Creating Immersive Virtual Training Scenarios from 2D Inputs Using Artificial Intelligence

Larry Moralez, Sam Haddad, Katie DelGiudice, and Jennifer Riley

Design Interactive, Inc.

Orlando, FL

Larry.Moralez@DesignInteractive.net
Sam.Haddad@DesignInteractive.net
Katie.DelGiudice@DesignInteractive.net
Jennifer.Riley@DesignInteractive.net

ABSTRACT

Augmented Reality and Virtual Reality (AR/VR) are critical technologies for future training initiatives. However, creating sufficiently rich context and content for these platforms requires a significant cost in terms of time, labor, and technical expertise. Designing the environmental setting, implementing relevant actions to simulate real-world events and interactions, designing, and creating instructional content, and integrating it all within a digital form requires substantial effort. Creating multiple, varied, and useful training scenarios rapidly and repeatedly is therefore a significant barrier preventing widespread adoption and utility in DoD and commercial training programs. Recent advances in computer vision and artificial intelligence (AI) can be used to generate quality training scenarios automatically from 2D video. In this paper we describe the Generative Immersive Scenario Testbed (GIST), an AI driven approach to generate dynamic 3D virtual content from 2D videos. We outline GIST's integrated system of technologies for generating scenarios which recreate events from videos, including entity positions and high-level human behaviors. We describe the challenges and outcomes from our approach against 3 performance metrics: time to complete scenario generation, number of possible scenario permutations, and the efficacy of the generated content (ability to detect, track, and translate critical cues) for training. Our results suggest that AI driven solutions hold great promise for the rapid creation of 3D training content.

Creating Immersive Virtual Training Scenarios from 2D Inputs Using Artificial Intelligence

Larry Moralez, Sam Haddad, Katie DelGiudice, and Jennifer Riley

Design Interactive, Inc.

Orlando, FL

Larry.Moralez@DesignInteractive.net
Sam.Haddad@DesignInteractive.net
Katie.DelGiudice@DesignInteractive.net
Jennifer.Riley@DesignInteractive.net

INTRODUCTION

Government and industrial organizations are increasingly concerned with the impact of large-scale workforce transitions. More specifically, they are challenged with capturing, sharing, and transferring the critical, experience-based knowledge and skills leaving the workforce as personnel retire (Basque et al., 2014). Additionally, with the expansion of the distributed workforce, organizations must be able to build expertise across multiple and potentially distant locations. Experts are traditionally asked to put what they have learned over the course of their careers into documents and slide decks that are used to support classroom-based training, or to support the development of lengthy manuals and procedures. The effectiveness of this type of passive learning can be limited, however. To progressively increase skill in operations requiring decision making and mastery of complex competencies, personnel need to "learn by doing" in representative environments and situations (Schunk, 2012). Learning that supports practice and putting declarative knowledge into action has been demonstrated to increase knowledge and skill transfer. Live, interactive training can be expensive, though; and for dangerous operations, it can be risky.

Training in simulated environments can help to alleviate concerns with the time, expense, and risk of interactive, live training. With the advancement of virtual reality (VR) and augmented reality (AR) technologies, the potential to provide interactive training for sharing and building expertise with novices is vastly increased. A trainee's presence in the virtual or augmented space offers an immersive experience within which to explore and engage with realistic operational settings, instruments, actors, challenges, and situations. However, a pervasive challenge is the time, labor, and technical expertise required to author and create VR or AR scenarios that sufficiently provide rich context and content. The effort to design the environmental setting, implement relevant actions to simulate real-world events and interactions, design and create instructional content, and integrate it all within a digital form is substantial. Doing so rapidly and repeatedly to make multiple, varied, and useful training scenarios is an even greater challenge.

Emerging technologies such as artificial intelligence, specifically object recognition and object tracking, are opening the aperture here. The technologies are maturing for rapidly and directly converting observed reality into digital experiences. Currently, these tools are not integrated within a single application. Instead, they are used to produce related but segregated technical capabilities and digital outputs. But with the appropriate data translation techniques, the detailed data outputs from these tools can be converted into virtual or augmented worlds explored via game engines such as Unity. Data on patterns of human behavior can generate human behavior models (HBMs) and applied to non-playable characters (NPCs) in the scene to generate multiple permutations of the original input. The integrated virtual worlds that can result from these approaches—the environment/setting along with the avatar/animated humans—provides the foundation for training scenarios that are based upon digital captures. Furthermore, it unlocks the potential for automating aspects of the scenario generation process, including the rapid blending or permutation of environmental models and HBMs to create new content. There is an opportunity to consolidate these interrelated capacities.

This paper outlines a technology integration framework that extracts data from 2D videos using AI and translates it into 3D virtual environments using the 3D game engine Unity. We provide a basic introduction to the AI tools employed in the system in the subsequent section. We then detail the inner workings of the system and the process it follows from 2D video input to 3D virtual scene output. Next, results from testing the system against three key benchmarks—time to completion, number of permutations, and object detection/tracking efficacy—are presented to show the potential that the system has at significantly reducing the resources required to develop 3D training scenes. We also present the results from a usability study that evaluated the ease-of-use of the system. Lastly, we highlight the current limitations of the system and discuss how it can be improved by integrating additional technologies.

BACKGROUND

Research in the field of computer vision has advanced at a rapid pace in the past decade. Two key areas of interest within this field are computer automated object detection and object recognition. These tools allow for the extraction of data from 2D images over time, data that can be used as input into a 3D game engine such as Unity to re-create 3D virtual experiences that mimic the original video. In this section, we provide a brief introduction to these tools and follows with their integration into in the subsequent section.

Object Recognition

Object recognition is a computer vision technique whereby computer algorithms detect and categorize objects contained in an image or video. It is an extremely important task in computer science and plays a crucial role in many real-world applications such as surveillance, biometric recognition, content-based image retrieval, medical analysis, industrial inspection, robotics, and smart vehicle systems. (Dev Singh, Mittal, Bhatia, 2018; Rani & Gupta, 2019). Such algorithms typically consist of artificial neural networks that are comprised of a series of matrices that are trained on a set of images (Hassoun, 1995). These matrices are tuned at each step of the training process, typically several thousand times, until they can accurately detect (is there an object and where is it) and categorize (what type of object is it) the objects to satisfactory performance.

There are, however, several key difficulties in facing the process of object detection (Rani & Gupta, 2019). One is the quality of the image being detected. Images vary widely in their positioning, framing, resolution, contrast, and lighting. While detecting objects in these types of images may be relatively simple for humans, they can make it difficult for the algorithms to accurately extract the appropriate features for prediction. Another key problem is that of obstruction: images may only partially display the object, occluding crucial features that the algorithm depends on for detection. Fortunately, object detection and recognition has seen a significant increase in research interest as computer vision hardware and sensors continues to improve. There remains a strong demand for novel recognition systems that increases the speed, accuracy, and number of objects that can be detected, and is therefore an area of research that is likely to see major advances in the future. As these algorithms advance in their efficiency, we are likely to see wide-spread adoption across a larger set of industries.

Object Tracking

Object tracking is an extension of object recognition and is typically used to track multiple objects in a video over time. Early object tracking algorithms were designed for Single Object Tracking, where the target object is known a priori (Ciaparrone et al., 2020). However, many videos contained multiple objects that had to be differentiated and tracked simultaneously as they leave and enter the scene. Therefore, Multiple Object Recognition (MOR) was developed to overcome these problems. MOR algorithms oftentimes track pedestrians, vehicles, sports players, and animals simultaneously, and can do so despite high variability in video quality and object occlusion (Xu et al., 2019). This is achieved by what is referred to as tracking-by-detection, whereby bounding boxes are used to define the area containing the object and associations are formulated between the frames to continuously track the object over time. The majority of the MORs consist of the following stages (Ciaparrone et al., 2020):

- Detection Stage: Object detection algorithms detect the existence of an object in the scene and then categorize them.
- Feature Extraction and Motion Prediction stage: Feature extraction algorithms analyze the frame to detect appearance, motion, and interactions.

- Affinity stage: features and motion predictions are used to create a similarity score between frames
- Association stage: Similarity scores calculated in the Affinity stage are used to associate detected objects in frames and IDs are assigned to detections of the same target.

Object tracking algorithms train the object detector prior to being used in object tracking and are therefore constrained to detecting and tracking a small set of targets, typically pedestrians, vehicles, and various animals like cats and dogs (Luo et al., 2021). As such, the performance of the object tracking system is constrained by the efficacy of the object detection algorithm it employs. If the object detector struggles to effectively identify and categorize targets consistently, it will perform poorly as an object tracker. However, detection-free tracking also exists and allows the user to initialize the objects to be tracked at the onset of tracking. Detection-free trackers can be used in cases where the objects that need to be detected were not objects that the object detector was trained to identify, though they can run into problems when new objects in the scene appear. Nonetheless, as research into object detection and recognition advances, so too will the efficacy of object tracking.

Video Game Engines

Video games have been around for a half a century, beginning with Spacewar in 1961 (Anderson et al., 2008). When the industry was young, developers created games from the ground up with little to no reliance on pre-defined game engine architectures and reusable components. Today, however, many games are created with 3D game engines environments such as Unity, Unreal, Gadot, and GameMaker Studio. These environments feature user interfaces that are integrated with various software programming tools and a 3D view of the virtual space being developed, allowing developers to build games with a suite of visual development tools. They offer the developer core functionalities such as graphic rendering, physics engines, collision detection, audio, and animation that do not have to be generated from scratch. These tools allow the user to work within an integrated development environment and enables easy to use, rapid development of games using state-of-the art technologies.

GENERATIVE IMMERSIVE SCENARIO TESTBED

The GIST system was conceptualized to be a single integrated software ecosystem whereby data (object identification, position, trajectory) from 2D video recordings were extracted and translated into 3D virtual environments. The main goals of the system were to significantly reduce the time and resources required to generate immersive VR training scenarios and quickly generate novel permutations. This section describes the process for doing so.

Data Selection and Input

The first step in the data acquisition process is to identify a video scene that visually represents the actors and training cues needed for the learning objective. This video will serve as the main data input to the system. The user then has the opportunity identify objects that the system should track if the object detection algorithm cannot identify it. This detection-free tracking is useful in the case that the entities in the scene are not entities that the algorithm has been trained to categorize. Moreover, in many cases video quality and occlusion can lead to non-detection, and this manual input affords a work around in these conditions. In some cases, again perhaps due to poor video quality or object occlusion, the user can reidentify objects that the system has lost.

Metadata Extraction

Following data acquisition, the scenario re-creation process begins with metadata extraction which gathers information about the objects in the scene, their types and locations, and begins the translation into 3D coordinate space.

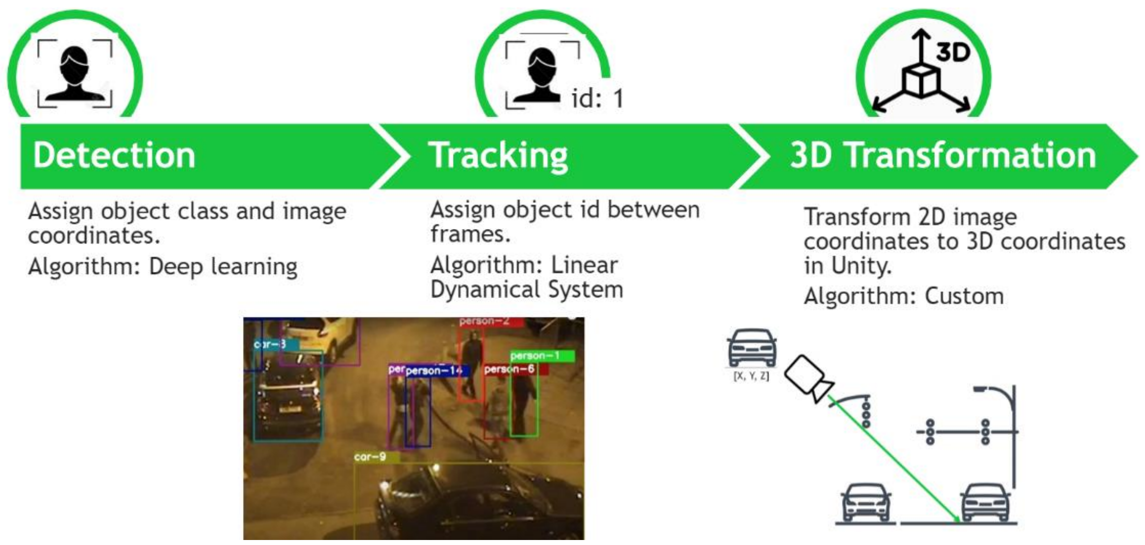
id: 1
3D
Detection
Tracking
3D Transformation
Assign object class and image coordinates.
Algorithm: Deep learning
Assign object id between frames.
Algorithm: Linear Dynamical System
Transform 2D image coordinates to 3D coordinates in Unity.
Algorithm: Custom
car-3
person-2
person-1
person-14
person-6
car-9
[X, Y, Z]

Figure 1. Metadata Extraction Stages and Components

Extraction of metadata consists of the application of several machine learning tasks. The current pipeline contains tools which are focused on extracting the object type using a pre-trained object detection algorithm, YOLOv3. This real-time object detection algorithm can detect 20 different objects. It divides an image into cells, or bounding boxes (Figure 1), and computes the confidence that an object is contained within it. It then categorizes the object based on the 20 objects it's been trained to detect. YOLOv3 is used in concert with object tracking to extract the trajectory of the objects in the video over time. Object tracking is tackled in two different ways in the current pipeline. The first approach is centered on extracting all possible information for every type of object. The benefit of this approach is it does not require the user to specify which objects to track. The downside of this approach is it is limited to tracking objects that have been seen in previous datasets or scenarios. In many cases, content creators are interested in tracking unique objects or only a few select objects in a video. For this case we have developed a second approach that allows the user to manually select objects in a scene by dragging bounding boxes over them before and during the metadata extraction process. This second process allows the system to track any item at any fidelity but requires more input from the content creator.

Once the source video is processed using object tracking, their positions are provided frame-by-frame in the 2D image space relative to the canvas size, using pixels as units. Additionally, the objects are categorized as *entities* and given a unique ID and type categorization. For example, a detected entity may consist of an *ID* of 2, and a *Type* identification of *Person* (Figure 2). This data is compiled into a JSON format as a single file to be parsed by a 3D content generator such as Unity.

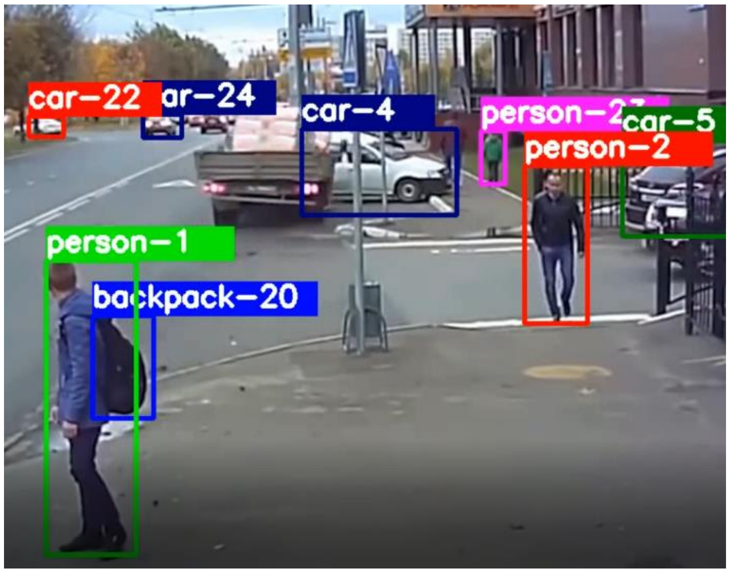

Figure 2. Sample Source Video Processed for Entities (Type followed by ID)

Scenario Generation

After the source data has been processed into the JSON file format, the next stage in the pipeline is executed. This processing stage collects the library of entities and their associated information into a data model that is programmatically designed for use in real-time applications such as serious games or XR. This includes processing the entities into a local database that includes some original source data such as the ID and the entity type. This is then used to track the entity's state throughout the re-creation process. Because the source material is 2D relative to a camera-sourced resolution (e.g., 1280x720 for a 720p video capture), the 2D positions need to be 'reproduced' in a 3D environment that is contextually relevant to the trainee. This presents a significant challenge, given the dynamic nature of the re-created environment. For example, if the source material was recorded in an empty city block at dawn, but the instructor would like to re-create the scenario in a fully populated city during rush hour, the desired scenario destination does not necessarily match the source material. Additionally, the source material is often missing the 3rd depth dimension if it was not captured using a LiDAR or similar depth sensor device.

We developed a unique pipeline (Figure 3) to address these challenges and have prototyped several techniques to accurately recreate most of the scenario in 3D dynamic environments using 2D source material. The main technique of this pipeline is to translate the 2D coordinates into 3D space using a similar positioned virtual camera and ray casting, The pipeline repeats this process for all the detected coordinates, and velocities are calculated in 3D space using the distance between points (delta position) / the change in time (delta time). Once all this is processed, calculate and stored, the scene is now ready to be mapped to AI behaviors that will re-create the essential elements of the original 2D media.

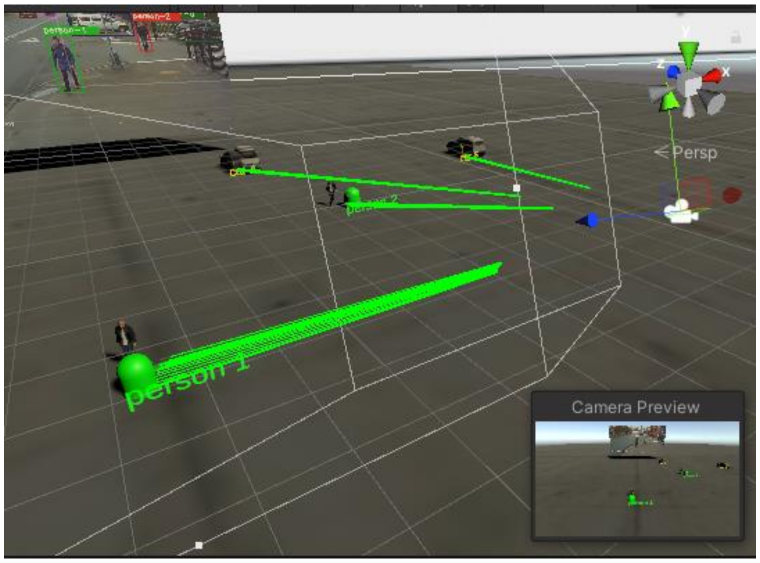

Figure 3. Pipeline to re-create detected objects into dynamic 3D space in real time. The grey cone represents the camera viewing frustrum and the near clip plane. The green lines represent ray casts into the 3D space. They originate from the virtualized camera's 'pixel' coordinates as represented by the original source material capture position.

Dynamic Re-creation and Scenario Permutation with AI Behaviors

The original source data that is now processed and recreated in 3D essentially captures the essence of the original content and can be immediately experienced in select HWDs such as the Oculus Quest 2. However, the scene remains sterile considering the possible training use cases that the intended source video was intended to reflect. For example, if the original source material contains a depiction of a car accident on a busy intersection at rush hour but the user intends to use the scenario for after dark training, different behaviors and bystander reactions that more accurately reflect the context would be desire.

By 'upgrading' the generated entities to be AI agents and adding AI behaviors while still maintaining their original source behavior and movement, the system can take the original scenario and reproduce novel permutations with exponentially increasing environments and contexts. The AI agents can use known algorithms such as pathfinding and NavMeshes to decide, based on their source material and desired destination, the ideal path to take to get to its desired endpoint in whatever dynamic environment the scene is recreated in. In final versions of the system, the user may be able to specify entities that they want to remain true to the source video, while selecting AI behaivors for other NPCs to add context relevance.

AI behaviors such as flee, flocking, and pursue can all be applied to these newly empowered agents, that allow them to react and decide what behavior should be taken given the context of the recreated scene. For example, if the car accident was recreated at rush hour on a busy city block, those agents in proximity to the crash should trigger the 'flee' behavior to avoid getting injured. Agents not in direct proximity but in the same general area should have their attention brought to the car crash. Other more 'curious' agents may decide to walk towards the crash to help/intervene. Many other agents may simply continue with their day. By taking the final step of processing entities into AI agents with rich behaviors, the system can recreate the scene in innumerable contexts and scenarios, with little to no intervention by the instructor. If desired, a UI can be used in later iterations to provide content creators with some control over AI entity behavior to adjust the difficulty of the scenario (e.g., causing additional bystanders to flock to the scene of an accident, frustrating rescue and scene control efforts).

Several core technologies (Table 1) were integrated into the GIST system. The integration of these technologies allows for the extraction of scene information that is then reproduced within the 3D game engine Unity. The following section details the integrated system's performance against several key metrics that highlight its potential to aid in the development of 3D virtual training scenarios.

Table 1: Key Technologies used in the GIST testbed process architecture

Technology	Description
Object Detection/ Recognition with Yolov3	Provides object detection and recognition for populating the 3D scene with the appropriate objects (e.g., identification of discrete objects and labeling of each by type and unique ID); no additional training necessary for new objects
Recording of Information Model for Scenario Data with JSON	Records and stores data required for 3D scene re-creation, including, for each object in the scene, a unique ID, detected type, starting point, movement waypoints, and end point; this file can be passed to a game engine for scene re-creation
Scene Extension and Dynamic Adaptation with Finite State Machines	Leverages an AI behavior library, created in Unity, that allows for each object to exhibit dynamic behavior based on the characteristics of the environment and activities in the scenario; these behaviors can be used to extend the scenario upon completion or to change the initial design of the scenario without requiring the instructor to code relevant object behaviors
3D Scene Delivery with Oculus	Supports 3D scene viewing by using the Oculus VR headset; as the scenario is created using the Unity game engine, it can be exported and viewable on different VR/AR headsets

TESTING AND RESULTS

There were three key performance metrics for the system: time to completion, automated permutation generation, and content efficacy. Time to completion was identified as a key goal of the system due to the extensive time typically associated with created virtual training scenarios. It was therefore crucial that the system be able to go from the initial user video input to completed virtual scenario in less than 5 days. Another crucial requirement for the system was that of automated generation of permutations of the original scene. This affords the system the ability to generate novel training scenarios that differ from the initial video such that they can meet the varying instructional needs of the user. The final key metric was content extraction efficacy. Content efficacy refers to the system's ability to accurately detect and translate entities in the training scenario to the virtual training scene. This ensures that the system delivers on its promise to produce fruitful training scenarios that faithfully represent the entities of the original scene.

Time to Completion

Given the large amounts of time typically needed for standard VR training scene development, one of the key performance goals for the ecosystem was the automatic creation of 3D immersive scenarios based on 2D input (videos or images) within 5 days. Results from testing on a use case video of an arrest at a hospital are scene in Table 4. The total length required for the system to analyze the 40 second video was 8.15 Minutes, or roughly 12.23 minutes for every minute of video.

Video Length	FPS	Time to Completion
40 Seconds	15	489 Seconds

Time to translate analytical results into a 3D scenario in Unity was negligible, even when applying AI-based entity behaviors. Time to integrate results into AR/VR technologies is estimated at 1 hour.

Generating Permutations

The second key performance goal for the ecosystem was rapid permutation or auto-generation of content from a single source video. More specifically, the goal was to be able achieve 1-100 permutations of content from a single 2D video. Scenario permutations of 1-10 were reliably generated from a single input by: Modifying the entities GIST tracks from the original video; Using game AI to alter the number of entities in the scenario or modifying the position and/or behavior of tracked entities; Using game 3D models to modify/enhance the virtual scenery. This allowed for virtually an infinite number of VR permutations of the original source video to be generated.

Content Efficacy

The final key performance goal was to demonstrate efficacy of faithfully replicating the original 2D source video as an immersive 3D scenario for training with altering permutations yet to be applied. Entities associated with critical cues for situation awareness and decision making can be automatically detected, tracked, and translated into the 3D scenario. However, due to current limitations of object detection capabilities (can only categorize 20 different entities), the generalizability of the system to different types of videos with more nuanced visual cues remains an open question. This concern is discussed more in the following section.

Usability Evaluation

Preliminary UI wireframes for GIST were reviewed by six participants, each matching a potential user group for GIST: 1 VR engineer/programmer, 3 UI/UX designers, 1 instructional systems designer, and 1 SME in the law enforcement domain. Participants were first given a walk-through of the UI wireframes and provided an opportunity to ask questions as well as provide feedback on each wireframe. Following the wireframe walkthrough, participants completed a heuristic review of the wireframes and answered a system usability questionnaire.

To complete the heuristic review, participants responded to a worksheet with 49 heuristic items from the Xerox/Neilsen 13 Usability Heuristic set. This set includes multiple items for each evaluation category and is one of several heuristic sets that are commonly used to evaluate computer software and VR systems. Heuristic items from the Xerox/Neilsen 13 Usability Heuristic set were down-selected to those relevant to the GIST wireframes and included items in the following categories:

- Visibility and System Status (16 items)
- Match Between System and the Real World (3 items)
- User Control and Freedom (6 items)
- Consistency and Standards (9 items)
- Help Users Recognize, Diagnose, and Recover from Errors (2 items)
- Recognition Rather than Recall (9 items)
- Aesthetic and Minimalist Design (4 items)

Participants responded to each item set on a scale of 1 (Strongly Disagree) to 5 (Strongly Agree). The complete heuristic set is provided in Appendix A. The average for each category is provided in Table 2.

Table 2: Participant Results from Heuristic Review

Heuristic Category	**Average Score (Maximum of 5) and Std. Deviation**
Visibility of System Status	**4.48** (SD = 0.27)
Match Between System and the Real World	**4.33** (SD = 0.14)
User Control and Freedom	**3.92** (SD = 0.53)
Consistency and Standards	**4.67** (SD = 0.25)
Help Users Recognize, Diagnose, and Recover from Errors	**3.00** (SD = 0.00)
Recognition Rather than Recall	**4.74** (SD = 0.25)
Aesthetic and Minimalist Design	**4.74** (SD = 0.25)

Items averaging lower than 4.0 in each category indicate areas of improvement for the GIST UIs in additional development phases. Items meeting this criteria in the "Visibility of System Status" category included "Response times are appropriate to the task." Users indicated that they would like to be kept informed of the system's progress, especially for longer operations. To address these items, progress bars will be provided in all areas where operations are expected to take longer than 1 minute. When operations are expected to take between 10 seconds to less than 1 minute, a “busy” indicator icon will be used to indicate system processing time. The only other items meeting the less than 4.0 threshold criteria were in the “User Control and Freedom” category and included “Users are prompted to confirm commands that have drastic, destructive consequences” and “There is an undo function at the level of a single action, a data entry, and a complete group of actions.” To address these concerns, future versions of the GIST UI will include the ability to undo actions at each step of the workflow, as well as confirmation of each deletion and undo action when the user requests them.

A 10-item System Usability Scale (SUS) was used to capture quantitative ratings on aspects of the usability of a system or interface (Brooke, 1986). Participants provided a response to each item on a scale of 1 (strongly disagree) to 5 (strongly agree). Scores over 60 represent high usability, with scores greater than 80 preferred. Results for each user group are provided in Table 3.

Table 3 Participant Results from SUS Questionnaire

User Group	**Average SUS Score (>60 desired)**
VR Engineer/Programmer	**60**
UI/UX Designers	**92**
Instructional Designer	**93**
SME	**70**

The lowest SUS score was given by the VR Engineer/Programmer. The main concern was that a non-technical end user may not understand what is occurring and the consequences of their actions at each step in the workflow. They desired much more transparency and assistance at each step of the workflow to help the user decide the best course of action. Nonetheless, the overall usability of the system was viewed favorably and indicates that the system is likely to be highly usable regardless of the complexity of the technologies used in the system.

CHALLANGES

One key challenge for the system is that which commonly plagues many AI driven solutions: inferring the user’s intentions. AI is great for a wide array of tasks but remains poor at inferring what to do with the outputs (Pearle & Mackenzie, 2018). As such, the system outlined in this document currently struggles with the task of automating learning objective specific instructional content into the 3D scenarios. A video used as input for the system could potentially have myriad learning objectives associated with it, and it is difficult for the system to infer why that video was selected and what training objectives could be associated with it. This becomes a larger issue as the system creates outputs that are permutations of the original video, as a learning objective associated with the original video may lost in a permutation. For instance, training a law enforcement officer to conduct a traffic stop during the nighttime would require attention to a different set of sensory cues to be trained for than during daytime. Therefore, it is not likely that the system will be able to automate this content from the video input alone and would instead need to be input by the user. Human-centered AI provides a potential process for developing solutions to this problem by providing the user the opportunity to make key edits that add relevant context to the scene. This leads to another key challenge, that of user control. The system currently exists as a functional prototype but will require a highly intuitive and frictionless interface that guide the user from 2D video input to 3D virtual scene output. What’s more, it can be assumed that non instructional system design (ISD) professionals will use the system, therefore the system may also need to guide the user to outcomes that are consistent with crucial ISD principles. A user-centered design process can be employed here to ensure that the system is easy to use while providing maximal value to the various users.

Accurate recreation of the environment from the video scene also presents a significant challenge. As of now, the system cannot recreate the environmental layout, nor the visual details associated with it. Instead, objects that are detected in the video and placed in the virtual environment are not direct copies of the video objects, but rather are pre-defined digital assets that are place according to the location and trajectories extracted from the video. Photogrammetry, laser scanning, ambient sensors, and crowd-sourced 3-D modeling are emerging technologies that can be implemented in future iterations of the system to overcome this obstacle. However, limitations with current object detection algorithms will continue to limit the number objects that can be detected, entailing that crucial training cues may be missed and subsequently leading to poor training outcomes and field readiness. This concern will be mitigated as object detection algorithms continue their rapid advances. Additionally, the user-centered design process may reveal a way in which the user can quickly insert training cues that currently cannot be detected by state-of-the-art object detection algorithms.

CONCLUSION

The GIST system is a functional proof-of-concept prototype that is capable of converting 2D videos to 3D virtual environments. Leveraging a blend of state-of-the-art technologies—object detection and tracking, Kalman filters, finite state machines, video game engines, and VR head worn displays—the integrated technological ecosystem will significantly decrease the time to generate novel training scenarios. Future iterations of the system can be improved by integrating photogrammetry, laser scanning, ambient sensors, human behavioral modeling, and crowd-sourced 3-D modeling. What's more, advances in object detection and tracking will improve its performance further. This technology has the potential to rapidly accelerate the adoption of immersive training and learning applications, facilitating a transition to a novel learning paradigm suited for the 21st century.

References

Anderson, E. F., Engel, S., Comninos, P., & McLoughlin, L. (2008). The case for research in game engine architecture. In Proceedings of the 2008 Conference on Future Play: Research, Play, Share (pp. 228-231).

Basque, J., Paquette, G., Pudelko, B., & Léonard, M. (2014). Collaborative knowledge modelling with a graphical knowledge representation tool: A strategy to support the transfer of expertise in organisations. In Knowledge cartography (pp. 491-517). Springer, London.

Ciaparrone, G., Sánchez, F. L., Tabik, S., Troiano, L., Tagliaferri, R., & Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. Neurocomputing, 381, 61-88.

Hassoun, M. H. (1995). Fundamentals of artificial neural networks. MIT press.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T. K. (2020). Multiple object tracking: A literature review. Artificial Intelligence, 103448.

Pearl, J., & Mackenzie, D. (2018). AI can't reason why. Wall Street Journal.

Rani, M., & Gupta, E. A. (2019). Recognition and Detection of Multiple Objects from Images: A Review. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 8(12), 524-527.

Schunk, D. H. (2012). Learning theories an educational perspective sixth edition. Pearson.

Singh, R. D., Mittal, A., & Bhatia, R. K. (2019). 3D convolutional neural network for object recognition: a review. Multimedia Tools and Applications, 78(12), 15951-15995.

Xu, Y., Zhou, X., Chen, S., & Li, F. (2019). Deep learning for multiple object tracking: a survey. IET Computer Vision, 13(4), 355-368.