load.

The second comparison task questionnaire asked the participants which system they preferred overall and why, as shown in Listing 3. The final comparison task questionnaire provided participants with descriptions of how new features (e.g., adding music, animating objects, etc.) would work if they were added to System A, B or C, and asked which version participants would prefer and why. This is shown in Listing 4.

Listing 2. The first comparison task questionnaire, which participants completed after every trial of a new system feature.

```
1. Which feature did you just try?
[Options:
1 Brainstorming Ideas
2 Choosing / Creating 3D Virtual Object(s)
3 Object Placement
4 Object Selection & Modification
5 Object Selection & Remove Object
]

2. Which version did you use? (See app for the feature version)
[Options: A, B, C]

[If "1 Brainstorming Ideas" was chosen:]
3. In that brainstorming session, how do you feel about the end result, creatively speaking?
[Options: Not at all creative to Very creative on a 5-point scale]

[If "2 Choosing / Creating 3D Virtual Object(s)" was chosen:]
4. When you chose or created an object, how did you feel about the end result, creatively speaking?
[Options: Not at all creative to Very creative on a 5-point scale]

[UMUX-LITE: Both questions]

[NASA-TLX: The Mental Demand, Performance, Effort, and Frustration questions]
```

Listing 3. The second comparison task questionnaire, which participants completed after they finished all of the trials. This questionnaire compared the systems overall (as opposed to each feature).

```
1. Which system did you prefer? (Please choose one, if possible)
[Options:
System A: Manual (e.g. tap to place)
System B: AI system with multiple responses/options (e.g. select one of the AI's object placements)
System C: AI system with single response/option (e.g. AI system places an object)
]

2. Why did you prefer the above system? (Please write at least 1-2 full sentences.)
[Long answer]
```

Listing 4. The third comparison task questionnaire, which asked participants to brainstorm about new features that had not yet been added to the app.

```
1. Imagine you were adding music to your AR experience. How would you prefer to do this? (Please
    choose one, if possible)
[Options:
System A: Manually scroll through a list of music files and choose one.
System B: Ask the AI for music (e.g. "Can you add upbeat music?"), and it gives you 3 options.
System C: Ask the AI for music (e.g. "Can you add upbeat music?"), and it adds that song.
Other - Write in
]

2. Why did you prefer the above? Please write 1-2 full sentences.

3. Imagine you were adding sound effects to your AR experience. How would you prefer to do this? (
    Please choose one, if possible)
[Options:
System A: Choose an object (e.g. a dog), then tap "add sound effects", then manually scroll through a
    list of sound effect files, and choose one.
System B: Choose an object (e.g. a dog), then ask the AI for a sound effect (e.g. "Can you add a
    barking sound effect?"), and it gives you 3 options.
System C: Tell the AI to generate a sound effect with respect to an object (e.g. "Can you add a
    barking sound effect to this dog?"), and it adds that sound effect.
Other - Write in
]

4. Why did you prefer the above? Please write 1-2 full sentences.

5. Imagine you were animating objects in your AR experience. How would you prefer to do this? (Please
    choose one, if possible)
[Options:
System A: Choose an object, then tap "animate". Tap a start location and end location for the object's
     journey. Choose which type of animation (e.g. bounce, walk, run, etc.).
System B: Choose an object, then ask the AI system to generate an animation (e.g. "Make it walk to the
     dog bowl"), and it gives you 3 options proposing a path and walk style.
System C: Tell the AI system to generate an animation (e.g. "Make it walk to the dog bowl"), and it
    generates that animation.
Other - Write in
]

6. Why did you prefer the above? Please write 1-2 full sentences.

7. Imagine you were preparing an object to react to an event in your AR experience.
***An event*** can be triggered by a person pressing a virtual button, standing near a virtual object,
     making a noise or saying a trigger word, etc.
***A reaction*** to an event could be an object animating or changing colour.
How would you prefer to prepare an object to react to an event? (Please choose one, if possible)
[Options:
System A: Choose the reacting object, then tap "events". Choose an event from a dropdown list. Choose
    how the object should react from a list.
System B: Choose the reacting object, then tell the AI system what event should trigger what
    corresponding reaction. The AI gives you 3 options proposing various reactions.
System C: Tell the AI system that an event should trigger a reaction. The AI provides that reaction to
     that event.
Other - Write in
]
```

```
8. Why did you prefer the above? Please write 1-2 full sentences.

9. Imagine you were pinning a virtual object to a real one in your AR experience. E.g. pinning a
     virtual hat to a statue.
How would you prefer to do this? (Please choose one, if possible)
[Option:
System A: Choose the real object, e.g. the head of a statue. Tap "pin" and then choose a virtual
     object from a list, e.g. a hat. Adjust the virtual object's location/scale/orientation to fit the
      real object.
System B: Choose the real object, then tell the AI system what virtual object to pin. The AI gives you
      3 options proposing various objects and locations/scales/orientations.
System C: Tell the AI to pin a virtual object to a real object, and it pins it.
Other - Write in
]

10. Why did you prefer the above? Please write 1-2 full sentences.
[Long answer]

11. Any other comments?
[Long answer]
```

5.3 Part 2: Free-Form Authoring Task Questionnaire

After participants completed the free-form authoring task in Part 2 of the study, they completed the Creativity Support Index (CSI) questionnaire [1]. This can be found in Cherry and Latulipe's paper [1]. Note that we did not include the Collaboration factor, as our system is not collaborative.

5.4 Part 3: Semi-Structured Interview Questions

Listing 5 contains the main interview questions we asked participants. Since it was a semi-structured interview, we additionally asked follow-up questions about certain responses, e.g., for clarity.

Listing 5. The main interview questions we asked participants in Part 3 of the study.

```
~ Questions about the tool ~
1. What did you build and who did you build it for?

2. What features did you use to build it? Why?

3. Did any of the features you used while building stand out to you? Why?
  a. What was your favourite feature? Why?
  b. What was your least favourite feature? Why?

4. What were some of the pros/cons to using the AI agent vs manual process?

5. Was there anything that you found particularly enjoyable or rewarding?

6. Was there anything that you found particularly stressful?
  a. (E.g. social stressors)

7. Imagine that your job was to create AR experiences (e.g. for the audience you chose in Q1). Would
     you feel confident using this as an initial brainstorming/prototyping tool? Why?
```

```
~ Brainstorming questions ~
8. Imagine you could make this tool infinitely better. How might you improve it? What two features
     might you add?

9. Imagine you had this "better tool" and much more time, and you could build an AR experience in any
     location. What AR experience might you build and where?

~ Final question ~
10. Any other thoughts or comments?
```

References

[1] Erin Cherry and Celine Latulipe. 2014. Quantifying the Creativity Support of Digital Tools through the Creativity Support Index. *ACM Trans. Comput.-Hum. Interact.* 21, 4, Article 21 (jun 2014), 25 pages. https://doi.org/10.1145/2617588

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 5828–5839.

[3] Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage publications Sage CA: Los Angeles, CA, 904–908.

[4] Rui Huang, Songyou Peng, Ayca Takmaz, Federico Tombari, Marc Pollefeys, Shiji Song, Gao Huang, and Francis Engelmann. 2024. Segment3D: Learning Fine-Grained Class-Agnostic 3D Segmentation without Manual Labels. In *European Conference on Computer Vision (ECCV)*. Springer-Verlag, Berlin, Heidelberg.

[5] James R. Lewis, Brian S. Utesch, and Deborah E. Maher. 2013. UMUX-LITE: when there's no time for the SUS. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2099–2102. https://doi.org/10.1145/2470654.2481287

[6] Lucas Ostrowski. 2024. What is the difference between system, User, and Assistant roles in ChatGPT? https://lostrowski.pl/tips-news/what-is-the-difference-between-system-user-and-assistant-roles-in-chatgpt.

[7] Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. 2022. Highly Accurate Dichotomous Image Segmentation. In *ECCV*.

[8] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* 1, 2 (2022), 3.

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 10684–10695.

[10] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. 2023. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042* (2023).

[11] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. 2023. OpenMask3D: Open-Vocabulary 3D Instance Segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[12] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. 2024. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191* (2024).

[13] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024. Depth Anything V2. *arXiv:2406.09414* (2024).