A human-in-the-loop interface in virtual reality

[1]Meaghan Boykin,
[2]Guilherme Melo dos Santos, and
[3]Daniela Ushizima

[1]Sonoma State University, Rohnert Park, CA
[2]Universidade Federal de São Paulo, São Jose dos Campos, SP, Brazil
[3]Lawrence Berkeley National Laboratory, Berkeley, CA

August 6, 2025

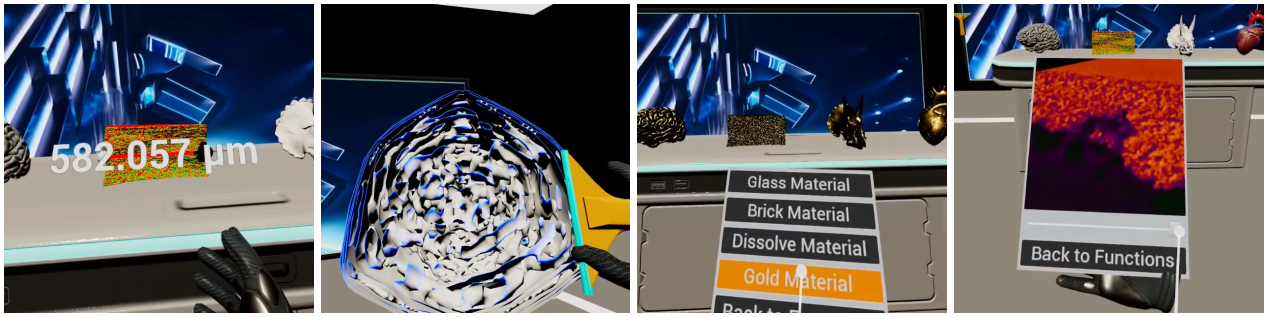
ABSTRACT

ASCRIBE-VR (Autonomous Solutions for Computational Research with Immersive Browsing & Exploration in Virtual Reality) is a virtual reality application designed to facilitate research with tools for enhanced data visualization and exploration. It provides an alternative to two-dimensional interfaces, which can be difficult to use when working with 3D and multilayered data. A goal of ASCRIBE-VR is to enable human-in-the-loop decision-making for experimental validation. This project develops ASCRIBE-VR's first human-in-the-loop interface using Unreal Engine 5.5.4 and its Blueprints Visual Scripting system. The interface is designed for the review of image data and image classifications. It allows users to import images into the application via a CSV file containing the images' file names, x-, y-, and z-positions in the virtual environment, and, optionally, classifications. Within the application, users can define 3D areas that contain images of the same classification. Users can update images' classifications by moving them in and out of these areas, and the updated data can be exported as a CSV file. These functions provide a way for researchers to quickly sort image data and provide feedback on image classifications produced by autonomous systems.

I. INTRODUCTION

Human-in-the-loop (HITL) is an approach to the operation or training of automated systems, such as machine learning models, that incorporates human knowledge and intuition. In this approach, humans can perform various tasks, including iterative data annotation, identifying critical data samples, and providing feedback on results produced by automated systems.[1,2] In some cases, humans may even demonstrate the proper way to complete a task to the system.[3] A survey of papers on HITL for machine learning found that the primary motivation for following this approach is to boost the performance of machine learning models, but HITL can also improve the transparency of systems that otherwise have details of their operation obscured from the user.[1] Despite these benefits, the HITL approach is not common in all fields of research. For example, it is not often utilized in the computer vision field, seemingly due to the lack of an ideal method of integrating human knowledge into image processing.[1] The addition of a HITL interface to ASCRIBE-VR addresses this problem.

ASCRIBE-VR is a virtual reality (VR) application designed for data visualization and exploration in 3D, providing a solution to the limits of 2D interfaces. Its development began in December 2024, and by June 2025 it was equipped with many tools for the exploration of data, outside of the context of HITL. It allowed users to import 3D data into the virtual environment as meshes and explore these meshes with tools for linear measuring (Figure 1a), slicing (Figure 1b), resizing, and changing materials (Figure 1c). It also allowed users to view 2D images in the 3D environment using a menu with a scroll bar (Figure 1d). In this project, we begin ASCRIBE-VR's expansion into HITL functionalities, starting with image data. To create an intuitive HITL experience, we developed a new way to interact with images within ASCRIBE-VR, taking full advantage of its 3D nature.



(a) Linear measurement. (b) Mesh slicing. (c) Changing materials. (d) Viewing image stacks.

FIG. 1: Pre-existing functions of ASCRIBE-VR.

II. MATERIALS AND SETUP

Like the rest of ASCRIBE-VR, we developed the human-in-the-loop interface in Unreal Engine 5.5.4 using its Blueprints Visual Scripting system. Blueprints is a visual programming interface that provides most of the functionalities offered by development in C++.[4] It expedites development by providing a context-sensitive mode that filters the most relevant methods.[5] After each phase of development in Unreal Engine, we packaged the project as an Android Package (APK) and tested new features in the Meta Quest 3 or Meta Quest 3S, as shown in the development cycle outlined in Figure 2. We used the Meta Quest Developer Hub to upload APKs to these headsets.

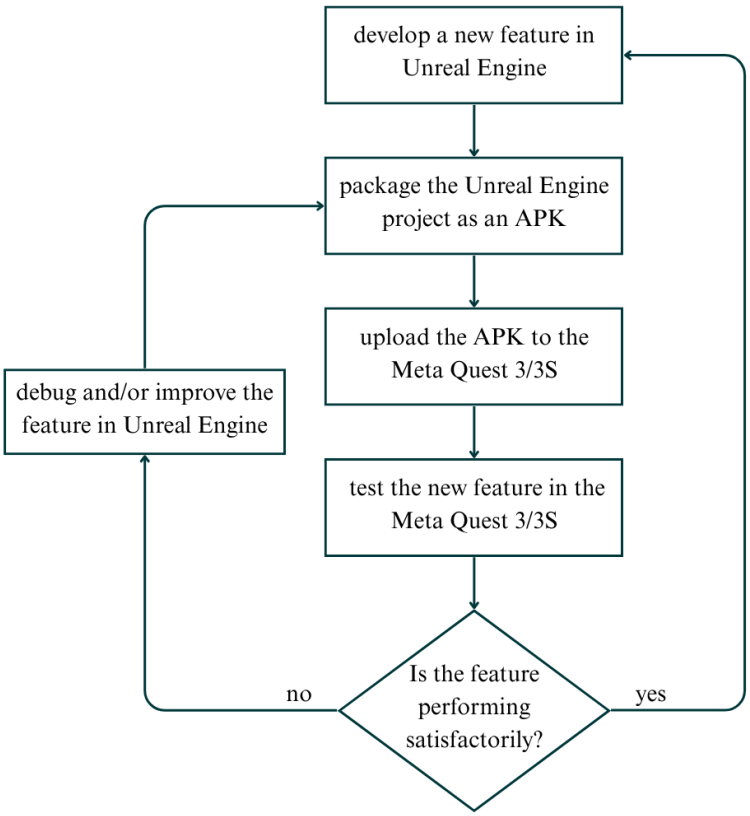


FIG. 2: ASCRIBE-VR application development cycle.

We used 49 images of leaves of size 720 x 960 pixels as sample image data. A currently unpublished machine learning script generated classification data for the images. This script, referred to as the classification script, analyzes the similarity of images to classify them and calculate their locations in the virtual environment, with similar images located closer together and dissimilar images located farther apart. The script outputs the data, referred to as the images' metadata, in a CSV file in the following format:

```
Image File Name, x-Position, y-Position, z-Position, Classification
```

While we designed ASCRIBE-VR's HITL interface for CSV files generated by the classification script, any CSV file following the same format is valid for use in the application. The first line in the CSV file, which defines the format of the CSV file, is ignored. Image file names must include their extension, for example, `image1.png`. (ASCRIBE-VR's HITL interface currently accepts PNG, JPEG/JPG, and BMP image formats.) The x-, y-, and z-positions must be floats, and ideally they should be normalized to fit within ASCRIBE-VR's virtual environment; we define the recommended maximum and minimum values for each axis in Figure 3. Classifications are optional to include in the CSV file, but if they are included, they must be strings containing numbers, letters, spaces, underscores, and/or hyphens. To access the source images and related CSV file while in the virtual environment, users must place them in the following location (or in a subdirectory of it) on the Meta Quest 3/3S:

```
[Game Source Directory Path]/HITL/
```

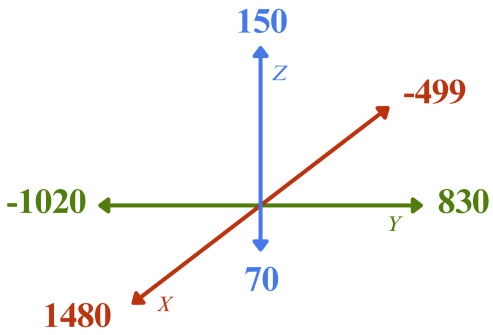


FIG. 3: Recommended ranges for x-, y-, and z-positions of images within ASCRIBE-VR.

The game source directory path, which leads to the `Source` directory, varies by device. For example, after the Meta Quest is connected via a USB-C to a desktop computer or laptop running Windows, users can access the `Source` directory in the following location on the computer:

```
This PC/Quest 3S/Internal shared storage/Android/data/
com.Ascribe.LBNL/files/UnrealGame/LBNL/LBNL/Source/
```

In some cases, the ASCRIBE-VR application must be run at least once in the Meta Quest in order for the `com.Ascribe.LBNL` directory to appear.

Lastly, we used the Real Time Import/Export Plugin, a plugin to Unreal Engine, to read and write CSV files and import source images. The plugin enables sounds, images, and meshes to be imported into Unreal Engine applications during runtime, as well as text files to be read and written during runtime.[6] Utilizing this plugin allowed our development to focus on the HITL interface and user experience, rather than on basic file input and output. Initially, the File System Library plugin[7] was used for these functions. This plugin functioned properly when the application was run through the Unreal Engine editor, but it caused the application to crash when reading or writing files while running on the Meta Quest.

III. METHODS

We developed three Blueprint classes for the human-in-the-loop interface: image actor, image manager, and cluster area. All are child classes of Unreal Engine's Actor class. Actors are objects that can be placed in the virtual environment through the Unreal Engine editor during development or spawned in during run time.[8]

A. Image actor

The image actor class allows users to see images and move them throughout the virtual environment. Each image imported into the virtual environment is an instance of the image actor class. This class contains an invisible static mesh with an attached grab component, which allows the user to grab and move the image. The grab component is accessible through Unreal Engine's VR Template, which provides basic assets for the development of VR applications.[9] The class also contains a widget component, a component provided in the VR Template that allows 2D user interface (UI) elements to be displayed in the 3D environment.[10] The widget component is used to display the image and its metadata.

The image is always visible, but the visibility of the metadata toggles on when the actor is grabbed and off when it is released to reduce the visual clutter in the scene; we accomplished this using the grab component's `On Grabbed` and `On Dropped` events to set the visibility of the metadata. The class Blueprint is shown in Figure 4 with the static mesh and metadata visible for easier visualization. We used Unreal Engine's `Find Look at Rotation` function to enable image actors to automatically rotate to face the user as they move. However, if the user grabs an image, the image will instead rotate with the user's hand, mirroring how a grabbed object behaves in real life.

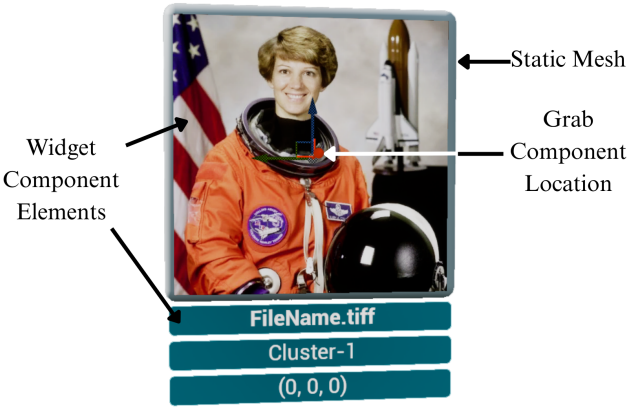


FIG. 4: Image actor Blueprint.

B. Image manager

The image manager class is an invisible actor in the virtual environment. Its location is shown in Figure 5. This actor imports images, exports metadata, and clears images from the virtual environment when prompted by the user. It uses the Real Time Import/Export Plugin's `Read String From File` function to read images' metadata from CSV files. For every image name read, the image manager spawns an image actor in its assigned location, sets its classification, and sets the image in its widget component using the Real Time Import/Export Plugin's `Load Image File` function. The image manager maintains an array of every spawned image so that the metadata of each can be exported to a CSV file when prompted by the user.

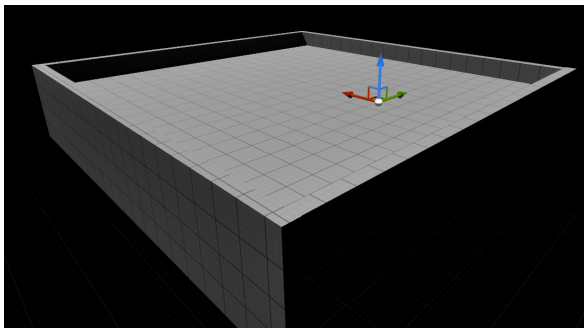


FIG. 5: Location of the image manager in the virtual environment.

C. Cluster area

The cluster area class allows users to change the classification of images. This class contains a box collision component that defines a 3D area containing images of the same classification, which are referred to as a cluster. The box collision component is visible in the Unreal Engine editor, as seen in Figure 6, but is not visible to users in the virtual environment. Using Unreal Engine's `On Component Begin Overlap`[11] and `On Component End Overlap`[12] events, we enabled the box collision component to change the classification of an image. When an image actor enters the box collision component, the `On Component Begin Overlap` event triggers the image actor to update its classification to the cluster's classification. When an image actor exits the component, the `On Component End Overlap` event triggers the image actor to update its classification to its original classification as defined in the source CSV file. To help users visualize the cluster area, this class contains a colored plane outlining its boundaries on the ground and a text render object of the cluster name.

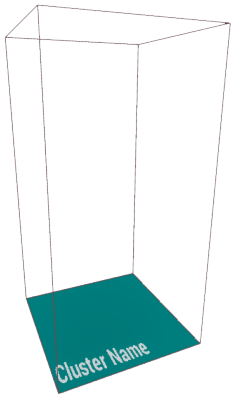
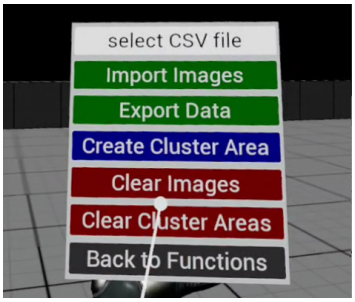


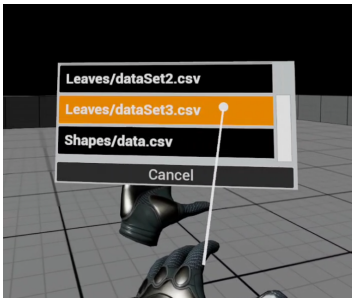
FIG. 6: Cluster area Blueprint.

IV. RESULTS

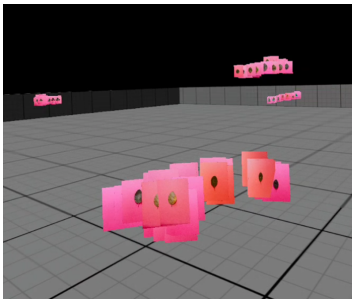
Users can access HITL functions through the menu shown in Figure 7a. They can select a CSV file using the "select CSV file" button shown in Figure 7a; this takes them to the menu shown in Figure 7b, which lists the CSV files in the `HITL` directory and its subdirectories. After selecting a CSV file, users can import the images listed in it by pressing the "Import Images" button in Figure 7a. This prompts the image manager to read the selected CSV file and spawn image actors for each item in the file. Figure 7c shows imported images from the user's point of view. Users can immediately grab the imported images to move them or view their metadata, as shown in Figure 7d. The metadata updates automatically when an image is moved or reclassified. If an image fails to spawn, the entire import is canceled, and the text on the "Import Images" button shown in Figure 7a changes to an error message that states why the import failed. Users can clear all imported images by pressing the "Clear Images" button in Figure 7a. After doing so, they are prompted to confirm this action and warned that data will not be saved. They are given a similar prompt if they try to import images while there are already images in the scene.



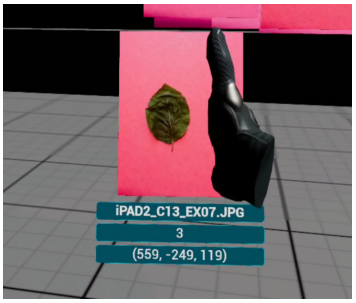
(a) Menu with HITL functions.



(b) CSV file selection menu.



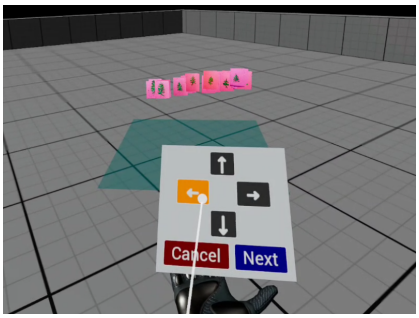
(c) Imported images.



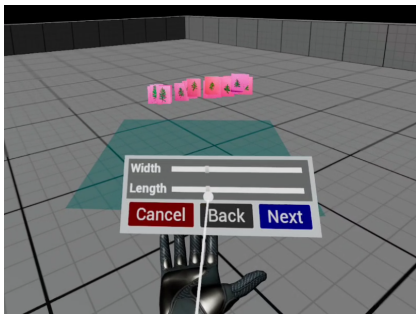
(d) A grabbed image.

FIG. 7: Importing images using the menu.

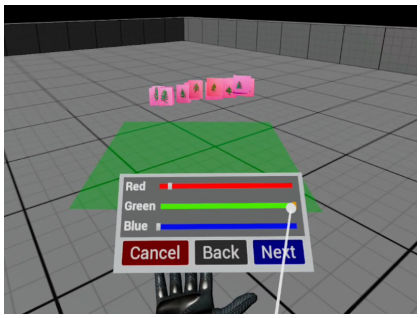
Users can create a cluster area by pressing the "Create Cluster Area" button in Figure 7a. The steps of cluster area creation are shown in order in Figure 8a-d, and Figure 8e shows a completed cluster area. Users can go back a step to edit their inputs or cancel the creation of a cluster area at any point in the process. During creation, the cluster area will trigger any image actors it overlaps to update their classification. However, if the creation is canceled, the images' classifications are reset.



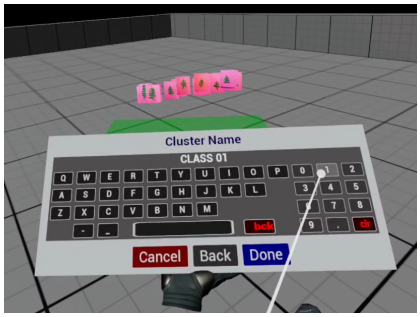
(a) Positioning the cluster area.



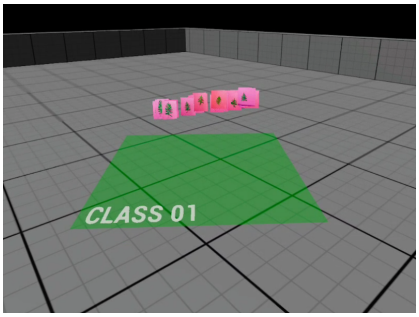
(b) Sizing the cluster area.



(c) Setting the cluster area's color.



(d) Naming the cluster.



(e) Resulting cluster area.

FIG. 8: Creating a cluster area.

After creation, users can manually move images into and out of cluster areas to update their classifications, as shown in Figure 9a-b. Users may clear all cluster areas using the "Clear Cluster Areas" button shown in Figure 7a, which also resets all images' classifications after prompting the user to confirm this action.

(a) Image with original classification. (b) Image placed in cluster area.

FIG. 9: Image reclassification using a cluster area.

Users can press the "Export Data" button shown in Figure 7a at any time. This prompts the image manager to write the current metadata to a new CSV file, which is named with a timestamp. Figure 10a-b shows an imported CSV file generated by the classification script compared to a CSV file exported from the application after a subset of the images was moved and reclassified. If there are no images in the virtual environment when the user attempts to export data, the image manager does not create a new CSV file, and the text on the "Export Data" button changes to "nothing to export" for four seconds to warn the user.

```
iPAD2_C19_EX01.JPG,169.82611,-96.58374,73.14529,4
iPAD2_C19_EX02.JPG,168.61493,-107.30237,73.6191,4
iPAD2_C19_EX03.JPG,171.11897,-98.19336,73.89965,4
iPAD2_C19_EX04.JPG,145.08455,-114.95212,76.60817,5
iPAD2_C19_EX05.JPG,150.29272,-109.365204,74.41318,5
iPAD2_C19_EX06.JPG,101.0,-146.8248,80.349525,6
iPAD2_C19_EX07.JPG,125.26611,-130.38882,77.75633,6
iPAD2_C19_EX08.JPG,128.23593,-128.72568,81.42961,6
iPAD2_C19_EX09.JPG,117.13612,-139.60129,79.54898,6
iPAD2_C19_EX10.JPG,140.5929,-115.85019,78.50252,5
iPAD2_C19_EX11.JPG,112.78169,-137.55774,80.96357,6
iPAD2_C19_EX12.JPG,109.43848,-147.04675,83.050644,6
```

```
iPAD2_C19_EX01.JPG,169.82611,-96.58374,73.14529,4
iPAD2_C19_EX02.JPG,-239.023115,-592.722795,84.569763,CLASS 01
iPAD2_C19_EX03.JPG,-109.9009,-575.168808,90.834594,CLASS 01
iPAD2_C19_EX04.JPG,145.08455,-114.95212,76.60817,5
iPAD2_C19_EX05.JPG,150.29272,-109.365204,74.41318,5
iPAD2_C19_EX06.JPG,101.0,-146.8248,80.349525,6
iPAD2_C19_EX07.JPG,-238.000768,-523.937125,82.378481,CLASS 01
iPAD2_C19_EX08.JPG,-209.886853,-677.614041,83.38612,CLASS 01
iPAD2_C19_EX09.JPG,-241.225572,-705.890488,85.939131,CLASS 01
iPAD2_C19_EX10.JPG,140.5929,-115.85019,78.50252,5
iPAD2_C19_EX11.JPG,112.78169,-137.55774,80.96357,6
iPAD2_C19_EX12.JPG,-271.874794,-540.705804,97.153918,CLASS 01
```

FIG. 10: Input CSV file (left) and output CSV file after reclassifying images (right). Items with changed values are bolded.

A. Performance

We measured the time taken to import images into the ASCRIBE-VR application using Unreal Engine's `Get Accurate Real Time` function, which returns the time in seconds since the application was started.[13] We called this function before and after importing images and calculated the difference to determine the duration of the import. From a set of 300 unique images, we imported subsets of 10, 50, 100, 150, 200, 250, and 300 images. Initially, the images were 1536 x 1024 pixels. However, upon importing 100 images of this size, there was an error in the application stating "Texture Streaming Pool Over Budget," indicating that the images took up too much memory. Because of this, we did not collect data for subsets of over 100 images of this size. Instead, we repeated data collection after downsizing the images to 768 x 512 pixels.

The performance of the application varies with the number of images imported and the size of the images. We imported each subset five times, restarting the application between each import. The average time taken to import each subset is shown in Figure 11. Based on our data, there is a positive correlation between the number of images imported and the average import time. The time taken to import images is important because the application freezes during the duration of the import from the user's perspective. In addition, a mild lag in the application began after importing 150 images, but it became severe after importing 250 images, which may cause cybersickness for some users.

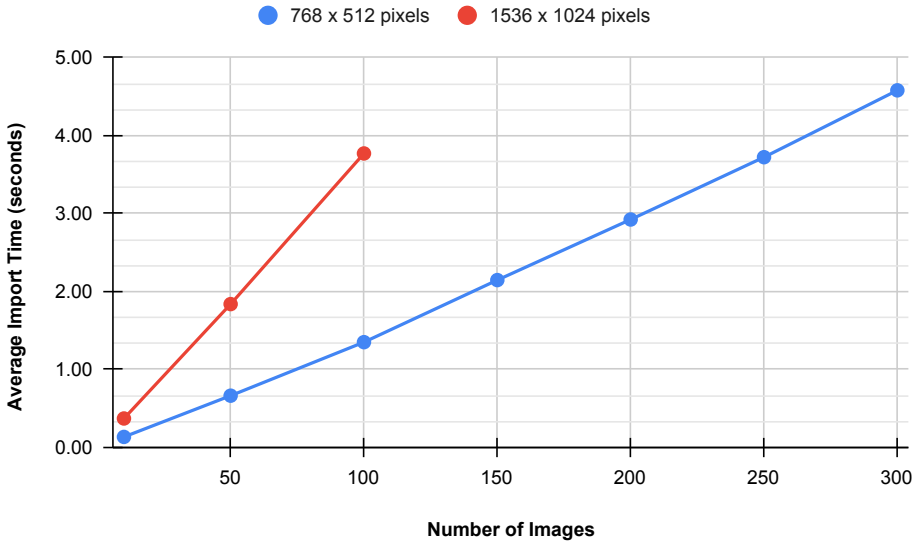


FIG. 11: Average time taken to import images.

V. DISCUSSION

The interface we have developed in this project begins ASCRIBE-VR's expansion into human-in-the-loop functionalities. It allows users to intuitively review image data by taking full advantage of the 3D nature of ASCRIBE-VR. Users can physically grab and move images to sort and classify them, reflecting how humans observe and interact with objects in real life. They can take into account the similarity of images when making decisions by observing the proximity of the images in three dimensions. These functions are especially applicable to machine learning for computer vision research; they enable users to validate image classifications produced by machine learning models and export corrected classification data as a CSV file. Such CSV files are intended to be provided as feedback to the model. Future work may explore ways to effectively integrate these CSV files into the training of machine learning models and other automated systems. In addition to validating image classifications, the interface can be used to perform data quality control and isolate critical data samples.

More broadly, the interface contributes to ASCRIBE-VR's mission to provide a comprehensive platform for research that addresses the limitations of 2D interfaces.[14] It adds another function to ASCRIBE-VR's tool set, enabling the application to address another research-relevant use case. While the development of ASCRIBE-VR and its HITL interface is still ongoing, this project is a major advancement towards ASCRIBE-VR's goal of enabling HITL for experimental validation.

A. Future features

During the next phase of development, we plan to implement several features that will expand the interface for use with more types of image sets:

- Support for importing images in TIFF format, which currently cannot be imported due to limitations of the Real Time Import/Export Plugin.
- Support for importing images that do not have a CSV file that dictates their locations, allowing users to work with unsorted data. To accomplish this, we may potentially integrate the ability for the user to run the classification script in real time through the ASCRIBE-VR application.
- The option for users to import subsets of the images listed in a CSV file, allowing them to sort through large image sets while maximizing the application's performance.
- A function that allows users to select and manipulate multiple images at once.

VI. CONCLUSION

This project introduces a new interface for human-in-the-loop for smart sorting of images, enabling fast overview of large amounts of 2D data and potentially accelerating the process of image annotation for classification tasks in computer vision. The interface enables users to review image data and validate image classifications produced by automated systems within a VR environment. The 3D nature of VR allows users to take into account the proximity of images when making decisions. This makes the interface especially suitable for validating image sets that have been clustered in three dimensions based on their similarity, such as those produced by our machine learning script. After validating and correcting image classifications, users can export the updated data as a CSV file that can be provided as feedback to the originating automated system. Overall, the interface is a significant contribution to ASCRIBE-VR's expansion into HITL, adding another function to its comprehensive tool set. During future development, we will implement features that expand the interface for use with large, unsorted, and TIFF-formatted image sets.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship (SULI) program.

References

[1]X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning", Future Generation Computer Systems **135**, 364–381 (2022).

[2]A. van der Stappen and M. Funk, "Towards guidelines for designing human-in-the-loop machine training interfaces", in Proceedings of the 26th international conference on intelligent user interfaces, IUI '21 (2021), pp. 514–519.

[3]E. Yigitbas, K. Karakaya, I. Jovanovikj, and G. Engels, "Enhancing human-in-the-loop adaptive systems through digital twins and vr interfaces", en, in Proceedings of the 2021 international symposium on software engineering for adaptive and self-managing systems (2021).

[4]*Blueprints visual scripting*, https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine, Accessed: 2025-07-01.

[5]*Blueprint basic user guide*, https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprint-basic-user-guide-in-unreal-engine, Accessed: 2025-07-01.

[6]*Real time import/export plugin*, https://www.fab.com/listings/60321fa8-c82f-436d-a372-4035207f7eb9, Accessed: 2025-07-01.

[7]*File system library*, https://www.fab.com/listings/ec8e92ff-074b-4246-8fd1-e7f2c8510d76, Accessed: 2025-07-16.

[8]*Actors*, https://dev.epicgames.com/documentation/en-us/unreal-engine/actors-in-unreal-engine, Accessed: 2025-07-02.

[9]*Vr template*, https://dev.epicgames.com/documentation/en-us/unreal-engine/vr-template-in-unreal-engine, Accessed: 2025-07-02.

[10]*Widget components*, https://dev.epicgames.com/documentation/en-us/unreal-engine/widget-components-in-unreal-engine, Accessed: 2025-07-02.

[11]*On component begin overlap*, https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Collision/OnComponentBeginOverlap, Accessed: 2025-07-02.

[12]*On component end overlap*, https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Collision/OnComponentEndOverlap, Accessed: 2025-07-02.

[13]*Get accurate real time*, https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Utilities/Time/GetAccurateRealTime, Accessed: 2025-07-28.

[14]G. M. dos Santos, M. Boykin, R. C. Coelho, C. M. G. de Godoy, J. Donatelli, and D. Ushizima, "Immersive morphometric analysis: interactive VR exploration and scientific 3D visualization using the Meta Quest", under submission to IEEE, 2025.

[15]D. Ushizima, G. M. dos Santos, Z. Sordo, R. Pandolfi, and J. Donatelli, "Ascribe new dimensions to scientific data visualization and interactive exploration with VR", DOI:000000/11111, 2025.

[16]*Textures*, https://dev.epicgames.com/documentation/en-us/unreal-engine/textures-in-unreal-engine, Accessed: 2025-07-02.