2014

A software framework for initializing, running, and maintaining mixed reality environments

Kenneth Edward Kopecky II
Iowa State University

A software framework for initializing, running, and maintaining mixed reality environments

by

Kenneth Edward Kopecky II

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Human-Computer Interaction

Program of Study Committee:

Eliot Winer, Major Professor

James Oliver

Stephen Gilbert

Song Zhang

Akhilesh Tyagi

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

ABSTRACT

Immersive environments have become increasingly important in the past decade in areas such as training, data visualization, and even entertainment. While terms such as virtual reality (VR), mixed reality (MR), and augmented reality (AR) are often used to describe different types of immersion, it should be noted that these technologies exist along a continuum defined by level of immersion (Milgram and Kishino 1994). VR is at the most immersive end of this continuum, consisting of experiences where most or all of the user's field of vision is replaced by computer-generated imagery (CGI). At the other end, AR refers to text or images overlaid onto the the user's view of the world, such as the heads-up display of an airplane. The user is immersed only in the real world. Finally, MR is the term used for immersive environments that can't be clearly defined as either AR or VR. An example of MR discussed several times in this article is that of a military training environment consisting of physical sets and props. Displays mounted inside windows and doors show 3D images of the larger, virtual world in which the training takes place. To make MR work, many technologies, such as computer generated (CG) graphics, positional tracking, and input devices are all required. Graphics typically consist of animated 3D models and special effects representative of what a trainee would encounter in an actual situation. They can be front- or rear-projected onto screens or use video displays for presentation to a trainee. Each display can be connected to the primary computer running the simulation, or an application can be clustered, in which multiple computers each generate graphics for connected displays. Position tracking, or just "tracking", is a term for the real-time collection of data specifying the position and

orientation of objects involved in a simulation. Tracking data usually describes all six degrees of freedom (DOF) of an object in space, but for certain situations, only three DOF are necessary, such as an object that rests on the ground and only moves along two axes while rotating about a single axis. This data can be acquired in many ways. Trackers can use image analysis, electromagnetic fields, solid state accelerometers and gyroscopes, ultrasonic bursts, or a combination of these to determine an object's position and orientation. Finally, input devices act as a proxy for the real tools a trainee would use on the job. Gamepads, smartphones, and simulated weapons can stand in for switches, control panels, or real weapons. These devices send signals back to the computer(s) running the simulation describing their state (e.g., if a button is pressed) and can be tracked if their position and orientation are relevant to how their real-life counterparts are used.

One particular type of simulation paradigm called LVC (Live, Virtual, Constructive), involves live participants interacting with virtual players (i.e. on laptop computers), as well as computer-simulated (constructive) players. When LVC training is combined with VR and MR, a new type of simulation paradigm called MRVC, (Mixed Reality, Virtual, Constructive) emerges. MRVC training allows live participants to be immersed in a mixed-reality simulation while other trainees can can interact with the simulation in a less immersive manner. MRVC often includes physical objects that, to play their proper roles in a scenario, must have their position known by the computers running the simulation.

Bringing all the information gathered from input devices and tracking systems requires many programs to work together. This often happens behind the scenes, so that simulation programmers are spared the trouble of configuring their programs to work with the specific devices at hand. However, this approach makes diagnosing problems much more difficult. For example, if a trigger is pulled on a simulated rifle and there is no visible response in the virtual world, the problem may lie in several places such as: the electronics of the simulated rifle, an object occluding the view from a camera of an optical tracking system, a failure in the software driver for the trigger's signal, or a mistyped line in a configuration file. Typical VR systems often have only two devices other than the image generators: a head tracker and an input device (e.g., a wand with buttons or a gamepad). When problems occur, they can be challenging to diagnose and solve, but there are a limited number of components in which the fault may lie. MR systems, especially when part of an MRVC system, can use multiple tracking systems, input devices, and displays. More systems working together increases the number of possible failure points, making it more difficult to fix runtime problems.

This dissertation focuses in particular on mixed reality training in reconfigurable environments. Environments such as these consist of movable props, such as walls and doors, that can be rearranged to form different buildings or rooms. An endless number of training scenarios can be theoretically created by moving the props into different arrangements. However, this configurability must also be present in the simulation. Often, 3D artists must re-create physical objects for the scenarios using 3D modeling tools. The real-life objects must then be positioned to precisely match the 3D models.

Changing the scenario then means changing the type, position, and orientation of all 3D models. This is a time-consuming process that usually involves exporting data from a modeling package and converting it to a form that the simulation software can understand. This barrier to reconfiguration significantly decreases the ease with which new training scenarios can be created.

To address these complex problems, a set of tools was developed to more clearly show the flow of data in the MR environment. These tools have made it much simpler to diagnose and correct problems that crop up in mixed reality environments, as well as VR and AR systems. The Mixed Reality Toolbox (MRT) is composed of four tools: GadgetProbe, MetaTracker, Launcher, and Overview. All four of these tools provide "inspection portals" showing the state of simulation computers, devices and tracking systems at different places in a simulation data flow. In addition, MetaTracker and Launcher help to reduce the complexity of specific aspects of distributed mixed reality simulations: MetaTracker simplifies the use of multiple tracking systems while Launcher eases the task of reconfiguring and launching cluster-based applications.

The GadgetProbe component of MRT was built to visualize data relating to hardware configuration and input device state (e.g. positions, orientations, and button presses) at the last stage of the simulation data flow: the immersive application itself. GadgetProbe acts as an extra image generator node for a cluster-based application. It receives its input data after it has been processed by the VR software framework, allowing the simulation

operator to verify that data from a given device is showing expected values and is arriving through the “channels” expected by the application developer.

In simulations that requires trainees to move through walls and rooms, 3D tracking of participants can be challenging for several reasons, such as occlusion from walls, a large number of objects to track, and a need for a wide area of coverage. Using multiple tracking systems is an effective way to get around these issues by concentrating tracking coverage in areas where it is the most difficult. However, it also adds complications, such as using the correct hardware drivers and application programming interfaces (API) with each system, and selecting which system’s data to use for a given tracked object at a particular moment in time. The MetaTracker component of MRT was designed to abstract the task of integrating multiple tracking systems into a mixed reality simulation by allowing the separate systems to all be treated as a single tracking system. MetaTracker also provides a window for viewing the tracking data it processes, allowing the operator to verify that each system is working as expected.

Another issue associated with MRVC systems is cluster and device selection. Different training scenarios may use different input devices or different nodes in a graphics cluster. If a clustered application crashes (or even exits properly), copies of that application might remain running on various nodes. MRT’s next component is Launcher, a graphical user interface (GUI) for application configuration and launching. Launcher allows a user to quickly configure and launch an application on specified cluster nodes with different input devices and tracking systems. Once the application is launched, the

user can view runtime log files on the various nodes. Finally, after finishing with the application, he or she can confirm that the application was properly terminated on each cluster node.

While viewing of tracking and device data is useful for verifying device operation and monitoring a simulation as it runs, the data provided by MetaTracker and GadgetProbe is very abstract. For example, tracking data is displayed as numbers and as an arrow moving over a grid. The final component of MRT is Overview. Overview combines data from MetaTracker and GadgetProbe with 3D models, using it to reconstruct a view of the simulation that shows the relationship of virtual and real objects, independent of the particular application being run. Evaluation of MRT consisted of running VR/MR applications both with and without MRT. MRT was found to greatly speed the process of application launching and configuration, as well as reduce the time needed to isolate run-time problems compared to running applications without using MRT's tools. Two components of MRT also played a key role in two user studies: MetaTracker, by allowing a large area to be tracked that otherwise could not have been, and Launcher, by enabling the operator of a study to quickly switch configurations of an immersive application.

In addition to a qualitative evaluation of MRT's components, MetaTracker was qualitatively evaluated in two areas. The first of these was the amount of latency introduced when data was sent first to the MetaTracker server and then to the simulation application. When tracking 10 or fewer objects, MetaTracker was found to introduce

under 2 ms of latency compared to the case of a tracking system communicating directly with the simulation computer, and 3.5 ms when tracking 25 objects.

The second quantitative evaluation of MetaTracker related to situations in which two tracking systems were tracking the same object, but reporting different positions due to calibration error. MetaTracker's original behavior in this situation was to simply report last position received from either tracking system. This resulted in objects appears to jump back and forth erratically, a behavior referred to as "jitter". To reduce jitter, MetaTracker has an error resolution system that filters out data from all but one tracking system in a situation such as this. Testing with in cases with two tracking systems showed that MetaTracker's error resolution reduced the appearance of jitter by 70% in the worst case, and 91% in the best case.

CHAPTER 1 - INTRODUCTION

Technology-aided Training

In today's society, training is required for more and more jobs. Whether painting products on an assembly line, fighting fires, or performing surgery, a large amount of training is needed to prepare. Many times, training involves expensive consumable materials. This is evident in the case of a fire fighter, where training often takes place in "burn buildings": structures where fires are started, fueled either by the building itself or by gas or other combustibles (Hodge 2003). A second example is training soldiers (warfighters), which accompanies the consumption of expensive munitions, fuel, and targets (Shufelt Jr. 2006). To reduce the financial and labor cost of training, many companies and organizations have turned to CGI-based (Computer-generated Image) simulation technologies, such as virtual reality (VR) (Brooks 1999), augmented reality (AR) (Van Krevelen and Poelman 2010), and mixed reality (MR)(Costanza, Kunz et al. 2009). It is impossible to cleanly differentiate between these three categories, and many

Figure 1: Augmented reality involves overlaying data onto our view of the real world. Photo Credit:(Google 2013)

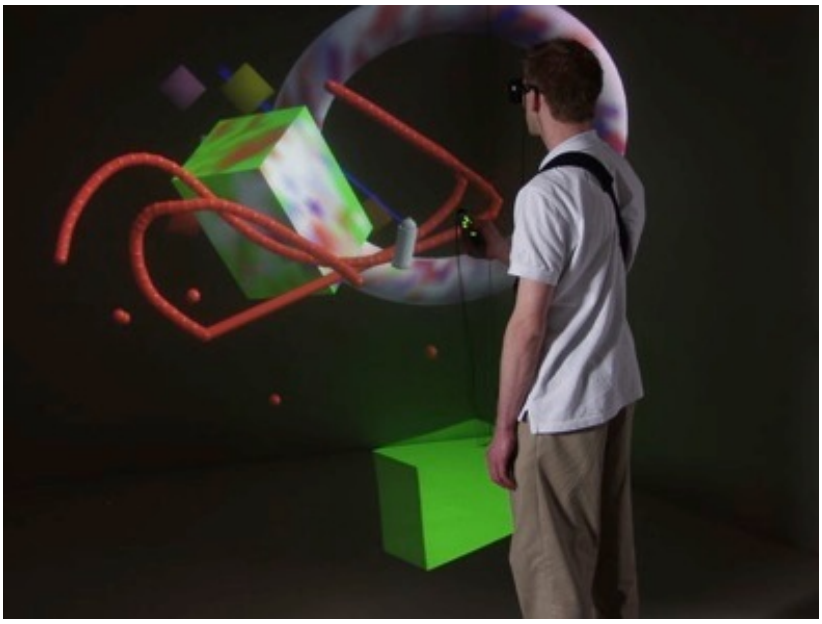

Figure 2: Virtual reality immerses the user in a virtual world.
Photo Credit: Kevin Teske

of the works cited here offer overlapping definitions, but we must attempt to do so. Working definitions for these technologies are as follows: AR is considered to be images or data overlaid onto the user's view of the real world, as in Figure 1. Virtual reality, seen in Figure 2, is considered to be instances where all or nearly all simulation-related visuals are computer-generated, with the exception of input devices, and take up all or nearly all of the user's visual field. Additionally, input devices are usually actual or simulated versions of their real-life counterparts, as opposed to a keyboard or mouse. Finally, MR simulations are defined as those where the real world is extended and enhanced by computer displays depicting virtual elements of the simulation. More specifically, these virtual elements appear to occupy positions in the real world and can even be composited into real world props including walls, doorways, and other physical objects.. In Figure 3, the tablet computer shows an image of a helicopter overlaid onto the real world. The

Figure 3: Mixed reality uses displays such as this tablet computer to add virtual content to the real world.

helicopter has a specific position in real 3D space, and its representation on the tablet is adjusted to make it appear at that location.

How VR can Benefit Training

In many areas where a significant part of training expense comes in the form of consumable materials, VR training can provide an immersive experience that simulates, to a certain degree, the desired situation. Virtual spray painting, for example, can combine a real input device (a modified spray gun) with a virtual part on a stereo display. As in Figure 4, a virtual part responds to the paint gun as a real part would, by changing color, but no paint is wasted, no toxic paint fumes are released into the air, and no materials need to be cleaned, sanded, and re-primed (VRSim 2012). For firefighters, virtual buildings can be modeled and navigated in a virtual environment (VE) while

participants fight a numerically-simulated fire, as proposed by Lee, et al.(Lee, Cha et al. 2010).

Figure 4: The trainee's view in SimSpray (VRSim 2012)

Military Training with VR and MR

The United States Military budgeted $172 billion for "training and readiness" (DOD 2012). This number alone is enough to illustrate that military training is a very expensive endeavor. Crafting recruits into soldiers, as well as keeping the skills of experienced soldiers sharp, traditionally involves a large amount of material consumption. To reduce costs, all sorts of training simulations have been developed. (Summers 2012). Because military training is conducted on such a large scale, it will become the application field for which the methods in this dissertation will be developed. However, the issues therein extend to almost any industry requiring some form of specialized training.

Figure 5: A flight simulator from 1910 (Moore 2006)

One obvious success story of virtual training is the flight simulator. Although flightless training for aircraft has been around almost as long as aircraft themselves (Figure 5), the sort of device we would recognize as a flight simulator was first built in 1941 (Moore 2006). Modern flight simulators, such as the Boeing 787 simulator shown in Figures 6 and 7, combine advanced VR displays with real cockpits, realistic aircraft sounds, and a motion base that can tilt the aircraft to simulate the forces a pilot feels (Choi 2010). Simulator training can cost as little as a tenth as much as flying a real aircraft. It also saves time and eliminates the risk of crashing a real aircraft. (Currie 2008) This training is so effective that the first time a pilot flies a new aircraft, it may be as his or her first actual mission with it as training in a simulator is accurate for all pilot skills acquisition.

Figure 6: External view of the Boeing 787 simulator. (Choi 2010)

Figure 7: Internal view of the Boeing 787 simulator. (Choi 2010)

Closer to the ground, mixed reality virtual convoy simulators (VCS), like that in Figure 8, simulate the dangerous task of navigating roads in hostile parts of Afghanistan

(Shufelt Jr. 2006). Up to 30 soldiers atop five HUMVEE chassis are surrounded by video screens immersing them in the simulation, firing simulated rifles and turret-mounted guns at attackers. The rifles recoil with each round fired, and hits are registered on the screens and sent to the simulation controller (USMC 2012; Boothe 2008).

Figure 8: A soldier participates in a training exercise using the Virtual Convoy Simulator. (Boothe 2008)

Virtual training for dismounted soldiers is more varied, as different skills and situations require completely different training environments. "Shooting gallery" type trainers typically involve a single large display and a weapon equipped with infrared lasers, such as Camp Pendleton's Indoor Simulated Marksmanship Trainer (USMC 2008; USMC 2012). These trainers can simultaneously support five soldiers using different types of weapons, from pistols to mortars, and even support infrared vision. In addition

to marksmanship training, they are useful for shoot/no-shoot judgement and employment tactics training. In South Africa, the Laser Shot training system (Figure 9) has significantly reduced the cost of marksmanship and other types of training, such as dealing with hostage situations (Martin 2013).

Figure 9: South African soldiers train using Laser Shot (Martin 2013)

MOUT

Another type of dismounted warfighter training is military operations in urban terrain (MOUT). MOUT training takes place in physical environments designed to mimic real-life urban areas. Buildings, roads, and other features are used to construct a training site that can be the size of a small town. These towns are then populated by paid actors, as well as warfighters, to emulate the local population. Finally, pyrotechnics and other

props are added, creating a realistic, highly immersive simulated environment. Soldiers training in some MOUTs even report experiences very close to those of actual combat (Filkins and Burns 2006). These MOUT scenarios also offer the advantage of cultural, as well as physical, immersion. This is an important aspect of training, as a major part of a soldier's duties include directly interacting with civilians of an occupied territory (Benedetti 2008). While MOUT training is extremely engaging and realistic, it requires a tremendous amount of work. Each time it is used, a MOUT site must be prepared, actors must learn their roles, don their costumes, and take their positions. Pyrotechnic effects must be primed. An entire town's worth of props must be put into place. After the exercise, everything must be cleaned up--explosion residue, shell casings, props, and people (Visser 2007). In short, conducting a MOUT exercise is very costly and time consuming.

LVC and MRVC

LVC, or Live, Virtual, and Constructive training is on the forefront of training technology. LVC training combines different simulations into a single shared exercise. Live entities are those in which trainees use real or simulated equipment and perform tasks as if they were real. Virtual entities are humans interacting through computers, often via first-person shooter video games. Finally, constructive entities are simulated forces that are not under direct control of human operators, but rather computer generated and controlled. LVC scenarios can involve multiple types of technology, from remotely-operated vehicles (ROV)s, to mobile devices, to PC training games. Live forces used in LVC training can be dismounted infantry, aircraft, or other military vehicles. The

overarching theme, however, is that the training takes place in a shared reality, common to all forces, whether live, virtual, or constructive. LVC is often distributed, meaning that the forces that are working together (or against each other) are in separate physical locations.

LVC Training is a popular paradigm for training because it can connect multiple forces in a single, virtually contiguous scenario. This allows for more complex training situations, such as those where air and ground forces must work together to achieve an objective, or where a real aircraft works with a virtual wingman to combat constructive hostile aircraft (Lechner and Huether 2008). However, LVC training is not without its challenges. The most common of these is communication among components. Fortunately, protocols and standards, such as Distributed Interactive Simulation (DIS) and High Level Architecture (HLA), exist to help components communicate in a common language. Although these standards don't completely eliminate communications problems, they can simplify the process of getting simulation components operating together.

Some LVC training setups combine mixed reality and physical sets with the distributed simulation. This is the next step for military training and is being termed MRVC (Mixed Reality, Virtual, and Constructive) training. In addition to the basic elements of LVC, MRVC is characterized by virtual reality-type technologies, such as 3D tracking systems, stereo displays, and instrumented hardware. The latter category includes input devices like simulated weapons with sensors connected to the triggers, as

well as other physical objects with virtual analogs, like movable walls. The goal of MRVC is to create a flexible, immersive, multimodal training setup that combines the benefits of LVC, mixed reality, and MOUT sites, approaching the sort of training described by (Dean, Garrity et al. 2004). MRVC scenarios can include the participants of traditional LVC exercises: players on PCs interacting via video games engines, live trainees, and users of other simulated vehicles and hardware. One training system currently in use that could be classified as MRVC is The Marine Corps' Immersive Infantry Trainer at Camp Pendleton. It is based on systems developed by Flatworlds. (Pair, Neumann et al. 2003; Schwetje 2009) and features live actors, pyrotechnics, and constructive entities projected onto walls.

Flexibility is the most important benefit of MRVC. Scenarios can, in theory, be quickly reconfigured depending on the training needs of the users. For example, before a squad performs a real-life raid on a building occupied by hostile forces, the building could be mimicked using movable walls in an MRVC facility, allowing the squad to practice against simulated opponents in it. However, MRVC's flexibility does have a cost. Technical difficulties inherent to LVC can be encountered in addition to those associated with MR systems. In the example scenario of recreating a building prior to a raid, the virtual world must be carefully reconfigured to match the real world. Depending on the application, this could be a matter of rearranging 3D models within an editor program, or creating new 3D models higher up the content pipeline.

The VR and MR aspects of MRVC add further complications to training. A typical VR system with fixed screens, one or two input devices, and a relatively small tracking area can take days to set up and calibrate. However, once set up, the static nature of a VR facility means that, barring hardware failures, it will be fairly reliable. A multi-user MRVC environment, on the other hand, changes often. Different scenarios may require displays to be added, removed, or repositioned, and input devices to be used. Configuration files or hard-coded scenario information must be updated to match. Runtime problems are also a daunting obstacle. The physical nature of military training necessitates a high level of ruggedness that is difficult to achieve in most commercials off the shelf (COTS) input devices or those created in a research laboratory. Finally, every visible action or event in an MRVC environment is preceded by a chain of events, each of which must happen correctly. Figure 10 illustrates some of the factors that come into play when moving from VR to MR and finally to MRVC. Note that like in the actual implementation of these concepts, the divisions are not discrete. In summary, there are many critical things that can go wrong while a scenario is running:

- Software problems
- Configuration problems
- Latency
- Hardware issues

While many of these issues are are a challenge in and of themselves, determining exactly where, when, how, and why the chain of events broke is often an even more complex challenge. Examples of this problem, along with potential solutions, will be

presented in depth later in this dissertation, as will a testbed MRVC environment, Iowa State University's MIRAGE.

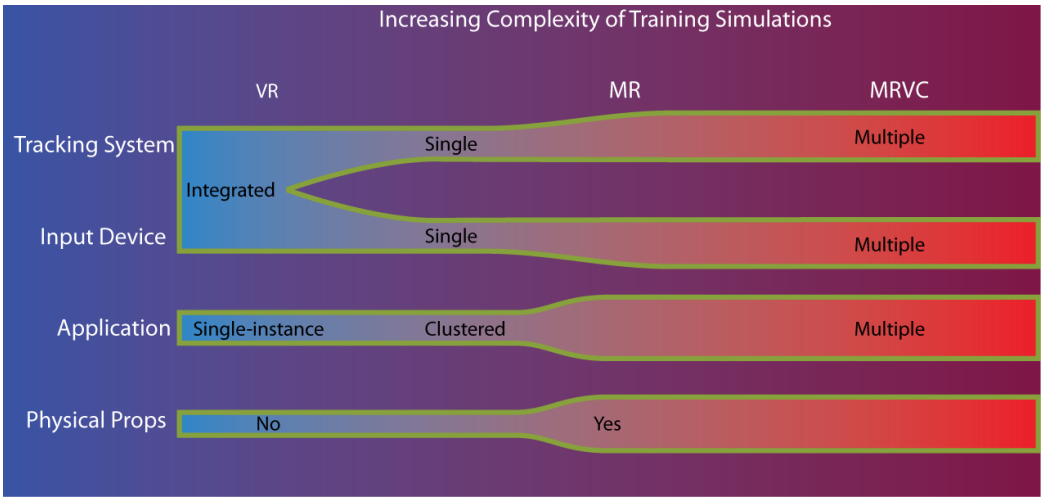

Figure 10: Increasing complexity of training simulations

Dissertation Organization

In this introduction, this paper discussed several types of military training. It has gone more in-depth in LVC and MRVC-based scenarios, such as the immersive training systems at Camp Pendleton. Chapter two will provide background on VR, AR, MR, military simulations, and the technologies that support them. Chapter three will introduce the MIRAGE, an MRVC system that serves as a testbed for this research. Chapter 4, the first of two journal articles composing this dissertation, presents the methodology and evaluation behind the MRT system. It provides details on each individual component, as well as the outcome of their use in the MIRAGE. Chapter 5, also a journal article, focuses on one component of MRT: MetaTracker, a tool for managing and combining data from multiple tracking systems. Finally, Chapter 6 will have conclusions of this research and discuss what areas still need improvement.

CHAPTER 2 - LITERATURE SURVEY

This chapter will provide background information on the fundamental technologies needed for advanced MRVC systems.

VR

VR comes in several forms, such as head-mounted displays (HMDs), Cave Automated Virtual Environment (CAVE™) systems, and other projection configurations. An HMD, which is a single- or dual-screen display attached to the user's head, is the only form of VR that can completely remove all outside visual cues. HMDs have the advantage of being small and portable, allowing multiple users to see their own personal view of the virtual environment. While this can, in theory, provide a very immersive user experience, HMDs tend to have low angular display resolution. Even modern HMDs, such as Sensic's zSight (Sensics 2012), have an angular resolution of around 20 pixels/degree, much lower than the human vision threshold of around 60-80 pixels/degree (Deering 1998; Huard 2010). In addition, an HMD is much more sensitive to latency in orientation tracking than a CAVE™, due to the plane of the display rotating with the user's head (Cruz-Neira, Sandin et al. 1993). This can be mitigated by the inclusion of built-in position and orientation tracking hardware included in some HMDs, such as the aforementioned zSight.

The CAVE™ is a type of VR display in which the user is surrounded by up to six projected stereo screens capable of higher angular resolution than HMDs. Iowa State University's C6 features 10 foot wide screens spanning 4096 pixels (VRAC 2012), or

about 410 pixels per foot. A user standing in the center of the C6--five feet away from the front screen--would therefore see 2048 pixels per radian at the center of the screen, or about 36 pixels per degree. Another advantage of a CAVE™ is nearly complete immersion. A four-sided (front, left, right, floor) CAVE™ fills nearly all of a user's visual field, while a six-sided CAVE completely blocks the outside world while still allowing the user to see his or her own body. Finally, unlike HMDs, CAVE™ systems provide limited support for multiple users, with somewhat distorted views available for users other than the primary (tracked) user. The primary disadvantages of a CAVE™ are its cost and its size. A CAVE™'s cost can reach several million dollars depending on the projectors and tracking technology used. The size of a CAVE™ is a two-fold disadvantage. The rear-projected nature of CAVE™ screens means that the CAVE™ facility takes up a large amount of space. However, the usable volume of a CAVE™ is much smaller. Iowa State University's C6 CAVE™ facility is currently the highest-resolution VR installation in the world, with over 100 million pixels. However, it occupies approximately 7000 cubic feet of space to house the entire visual system, but offers only 100 square feet of trackable area, and 600 square feet of display area. This is not enough space for training in any sort of skill that requires one to move on their feet.

A powerwall is a VR display device that is similar to a movie theater: A single large screen for multiple users. Powerwalls are less expensive and take up less space than CAVE™s, but provide much less immersion. The primary advantage of powerwalls lies in their use for groups of people, rather than single-user HMDs and relatively smaller user areas of CAVE™ systems.

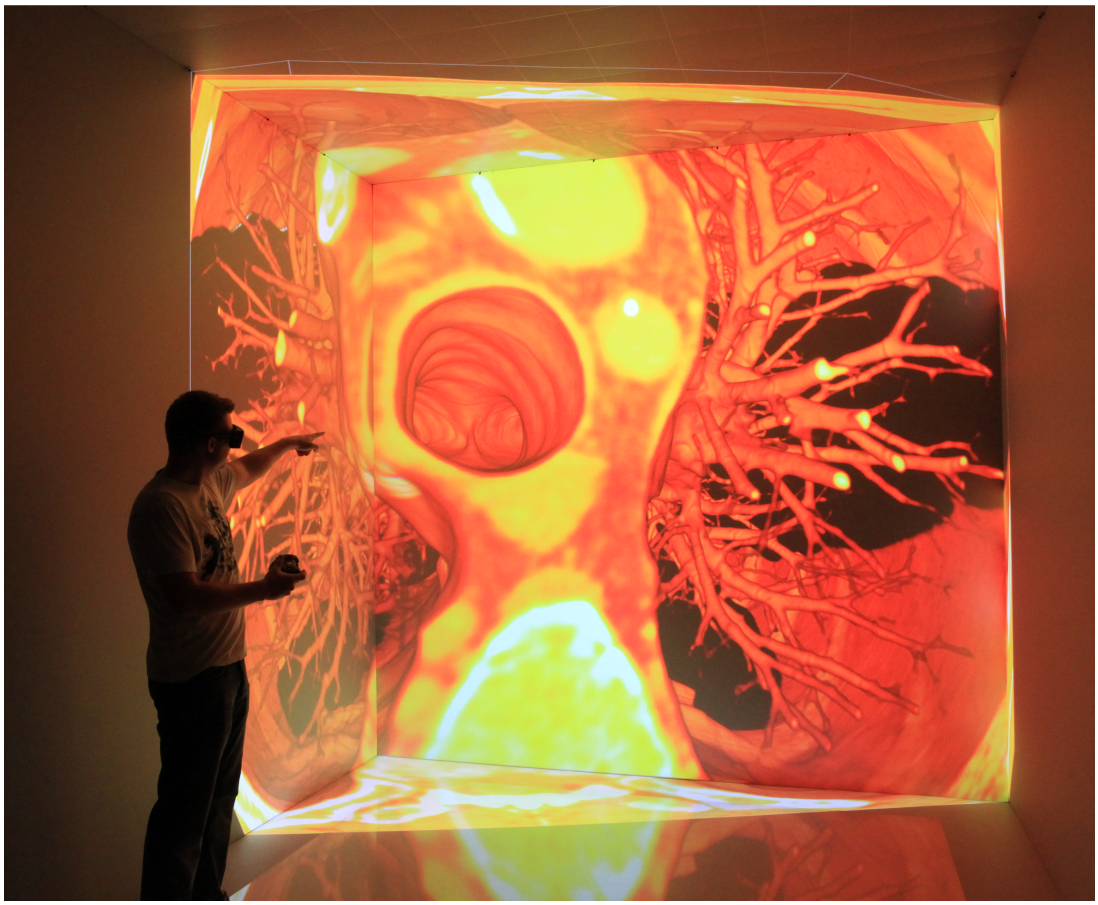

Figure 11: Visualization of medical data in a CAVE™(Noon 2012).

Other types of VR systems exist including projected domes (Baar, Willwacher et al. 2003) and flat-panel tiled systems (CVL 2012), but the scope of this dissertation cannot accommodate every variation on VR display. They are different in configuration but still use the same concepts of flat panel displays and/or projectors to create the virtual worlds.

While VR has many potential applications, some of its true strengths currently lie in data visualization, training, and engineering design. Virtual reality provides a powerful platform for presenting three-dimensional data in a clear way. Medical data is commonly visualized in VR or on stereoscopic displays (Noon 2012) like in Figure 11, as is

engineering data, such as data from computational fluid dynamics (CFD) simulations (Fu, Wu et al. 2010).

Virtual reality is a very powerful technology, but it is not without its limitations. The primary limitation is that in VR, most, if not all, of what the user sees is computer-generated. Creating new, realistic CG content can be time consuming and expensive. VR also lacks a sense of physical presence, which can reduce the transfer of experiences from VR to reality.

VR For Consumers

Until very recently, the price of VR has been the primary factor in its relative rarity outside of the laboratory. Even with the revolution in 3D graphics hardware this century, the input and display hardware required for VR has remained a significant barrier to entry. However, in late 2012, Oculus VR began shipping prototypes of its $350 Rift DK1 HMD to developers(OculusVR 2014). The DK1 featured a 1280x800 display divided between the user's two eyes and built-in, low-latency head orientation tracking. Less than two years later, in March of 2014, a higher-resolution display (1920 x 1080) and head position tracking were added to the Oculus Rift, then called the DK2. As of this writing, the final specifications for the consumer model of the Oculus Rift have not been announced, but an in-house prototype, dubbed Crescent Bay, was unveiled in September 2014, with higher specifications than the DK2.

New input devices, such as the Razer Hydra(Razer 2014) and Sixense's upcoming STEM(Sixense 2014) system both use magnetic tracking to provide 3D input to VR

applications. Gestural input can be obtained from a Microsoft Kinect (Microsoft 2014) or a Leap Motion (LeapMotion 2014) for a price that's within reach of a much wider audience: between $100-$600, depending on the type of input and number of tracked features needed. Together, the Oculus (as well as other low-cost HMDs, such as Sony's upcoming Morpheus(Sony 2014)) and accompanying input devices are making virtual reality available to small companies and individual consumers in a way that is currently impossible for CAVE™s or powerwalls.

AR

Augmented reality (AR) is a type of reality technology in which the user's normal vision is overlaid by computer-generated information in the form of text, images, and 3D objects. AR is typically done in a see-through or opaque manner. See-through displays

FIgure 12: Heads-up display in the space shuttle Atlantis (Harwood 2010)

overlay graphics on the user's actual field of vision, while opaque displays use cameras to recreate the user's vision on the display itself. AR is not intended to be immersive; rather, it is intended to keep the user immersed in the real world while augmenting their vision with extra information.

One familiar example of AR is the heads-up display, or HUD, in commercial and military aircraft as well as the space shuttle, as seen in Figure 12. The HUD displays important information, such as velocity vector and radar targets, on a transparent display in front of the pilot. The image drawn by a HUD display is manipulated so as to lie at a focal distance of infinity, through a process called collimation (Spitzer 2001). Thus, a box drawn around the calculated position of an enemy aircraft will always be drawn around that aircraft, regardless of where the pilot's head moves. Today, HUDs can also be found in some higher-end cars, such as the Cadillac XTS (General_Motors 2013), where they deliver information such as vehicle speed and upcoming navigational instructions without requiring the driver to look away from the road. Recently, modern smartphones have begun to offer AR applications that use the phone's camera and position sensors (GPS, compass, gyroscope, accelerometer) to display information on top of the camera's video stream. Applications such as Wikitude (Wikitude_GmbH 2012), shown in Figure 13, connect to the internet to provide the user with information on nearby hotels, tourist sites, and location-tagged Wikipedia entries that are visible in the camera's field of view.

With promises of "always there" data display capabilities, such as those of Google's Project Glass (Parviz, Lee et al. 2012; Google 2013), AR may soon play a prominent role in our everyday lives, but research is also ongoing in medical and industrial areas. One very promising application of AR is in medicine, where medical data and imagery can be overlaid on a patient during surgery, providing a map for surgeons to follow with their scalpels. In one study, (Wacker, Vogt et al. 2006) investigated and found benefits for AR-assisted biopsy using overlaid MRI (magnetic resonance imager) data. Other potential applications of AR include assistance for vehicle maintenance (Henderson and Feiner 2009) and visualization of underground infrastructure, such as electric and telephone lines (Schall, Mendez et al. 2008).

FIgure 13: Wikitude (Wikitude_GmbH 2012)

MR

Milgram and Kishino (Milgram and Kishino 1994) define mixed reality (MR) as "a particular subset of Virtual Reality related technologies that involve the merging of real

and virtual worlds somewhere along the ‘virtuality continuum’ which connects completely real environments to completely virtual ones” Strictly speaking, this includes VR, AR, and everything in between, but for the purpose of this dissertation has been defined as the real world extended and enhanced by computer displays providing a viewport into the virtual world. As noted in the introduction section, this can include cases where 3D objects are actually composited into a view of the real world. MR is perhaps best exemplified in theme park attractions, such as Universal Orlando’s ride, *The Amazing Adventures of Spider-Man*, which uses high-definition, stereo displays to thrill riders in seats mounted on a motion base. (Universal_Studios 2012) Flight simulators also fall under this definition of mixed reality, due to their combination of realistic cockpits and VR displays, such as in training simulators offered by (SIM_Industries 2013).

A more complex MR implementation can be found in FlatWorld (Pair, Neumann et al. 2003), which was developed between 2001 and 2005. FlatWorld mixed the techniques of Hollywood set design and special effects with cutting-edge display technology. FlatWorld took its name from the modular panels (“flats”) that are used to create movie sets. When these flats are combined with physical props and “digital flats”, a set can be created that successfully combines both the real and virtual worlds into a seamless experience. A real door, for example, could be opened to reveal a vast virtual landscape. FlatWorld further enhanced the sense of presence with strobe lights, overhead fans, and other effects.

Mixed reality can use either projected or mounted displays, such as the above examples, or see-through HMDs, as suggested by (Dean, Garrity et al. 2004). Combining multiple pieces of mixed reality hardware with an LVC simulation system results in the aforementioned MRVC style of training environment. These systems play an important role in current and emerging types of training.

VR and MR in Simulation-Based Military Training

Military training has long been a driving force in simulation design. A 1977 report by (Orlansky and String 1977) for the Institute for Defense Analysis on the effectiveness of flight simulators found the median simulation-to-real cost ratio for various flight simulators was 12%.The report found a favorable degree of training transfer from simulation to aircraft but warned that flight simulators with visual systems can cost 5-10 times as much as those with just a motion base. This, of course, has changed drastically since 1977, and advanced visualization of simulations has become the norm. This section will discuss several uses of virtual and mixed reality as training aids for Warfighters.

Aircraft Simulators

Flight simulators have been in development for over a century, with modern versions almost identically matching the aircraft they simulate. For example, Boeing's AH-64D Apache Longbow Crew Trainer simulation features multiple high-resolution displays as well as an integrated helmet display (Boeing 2007). It can connect to and interact with other simulators with the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) communication protocols. Another important feature of these

simulators is mobility. These trainers can be deployed around the world, including active combat theaters. This capability allows pilots to not only train while deployed, but to actually rehearse missions before they fly them. (Sheridan 2010)

The KC-135 Stratotanker is, according to the United States Air Force, the primary aerial refueling aircraft for the USAF and has been in operation since 1957. (USAF 2011) CAE USA currently is the prime contractor for the KC-135's Aircrew Training System (ATS). Its training system includes stations not only for pilots, co-pilots, and navigators, but also one for the refueling boom operator (CAE 2011). The Boom Operator Weapon System Trainer (BOWST) replaces traditional in-flight training at a cost savings of 98.8% (Storms 2012). This simulator can communicate with other simulators for joint operations. (Jean 2008)

Ground-Based Simulators

VR and MR simulations for ground-based Warfighters take many forms. The most basic is a shooting gallery-type trainer. In this type of simulator, participants wield mock ballistic weapons, firing them at images on a screen. The weapons, such as M-16 rifles or mortar launchers (White, Carson et al. 1991; USMC 2012) can be fitted with CO_2 gas systems to provide a realistic recoil. Sound systems can provide appropriate weapon reports. Participants' aim is usually determined by infrared laser sites.

Convoy simulators

Several simulators exist for training military convoy exercises. At Fort Bragg, North Carolina, the Special Operations Reconfigurable Vehicle Tactical Trainer helps new

soldiers learn the basics of convoy operations. Each of four live-sized Humvees is surrounded by projected video screens showing a military scenario, and is equipped with a field radio, as well as weapons such as machine guns, grenade launchers, and anti-tank devices. (MacLeod 2013) Currently, the trainer is used primarily with new soldiers who have never been deployed. It teaches them the basics of maneuvering the convoy vehicles, as the teamwork skills that are necessary for military work.

Game and Simulation Engines

An effective method to get graphical assets, artificially intelligent constructive agents, and other enhancements to a simulation is to use a pre-existing game engine. Several are available, and offer varying sets of features.

Bohemia Interactive Simulations' Virtual BattleSpace 2 (BohemiaInteractive 2013) is a training and simulation platform featuring advanced graphics and artificial intelligence (AI) functionality, as well as scenario authoring capabilities, after-action review, and an extensive library of models and other assets. It can be used for controlling constructive entities through AI as well as providing an interaction point for virtual trainees, who can participate in a simulation as a first-person shooter on a desktop or laptop computer. The platform has also been integrated with more specialized systems, such as flight simulators and gunnery trainers. .

While not technically an engine, LVC Game by Calytrix Technologies (Calytrix 2013) is simulation middleware that allows other commercial game engines to easily integrate with DIS and HLA-based simulations. It supports a host of modern game

engines, such as Havok (Havok 2013), CryENGINE(CRYTEK 2013), and X-Plane (X-Plane 2013).

Ternion Corp.'s FLAMES (FLexible Analysis, Modeling, and Exercise System) software package is "a framework for composable constructive simulations and interfaces between live, virtual, and constructive simulations" (Ternion 2013). FLAMES is not intended to supply an immersive 3D view into the simulation. Rather, it provides advanced simulation for everything from the motion of an individual bullet to the behavior of a battalion of tanks. This data can be visualized in both 2D and 3D views. FLAMES can also communicate with other simulations using DIS and HLA.

Solutions to LVC and MRVC Challenges

A great deal of research has been put into LVC, and later into MRVC technology to address the issues related to making different components work together. Despite the existence of standardized communication protocols, such as HLA, DIS, and TENA, not every application used in an LVC/MRVC environment will be able to speak to every other. It could be compared to a room in which each participant is speaking in his or her native language. While many integrated systems exist for collecting data from different locations and processing it at a central location, these systems are not generally real-time in the same way a training simulation must be. For example, a delay of half a second between data acquisition from an instrumented trigger and processing the simulated bullet's trajectory can render a firefight training simulation all but useless.

Traditional methods for application integration can be classified as application-centric integration and interface-centric integration (Gustavsson and Wemmergård 2009). Application-Centric Integration works best on just two systems, with individual applications being adapted to work with a selected protocol. Interface-Centric Integration (ICI) is used when more than two systems must be connected. ICI works by identifying a set of data elements in a common format and protocol. Both of these methods suffer from several problems, primarily that each application must be upgraded to keep up with changes to the protocol or changes to the information required by the training system. This is similar to every person in the previously mentioned room being required to learn Esperanto, and carrying on all communication with it. Gustavsson, et. al. go on to advocate the use of "information-centric integration." Information-centric integration is effectively manifested as a universal translator. Each application can broadcast its data to the translator in its native protocol. The translator will then convert this information into the native protocol of every other device and send it to those that are subscribed to that data. This way, when changes must be made to protocols or simulations, only a single application needs to be modified.

Thin Translation Layer

Thin Translation Layer (TTL) is a name given to systems where data is translated into a common format as soon as possible after it is acquired. VRJuggler is one framework that uses this approach with its Gadgeteer component for integrating with input devices. Other examples of TTL include VRPN and TrackD. This is different from ICI in that the

translation to a common language occurs after the data has been received from the device in question, but before it is received by the application that needs it.

VRPN

The open source VRPN (Virtual Reality Peripheral Network) project (VRPN Team 2013) aims to simplify the use of different input devices for virtual reality and, consequently, for non-VR applications. It uses a client-server model to allow a VR app, the client, to connect to servers that interact with peripheral input devices, such as trackers and game pads. In addition to making the devices available to multiple computers, it also abstracts the software interface for each device to a simple, generic API. VRPN currently supports several dozen input devices and trackers.

TrackD

TrackD, commercial software available from Mechdyne (Corporation 2013), performs a similar function as VRPN and is compatible with many commercial visualization products.

Protocols

In addition to device integration software, several protocols have been created for simulation component communication. These include DIS (Distributed Interactive Simulation, HLA (High Level Architecture), TENA (Testing and Training Enabling Architecture), CIGI (Common Image Generator Interface), and others.

DIS

The DIS protocol was originally developed by the University of Central Florida's Institute for Simulation and Training. It allows simulations to be carried out on multiple devices, without a central "simulation authority". Simulation events and meta-events are packed into protocol data units (PDUs) and sent over the network, usually via UDP datagrams. For example, an entity-state PDU contains information pertaining to the current state of any simulation entity. For example, it could describe the exact position and orientation of an M1A1 tank, as well as whether its hatch is open and whether or not the tank has sustained damage. Every entity involved in a DIS exercise is owned and controlled by one of the simulation applications. When the state of an entity changes, those changes are broadcast by its controlling application. In order to reduce network traffic, the DIS protocol supports dead reckoning, allowing entity states to be reported less often (McCall 2010). A wide variety of programs, such as the aforementioned VBS2 and LVC Game, use DIS as a means of communicating simulation events.

HLA

In the late 1990's, a new simulation protocol, HLA was developed as a successor to DIS. It is significantly different from DIS in that the HLA protocol is defined as an application programming interface (API) rather than a network data format. The actual data that is transmitted over the network is dependent on the particular implementation of HLA, called the Run-Time Infrastructure (RTI). While HLA also defines rules that simulations must obey and describes what data must be transmitted for simulation objects, it does nothing to ensure compatibility among HLA-speaking applications.

Dahmann, et. al.(Dahmann, Fujimoto et al. 1997) describe HLA as being based on three major components: Federates, the Runtime Infrastructure (RTI), and the runtime interface. A federate can be described as an individual simulator or other connected entity, such as a human player interface or a data collector. The RTI is responsible for the flow of information between federates. It handles network communication and sends data only where it needs to go, reducing network traffic. Finally, the runtime interface facilitates interactions between the RTI and the federates.

TENA

TENA has been in development since around 2002 (Powell and Noseworthy 2012). TENA is different from DIS and HLA in that TENA supplies tools for code-based specification of data that will be used in a simulation. This ensures a certain robustness to its data type specifications, but reduces the flexibility of the compiled application. TENA also has a single implementation of its middleware that is maintained by the TENA Software Development Activity. This implementation is available for many common platforms, including Windows, Mac OS, and Linux, and, while not open source, is freely available. The Tena Middleware handles all aspects of TENA use while allowing for customizability with user-created data types.

CIGI

The CIGI protocol was developed by Boeing starting in 2001. (Lechner and Phelps 2002) While it is not a protocol for communication and data-sharing among simulations, it occupies a useful role in simulation technology by maintaining a description of the

simulation from a visual standpoint. This description is shared, in realtime, with an image generator, or IG. The IG bears no responsibility for carrying out the simulation, instead, it is dedicated to rendering the scene. This decoupling of simulation from graphics rendering makes it possible to use a single simulation program with many types of display devices, from a laptop to a CAVE. Many commercial, and several open source implementations of CIGI exist, such as MetaVR's implementation(MetaVR 2013).

Summary and Next Steps

While many technologies exist to support and enhance LVC and MRVC training, it is clear that there are still areas where the state-of-the-art is rather weak. The primary of these is reconfigurability. Most training environments lack any sort of physical reconfigurability, instead relying on CGI content to tailor the scenario to the training need at hand. In environments that are reconfigurable, the limitations generally come from the software. Adjusting virtual environments to match real environments is often tedious. It often requires changes to 3D content necessitating the labor of artists, programers, trainers, and system operators to dismantle one training scenario and setup another. When conducting training, the person in charge needs to be able to adapt to the trainees in realtime. To make immediate changes to a scenario and run it again, quickly. Currently, this is simply not possible in an MRVC environment in real-time.

The other major challenge is robustness. The more components in a training system, the more opportunities for things to go wrong, and finding that point of failure is very difficult without tools to inspect at a low level during initialization, testing, and runtime.

Capturing data flow and aggregating it at these stages can provide a value inspection portal for runtime simulation data. These portals, if properly tapped, can provide easily accessible information on the shared reality in a MRVC system.

Research Issues

The research issues concerning MRVC simulations that will be addressed in this dissertation are:

1. Application initialization, launching, and reconfiguration in an MRVC System

Applications are the most accessible when they require little training to run. If run on a cluster, an application must be launched on each computer, and sometimes different commands must be sent to different computers. A GUI-based launcher would help to streamline the process of launching applications and make training exercises more efficient. Real-time checking of device availability and readiness would further improve the application initialization process.

Changing the setup and configuration of an MRVC simulation can mean a variety of changes. Applications must be re-launched with new settings and command-line arguments. Physical props must be moved into places that correspond to their new virtual positions. Hardware devices such as tracking systems may also need to be reconfigured to work with a different set of objects. Streamlining these tasks can turn downtime into training time.

2. Run-time device status monitoring in an MRVC System

MRVC scenarios are composed of multiple devices which rely on each other to function properly. If a problem does occur with a component, the task of finding the point of failure might be much more daunting than fixing the problem--a dead battery, loose wire, or unplugged Ethernet cable. Thus, when a problem occurs, it is important to be able to quickly identify the cause.

3. Unifying and abstracting 3D tracking data from heterogenous hardware systems

MRVC simulations often take place over a larger area than can be effectively tracked with a single 3D tracking system. Moveable props, such as walls, can occlude areas, reducing the reliability of tracking. Using multiple tracking systems can increase coverage, but this introduces challenges in terms of combing data from each system. Lowering the barriers to using multiple tracking systems can lead to more effective use of hardware that is already available and allow new uses of existing MRVC systems.

CHAPTER 3: THE MIRAGE

The research in this dissertation was tested on an MRVC system at Iowa State University, the Mixed Reality Adaptive Generalizable Environment, or MIRAGE system, shown in Figures 14 and 15. The work is designed to be applicable to other systems, but basic knowledge of the MIRAGE system is beneficial for understanding the design and tools of MRT.

Figure 14: Photo of the MIRAGE LVC training environment showing moveable wall sections, a window-mounted display, and a simulated Jersey barrier.

Figure 15: Reconfigurable rooms allow for flexible scenarios.

The MIRAGE is a complex, yet flexible MRVC environment. Developed in cooperation with the U.S. Army and taking cues from Hollywood set design, it features twelve moveable wall sections with interchangeable facades. These sections can be put together to form a variety of rooms and urban scenarios in the 3D-tracked 40' x 40' physical environment. A photo of this is shown in Figure 15. The virtual world is presented on multiple stereo displays, including several mobile displays that simulate windows as well as a large, fixed 39' x 12' stereo display that forms the back wall of the room. Simulated weapons come in the form of modified airsoft rifles. These rifles are fitted with radio frequency (RF) transmitters that relay trigger information to a central receiver. The rifles, along with other movable objects in the MIRAGE, are tracked using multiple optical tracking systems. It is this tracking that is one of the biggest challenges

facing the MIRAGE. Movable walls mean any part of the MIRAGE can be visually occluded from nearly any direction. To alleviate this problem, several tools, which will be discussed later in this chapter, have been created. Intended to be a flexible, extensible system, the MIRAGE currently incorporates two game engines linked together by a central communications server.

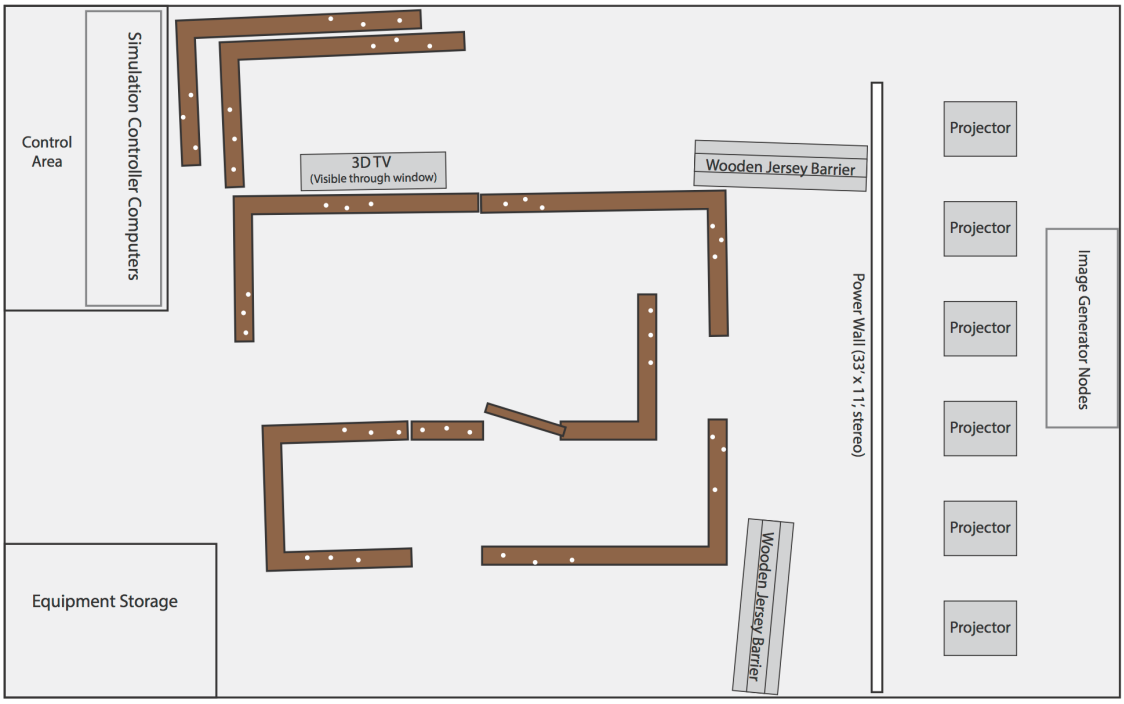

Figure 16: Sample layout of the reconfigurable MIRAGE. The small white dots are infrared tracking markers that allow the wall sections to be seen by the optical tracking system attached to the ceiling (not shown).

A diagram of the MIRAGE is shown in Figure 16. This is just one of many possible configurations that can be created in the facility. The white dots represent active (light-emitting) 3D tracking markers. These markers can also be seen in Figure 14, on top of the wall sections on the left side of the image.

DeltaJug

The first of the MIRAGE's game engines, DeltaJug, runs the VR and MR aspects of a simulation. DeltaJug is a hybrid of two platforms: Delta3D and VRJuggler. Delta3D is based on OpenSceneGraph (openscenegraph.org 2013) and other open source technologies and was developed specifically with simulation and training in mind. Its modular design incorporates components for basic model loading and rendering, terrain generation, particle effects, physics simulation, networking, and many other important aspects of the simulation. (Delta3D 2013)

VRJuggler (VRJuggler_Team 2013), originally developed at Iowa State University, is powerful, multi-purpose virtual reality software. It handles the communication and integration between nearly every piece of hardware involved in a VR application. Using its "Gadgeteer" component, it can interface with a wide variety of input devices, such as tracking systems, gamepads, and keyboards. It is also responsible for running applications as a cluster and making sure every viewport of a shared application is drawn with the correct perspective for the user. Because VRJuggler is based on configuration files describing the hardware that will be used with an application, cluster nodes, screens, input devices, and tracking systems can all be added or removed from a simulation without having to recompile source code. This feature makes VRJuggler an extremely flexible tool, but it also adds another point of failure. Problems in configuration files are often very difficult to track down, and can result in an application behaving in strange ways or not running at all. Furthermore, VRJuggler does not, by design, provide access to information regarding the hardware configuration. Such access understandably goes

against the principles behind VRJuggler, but is also necessary in some situations, , such as creating an application that is resistant to hardware failure. These shortcomings of VRJuggler are addressed later in the methodology section.

Their powers combined, VRJuggler and Delta3D were used as the foundation upon which to build DeltaJug. The open source nature of both of these libraries allowed for tight integration that preserved the desired strengths and attributes of each.

Communications Server

The final component of the MIRAGE system is the communications server. (B. Pollock 2011) This application acts as a go-between for the different applications running in the MIRAGE. Currently, the communications server distributes entity information between DeltaJug's master node and all instances of VBS2 using the DIS protocol. It was designed to be extendable, however, and can easily be upgraded to handle more protocols and applications.

CHAPTER 4: A SYSTEM OF SYSTEMS FOR MANAGING INCREASINGLY COMPLEX VIRTUAL AND MIXED REALITY TRAINING SIMULATIONS

A paper to be submitted to *Information Visualization*

Ken Kopecky

Ph.D. candidate, primary researcher and author

Eliot Winer

Associate Professor, author for correspondence

Virtual Reality Applications Center, Iowa State University

Abstract

With the advent of cheaper mixed reality (MR) and virtual reality (VR) equipment, immersive simulation technology has seen use in many industries for training purposes. As this training becomes more complex and realistic, so does the hardware and software supporting it. For example, mixed reality military training may run on a clustered computer system and occur over a large area, requiring multiple 3D motion tracking systems to monitor user positions. Simulated weapons communicate wirelessly to a master simulation server, and multiple stereo video screens show trainees a view into a virtual world where their training will occur.

In a complex scenario such as this, it is necessary to have tools to help with the tasks such as: 1) input parameter initialization (e.g., scenario, number of participants, etc.), 2) the

real-time status of individual systems, 3) data flow and timing between systems, and 4) software and hardware malfunctions.

In this article, the Mixed Reality ToolBox (MRT), developed to perform these operations, is presented. A review of current research and available products will be presented, with a focus on military training. Next, some of the issues that arise when combining physical, VR, and MR environments will be discussed. Finally, the components of MRT will be described, along with results showing MRT's effectiveness enabling different tasks. The ways in which MRT has greatly simplified tasks such as launching clustered applications, tracking down configuration problems, and dealing with 3D tracking in large mixed reality environments will also be discussed.

Introduction

Technology-aided training

In today's society, training is required for more and more jobs. Whether painting products on an assembly line, fighting fires, or performing surgery, a large amount of training is needed to prepare. Many times, training involves expensive consumable materials. This is evident in the case of a fire fighter, where training often takes place in "burn buildings": structures where fires are started, fueled either by the building itself or by gas or other combustibles [1]. A second example is training soldiers (warfighters), which accompanies the consumption of expensive munitions, fuel, and targets[2]. To reduce the financial and labor cost of training, many companies and organizations have turned to CGI-based (Computer-generated Image) simulation technologies, such as virtual reality (VR)[3], augmented reality (AR)[4], and mixed reality (MR)[5].

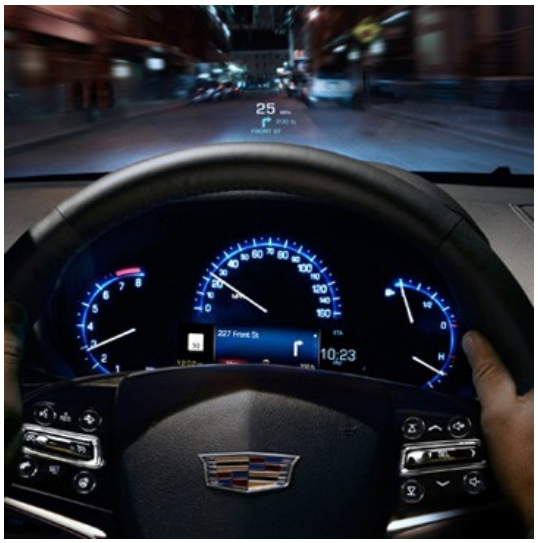

Figure 1: AR can take the form of a heads-up display, as seen on this Cadillac ATS [32].

It is impossible to cleanly differentiate between these three categories, and many of the works cited here offer overlapping definitions. Working definitions for these technologies are as follows: AR is considered to be images or data overlaid onto a user's view of the real world, such as the text overlaid on the windshield of the Cadillac in Figure 1. Virtual reality is considered to be instances where all or nearly all simulation-related visuals are computer-generated, with the exception of input devices, and take up all or nearly all of the user's visual field. Additionally, input devices are actual or simulated versions of their real-life counterparts, as opposed to a keyboard or mouse. A medical data visualization VR application is shown in Figure 2. Finally, MR simulations are defined as those where the real world is extended and enhanced by computer displays depicting virtual elements of the simulation. Specifically, these virtual elements appear to occupy positions in the real world and can even be composited into real world props including walls, doorways, and other physical objects. In Figure 3, the tablet computer shows an image of a helicopter overlaid onto

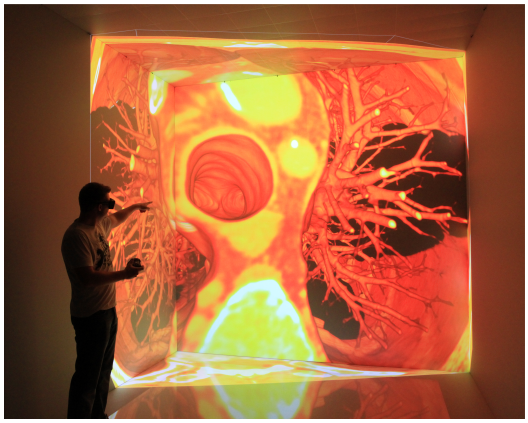

Figure 2: Visualization of medical data in a CAVE™ VR system[29].

the real world. The helicopter has a specific position in real 3D space, and its representation on the tablet is adjusted to make it appear at that location.

Figure 3: Mixed reality uses displays such as this tablet computer to add virtual content to the real world.

How VR can benefit training

In many areas where a significant part of training expense comes in the form of consumable materials, VR training can provide an immersive experience that simulates, to a certain degree, the desired situation. Virtual spray painting, for example, can combine a real input device (a modified spray gun) with a virtual part on a stereo display. As in Figure 4, a virtual part responds to the paint gun as a real part would, by changing color, but no paint is wasted, no toxic paint fumes are released into the air, and no materials need to be cleaned, sanded, and re-primed [6]. For firefighters, virtual buildings can be modeled and navigated in a virtual environment (VE) while participants fight a numerically-simulated fire, as proposed by Lee, et al.[7].

Figure 4: The trainee's view in SimSpray [6]

Figure 5: Internal view of the Boeing 787 simulator[10].

Military Training with VR and MR

In 2011, the United States Military budgeted $172 billion for "training and readiness" [8]. This number alone is enough to illustrate that military training is a very expensive endeavor. Crafting recruits into soldiers, as well as keeping the skills of experienced soldiers sharp, traditionally involves a large amount of material consumption and other resources. To reduce costs, all sorts of training simulations have been developed [9]. Because military training is conducted on such a large scale, it will become the application field for which the methods in this article will be developed. However, the issues therein extend to almost any industry requiring some form of specialized training.

One obvious success story of virtual training is the flight simulator. Modern flight simulators, such as the Boeing 787 simulator shown in Figure 5, combine advanced VR displays with real cockpits, realistic aircraft sounds, and a motion base that can tilt the aircraft to simulate the forces a pilot feels[10]. Simulator training can cost as little as a tenth as much as flying a real aircraft while eliminating the risk of crashing[11]. This training is so effective that the first time a pilot flies a new aircraft, it may be as his or her first actual mission with it as training in a simulator is accurate for all pilot skills acquisition.

Closer to the ground, mixed reality virtual convoy simulators (VCS), like that in Figure 6, simulate the dangerous task of navigating roads in hostile parts of Afghanistan[12]. Up to 30 soldiers atop five HUMVEE chassis are surrounded by video screens immersing them in the simulation, firing simulated rifles and turret-mounted guns at attackers. The rifles recoil with each round fired, and hits are registered on the screens and sent to the simulation controller[13,14].

Virtual training for dismounted soldiers is more varied, as different skills and situations require completely different training environments. "Shooting gallery" type trainers typically involve a single large display and a weapon equipped with infrared lasers, such as Camp Pendleton's Indoor Simulated Marksmanship Trainer[15, 16]. These trainers can simultaneously support five soldiers using different types of weapons, from pistols to mortars, and even support infrared vision. In addition to marksmanship training, they are useful for shoot/no-shoot judgement and employment tactics training.

LVC and MRVC

LVC, or Live, Virtual, and Constructive training is on the forefront of training technology. LVC training combines different simulations into a single shared exercise. Live entities are those in which trainees use real or simulated equipment and perform tasks as if they were real. Virtual entities are humans

Figure 6: A soldier participates in a training exercise using the Virtual Convoy Simulator[14].

interacting through computers, often via first-person shooter video games. Finally, constructive entities are simulated forces that are not under direct control of human operators, but rather computer generated and controlled. LVC scenarios can involve multiple types of technology, from remotely-operated vehicles (ROV)s, to mobile devices, to PC training games. Live forces used in LVC training can be dismounted infantry, aircraft, or other military vehicles. The overarching theme, however, is that the training takes place in a shared reality, common to all forces, whether live, virtual, or constructive. LVC is often distributed, meaning that the forces that are working together (or against each other) are in separate physical locations. LVC Training is a popular paradigm for training because it can connect multiple forces in a single, virtually contiguous scenario. This allows for more complex training situations, such as those where air and ground forces must work together to achieve an objective, or where a real aircraft works with a virtual wingman to combat constructive hostile aircraft[17].

Some LVC training setups involve detailed physical sets and combine mixed reality with the distributed simulation. This is the next step for military training and is being termed MRVC (Mixed Reality, Virtual, and Constructive) training. The hallmarks of MRVC, in addition to the basic elements of LVC, are virtual reality-type technologies, such as 3D tracking systems, stereo displays, and instrumented hardware. The latter category includes physical objects with virtual analogs, like movable walls, as well as simulated weapons with sensors connected to their triggers. MRVC attempts to combine the benefits of LVC, mixed reality, and MOUT [18, 19] sites into a flexible, immersive, multimodal training setup, approaching that envisioned byDean, Garrity, et al[20]. Like in

traditional LVC, participants in an MRVC scenario can include players on PCs interacting via video games engines, live trainees, and users of other simulated vehicles and hardware. The Marine Corps' Immersive Infantry Trainer is an MRVC system at Camp Pendleton. It is based on systems developed by Flatworlds[21, 22], and features live actors, pyrotechnics, and constructive entities projected onto walls.

The most important benefit of MRVC is its flexibility. Scenarios can, in theory, be quickly reconfigured to suit the immediate training needs of the users. For example, before a squad performs a real-life raid on a building occupied by hostiles, the building could be fabricated using movable walls in an MRVC trainer and the squad could practice against simulated opponents. This flexibility does have a cost. In the case of MRVC, the aforementioned technical difficulties inherent to LVC can be encountered, as well as those associated with MR systems. For example, the scenario of recreating a building prior to a raid involves physical changes to the real world. The virtual world must then be reconfigured to precisely match those changes or the effectiveness of the training will be compromised. This could mean simply switching out 3D models of different physical layouts, but these 3D models must be created and customized for each scenario. What's more, once the 3D models are created, their virtual counterparts must be carefully repositioned each time to match the physical world.

The VR and MR aspects of MRVC provide additional complications to training. A traditional VR system with fixed screens, one or two input devices, and a relatively small tracking area requires many hours, if not days, to set up. However, once set up, they are

fairly reliable and unchanging. A multi-user MRVC environment is constantly in flux. Displays and input devices alike are added, removed, and repositioned, and configuration files or hard-coded scenario information must be updated to match. In addition to setup tasks, runtime problems are also a daunting obstacle. The physical nature of military training necessitates a level of ruggedness that is difficult to reach in most commercials off the shelf (COTS) input devices or those created in a research laboratory. Finally, every action or event in an MRVC environment is preceded by a chain of events, each of which must work correctly. While many of these steps are individually simple and their problems easy to resolve, determining exactly where, when, how, and why a chain of events broke is a challenge in and of itself. Examples and potential solutions to this problem will be presented in depth later in this article.

Background

This chapter will provide background information on the fundamental technologies needed for advanced MRVC systems.

VR. VR comes in several forms, such as head-mounted displays (HMDs), Cave Automated Virtual Environment (CAVE™) systems, and other projection configurations. The CAVE™ is a type of VR display in which the user is surrounded by up to six rear-projected stereo screens capable of high angular resolution; A user standing at the center of Iowa State University's C6 [23], sees an angular resolution of 36 pixels per degree. Another advantage of a CAVE™ is nearly complete immersion. A six-sided CAVE™ completely blocks the outside world while still allowing the user to see his or her own

body. CAVE™ systems provide limited support for multiple users, with somewhat distorted views available for users other than the primary (tracked) user. The primary disadvantages of a CAVE™ are its cost and its size. A CAVE™'s cost can reach several million dollars depending on the projectors and tracking technology used. Finally, a CAVE™ usually requires a large amount of space for items such as projectors, screen supports, and a computer cluster to support a much smaller useable area.

A powerwall is a VR display device that is similar to a movie theater: A single large screen for multiple users. Powerwalls are less expensive and take up less space than CAVE™s, but provide much less immersion. The primary advantage of powerwalls lies in their use for groups of people, rather than single-user HMDs and relatively smaller user areas of CAVE™ systems.

An HMD, which is a single- or dual-screen display attached to the user's head, is the only form of VR that can completely remove all outside visual cues. HMDs have the advantage of being small and portable, but have a relatively low angular display resolution. Modern HMDs, such as Sensic's zSight [24], have an angular resolution of around 20 pixels/degree, much lower than the human vision threshold of around 60-80 pixels/degree [25, 26]. In addition, an HMD is much more sensitive to latency in orientation tracking than a CAVE™, due to the plane of the display rotating with the user's head [27]. This can be mitigated by the inclusion of built-in position and orientation tracking hardware included in some HMDs, such as that included with the Oculus Rift [28]. HMDs

are also much cheaper than other forms of VR. The current model of the Oculus Rift, the DK2, costs only $350.

While VR has many potential applications, some of its true strengths currently lie in data visualization, training, and engineering design. Virtual reality provides a powerful platform for presenting three-dimensional data in a clear way. Medical data is commonly visualized in VR or on stereoscopic displays [29] like in Figure 2(in the previous section), as is engineering data, such as data from computational fluid dynamics (CFD) simulations[30].

Virtual reality is a powerful technology, but is not without its limitations. The primary limitation is that in VR, most, if not all, of what the user sees is computer-generated. Creating new, realistic CG content can be time consuming and expensive. VR also lacks a sense of physical presence, which can reduce the transfer of experiences from VR to reality.

AR. Augmented reality (AR) is a type of reality technology in which the user's normal vision is overlaid with computer-generated information in the form of text, images, and/ or 3D objects. AR is done in either a see-through or opaque manner. See-through displays overlay graphics on a transparent screen within the user's actual field of vision, while opaque displays use cameras to recreate the user's field of vision on the display itself. AR is not intended to be immersive. Rrather, it is intended to keep the user immersed in the real world while augmenting their senses with extra information.

One familiar example of AR is the heads-up display, or HUD, in commercial and military aircraft as well as the space shuttle. The HUD displays important information, such as velocity vector and radar targets, on a transparent display in front of the pilot. The image drawn by a HUD display is manipulated so as to lie at a focal distance of infinity, through a process called collimation [31] Thus, a box drawn around the calculated position of an enemy aircraft will always be drawn around that aircraft, regardless of where the pilot's head moves. Today, HUDs can also be found in some higher-end cars, such as the Cadillac ATS [32] where they deliver information such as vehicle speed and upcoming navigational instructions without requiring the driver to look away from the road. Recently, modern smartphones and tablets have begun to offer AR applications that use the phone's camera and position sensors (i.e. GPS, compass, gyroscope, and accelerometer) to display information on top of the camera's video stream. Applications such as Wikitude [33] connect to the internet to provide the user with information on nearby hotels, tourist sites, and location-tagged Wikipedia entries that are visible in the camera's field of view.

MR. Milgram and Kishino [34] define mixed reality (MR) as "a particular subset of Virtual Reality related technologies that involve the merging of real and virtual worlds somewhere along the 'virtuality continuum' which connects completely real environments to completely virtual ones" Strictly speaking, this includes VR, AR, and everything in between, but for the purpose of this article has been defined as the real world extended and enhanced by computer displays providing a viewport into the virtual world. As noted in the introduction section, this can include cases where 3D objects are

actually composited into a view of the real world. MR is perhaps best exemplified in theme park attractions, such as Universal Orlando's ride, *The Amazing Adventures of Spider-Man*, which uses high-definition, stereo displays to thrill riders in seats mounted on a motion base [35]. Flight simulators also fall under this definition of mixed reality, due to their combination of realistic cockpits and VR displays, such as in training simulators offered by [36] [2013].

A more complex MR implementation can be found in FlatWorld[22], which was developed between 2001 and 2005. FlatWorld mixed the techniques of Hollywood set design and special effects with cutting-edge display technology. FlatWorld took its name from the modular panels ("flats") that are used to create movie sets. When these flats are combined with physical props and "digital flats", a set can be created that successfully combines both the real and virtual worlds into a seamless experience. A real door, for example, could be opened to reveal a vast virtual landscape. FlatWorld further enhanced the sense of presence with strobe lights, overhead fans, and other effects.

Mixed reality can use either projected or mounted displays, such as the above examples, or see-through HMDs, as suggested by[20]. Combining multiple pieces of mixed reality hardware with an LVC simulation system results in the aforementioned MRVC style of training environment. These systems play an important role in current and emerging types of training.

VR and MR in Simulation-Based Military Training

Military training has long been a driving force in simulation design. A 1977 report [37] for the Institute for Defense Analysis on the effectiveness of flight simulators found the median simulation-to-real cost ratio for various flight simulators was 12 cents to 1 dollar. The report found a favorable degree of training transfer from simulation to aircraft but warned that flight simulators with visual systems can cost 5-10 times as much as those with just a motion base. This, of course, has changed drastically since 1977, and advanced visualization of simulations has become the norm. This section will discuss several uses of virtual and mixed reality as training aids for Warfighters.

Aircraft Simulators. Flight simulators have been in development for over a century, with modern versions almost identically matching the aircraft they simulate. For example, Boeing's AH-64D Apache Longbow Crew Trainer simulation features multiple high-resolution displays as well as an integrated helmet display [38]. It can connect to and interact with other simulators with the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) communication protocols. Another important feature of these simulators is mobility. These trainers can be deployed around the world, including active combat theaters. This capability allows pilots to not only train while deployed, but to actually rehearse missions before they fly them. [39]

The KC-135 Stratotanker is, according to the United States Air Force, the primary aerial refueling aircraft for the USAF and has been in operation since 1957. [40] CAE USA currently is the prime contractor for the KC-135's Aircrew Training System (ATS). Its

training system includes stations not only for pilots, co-pilots, and navigators, but also one for the refueling boom operator [41]. The Boom Operator Weapon System Trainer (BOWST) replaces traditional in-flight training at a cost savings of 98.8% [42]. This simulator can communicate with other simulators for joint operations. [43]

Ground-Based Simulators. VR and MR simulations for ground-based Warfighters take many forms. The most basic is a shooting gallery-type trainer. In this type of simulator, participants wield mock ballistic weapons, firing them at images on a screen. The weapons, such as M-16 rifles or mortar launchers [44] can be fitted with CO_2 gas systems to provide a realistic recoil. Sound systems can provide appropriate weapon reports. Participants' aim is usually determined by infrared laser sites.

Convoy simulators. Several simulators exist for training military convoy exercises. At Fort Bragg, North Carolina, the Special Operations Reconfigurable Vehicle Tactical Trainer helps new soldiers learn the basics of convoy operations. Each of four live-sized Humvees is surrounded by projected video screens showing a military scenario, and is equipped with a field radio, as well as weapons such as machine guns, grenade launchers, and anti-tank devices[45]. Currently, the trainer is used primarily with new soldiers who have never been deployed. It teaches them the basics of maneuvering the convoy vehicles, as well as the teamwork skills that are necessary for military work.

Solutions to LVC and MRVC Challenges

A great deal of research has been put into LVC, and later into MRVC technology to address the issues related to making different components work together. Despite the

existence of standardized communication protocols, such as HLA, DIS, and TENA, not every application used in an LVC/MRVC environment will be able to speak to every other. While many integrated systems exist for collecting data from different locations and processing it at a central location, these systems are not generally real-time in the same way a training simulation requires. For example, a delay of half a second between data acquisition from an instrumented trigger and processing the simulated bullet's trajectory can render a firefight training simulation all but useless.

Traditional methods for application integration can be classified as application-centric integration and interface-centric integration[46]. Application-Centric Integration works best on just two systems, with individual applications being adapted to work with a selected protocol. Interface-Centric Integration (ICI) is used when more than two systems must be connected. ICI works by identifying a set of data elements in a common format and protocol. Both of these methods suffer from several problems, primarily that each application must be upgraded to keep up with changes to the protocol or changes to the information required by the training system. Gustavsson, et. al. go on to advocate the use of "information-centric integration." Information-centric integration is effectively manifested as a universal translator. Each application can broadcast its data to the translator in its native protocol. The translator will then convert this information into the native protocol of every other device and send it to those that are subscribed to that data. This way, when changes must be made to protocols or simulations, only a single application needs to be modified[47].

Thin Translation Layer. Thin Translation Layer (TTL) is a name given to systems where data is translated into a common format as soon as possible after it is acquired. VRJuggler is one framework that uses this approach with its Gadgeteer component for integrating with input devices. Other examples of TTL include VRPN and TrackD. This is different from ICI in that the translation to a common language occurs after the data has been received from the device in question, but before it is received by the application that needs it.

VRPN. The open source VRPN (Virtual Reality Peripheral Network) project[49] aims to simplify the use of different input devices for virtual reality and, consequently, for non-VR applications. It uses a client-server model to allow a VR app, the client, to connect to servers that interact with peripheral input devices, such as trackers and game pads. In addition to making the devices available to multiple computers, it also abstracts the software interface for each device to a simple, generic API. VRPN currently supports several dozen input devices and trackers.

TrackD. TrackD, commercial software available from Mechdyne[50], performs a similar function as VRPN and is compatible with many commercial visualization products.

Protocols

In addition to device integration software, several protocols have been created for simulation component communication. These include DIS (Distributed Interactive Simulation, HLA (High Level Architecture), TENA (Testing and Training Enabling Architecture), CIGI (Common Image Generator Interface), and others.

DIS. The DIS protocol was originally developed by the University of Central Florida's Institute for Simulation and Training. It allows simulations to be carried out on multiple devices, without a central "simulation authority". Simulation events and meta-events are packed into protocol data units (PDUs) and sent over a network, usually via UDP datagrams. For example, it could describe the exact position and orientation of an M1A1 tank, as well as whether its hatch is open and whether or not the tank has sustained damage. Every entity involved in a DIS exercise is owned and controlled by one of the simulation applications. When the state of an entity changes, those changes are broadcast by its controlling application. In order to reduce network traffic, the DIS protocol supports dead reckoning, allowing entity states to be reported less often[51]. A wide variety of programs use DIS as a means of communicating simulation events.

HLA. HLA (High-Level Architecture) was developed as a successor to DIS. It is significantly different from DIS in that the HLA protocol is defined as an application programming interface (API) rather than a network data format. The actual data that is transmitted over the network is dependent on the particular implementation of HLA, called the Run-Time Infrastructure (RTI) 52.

TENA. TENA has been in development since around 2002[53]. TENA is different from DIS and HLA in that TENA supplies tools for code-based specification of data that will be used in a simulation. This ensures a certain robustness to its data type specifications, but reduces the flexibility of the compiled application. TENA also has a single

implementation of its middleware that is maintained by the TENA Software Development Activity.

CIGI. The CIGI protocol was developed by Boeing starting in 2001[54]. While it is not a protocol for communication and data-sharing among simulations, it occupies a useful role in simulation technology by maintaining a description of the simulation from a visual standpoint. This description is shared, in realtime, with an image generator, or IG. The IG bears no responsibility for carrying out the simulation, instead, it is dedicated to rendering the scene. This decoupling of simulation from graphics rendering makes it possible to use a single simulation program with many types of display devices, from a laptop to a CAVE. Many commercial, and several open source implementations of CIGI exist, such as MetaVR's implementation[55].

The MIRAGE

The research in this paper was tested on an MRVC system at ISU: the Mixed Reality Adaptive Generalizable Environment, (MIRAGE), system. The research outcomes were designed to be applicable to other systems, but basic knowledge of the system is beneficial for understanding what was completed.

The MIRAGE, shown in Figure 7 and described in previous work by Kopecky, et. al. [56], is a complex yet flexible MRVC environment. Developed in cooperation with the U.S. Army Research Laboratory and taking cues from Hollywood set design, it features twelve moveable wall sections with interchangeable facades. These sections can be put together

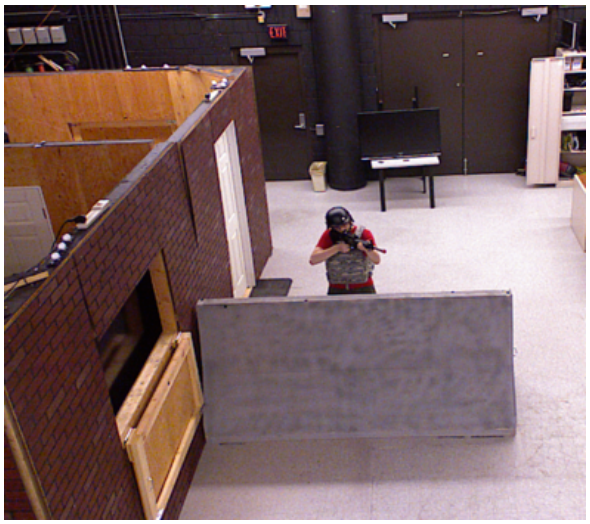

Figure 7: Part of the MIRAGE LVC training environment

to form a variety of rooms and urban scenarios in the 3D-tracked 40' x 40' physical environment. The virtual world is presented on multiple stereo displays, including several mobile displays that simulate windows as well as a large, fixed 39' x 12' stereo display that forms the back wall of the room. Simulated weapons come in the form of modified airsoft rifles. These rifles are fitted with radio frequency (RF) transmitters that relay trigger information to a central receiver. The rifles, along with other movable objects in the MIRAGE, are tracked using multiple optical tracking systems. Intended to be a flexible, extensible system, the MIRAGE currently incorporates a variety of software libraries and game engines.

DeltaJug. The first of the MIRAGE's game engines, DeltaJug, runs the VR and MR aspects of the simulation. DeltaJug is a hybrid of two platforms: Delta3D[57] and VRJuggler[58]. Delta3D is based on OpenSceneGraph and other open source technologies and was developed specifically with simulation and training in mind. Its modular design incorporates components for basic model loading and rendering, terrain generation,

particle effects, physics simulation, networking, and many other important aspects of the simulation.

VRJuggler. Most applications in the MIRAGE are based on the open-source VRJuggler VR software. Originally created at ISU, VRJuggler is designed to handle all aspects of VR on systems from desktop computers to fully immersive projective systems and Head-Mounted Displays (HMDs). VRJuggler can interface with a wide variety of hardware, such as tracking systems, gamepads, and keyboards, as well as manage data sharing and synchronization of clustered applications.

VRJuggler's strength lies in its use of configuration files. These describe the hardware upon which the application should be run, including tracking systems, screens, cluster nodes, and input devices. A single application can run on a desktop computer or on a clustered immersive system simply by changing the XML-based configuration files. This feature makes VRJuggler an extremely flexible tool, but it also adds another point of failure. Problems in configuration files are often very difficult to track down, and can result in an application behaving in strange ways or not running at all. These shortcomings of VRJuggler, and others, are addressed in the methodology section.

VBS2. Bohemia Interactive Simulations' Virtual BattleSpace 2[59] is a training and simulation platform featuring advanced graphics and Artificial Intelligence (AI) functionality, as well as scenario authoring capabilities, after-action review, and an extensive library of models and other assets. It can be used for controlling constructive entities through AI as well as providing an interaction point for virtual trainees, who can

participate in a simulation as a first-person shooter on a desktop or laptop computer. The platform has also been integrated with more specialized systems, such as flight simulators and gunnery trainers.

Communication Server. The final component of the MIRAGE system is the communications server[60]. This application acts as a go-between for the different applications running in the MIRAGE. Currently, the communications server distributes entity information between DeltaJug's master node and all instances of VBS2 (or other game engines or simulations) using the DIS (Distributed Interactive Simulation) protocol. It was designed to be extendable, however, and can easily be upgraded to handle more protocols and applications.

Putting It Together

As has been discussed in this section, simulated training environments are highly complex systems. Different systems may use different types of software and hardware (for example, 3D motion tracking systems in lieu of infrared lasers for weapon targeting). Still, many of the challenges encountered in setting up, running, and maintaining a simulation are common amongst them, such as detecting hardware failures, keeping components in synchronization, and obtaining simulation information at run-time. The rest of this article will present research aimed at addressing a subset of these challenges, while discussing concepts that can be applied more broadly to training simulations.

Methodology

While the MIRAGE is functional with the above systems, its complexity introduces problems when individual components must be added or removed or when scenario configurations change. This is largely due to the very nature of MRVC systems as outlined previously. Multiple pieces of hardware produce data that must be sent to multiple computers running their own simulations of the scenario at hand. A problem in any of these can have repercussions for the whole exercise. To make matters worse, these components often don't have a set way of communicating with each other. While a flight simulator and first-person infantry simulator might both communicate using the DIS or HLA protocols, other hardware and software, such as a position tracking system, might use a proprietary protocol to disseminate its data. Confronting the many challenges of MRVC environments requires approaching from two angles: centralization and abstraction. Much of the research in the previous chapter can be classified as one or both of these.

The MIRAGE's communications server provides an example of centralization, or assigning one computer or application as a master for a particular task or set of related tasks. Similarly, a master-slave approach to clustered VR applications, such as that used by VRJuggler, ensures that all input devices and other program control tasks are handled by a single computer, simplifying task ordering and execution. Abstraction is exemplified by VRPN software discussed previously, as it takes information from hardware devices in their native formats and translates it to VRPN's native format. This

allows any application that can interface with VRPN to use any hardware device that VRPN can communicate with.

Incorporating the technologies above can reduce the complexity of MRVC training, but difficulties still remain in many aspects running and maintaining these systems. A functioning MRVC environment requires monitoring of hardware devices and software, making sure everything is working properly. Everything must be correctly initialized at launch time, and shut down when the simulation is over. Finally, once an MRVC environment is functional, work can begin on enhancing that functionality. It is important

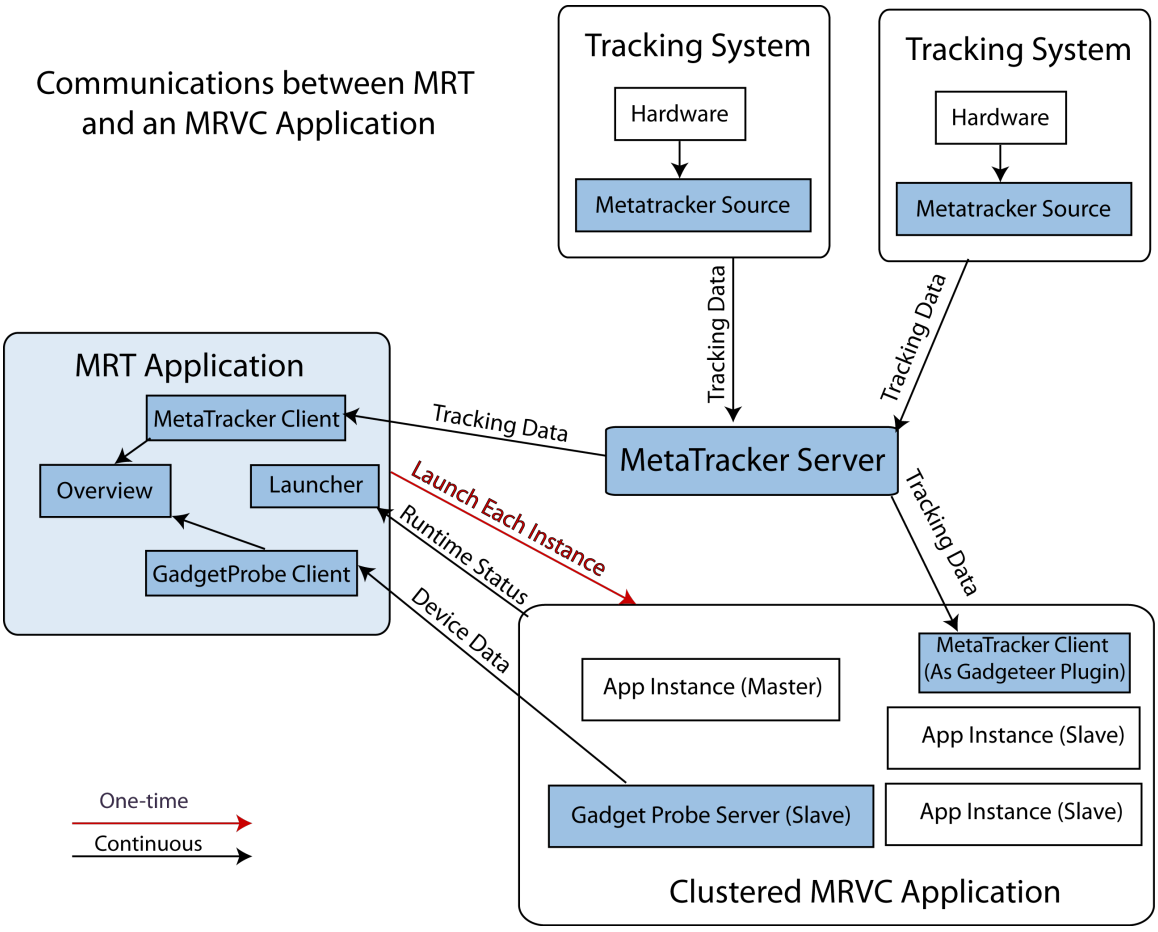

Figure 8: The major components of MRT and their interactions. Sections shaded in blue were created as part of this research.

here to differentiate between reconfigurability and flexibility. Flexibility refers to the gamut of situations for which a facility can be used. Reconfigurability refers to the time taken and ease of setting up the facility for a particular situation. For truly useful training, reconfiguration must be as easy as possible without sacrificing too much flexibility. Thus, the solution is neither a two-position switch that selects between predefined scenarios A and B, or a set of props and sets whose positions must be hand-entered into the computer each time the scenario is adjusted. Finally, the limiting factor in the flexibility of a training installation should be the physical, not virtual, elements of the simulation. To accommodate both goals, software can ease and sometimes even automate reconfiguration tasks. The Mixed Reality Toolbox, or MRT, was designed to handle all these complexities that hinder effective use of MRVC systems.

MRT is a set of software tools designed to solve many of the problems associated with MRVC systems. MRT includes four components:

1. MetaTracker, a system to abstract 3D tracking data away from the specific hardware that provides it, allowing multiple tracking systems to be treated as one.

2. Overview, a software module designed to use tracking data and virtual environment data, along with user-positioned 3D models, to create a real-time, application-independent 3D view of the training simulation.

3. Launcher, a software module for customizing and launching cluster-based applications without relying on the command line.

4. GadgetProbe, a VRJuggler-based application that reads the device configuration and input device data as it is received by the immersive application, displaying it on a screen.

Figure 8 shows how these components relate to each other.

MetaTracker

Traditional virtual reality systems are designed to support a single user in a limited area, and thus only need a single tracking system with two to three tracked objects, such as a pair of stereographic glasses and an input device. While some VR frameworks, such as VRJuggler, support the use of multiple tracking systems, none support the tracking of a single object by multiple systems simultaneously. In an MRVC simulation covering a wide area or tracking objects in areas occluded from several directions, multiple tracking systems may be needed to properly determine the positions of objects, and lack of support for this in the VR framework can be problematic. MetaTracker is a system designed not only to abstract tracking hardware from the application, freeing users from dealing with hardware drivers, but also to combine data from multiple 3D tracking systems, such that they are effectively acting as a single system.

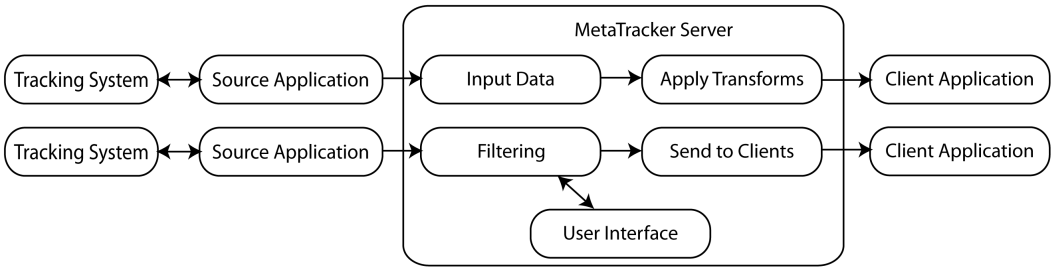

Figure 9. The components of MetaTracker, showing a pair of tracking systems whose data is accessed by multiple clients.

Components. MetaTracker itself is composed of three components, as shown in Figure 9. The first of these is the source application. This application utilizes the SDK and drivers supplied with a tracking system to read data from it in the tracking system's native format. A separate source application must be created for each tracking system's particular SDK. If the SDK contains appropriate functionality, the source application may also be able to control the tracking system to a certain extent, performing functions such as turning it on and off, or telling it which objects to track.

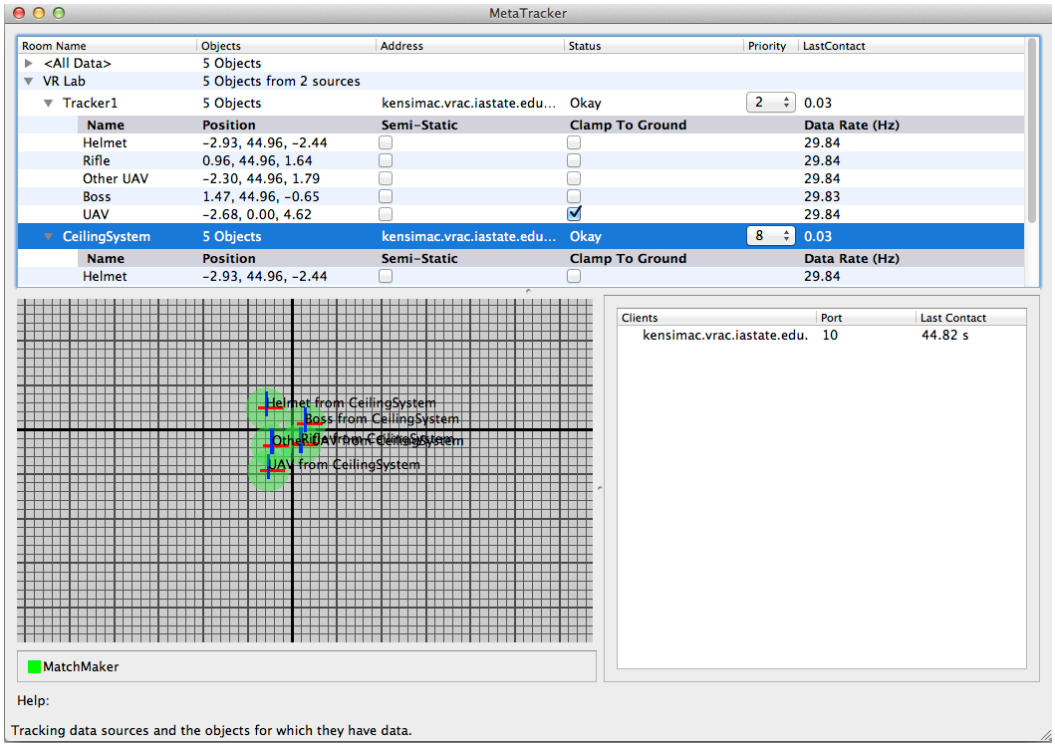

Figure 10. MetaTracker Server's user interface

The next, and most complex component of MetaTracker is the server, the user interface of which is shown in Figure 10. The server receives raw tracking data from each source and then sends that data to all connected MetaTracker clients. In most cases, the data is used as-is, and overwrites the previously known position and orientation data for that

object. However, in cases where the server is receiving data for a single object from two or more sources, a trust-based filtering algorithm is used to reduce discrepancies and smooth transitions as the object moves from one tracking system to the next. Data can also be modified based on known limited degrees of freedom (DOF). For example, an object that always rests on the ground, such as a faux concrete Jersey barrier, has only three degrees of freedom: the horizontal axes and rotation about the vertical axis. If tracking data for an object such as this is incorrect, and its position data is used for visualization, it can result in objects occupying positions that are highly implausible, such as a CG concrete barrier floating an inch off the ground. The MetaTracker server allows an object to be flagged with "clamp to ground", enforcing these degrees of freedom and reducing the appearance of these potentially distracting tracking errors.

The last component of MetaTracker is the client. The client is any application (or application plug-in) that registers itself with and receives data from the MetaTracker server. Clients can request data from specific tracking systems or rooms, and also specify what types of data they which to receive: 3 DOF positions, 6 DOF positions with orientations, or higher-DOF data for articulated objects.

MetaTracking. MetaTracker's name is derived from its ability to adjust the output of one tracking system according to the output of another, as illustrated in Figure 11. For example, in the MIRAGE, a ceiling-mounted MotionAnalysis tracking system is used to track the position and orientation of two self-contained ART SMARTTRACK systems. While the SMARTTRACK systems track objects relative to themselves, the known

positions of the SMARTTRACK systems can be used to determine the position and orientation of those tracked objects in the global coordinate system. If the SMARTTRACK systems are moved while the scenario is running, their new positions are continuously accounted for, allowing tracking coverage to be adjusted in real-time, while a training scenario is running. This allows for tracking to be dynamically extended to or improved in areas that are occluded or otherwise difficult to track with the ceiling-mounted system.

Persistence. In training simulations with larger objects, such as moveable structures, a scenario may require that these objects occupy positions in the virtual world that correspond to their real-world positions. In order to quickly and accurately update the

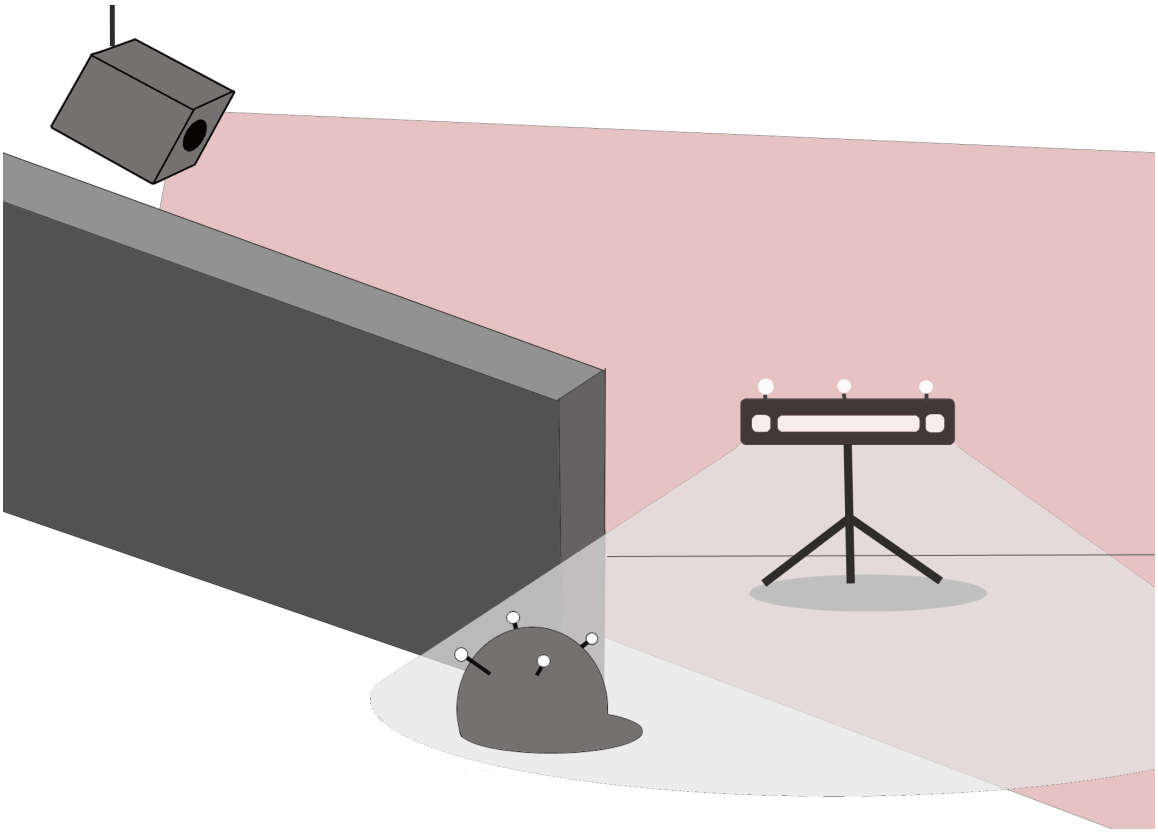

Figure 11: MetaTracking allows a portable tracking system to extend the range of a fixed system without recalibration or reconfiguration.

virtual world to reflect changes in the real world, tracking systems can be used to measure real positions and send them to training simulations in real-time. However, if an object is rarely moved, it doesn't make sense to have its position measured at 60 Hz. For cases such as this, MetaTracker is able to mark objects as "persistent". Until new tracking data for them is received, persistent objects are assumed to remain stationary, even across multiple launches of a training simulation or the MetaTracker server. When a client initially connects to the server, the last known positions of all persistent objects are sent to it, regardless of whether the object is currently being tracked.

Resolution of Tracker Disagreement. One issue encountered with MetaTracker's aggregation of tracking data occurred when two tracking systems could simultaneously track a single object. There is always some error in a tracking system's calibration, and early versions of MetaTracker simply passed on the most recent data they received, causing the object to seem to rapidly jump, or jitter, back and forth between the two reported positions. When this data was used for visualization of 3D objects, this jitter became erratic. Each frame, the object would appear whichever position was reported more recently.

Figure 12 shows the x-coordinate of the reported path of an object moving from the coverage area of one tracker, through an area of overlapping coverage, to the coverage area of a second tracker. Each tracker was calibrated to the global coordinate system, and then had a calibration error of 3 inches introduced. The "actual" position of the object was then measured by a third, ceiling-mounted tracking system that served as a reference.

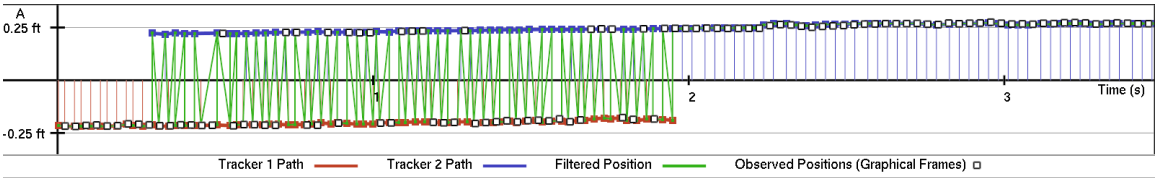

Figure 12. Positional error along the x-axis of a real tracked object relative to a "real" position measured by a third, wide-area tracking system.

The green line shows the actual data as reported by MetaTracker, and the white squares indicate the position of the object that would be observed in each graphical frame of the immersive application. While the green line shows a rapid, fairly regular oscillation between the two paths, the white squares show that a user of the system would experience a much more erratic motion.

In order to reduce this jitter, a data filtering system was created to allow objects to smoothly transition from the path measured by one tracker to the path measured by another. This method is based on the user assigning a confidence, or "priority" value to each tracking system. These confidence values are used to determine which tracking system is sending the most reliable data a particular tracked object. This filtering system is described in detail in Kopecky and Winer[61], along with a method to quantify the jitter, based on the total length of the incorrect reported movements of the object, called summed delta error (SDE). To test the effectiveness of the confidence-based error filtering, the above experiment was carried out with varying confidence values assigned to the two tracking systems. Cases where an object moved from a tracker of low confidence to a tracker of high confidence and vice versa, as well as between two

tracking systems with the same confidence values were tested. Visualizations of these tests are shown along with calculated SDE values in the results section of this article.

Latency Introduction. Another concern with the use of MetaTracker was the possibility of additional tracker latency being introduced to the system. Low tracker latency is critical to the performance of a VR or MR system, so any latency introduced by MetaTracker had to be very small compared to the pre-existing latency. Latency in MetaTracker was measured using both simulated and real-world tracking data. These experiments and their findings are discussed in the Results section.

GadgetProbe

The next integral component of MRT is GadgetProbe, a runtime input data inspection layer designed for VRJuggler-based immersive applications. While VRJuggler's Gadgeteer device integration system is well-engineered and provides easy access to many different input devices, it is difficult to configure and lacks runtime inspection capabilities. If, for example, the virtual pointer in a VR application isn't moving along with its real-life tracked counterpart, the cause may be a problem within the application itself, the VRJuggler configuration file, or the tracking system. Determining where the problem lies can be a time-consuming task. In short, it is very difficult to determine the state of VR input devices without knowing exactly what data to poll. GadgetProbe is an application written using VRJuggler that acts like an instance of the clustered VR application that is to be run on the system. It analyzes VRJuggler's configuration files used to run the application and then displays, in a graphical layout rather than a blur of

flying console text, information such as the exact state of digital buttons and analog triggers, positions and orientations reported by tracking systems, and the size and position of the virtual screens used to draw its 3D views (useful when creating a configuration file for a new system). Because it is built on VRJuggler, GadgetProbe is able to gather this data using the VRJuggler APIs. As Figure 13 shows, GadgetProbe presents a clear view of relevant information to help to quickly determine if data from external devices is successfully reaching the MRVC application and has the expected values.

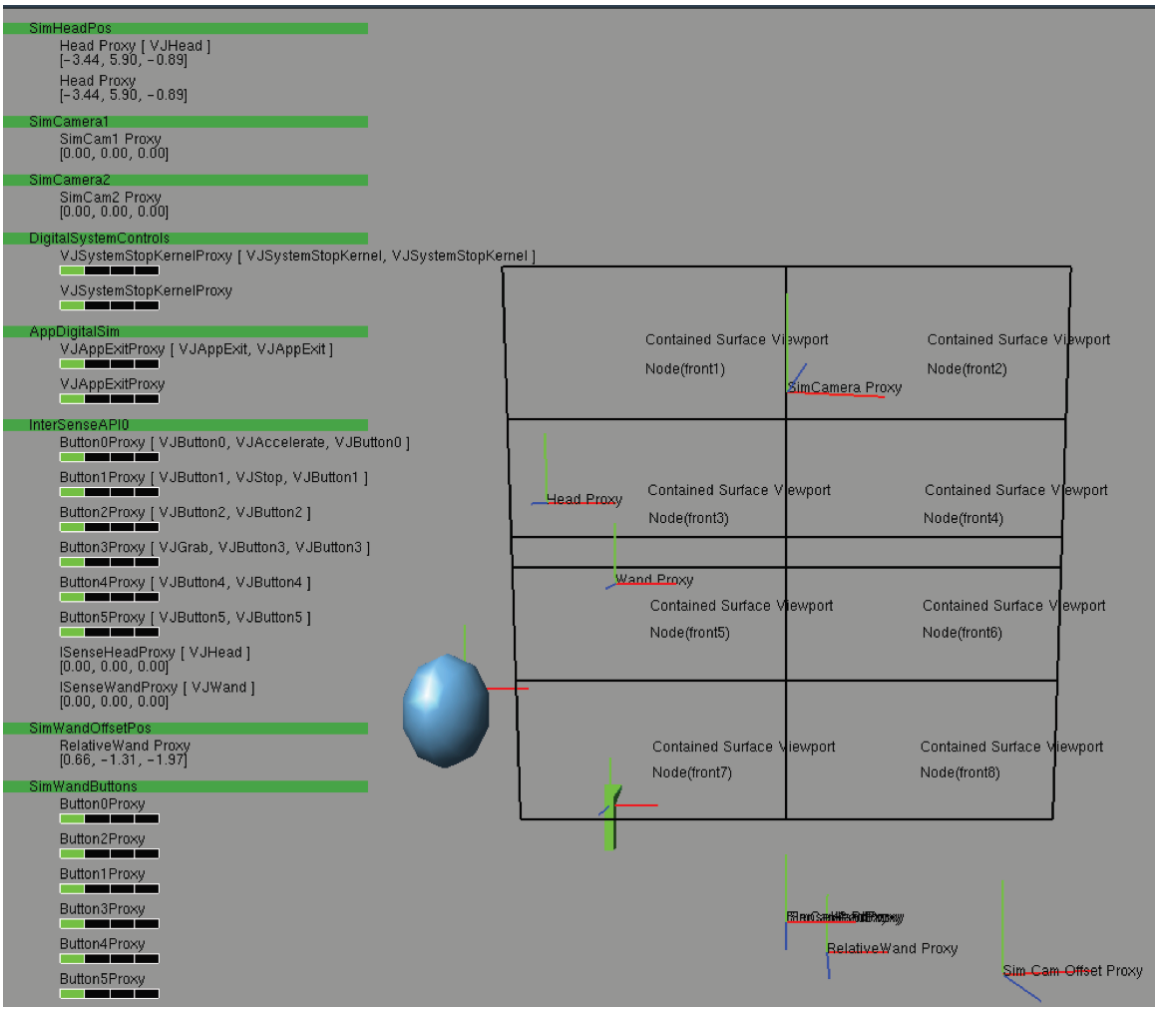

Figure 13: GadgetProbe's data display.

For example, it could provide confirmation that the “1” button on a gamepad is being correctly linked to the “ForwardButton” element of a program configuration. This can help diagnose problems that otherwise might only be fixed with hours of tweaking configuration files and viewing logs. In short, GadgetProbe shows the user exactly what VRJuggler knows and how VRJuggler knows it, all without any modification to the application itself. For external viewing of VRJuggler data, GadgetProbe runs a TCP server, allowing other applications to display its data in a form that’s best suited to the device at hand.

In its current form, GadgetProbe is built on VRJuggler, and can only work with VRJuggler applications. In the future, it could be adapted to other VR and MR APIs. The only requirement is that GadgetProbe is able to obtain, in realtime, information on the state of all connected hardware devices.

Overview

While both MetaTracker and GadgetProbe can display information regarding the state of tracked objects and other elements of the virtual and real environment, the data displayed in Figures 10 and 13 is presented in a generic, abstracted form that may reduce its usefulness. Overview, shown in Figure 14, was developed to help provide a more context-specific view of a VR or MR training simulation. Written using the OpenSceneGraph API[62], Overview visualizes data from GadgetProbe and MetaTracker to provide an overall view of a mixed reality simulation. Overview displays a scene composed of 3D models representing the major features of the physical room in which

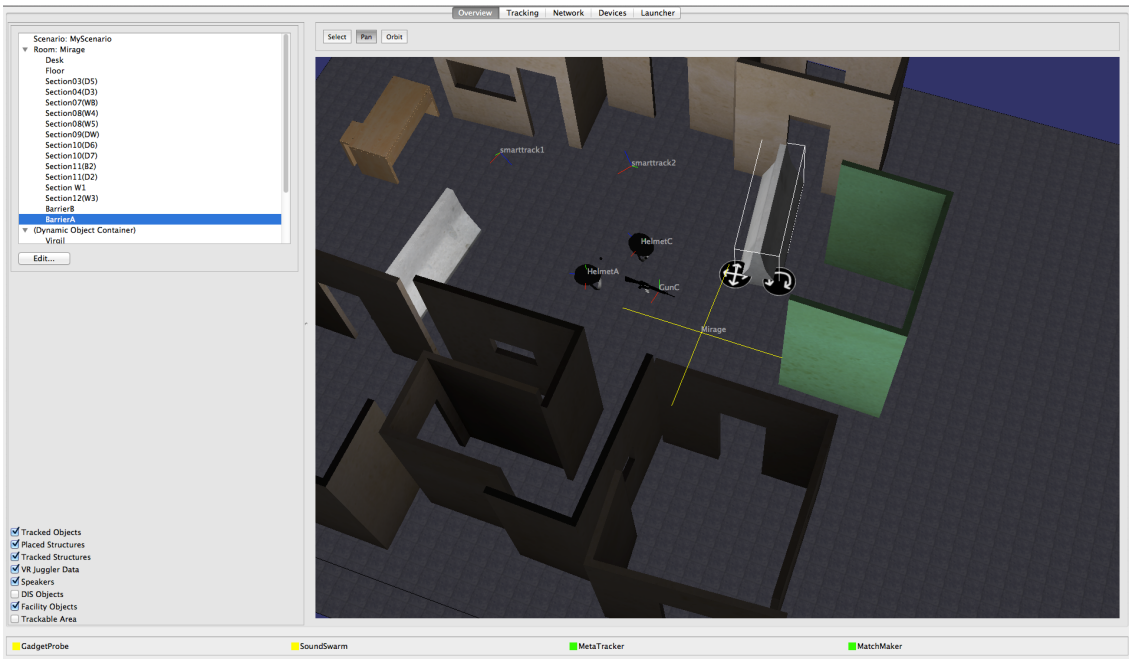

Figure 14: MRT's Overview. Objects are shown in both current and designated (green) positions.

the training scenario takes place. Data from MetaTracker is displayed in the form of 3D models of tracked objects in their real-life positions. Finally, information that GadgetProbe extracts from VRJuggler configuration files, such as the location and dimensions of display screens, is displayed within the 3D environment, helping the user to understand the relationship between the virtual and real worlds. GadgetProbe is also able to provide Overview with tracking data received by the MRVC application from tracking systems not connected to MetaTracker. By combining data from these two sources and associating 3D models with all of the objects in a scenario, Overview can automatically construct a 3D scene mimicking what is occurring in the training environment, regardless of the scenario or application that is running.

Data in Overview is stored in files corresponding to training scenarios and files corresponding to rooms. A scenario references the physical rooms in which the exercise

takes place. Each room represents a real, physical space in which a training exercise takes place, and can be shared among different scenarios. Associated with each virtual room are 3D models representing the real objects inhabiting the real room. These can include tracked, moveable objects as described in the "Persistence" section. In this case, they are shown at the positions last reported by MetaTracker. Each room is assigned a virtual position, corresponding to its relative location in the scenario's virtual world. In other words, even if two VR systems lie across the street from each other in real life, in the scenario they could be considered adjacent, or even a thousand miles apart. The scenario also stores its own set of 3D models. These are specific to the scenario and may or may not correspond to real-world objects. For example, objects stored in the scenario could include 3D buildings or vehicles that aren't directly interacted with in the simulation, but serve as visual references. Scenario files can also store preset positions for objects. This is particularly useful in MRVC systems where physical objects (e.g., walls, barriers, vehicles) may be positioned in different ways for different training scenarios. These objects can be laid out in Overview and will appear in their last-known actual locations while a transparent green version is displayed at its designated location, as in Figure 14. In brief, actual positions are stored with rooms, accessible to all scenarios, while desired objects are stored with scenarios. This setup is ideal for rapid creation and deployment of new training scenarios. In the case of tracked objects, it also reduces the need for marking the floor with tape, or worse, permanent marker.

To help Overview display new 3D objects in a scene with minimal user effort, an XML database of 3D model files corresponding to common objects for military training such as

guns, helmets, or moveable walls, called ModelDB was created. Each entry in ModelDB contains a path to a 3D model file, any necessary transformations to position, rotate, and scale the model to local space (so it appears in the proper orientation and scale relative to other 3D objects) and set of search tags. Overview uses this information to attach 3D models to new objects as soon as it receives information on them. For example, a tracking system might report (via MetaTracker or GadgetProbe) position data on an object named “rifleA”. Overview can combine this tracking data with a 3D model file obtained by searching ModelDB for “rifle” to add the rifle to the 3D view without any user intervention. By combining realtime and non-realtime positioning information with scenario planning capabilities, Overview can be a powerful tool for runtime inspection of MRVC applications.

Launcher

Modern VR facilities are increasingly turning to more numerous, higher resolution displays, driven by computer clusters. On VR systems that always run in the same configuration (that is, the same computers, input hardware, etc.), command line scripts can mostly automate the task of application launching. However, a reconfigurable MRVC system may use different sets of computers and different configuration files for different scenarios. To further complicate things, different applications may need other scripts to be run at launch-time to set environment variables and other settings. Finally, it is not uncommon for applications on a cluster to fail to correctly terminate. This can be difficult to detect, and the fastest fix is either rebooting the entire system or running yet another script to log into each node to make sure the application is no longer running.

The final component of MRT, Launcher, addresses these difficulties associated with clustered applications common in MRVC systems.

Launcher, seen in Figure 15, takes the tasks associated with running clustered applications and allows a user to perform them via a graphical user interface (GUI) rather than a command line. This tool makes running an MRVC application simple, yet retains much flexibility. To launch the application, a user selects its scenario file, choose desired cluster nodes, launch options (analogous to command line arguments), and configuration

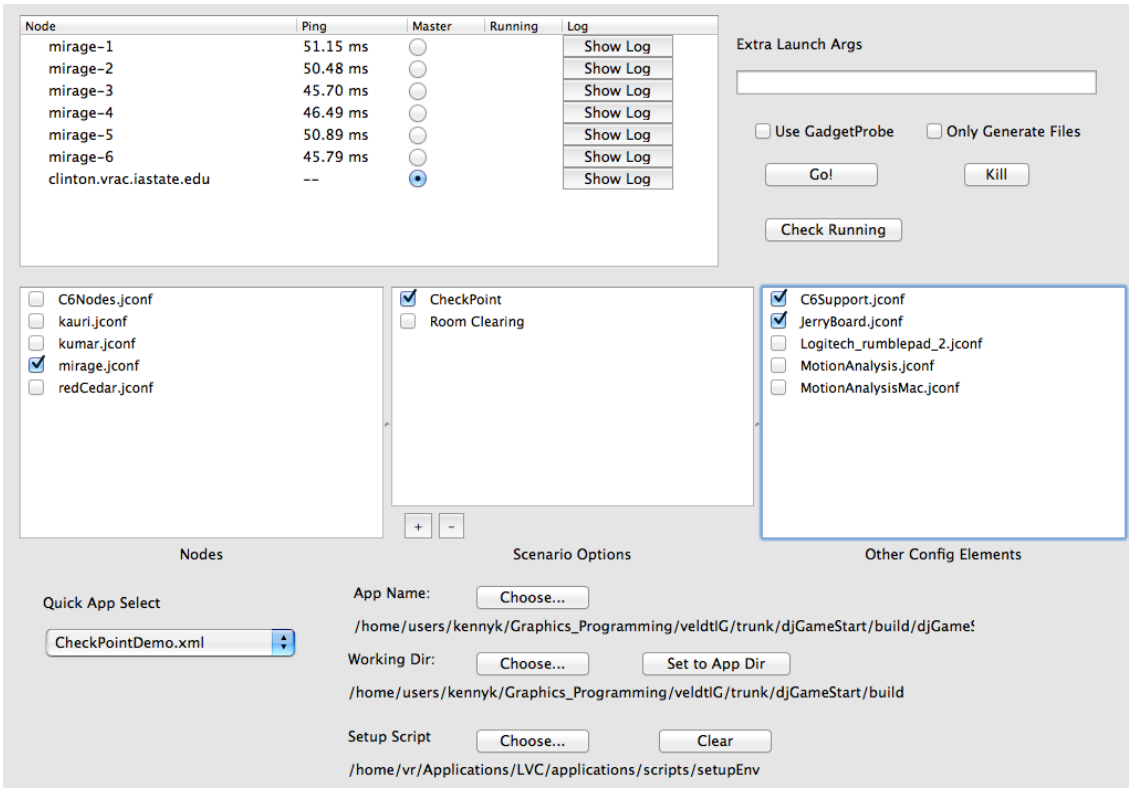

Figure 15: MRT's Launcher

files for other properties, such as specific tracking systems or input devices). This system reduces the redundancy of monolithic configuration files that include all cluster nodes, input devices, and tracking systems, and are then duplicated and modified for different

permutations of the scenario. Launcher also can automatically include GadgetProbe in the application's configuration and launch script.

When the user clicks "Go!", on the Launcher screen, a custom configuration is created, using options the user has specified. The necessary launch scripts are created and executed, and the clustered application is launched on all the selected cluster nodes. While the application runs, Launcher will occasionally ping each node, calculating a network latency time and determining if the application is still running on each node. Launcher also includes a kill button for reliable application termination. Finally, the user can check the output of the application on a particular cluster node by clicking on the "Show Log" button for that node. Launcher greatly streamlines the task of application configuration and running in these complex computing environments.

Results and Discussion

After over a year of use and testing (or more, depending on the component), MRT has, overall, greatly improved the process of performing certain VR- and MR-related tasks. Many of these improvements are difficult to quantify, such as the difference in difficulty of isolating a problem with a misbehaving VR application with and without GadgetProbe. MetaTracker, however, provided opportunities for measurement of both introduced network latency and for success in mitigating disagreements between tracking systems. This section will discuss the impact of MRT's components on their respective areas of focus, and will also briefly expand on the quantitative data obtained for MetaTracker.

GadgetProbe

The addition of GadgetProbe to the MIRAGE system, as well as its use with other VR systems, has been very beneficial for creating VRJuggler configuration files and verifying input device functionality. For example, user studies involving firing simulated weaponry occasionally run into problems with communications between the application and the weaponry. It was also common for the moveable walls to occlude areas, creating "dead spots" in the tracking coverage in the mixed reality environment. When pulling the trigger did not have the expected effect, GadgetProbe could tell the simulation operator with just a glance if the problem lay in the weapon itself or in the tracking. Even when MetaTracker is available to visualize tracking data, GadgetProbe's unique position at the very end of the data pipeline allows it to help verify that not only is the immersive application receiving the expected data, but that data is connected to the VRJuggler input channels on which the application is expecting it. Problems like these can now be solved in minutes, rather than the hours or days it had previously taken to solve this, and similar, problems.

Launcher

Launcher has been used extensively for multiple demonstrations and user studies, with VR systems as large as 49 cluster nodes. Prior to using Launcher, separate command line scripts were created for each application, which may have involved multiple versions of the same application but with different options. Modifying parameters or adding or removing cluster nodes required changing both command line scripts and configuration files. Searching through the hundreds (or even thousands) of lines in configuration files

could be a time-consuming and complex task, which often results in errors (e.g., typos or incorrect parameters). To make things worse, scripts often depended on a particular user's login account and environment variables, requiring that account to be used every time a particular application was used. With some initial setup, Launcher allows operators to quickly launch any available application, changing settings and configurations without modifying existing files. In cases where configurations and applications need to be switched and relaunched quickly, Launcher is frequently used to speed up the task. For example, a user study conducted at the VRAC required two different configurations of an application to be used alternately. The study operators used Launcher's GUI to modify the settings and relaunch the application in well under a minute. Likewise, visitors to the MIRAGE facility are often shown several applications. Launcher allows each application to be run in succession without requiring the operator to know the specific command line syntax for launching each application.

MetaTracker

MetaTracker has proven very useful in situations where a single tracking system cannot cover the entire area needed for a simulation. In a recent case, a user study required 20 feet of linear tracked space, and thus was to be performed in the MIRAGE. Shortly before the day of the study, the MIRAGE became unavailable, and the study was moved to a different room. Although no other available tracking system could track the needed 20 feet, two tracking systems were used with MetaTracker to accomplish the task. Using Metatracker, this was setup in a couple of hours in the new facility allowing the study to be conducted.

Figure 16: MetaTracker results for three tracking systems. Data points are color-coded by system. Thick lines indicate continuous tracking data, while thin lines indicate jumps between data points.

Tests with three tracking systems in the MIRAGE have shown visually that MetaTracker is capable of extending the tracking coverage of an existing system. In Figure 16, two SMARTTRACK systems, mounted on tripods, are used to track objects beyond the range of the ceiling mounted MotionAnalysis system. The observed motion was smooth, regardless of which system was used to track the object.

MetaTracker has opened up new possibilities for virtual and mixed reality applications by enabling larger, more complex areas to be tracked with existing hardware. Being hardware-agnostic, the MetaTracker client lets an application collect data from different systems without requiring any changes to the code. While integrating new types of tracking systems with MetaTracker does require creating a custom application to send data to the MetaTracker server, many tracking hardware vendors include an SDK with sample applications that can be modified to act as a MetaTracker source with ease.

Tracker Disagreement Results. The data filtering method created for MetaTracker were very successful in reducing the appearance of jitter in tracked objects. Figure 17 shows the reported motion of an object as it moves through the two overlapping tracked areas, using the data filtering method with various confidence levels assigned to the two

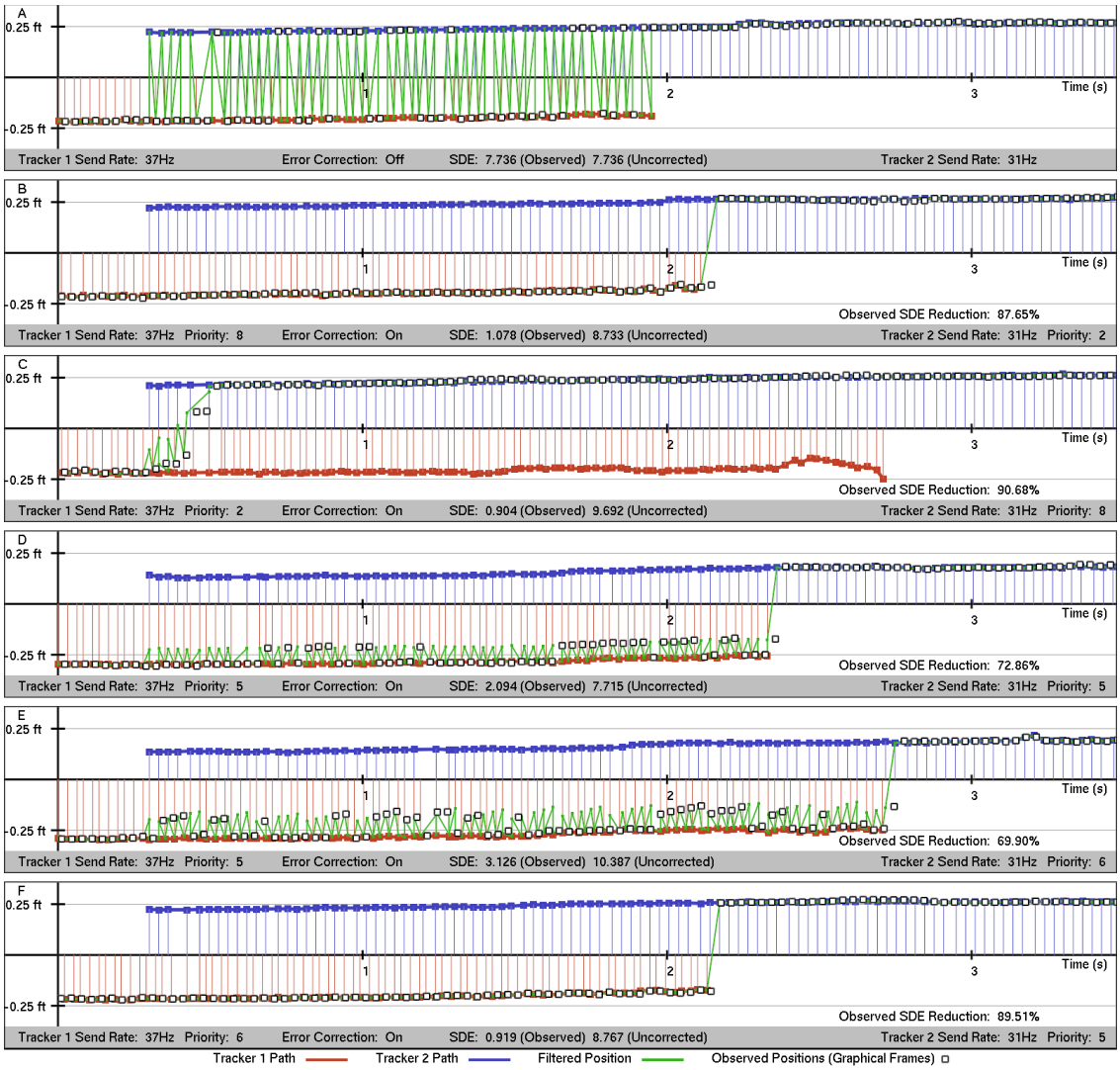

Figure 17(a-f). Positional error along the x-axis of a real tracked object relative to a "real" position measured by a third, wide-area tracking system.

tracking systems. A marked decrease in jitter, both for the path reported by MetaTracker and that observed in an immersive environment, can be seen. A similar decrease in the SDE (Summed Delta Error) value is also shown.

Latency Measurement and Results. Testing of MetaTracker with virtual data showed very little introduction of latency. Although each object within a given trial experienced a different amount of latency, the data shown in Figure 18 are for the object in a given run

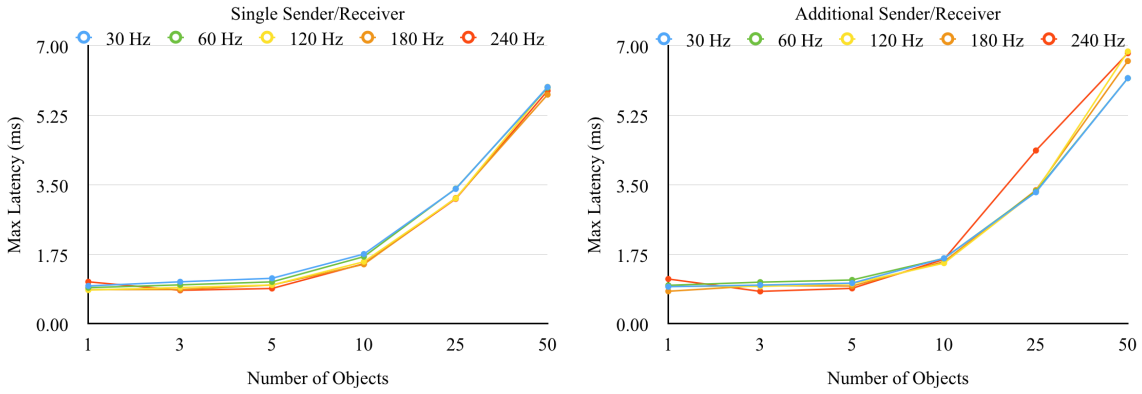

Figure 18. Average round-trip latency of a virtual object, by number of objects in the simulation and update frequency. Points shown are for the objects in each test with the highest average latency. Note: The second sender/receiver (right-hand graph) always ran with 5 objects at 60 Hz. The data shown is for the primary instance of the testing application.

with the highest average latency. As seen in the figure, for most situations the round-trip time for sending data to and receiving data back from the MetaTracker server was under 2 ms. This is a far shorter amount of time than the the overall system (from movement to image) latency values found by He, et al[63]. Even when 25 objects were tracked, latency was only around 3.5 ms. It's worth noting that tracking 25 unique objects is no small task. With an optical system, for example, a minimum of four markers may be required per object, and each target configuration must be unique[64].

To measure real-world tracking system latency, an experiment was devised wherein a pendulum, outfitted with retroreflective optical tracking markers, swung back and forth, as shown in Figure 19. Behind the pendulum stood a large screen that showed the most recent tracked positions for the pendulum, received both from the MetaTracker server and directly from the tracking system. A slow motion (120 frames per second) camera recorded video of the pendulum and its on-screen representations. For each swing of the pendulum, the number of video frames between the actual pendulum crossing the center

line and the virtual pendulums crossing the center line was counted, giving an estimate of the overall latency of the system. The results of the real-life pendulum experiment seemed to confirm the low latency times found in virtual testing. While the overall system latency (the time elapsed from when then pendulum crossed the center line to when its virtual counterpart did the same) was measured at 71-88 ms, no significant difference between the MetaTracker delay and the direct-from-tracking system delay was found. This is not to say there was no difference, but it was not detectable in the visual output of the display system.

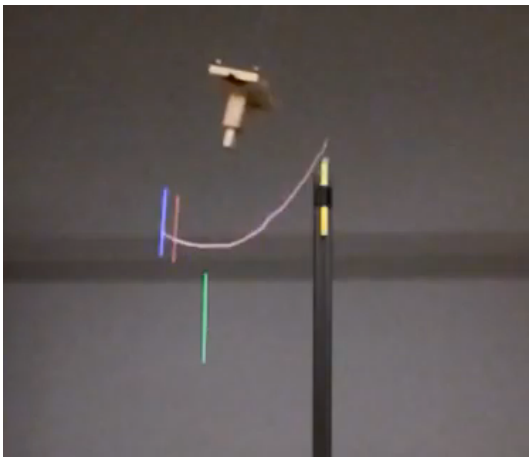

Figure 19. Captured video frame from the pendulum experiment. The red line indicates data from MetaTracker, and the blue line shows data sent directly to the application.

Overall, MetaTracker can bring increased flexibility to existing 3D tracking systems and allow easier access to the data provided by these systems. The data inspection capabilities provided by the server, as well as its clamp-to-ground and persistent object capabilities can help improve the quality of 3D tracking for virtual and mixed reality applications.

Summary

This article has laid out a number of challenges faced by advanced virtual and mixed reality training. These challenges include position tracking, input device management, initialization and launching of the training application. As virtual training advances, these

challenges will invariably become larger obstacles to simulation design, implementation, and use. The Mixed Reality Toolbox, or MRT, was designed to help minimize these difficulties through the principles of abstraction and centralization in its four components: Overview, GadgetProbe, MetaTracker, and Launcher. Testing and real-world use of these four tools has shown them to be very useful to the operation of VR and MR environments.

Future Work

The tools of MRT have proven to be very useful in real-world testing but there is still room for improvement. For example, future versions of Overview could be made to listen for DIS packets and use that information to visualize participants in simulations using that protocol and paint a more complete picture of the goings-on of the virtual world. GadgetProbe and Launcher currently only work with VRJuggler. Extending them to operate with other VR and MR frameworks would make MRT useful to a larger number of people.

Bibliography

1. Hodge RW. Back To Basics: Development Of Firefighter Training Burn Building Guidelines For The Vermont Fire Academy. 2003.

2. Shufelt Jr. JW. A Vision for Future Virtual Training. 2006.

3. Brooks F. What's Real about Virtual Reality? *IEEE Computer Graphics and Applications*. 1999; 19: 16-27.

4. Van Krevelen D and Poelman R. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*. 2010; 9: 1.

5. Costanza E, Kunz A and Fjeld M. *Mixed reality: A survey*. Springer, 2009.

6. VRSim. SimSpray. (2012, accessed September 9, 2012).

7. Lee J, Cha M, Choi B and Kim T. A Team-Based Firefighter Training Platform Using the Virtual Environment. *The 9th International Conference on Virtual Reality Continuum and Its Applications in Industry*. 2010.

8. DOD. The Federal Budget. In: Defense Do, (ed.). 2012.

9. Summers JE. Simulation-based military training: An engineering approach to better addressing competing environmental, fiscal, and security concerns. *J Wash Acad Sci, Spring*. 2012.

10. Choi B. Flying the 787 Flight Simulator. http://www.boeing.com/Features/2010/08/bca_787_flight_sim_08_26_10.html (2010, accessed April 16, 2012).

11. Currie A. McConnell now top KC-135 aircrew, follow-on training base. http://www.amc.af.mil/news/story.asp?id=123123815 (2008, accessed).

12. Shufelt JW. A Vision for Future Virtual Training. 2006.

13. USMC. Training Resources Online (Combat Convoy Simulator). http://www.marines.mil/unit/basecamppendleton/Pages/TrainingResources/ccs.html (accessed September 6, 2012).

14. Boothe D. Pendleton hosts newest DOD Combat Convoy Simulator http://www.marines.mil/unit/basecamppendleton/Pages/News/2008/PendletonhostsnewestDODCombatConvoySimulator.aspx (2008, accessed September 10, 2012).

15. USMC. Training Resources Online (Indoor Simulated Marksmanship Trainer). http://www.marines.mil/unit/basecamppendleton/Pages/TrainingResources/ismt.html (2012, accessed September 10, 2012).

16. USMC. Marine Corps Concepts and Programs. http://www.marines.mil/unit/pandr/Documents/Concepts/2008/CHPT3PRT7.htm (2008, accessed, 2012).

17. Lechner R and Huether C. Integrated Live Virtual Constructive Technologies Applied to Tactical Aviation Training. *The Interservice/Industry Training, Simulation, and Education Conference*. Orlando, FL2008.

18. Filkins D and Burns JF. Mock Iraqi Villages in Mojave Prepare Troops for Battle. *The New York Times*: (2006, accessed April 20, 2012).

19. Benedetti M. 102nd Security Forces: MOUT Training. *Seagull*. Otis ANG Base: 102nd INtelligence Wing, Massachusetts Air National Guard, 2008, p. 6-7.

20. Dean F, Garrity P and Stapleton C. Mixed reality: A Tool for Integrating Live, Virtual, & Constructive Domains to Support Training Transformation. *Interservice/ Industry Training, Simulation, and Education Conference*. Orlando, FL2004.

21. Schwetje C. Integrating Intelligence and Building Teams within the Infantry Immersion Trainer. Naval Postgraduate School, 2009.

22. Pair J, Neumann U, Piepol D and Swartout B. FlatWorld: Combining Hollywood set-design techniques with VR. *Ieee Computer Graphics and Applications*. 2003; 23: 12-5.

23. VRAC. Virtual Reality Applications Center. http://www.vrac.iastate.edu/c6.php (2012, accessed May 2012, 2012).

24. Sensics Inc. zSight™ Integrated SXGA HMD. http://sensics.com/products/head-mounted-displays/zsight-integrated-sxga-hmd/ (2012, accessed 9-10, 2012).

25. Deering MF. The Limits of Human Vision. *2nd International Immersive Projection Technology Workshop*. Ames, iowa1998.

26. Huard T. Angular Resolution: ASTR 288C: Lecture 6 presentation. 2010.

27. Cruz-Neira C, Sandin DJ and DeFanti TA. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. Anaheim, CA: ACM, 1993, p. 135-42.

28. OculusVR. The All New Oculus Rift Development Kit 2. http://www.oculus.com/dk2/ (2014, accessed, October 20, 2014).

29. Noon CJ. A volume rendering engine for desktops, laptops, mobile devices and immersive virtual reality systems using gpu-based volume raycasting. Iowa State University, 2012.

30. Fu D, Wu B, Chen G, et al. Virtual Reality Visualization of CFD Simulation for Iron/Steelmaking Processes. *ASME Conference Proceedings*. 2010; 2010: 761-8.

31. Spitzer CR. *The Avionics Handbook*. United State of America: CRC Press, 2001.

32. General_Motors. ATS Compact Sport Sedan. http://www.cadillac.com/ats-luxury-sport-sedan.html (2014, accessed September 10, 2014).

33. Wikitude_GmbH. Wikitude - World's Leading Augmented reality SDK. http://www.wikitude.com (2012, accessed September 10, 2012, 2012).

34. Milgram P and Kishino F. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems*. 1994; E77-D.

35. Universal_Studios. Amazing Adventures of Spider-Man Ride in 3-D at Islands of Adventure. http://www.universalorlando.com/rides/islands-of-adventure/adventures-of-spider-man-ride.aspx (2012, accessed September 10, 2012, 2012).

36. SIM Industries. Sim Industries | Visual-system. http://www.sim-industries.com/visual-system/ (accessed 1-29, 2013).

37. Orlansky J and String J. Cost-Effectiveness of Flight Simulators for Military Training. Institute for Defense Analyses--Science and Technology Division1977.

38. Boeing. Apache Longbow Crew Trainer(Product Card). 2007.

39. Sheridan A. BOEING DELIVERS RECORD FIVE APACHE LONGBOW CREW TRAINERS IN 2009. http://www.boeing.com/apachenews/2009/issue_01/news_s20_p2.html (2010, accessed September 10, 2012).

40. USAF. Factsheets : KC-135 Stratotanker. http://www.af.mil/information/factsheets/factsheet.asp?id=110 (2011, accessed September 10, 2012).

41. CAE, Inc. KC-135 Aircraft Training System (ATS). 2011.

42. Storms E. KC-135 Simulator to Help with Budget Cuts. http://www.milpages.com/blog/1748584 (2012, accessed Sepember 10, 2012).

43. Jean G. Success of Simulation-Based Training is Tough to Measure *National Device*: (2008).

44. White C, Carson J and Wilbourn J. Training and Effectiveness of an M-16 Rifle Simulator. *Military Psychology*. 1991; 3: 177-84.

45. MacLeod MJ. Virtual Convoy Simulator Maximizes Training Time. *Military.com*. 2013.

46. Gustavsson PM and Wemmergård J. LVC Aspects and Integration of Live Simulatio. *Fall Interoperability Workshop 2009*. Orlando, FL2009.

47. Pollock B, Winer E, Gilbert S and de La Cruz J. LVC interaction within a mixed-reality training system. *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2012, p. 82890K-K-10.

48. Team V. VRPN. http://www.cs.unc.edu/Research/vrpn/ (2013, accessed May 28, 2013).

49. VRPN_Team. VRPN. http://www.cs.unc.edu/Research/vrpn/ (2013, accessed May 28, 2013).

50. Corporation M. trackd the Device Driver Software for Immersive Displays. http://www.mechdyne.com/trackd.aspx (2013, accessed May 28, 2013).

51. McCall M. IEEE 1278 Distributed Interactive Simulation (DIS). 2010.

52. Dahmann J, Fujimoto RM and Weatherly RM. The Department of Defense High Level Architecture. *Winter Simulation Conference*. 1997.

53. Powell ET and Noseworthy JR. The Test and Training Enabling Architecture (TENA). United State Department of Defence Test Resource Management Center, 2012.

54. Lechner R and Phelps W. CIGI - A Common Image Generator Interface. *Interservice/Industry Training, Simulation and Education Conference*. Orlando, FL2002.

55. MetaVR. MetaVR Virtual Reality Scene Generator (VRSG)™ Features. http://www.metavr.com/products/vrsg/vrsg-feature-details.html (2013, accessed May 28, 2013).

56. Kopecky K, Gilbert S, Winer E, Civitate A and de la Cruz J. A Software Approach to Manage and Maintain Warfighter Training Systems. *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*. NTSA, 2013.

57. Delta3D. http://delta3d.org (accessed 2-17, 2013).

58. VRJuggler_Team. The VR Juggler Suite. http://vrjuggler.org/documentation.php (accessed 3-13, 2013).

59. BohemiaInteractive. VBS2 | Bohemia Interactive Simulations. http://products.bisimulations.com/products/vbs2/overview (2013, accessed).

60. Pollock B, Kopecky K, Gilbert S, de la Cruz J, Gonzalez H, and Winer E. Putting it Together: A System for Real-time, Reconfigurable, and Distributed Live, Virtual and Constructive Training. 2011.

61. Kopecky K and Winer E. MetaTracker: integration and abstraction of 3D motion tracking data from multiple hardware systems. *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014, p. 909507--11.

62. OpenSceneGraph. Home. www.openscenegraph.org (2014, accessed).

63. He D, Liu F, Pape D, Dawe G and Sandin D. Video-based measurement of system latency. *International Immersive Projection Technology Workshop*. Citeseer, 2000.

64. ART. Customized Targets. http://www.ar-tracking.com/products/markers-targets/customized-targets/ (2014, accessed October 15, 2014).

CHAPTER 5: METATRACKER: UNIFYING AND ABSTRACTING 3D MOTION TRACKING DATA FROM MULTIPLE HETEROGENOUS HARDWARE SYSTEMS

Submitted to IEEE Access

Ken Kopecky

Ph.D. candidate, primary researcher and author

Eliot Winer

Associate Professor, author for correspondence

Virtual Reality Applications Center, Iowa State University

***Abstract*—3D Motion tracking is one of the fundamental technologies enabling immersive virtual reality (VR) and mixed reality (MR). Correct 3D position and orientation detection is critical as both an input for interacting with virtual space and for properly displaying visual and audio output from a VR system. As the uses of VR and MR expand, new types of interaction are required for simulations, and place new demands on motion tracking systems. For example, a new simulation may require tracking in areas that are too large or have too much occlusion for on-hand hardware to perform effectively. In another case, a magnetic tracking system installed in a CAVE™ may need sensors to be attached to tracked objects to function, but the sensors cannot be easily attached to a new object that needs to be tracked in a simulation. Or a problem may simply arise with a simulation requiring tracking data from a tracking system that has no hardware drivers available on a particular hardware platform. The first and second of these problems require a second tracking system be used to track objects the existing tracking system cannot. The third requires either a different tracking system or an intermediary to query tracking data from the hardware and send it to the simulation. While frameworks exist that can allow a simulation to access multiple tracking systems (addressing the first two cases) and re-transmit it to a simulation (addressing the third case), none currently address the challenges arising from using multiple data sources, such as identifying which source to use at a particular moment, and what to do when the two sources disagree about an object's location. In this paper, a cross-platform solution for abstracting and combining tracking data from multiple tracking systems, and presenting the aggregate data as though from a single tracking system is presented. Individual objects are identified by name, and data is provided to simulations from a server application. This frees a simulation from having to account for multiple data sources, and allows tracked objects to transition seamlessly from an area covered by one tracking system to another. It also presents a unified API (Application Programming Interface) with a single data format, eliminating the need**

for hardware drivers. Finally, when self-contained tracking systems are used, those systems can themselves be tracked, allowing the trackable area to be adjusted in real-time. This allows simulation operators to leverage limited resources in more effective ways, improving simulation quality and opening new possibilities for VR and MR applications.

This article discusses the components and capabilities of this system and the API used to access its data. Several challenges encountered when designing the system are also covered, including resolution of conflicting position data from different tracking systems. Finally, in the course of developing the system, a great deal of development and testing was performed using simulated data inputs instead of real data. This article will also discuss the validation of those testing methods, showing that results obtained virtually are good predictor of real-life results.

***Index Terms*—Virtual Reality, Mixed Reality, Training, Motion Tracking, Distributed Systems, Military Training**

INTRODUCTION

Immersive environments have become increasingly important in the past decade in areas such as training, data visualization, and even entertainment. While terms such as virtual reality (VR), mixed reality (MR), and augmented reality (AR) are often used to describe different types of immersion, it should be noted that these technologies exist along a continuum defined by level of immersion [1]. VR is at the most immersive end of this continuum, consisting of experiences where most or all of the user's field of vision is replaced by computer-generated imagery (CGI). At the other end, AR refers to text or images overlaid onto the the user's view of the world, such as the heads-up display of an airplane. The user is immersed only in the real world. Finally, MR is the term used for immersive environments that can't be clearly defined as either AR or VR. An example of MR discussed several times in this article is that of a military training environment consisting of physical sets and props. Displays mounted inside windows and doors show 3D images of the larger, virtual world in which the training takes place.

As the technology needed for virtual reality and mixed reality advances, the variety of potential applications also increases. Long gone are the days when VR meant standing in one place, wearing a clunky head-mounted display physically connected to a single input device. A broad range of applications, including training, design review, data exploration [2], and of course entertainment [3] dictates a similarly wide range of physical equipment and hardware systems. For example, a CAVE™ system allows a user to look closely at a 3D object from different angles, walking around and examining it in a very natural way.

But training a firefighter to explore a burning building, looking for trapped inhabitants might require him or her to move around quickly, through real rooms, and interact with a combination of real and virtual objects. Such training would not be possible within the confines of even a large CAVE™. In fact, the virtual aspects of the simulation may not even be the most prominent, instead serving to complement the physical parts of the simulation. For example virtual windows could provide a view into the world outside the building or augmented reality equipment could be used to display virtual fire on burning objects.. This type of simulation, involving a mix of virtual and real elements scattered over a relatively large area, pushes the boundaries of virtual and mixed reality systems.

One of the most important technologies for VR and MR is 3D object tracking. Tracking allows 3D images to be drawn from the correct perspective and interaction devices to be used in a 3D space. It's also a critical link between the virtual and real worlds, ensuring the simulation knows the current position of props and other physical objects that interact in a simulation. The firefighter training scenario imagined in the previous paragraph presents many challenges to current tracking technologies, such as a wide area of coverage, occlusion of tracked objects by walls, and the high update rate required for AR images to accurately appear at the proper location in space. While no single tracking system available on-hand may be capable of the sort of tracking tasks the simulation requires, two tracking systems, used together, would be able to cover a wider area from more directions, reducing occlusion. A third tracking system, perhaps with a faster update rate but only capable of tracking two to three objects simultaneously, could be used to track a see-through HMD (head-mounted display) for displaying the AR images.

Unfortunately, using multiple tracking systems adds challenges of its own, such as having to deal with different hardware drivers and data formats, knowing which system is tracking specific objects, and handling disagreements between two tracking systems that can both see the same object. This article presents a software system for simplifying the use of multiple tracking systems by abstracting and aggregating their data, making it appear as though it originates from a single tracking system. In particular, it will focus on the measurement of latency introduced to the immersive system and how to handle situations where differing calibration errors lead to two tracking systems reporting different positions for the same object. Additionally, it will present methods developed to test this software using simulated data, rather than human resources.

BACKGROUND

A major area of application for virtual and mixed reality is training. In many areas where a significant part of training expense comes in the form of consumable materials, VR training can provide an immersive experience that simulates, to a certain degree, the desired situation. Virtual spray painting, for example, can combine a real input device (a modified spray gun) with a virtual part on a stereo display. As in Figure 1, a virtual part responds to the paint gun as a real part would with color changes, drips and overspray. However, no paint is wasted, no paint fumes are released into the air, and no materials need to be cleaned, sanded, and re-primed [4]. For firefighters, virtual buildings can be modeled and navigated in a virtual environment (VE) while participants fight a numerically-simulated fire, as proposed by Lee, et al.[5].

Figure 1: The trainee's view in SimSpray [26]

Virtual and mixed reality simulations also play an important role in military training. Physical training simulations can provide realistic training environments for warfighters. One of the most promising technologies available today is the LVC (Live, Virtual, Constructive) [6] training paradigm. In an LVC simulation, live players in a physical environment interact with virtual trainees controlling avatars and computer-controlled (constructive) entities across a single simulation. A major component of LVC is the informational link between the physical and virtual worlds. Keeping the virtual world in sync with the physical is essential for making sure all the players in an LVC exercise can properly interact with each other. One part of this process is the use of 3D tracking systems to relay the positions of objects and people to the simulation computers.

The Dismounted Soldier Training System (DSTS) [7] by Intelligent Decisions is a modular, mobile, and immersive VR training system where up to nine trainees don HMDs, instrumented weapons, and an array of other sensors and position trackers and participate in a virtual training exercise. An additional five trainees can participate via laptop stations. The DSTS can be adapted to a wide variety of scenarios, such as marksmanship training, squad/fire team collective tasks, and shoot/don't shoot judgement training. One potential disadvantage of the DST is the limited movement space for each participant: a four-foot diameter rubber pad. Concerns regarding negative training arising from this have been raised by Biagini, Trotta, and Joy [8].

Mixed Reality Training Systems

Mixed reality is a form of virtual reality in which a real environment, consisting of physical sets or props, contains screens that act as viewports into the virtual world. Several mixed reality training systems are currently in use by the U.S. Military. The Immersive Infantry Trainer at Camp Pendleton [9] features an environment consisting of multiple rooms equipped with large displays, audio systems, olfactory displays, and other special effects designed to simulate a combat environment. Trainees are able to interact with avatars that response to the movements and voice.

Figure 2. The Virtual Convoy Simulator[10]

The Virtual Convoy Combat Simulator (VCCS) [10] [11] seen in Figure 2 is designed to simulate the dangerous task of navigating roads in hostile parts of Afghanistan. Up to 30 soldiers atop five HUMVEE chassis are surrounded by video screens immersing them in the simulation, firing simulated rifles and turret-mounted guns at attackers.

Another mixed reality facility, the MIRAGE (MIxed Reality Adaptable Generalizable Environment) at Iowa State University, serves as a testbed for the work described in this article. Developed in cooperation

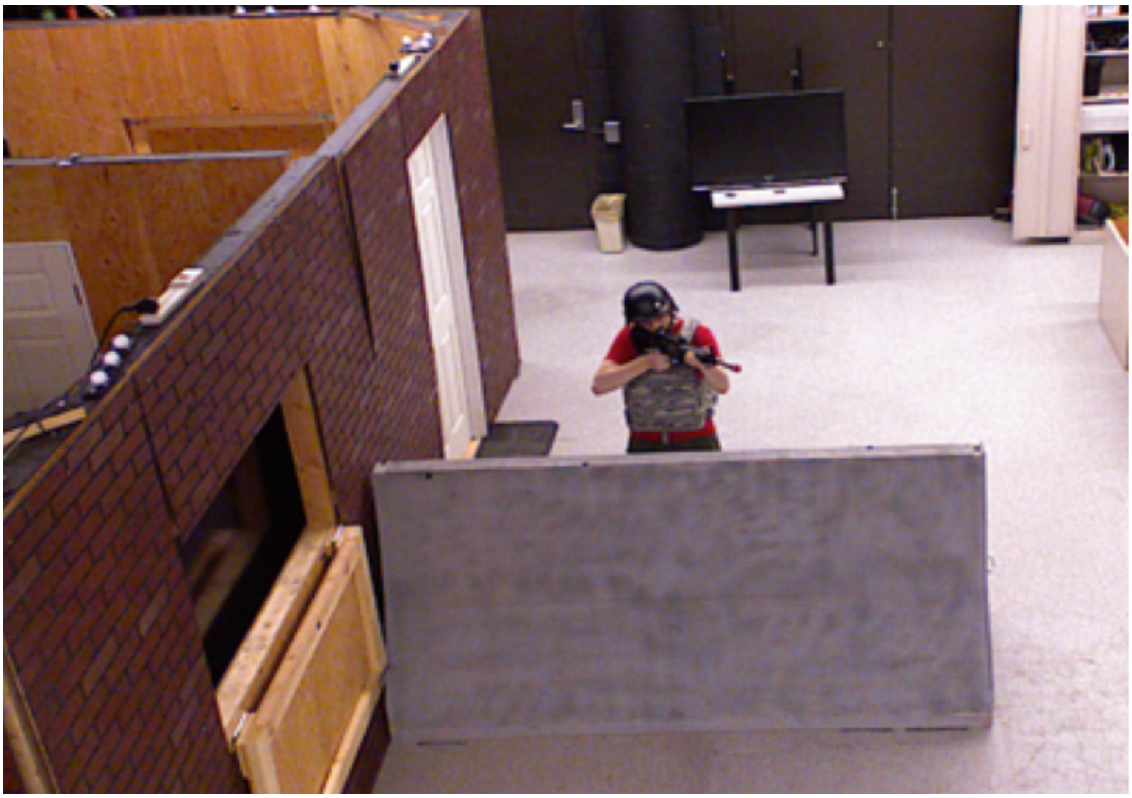

Figure 3: Part of the MIRAGE MR training environment

with the U.S. Army and taking cues from Hollywood set design, it features twelve moveable wall sections with interchangeable facades. These sections can be put together to form a variety of rooms and urban scenarios in the 3D-tracked 40' x 40' physical environment. The virtual world is presented on multiple stereo displays, including several mobile displays that simulate windows as well as a large, fixed 39' x 12' stereo display that forms the back wall of the room. A photo of the MIRAGE, showing moveable wall sections, is shown in Figure 3. Motion tracking in the MIRAGE is a challenge due to its large area and moveable walls. A configuration of its ceiling-mounted MotionAnalysis optical tracking system that is able to track nearly all areas of the room may become less much effective when the walls are rearranged, due to new areas being occluded to different tracking cameras. The challenge of tracking a dynamic environment is one of the issues the research presented in this article will address.

3D Tracking systems

There are a variety of technologies used for 3D motion tracking. Each has a number of advantages and disadvantages [12]. Magnetic tracking systems provide high accuracy that is unaffected by occlusion. However, stray magnetic fields and metal in the environment can degrade its accuracy [13]. GPS (Global Positioning System) works well outdoors and is not constrained to any particular area, but its precision is much too low for VR or MR applications [14]. Optical tracking systems, such as those by Advanced Realtime Tracking [15] require less work to set up and provide highly accurate, high-speed tracking, but are limited by occlusion of tracked objects by other objects in the tracked space, such as walls and people. Finally, hybrid tracking systems combine two or more types of tracking technology. Intersense's IS-9000 tracking system [16], for example, uses a combination of ultrasonic and inertial tracking. Unfortunately, while hybrid systems use multiple modes of data input to increase performance, they depend on receiving data from each mode to function properly. For example, ultrasonic systems are sensitive to occlusion as well as atmospheric conditions such as temperature and humidity [17], and thus the IS-900 is also subject to these limitations.

Tracking Challenges in Training Environments

Physically-based simulations present a challenging environment for modern tracking systems. Tracking for a physical simulation must have a high update rate to keep pace with quickly moving trainees. It must also be able to handle occlusion from walls and other objects that may be part of the training simulation. Accurate tracking is necessary for the realism of a simulation, and finally, a relatively large number of objects must be tracked: people, tools, simulated weapons, and other objects. Based on the strengths and

weaknesses of different types of tracking systems, it makes sense to use different types of tracking systems in the situations for which each is best suited. However, using multiple tracking systems means the simulation application(s) must be able to address different types of tracking hardware and be able to properly "follow" objects as they are "seen" by one tracking system and then by another. If positional data for an object is received from different tracking systems at the same time, a decision must be made regarding which data to trust, or if the measurements should be aggregated in some manner.

Abstraction Layers

Several software systems currently exist for abstracting tracking and other input data, freeing the user from dealing with drivers and SDKs (Software Development Kit). The VRJuggler virtual reality framework [18] is a set of libraries designed to abstract virtual reality applications away from the hardware in which it runs. One of its main components, Gadgeteer, is designed to interface with a variety of devices, including most popular 3D tracking systems. This input data is then accessed within a VRJuggler-based application by specifying a name that has been associated with a particular tracked object or other input channel. Similarly, the VRPN (Virtual Reality Peripheral Network) [19] also abstracts input device data, presenting the user with a single programming interface for obtaining that data within an application. Unlike Gadgeteer, VRPN does not require an application to use the VRJuggler framework. While they abstract and simplify the use of tracking systems, neither Gadgeteer nor VRPN aggregate tracking data. In other words, the user must specify which tracking system is to track a specific object. For multiple tracking systems to truly work together seamlessly, the abstraction system controlling the hardware must handle cases where an object might be seen by more than one tracker (alternately or simultaneously) in a way that doesn't require code-level user intervention.

METATRACKER

Overview of MetaTracker

MetaTracker is a system of systems (SoS) designed to not only abstract hardware, but to also combine data from multiple 3D tracking systems, such that they can be treated as a single system. As much as possible, it simplifies the process of getting 3D tracking data to an application, without requiring the application to be aware of the individual systems or their drivers. Figure 4 shows the three main components of MetaTracker and how they interact with one another. All three components communicate via UDP (User Datagram Protocol) packets sent over the local network.

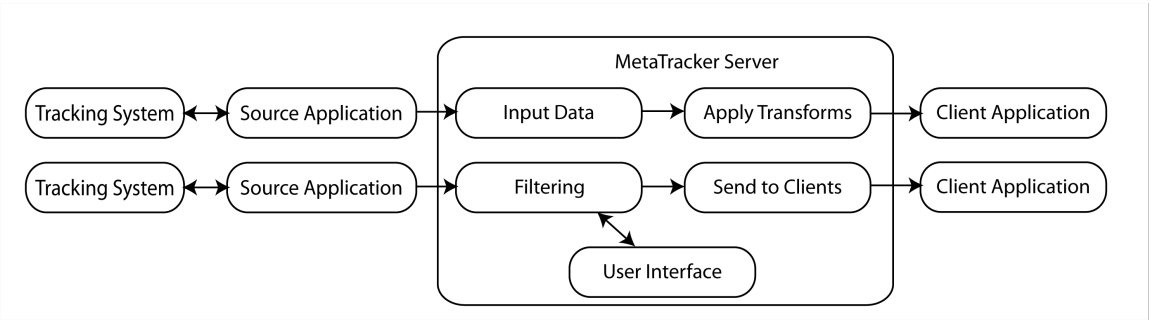

Figure 4. The components of MetaTracker, showing a pair of tracking systems whose data is accessed by multiple clients.

The first component of MetaTracker is the source application. This program utilizes the SDK and drivers supplied with the tracking system to read data from it in the tracking system's native format. A source application only needs to be created once for a particular type of tracking system. Position and orientation data for each object currently being tracked is set to the MetaTracker server as soon as it becomes available from the tracking hardware. In the case of articulated tracked objects, such as humans wearing motion capture suits, the transformation data for each "bone" is sent to the server. Depending on the functionality available in the tracking system's SDK, the source application may also be to control the system, performing functions such as turning it on and off or telling it which objects to track.

The next, and most complex component of MetaTracker is the server, the user interface of which is shown in Figure 5. MetaTracker's server receives raw tracking data from each source. The data is then sent out to all client applications that have requested data for the room containing the object. In most cases, the new data is used as-is, and overwrites the previously known position and orientation of the tracked object. However, if the server is receiving data for an object from multiple sources, it applies an error correction filter designed to smooth transitions as an object moves from one system's trackable area to the next. This is discussed further in the Challenges section below. Other situations where data is modified before use include when the source tracking system is using a different coordinate system than the other systems, or when an object is flagged as "clamp to ground." This flag is used for objects, such as moveable structures, that normally stay on the ground and are constrained to movement along the horizontal axes and rotation only around the vertical axis.

Another data flag available in MetaTracker is the "persistent" flag. This is for situations when a training simulation needs to know the position of a scenario object but the object may not actually move during the simulation, such as in the case of moveable walls and other props. Objects marked as persistent have their data stored in a file, and the data is reloaded when the server is launched. Additionally, new clients are

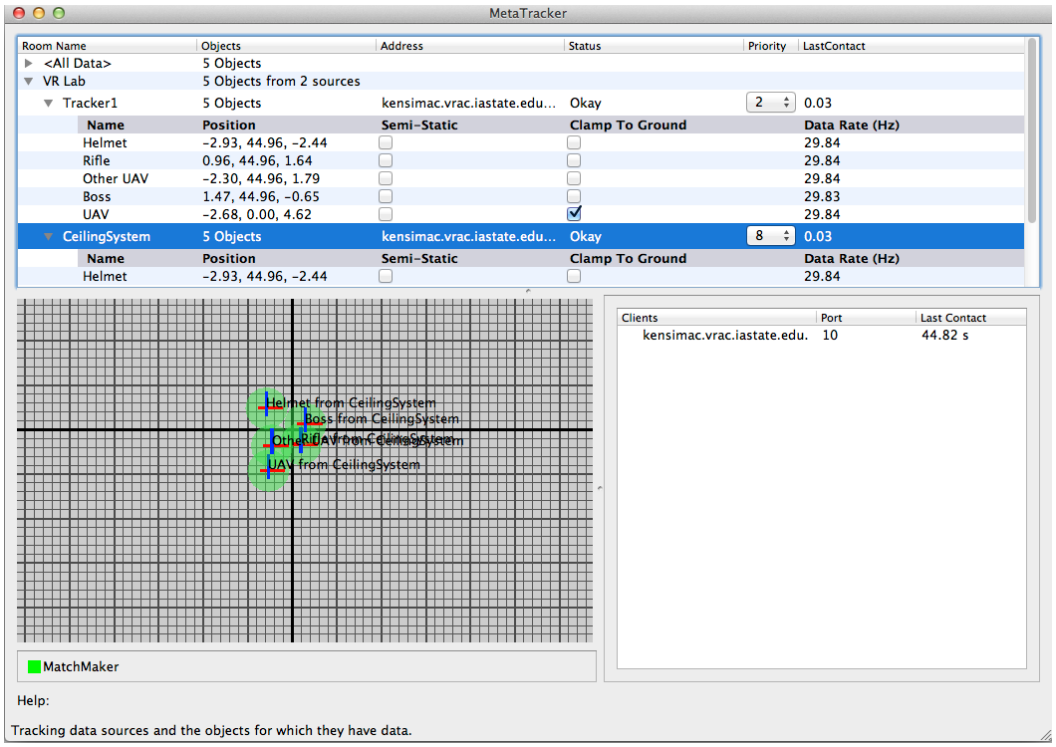

Figure 5. MetaTracker Server's user interface

automatically sent the current positions of persistent objects when they first connect to the server, regardless of whether any tracking system is currently tracking the object.

The last component in MetaTracker is the client. The client is any application which registers itself with the MetaTracker server and receives data from it. Clients are able to request data from specific tracking systems and specific rooms, or to get all data. In addition, they can specify which types of objects they want to track: 3 DOF (degrees of freedom) markers, 6 DOF props or articulated parts. New tracking data is sent to all clients as soon as it is received from the server.

From a programming standpoint, integrating the MetaTracker client software into an application is relatively simple. The MetaTracker client code, written in C++, can be compiled as a library file or simply compiled into the application itself. A sample program, showing the process of connecting to the MetaTracker server and printing tracking data to the console, is shown in the Appendix.

Usage Examples

Single Fixed System

This is a simple use case for MetaTracker. While the data aggregation aspects of MetaTracker aren't used here, MetaTracker can still provide multiple applications with simultaneous access to tracking hardware, as well as persistent tracking data and the clamp-to-ground functionality. In Iowa State University's MIRAGE training environment, which has moveable wall sections that can be used to create

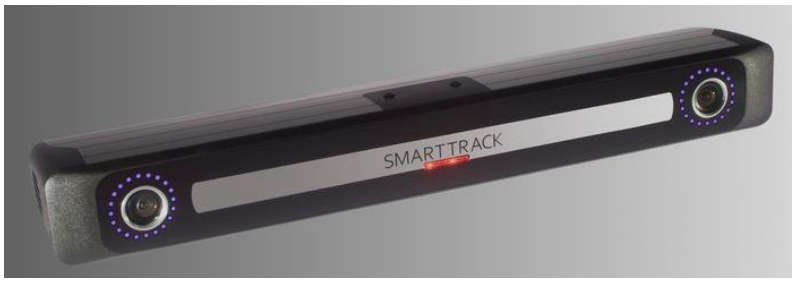

Figure 6. The ART SMARTTRACK is a self-contained tracking system [19]

different rooms and structures, a fixed, ceiling-mounted tracking system is able to track a large area of the room if few wall sections are in place to occlude objects. Training scenarios and user studies can then benefit from having access to tracking data without needing to connect directly to the tracking system. Furthermore, the MetaTracker server application's data display allows the simulation operator to monitor the tracking data and notice very quickly if problems arise due to occlusion or other issues with optical tracking markers.

Multiple Fixed Systems

MetaTracker's data aggregation becomes indispensable in a use case involving multiple tracking systems. If the area in which training takes place is relatively large or has areas occluded by walls or other structures, multiple 3D tracking systems may be needed to accurately track the players and other objects in the simulation. A user study conducted at ISU required users and simulated rifles to be tracked along a 20-foot long path [20]. No tracking hardware was on-hand that could perform this task, so three tracking systems were positioned adjacent to one another. In this case, each system was calibrated to the same coordinate system, and then MetaTracker received data from the different systems and supplied it to software clients as if it were from a single system.

Moveable Systems

Recently introduced single-piece tracking systems, such as Advanced Realtime Tracking's SMARTTRACK [21] system seen in Figure 6, can track objects relative to the hardware without any global calibration at all. Instead, they can measure positions relative to the center of the tracking system itself. By attaching tracking targets to one of these systems, it can itself be tracked by another tracking system. This allows the position data for objects tracked by the single-piece system to be transformed, via a simple matrix multiplication, into the coordinate system used by MetaTracker. As long as the single-piece tracking system can be seen by another tracking system, the trackable area of the simulation can be changed in real-time, without the need to reconfigure any tracking hardware. This is useful in cases where a change in the scenario, such as rearranging moveable structures, causes unexpected occlusion problems in part of the

tracked area. Moving a portable tracking system into the area can provide or restore motion tracking capabilities with minimal to no interruption to the scenario or simulation.

CHALLENGES

In the implementation and testing of MetaTracker, several challenges were encountered. This section discusses those challenges and the steps taken to address them.

Verification and Validation

The development of MetaTracker focused on using 3D tracking systems to measure the location of objects and sending that data from the source applications to the server and finally to the clients. In early development, when the basic networking and data sending was being tested, this could be accomplished with a tracking system repeatedly reading the position of a stationary object, and the source application sending that data to the MetaTracker server. However, when it came time to test other aspects of MetaTracker, such as the graphical visualization on the server, multiple moving tracked objects were needed. Initially, volunteers were recruited to walk around the trackable area, wearing helmets outfitted with optical tracking markers. While this proved feasible in the very short term, it quickly became clear that humans were not a viable source of repeatable tracking data.

To alleviate this problem, "virtual" tracking data was developed. A virtual tracking system was created that simulated objects moving in simple ways: either in circles, or wandering to a series of randomly generated waypoints. These simulated objects provided an approximation of the movements of the volunteers previously needed for testing. The virtual tracking system used the same MetaTracker client code as would be used with a real tracking system, sending position updates to the MetaTracker server. The virtual tracking system greatly helped speed up testing of the MetaTracker server by providing tracking data for many moving objects. This allowed an opportunity to refine and tweak many aspects of MetaTracker, such as the user interface and data display of the server component and management of network connections across all three components, from a single desk, and greatly reduced the amount of walking around needed for development.

Tracker Disagreement

One potential pitfall to using multiple tracking systems is when two tracking systems both see an object and report its position. Although each tracking system may be calibrated, there will usually be some difference in position measurements. The MetaTracker server helps to reduce problems caused by this error by smoothing object motion when multiple tracking systems are reporting data for the same object.

With error correction turned off, MetaTracker automatically sends to clients the last known location of an object. A motionless object seen by two tracking systems will, when displayed in a simulation, appear to jump back and forth between the two reported locations, as neither system will report the same location for the object at every time step. An object moving from an area tracked by one system to an area tracked by another will also jump back and forth while in the area of overlapping tracking coverage.

To address this problem, MetaTracker implements a trust-based data filtering system on a per-object level. Each object tracked by the server stores statistics on what tracking systems, or sources, it has received data from as well the average rate that those trackers send data. Each source is also assigned a trust value by each object, with the trust values from all sources summing to one. When an object is updated with data from a particular source, the trust value associated with that source is increased with all other sources losing a proportion of their trust. The resultant output position (D) is the linear interpolation of the previous and new positions, with an interpolation constant equal to the trust associated with the data source, as illustrated in equation 1:

$$\vec{D_1} = \vec{D_0}(1 - T) + \vec{D_{in}}(T) \tag{1}$$

where T is the trust value of the source from which the new data was received. Similarly, the new rotation is computed using a spherical linear interpolation (slerp) function. In situations where multiple trackers are used, it is useful to know which tracking systems are more likely to provide higher quality data than others. For example, a permanently installed, professionally calibrated tracking system will likely have better accuracy than a portable system. To support this, MetaTracker uses a priority system to guide trust assignment. Within the MetaTracker server's user interface, each tracking system is assigned a priority value, from 1 (lowest) to 10 (highest) with duplicate values not allowed. If the server is receiving data for a single object from more than one source, it may reject data from lower priority sources. Data is rejected if all of the following conditions are met:

1. The source the data came from has a lower priority than the object's most-trusted source
2. The most-trusted source has sent data for this object within 200 ms
3. The most-trusted source has not "missed" sending data three times in a row (based on the average rate at which it sends position data).

The first condition is in place to prevent an object from immediately trusting data from a new tracking system unless that system has a higher priority than the system from which it last received data. The

second and third conditions prevent a minor network hiccup or very brief software hang from causing an object to jump to another tracker's reported position. The numerical parameters used in conditions two and three were determined experimentally for smooth and accurate object motion when multiple trackers were present. Whether or not the data from a source is rejected, the source's trust value is still increased, to a maximum value of 1, by equation 2:

$$\Delta_{trust} = 0.1 + 0.02P \tag{2}$$

where P is the priority value of the data source. Following this, the trust values for each tracking system the object has received data from are reduced by using equation 3:

$$T_1 = T_0 \div (1 + \Delta\text{trust}) \tag{3}$$

to maintain a total trust value of one. As can be seen from equation 1, higher priority sources gain trust faster than lower priority sources. If an object is in an area of overlapping tracker coverage, it will place higher trust in the higher priority tracker and disregard information from the lower priority tracker. In the case that a tracking system's software reports it is no longer tracking an object, that message is relayed to the MetaTracker server, and the object in question's trust value for that tracking system is immediately set to zero.

Quantitative Evaluation

The other big challenge of implementing MetaTracker was finding quantitative methods to evaluate its functionality. While tracking data itself is quantitative, its effect on training simulations is difficult to measure. However, tracking data can be evaluated in terms of its accuracy and how up-to-date it is. Knowing this, the decision was made to evaluate MetaTracker based on how much error it introduced to the data on top of any pre-existing measurement error, and how much additional latency it introduced to the immersive system.

Latency Measurement

End-to-end latency in a simulation is the sum of four factors: Tracking system delay, application processing delay, image generation delay, and display system delay [22]. While latency introduced by MetaTracker will fall into the first of these categories, it is very difficult to directly measure a tracking system's delay, as it requires finding the time delay between an external event (an object begins to move) and an application event (data is updated). In order to avoid inaccuracies arising from physical measurements of moving objects, MetaTracker's latency was initially measured with purely digital objects.

A version of the virtual tracking system discussed earlier was modified to incorporate a MetaTracker client. The StressTester application, as it was called, simulated objects moving in simple patterns on the X-Z plane, and when sending data to MetaTracker, used the Y-coordinate of the object's position to store the precise time at which the object's data was sent. When tracking data was received by the client part of the application, the computer's current time was compared with the Y-coordinate of the incoming data to find the round-trip time. This process was repeated for a variety of conditions, and results are discussed in the next section of this article.

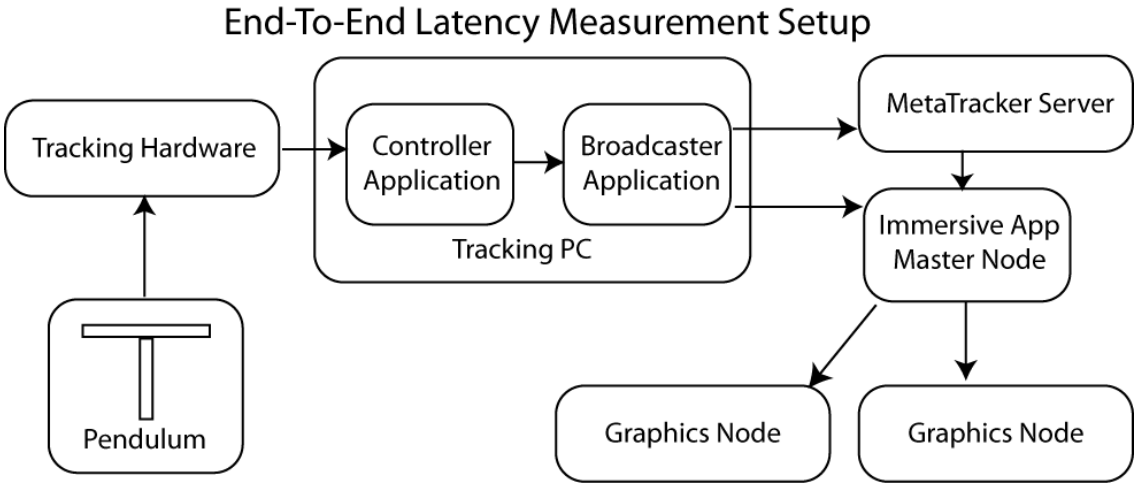

Figure 7. Network architecture used to test latency. Tracking data was sent from the broadcaster application to the immersive application with and without the MetaTracker server. Times were then compared.

Real-world latency testing of MetaTracker used an end-to-end latency measuring technique similar to that of Steed [23]. The tracking system used was built by MotionAnalysis, and was controlled by a dedicated Windows PC, as diagrammed in Figure 7. An application running on the PC received data from the tracking system where it could be sent over the network directly to an immersive application, to the MetaTracker server, or both A block of wood with optical tracking markers affixed to it was attached by four pieces of fishing line to an overhead mount in such a way as to create a pendulum. Behind the pendulum, a projection screen displayed the last measured position of the pendulum using both the MetaTracker and the directly received data, as seen in Figure 8. Both the real and virtual pendulum had marks to indicate when they crossed the lowest point in their path. The experiment was recorded with a slow motion (120 frames per second) camera. The resulting movies were analyzed by counting the number of frames of video between the real and virtual pendulums passing over their respective center points. The resulting value provided the end-to-end latency of the immersive system, both with and without MetaTracker. The additional latency introduced by MetaTracker could then be determined by finding the

difference in end-to-end latency for the two cases. Because the experiment used a single tracking system and a single application, the other sources of latency (internal tracking system delay, application processing, and image generation/display) did not factor into the difference in calculated latency as they were assumed constant for the two different data routes.

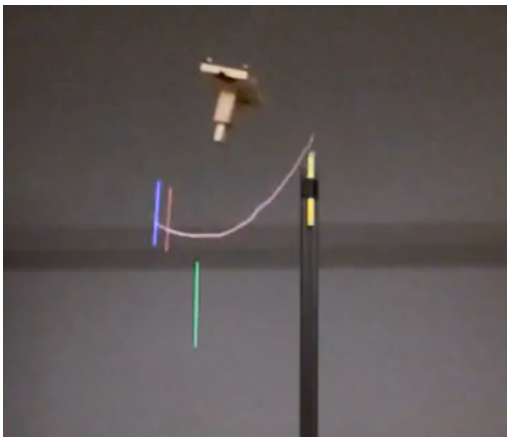

Figure 8. Captured video frame from the pendulum experiment. The red line indicates data from MetaTracker, and the blue line shows data sent directly to the application.

Error Measurement

Because MetaTracker does not currently make any attempt to correct errors in incoming data, the quality of its output data is dependent on the quality of its input data. Early testing of MetaTracker showed that when an object was visible to two tracking systems, it would be reported to be at two different alternating positions and appear to repeatedly jump back and forth between them. This jumping was clearly detrimental to a simulation, so the data filtering technique introduced in the "Tracker Disagreement" section above was implemented to reduce it. For simplification of visualization, only positional error was examined. Each tracking system was assumed to have a fixed "error vector" resulting from calibration error that was added to the actual position of each tracked object. Therefore, an object moving from the trackable area of one system to the next would start with its reported position offset from its actual position by the first system's error vector, and at the end of the movement, its reported position would be offset only by the other tracking system's error vector, as illustrated in Figure 9. The transition between these two error vectors should be as smooth as possible so as not to interfere with the immersive simulation. To quantify this transition, the magnitude of the change in the error vector between reported data points was calculated and summed for a pre-determined object motion. The sum of changes in the error vector was dubbed the summed delta error (SDE). An object jumping back and forth between two positions would hypothetically show a very high SDE while an object smoothly transitioning from one tracking system's error vector to the next would have a much lower SDE. Because tracker updates are not necessarily synchronized with visual updates of an immersive simulation, the SDE is calculated in two ways: For every position reported by MetaTracker (total SDE), and for every graphical frame displayed in the simulation (observed SDE).

Error Measurement with Virtual Data

Once again, virtual tracking systems were used to provide replicable object trajectories, but this time a set of simulated objects moved independently in a virtual area with several virtual tracking systems. To replicate the capabilities of real tracking systems, each virtual system had a built-in positional error offset, a fi*xed rate at which it sent data, and a limited area in which it could track objects. Thus, a given virtual tracking system would report d*ata for objects only when they were in its field of "view." By acting as a set of multiple tracker sources and a single client, this application, called AreaViz, could simultaneously display the actual path of an object, the paths of the object as reported by the virtual tracking systems, and the path as computed by the MetaTracker server. Recording this data allowed for the SDE to be easily calculated for the set of data points used.

Physical Error Measurement

To test MetaTracker's data filtering and verify the usefulness of the SDE calculations made with virtual tracking data, an equivalent experiment was needed that would provide the same data as had been obtained virtually, but with real objects. An object (a helmet outfitted with optical tracking markers) was placed on a cart and wheeled from an area covered by one tracking system to an area covered by another, crossing an area of overlapping coverage along the way, much like the shaded area in Figure 9. Both of these tracking systems were intentionally mis-calibrated by three inches along the x-axis, in addition to any calibration error. Because there was no feasible way to measure the "real" position of the object in real-time, a third tracking system, covering the entire experiment, was used to determine the tracked object's "correct" position. Finally, the MetaTracker server was configured to send the client not only its calculated position for each object, but the individual data points from each tracking system. This allowed the same information as had been recorded in the virtual experiment to be captured.

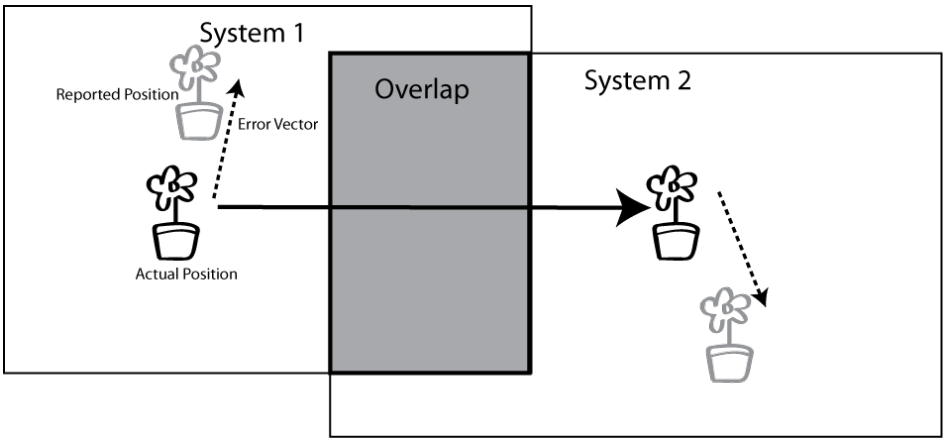

Figure 9. Illustration of the error in position measurement before and after an object passes through an area of overlapping tracker coverage.

Once the real-life data had been collected, the virtual experiment was re-run with its input parameters (tracker coverage area and object path) modified to more closely match the input parameters for the real experiment. This allowed for a direct comparison between the results of virtual and real testing, which are shown in the results section.

RESULTS

Error Measurement

To test the effectiveness of data filtering to reduce the side effects of tracker disagreement, the path of an object moving from the trackable area of one tracking system to that of another, with an area of overlapping coverage in the middle, was measured with both real and simulated tracking systems. The resulting position data from MetaTracker was shown, along with the raw data from the tracking systems and the "real" path of the object. Five different combinations of tracking system data priorities were used for the two tracking systems:

- 8(High) to 2(low)
- 2 to 8
- 5 to 5
- 5 to 6
- 6 to 5

Finally, the scenario was also tested without any data filtering. In each case, an object moved approximately 14 feet over approximately 8 seconds, with data sent at 37 Hz from the first tracking system and 31 Hz from the second. These send rates were chosen for two reasons: First, real-life tracking systems are unlikely to send data at exactly the same rate, so two prime-number rates were used to reduce the likelihood that the two virtual tracking systems would send data during the same iteration of the virtual testing program's running loop. Secondly, a data rate of 60 Hz (the peak rate available to the tracking systems) produced too many data points to visualize clearly. Both these rates, however, were higher than the minimum 30 Hz needed for real-time tracking, according to Ribo, Pinz, and Fuhrmann [24]. The movement was chosen to mimic a relaxed walking speed. Each tracking system had an error of 3 inches along the x-axis. The first in the negative direction, and the second in the positive direction. Images of these results are shown in Figure 10 for virtual data sent from the AreaViz program and Figure 11 for real-life data. The green line indicates the object trajectory as reported by MetaTracker and the white squares indicate object positions that would be seen in an immersive simulation running at 30 fps. SDE values are

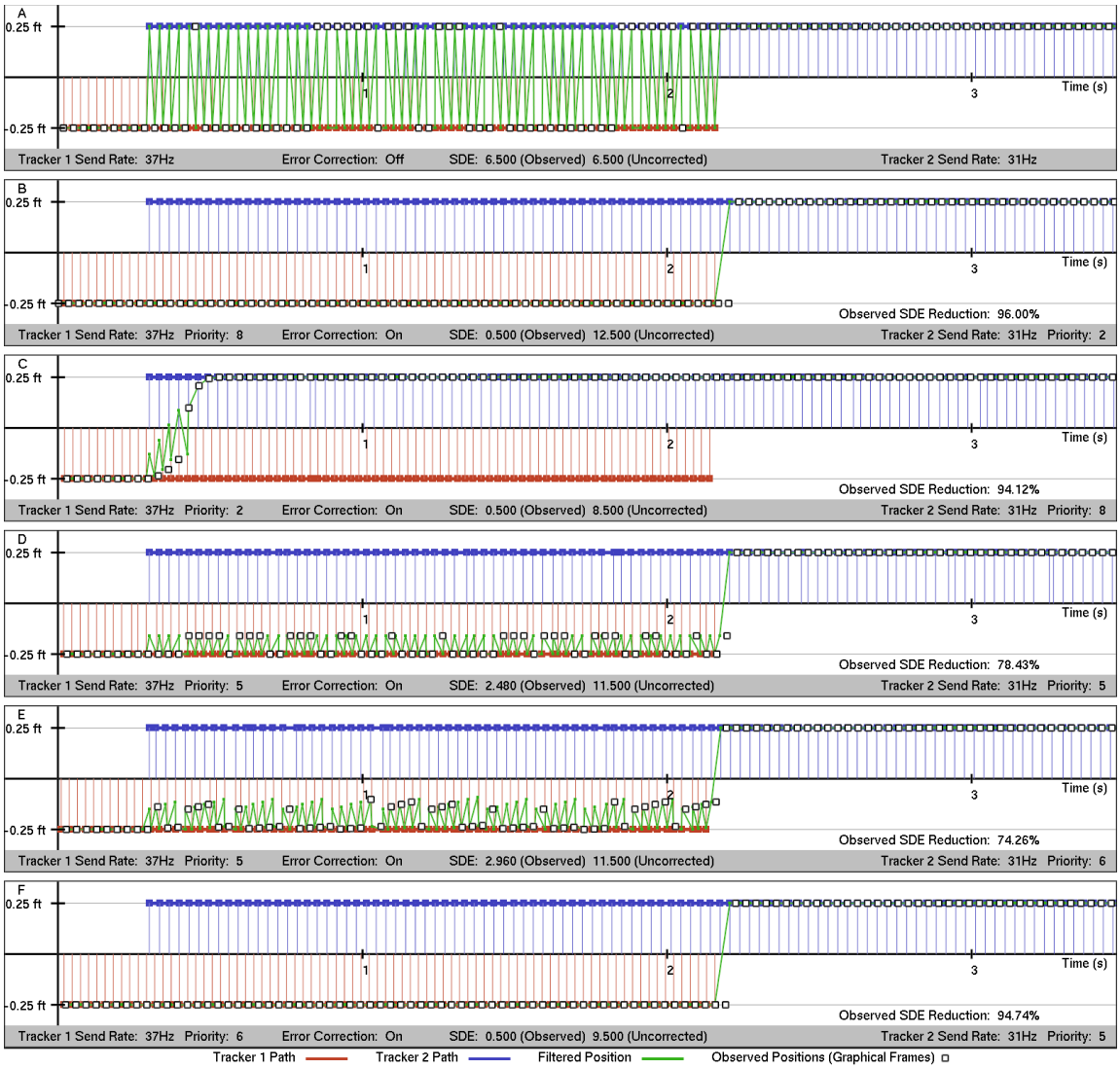

Figure 10a-f. Positional error of a virtual object over time as it moves along the z-axis and passes through an area of overlapping tracker coverage. The green path indicates the positions reported by MetaTracker, and the white squares indicate the position of the object during each graphical frame of the immersive simulation. The thin vertical lines show the temporal relationship of the tracker data. Finally, the priority values for each tracker are indicated on the diagram, along with the calculated observed SDE, the SDE without error correction, and the SDE reduction due to error correction.

shown in the center of the section for both the observed and non-error-corrected cases. The reduction in SDE with the use of error correction is indicated on each figure. The "jumping" behavior described above for objects in an area of overlapping tracker coverage without any data filtering is readily visible in part (a). In all cases in which data filtering is used, this jumping was reduced, if not almost entirely eliminated. This is further illustrated by the reduction in SDE, ranging from 70% to 91%.

In addition to the clear reduction in the jumping behavior, the real-life data shows very similar patterns to the virtual data gathered for the same object motion and tracked areas. For each trial, the resultant shape of the object's path and its SDE value are strikingly similar between the virtual and real trials.

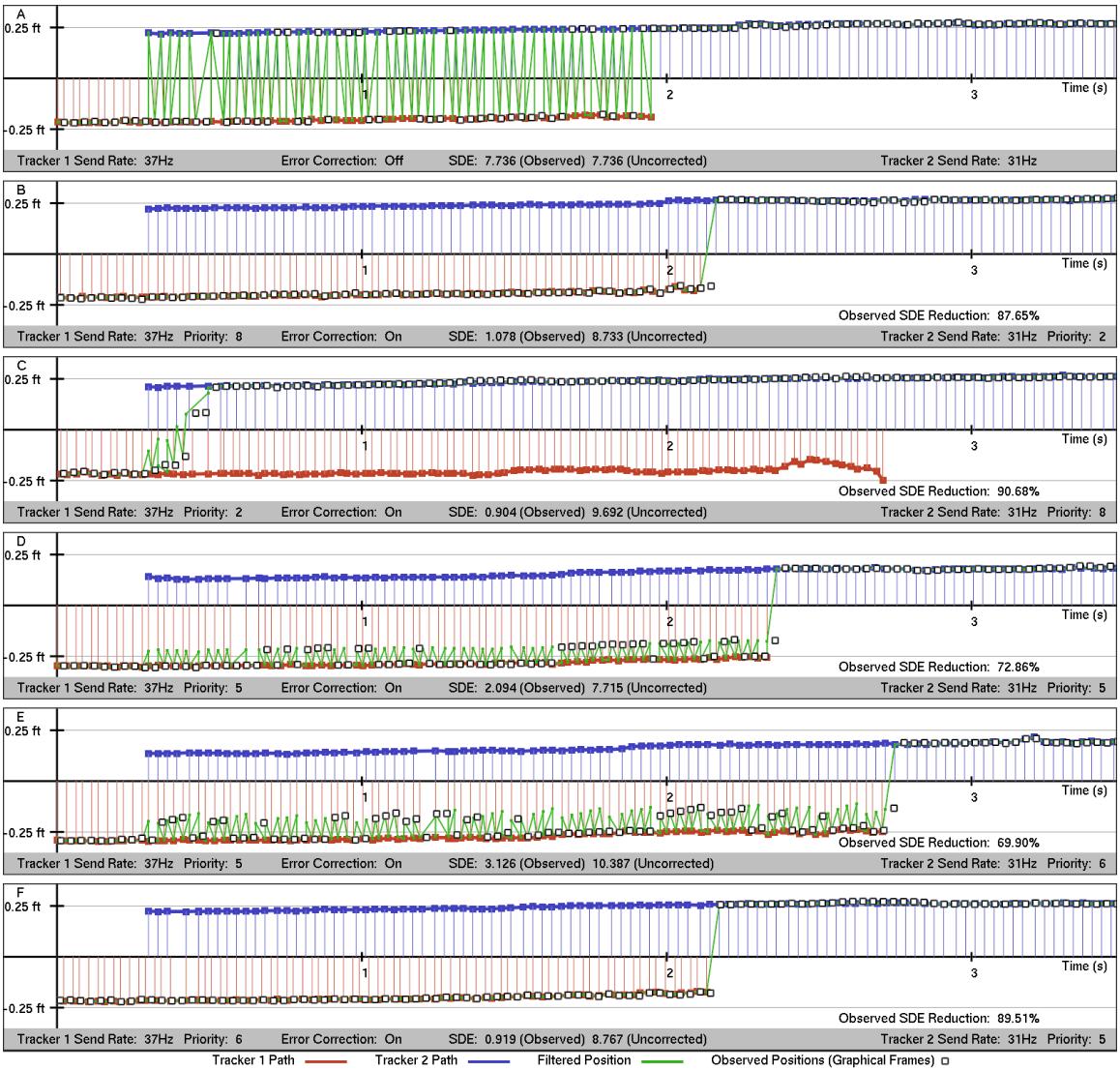

Figure 11a-f. Positional error along the x-axis of a real tracked object relative to a "real" position measured by a third, wide-area tracking system.

An unexpected feature of the data worth discussing is the behavior of the object in graph E of both figure 10 and 11. Although the second tracking system has a higher priority, the objects's reported position more closely matches that from the first tracking system. It is thought that the higher data rate of the first tracker caused its trust value for the tracked object to increase faster than that of the second tracker, despite the second tracker's slightly higher priority.

Latency Measurement

Latency measurement of the MetaTracker server consisted of using the StressTester application discussed above. Several conditions were tested, including varying numbers of objects from two to 50 (enough for a dozen simultaneous trainees, each with several tracked pieces of gear) and at varying update

rates. Trials were conducted both with and without other data sources sending tracking information to the MetaTracker server. Detailed results are shown in Figure 12 for one such case, with 25 objects tracked at 60 Hz, the maximum update rate provided by the ART SMARTTRACK [21] system. The MetaTracker server and client computers were connected over a non-isolated local area network (LAN) with a gigabit Ethernet connection. Several spikes in the data are visible, where latency-time increased very briefly. These number and location of these spikes appeared random over multiple runs of the simulation. As this experiment was run over a network used by many people, the spikes are believed to be external to the MetaTracker components. Although 25 objects were simulated and measured, for the sake of clarity the graph only shows the first 5 objects and the last object, in the order that their positions were sent to MetaTracker. Based on this graph, it becomes clear that the order in which object data was sent affects the its round-trip travel time. While the cause of this was not determined, this relationship is illustrated in Figure 12b. Further results with different numbers of simulated objects and different update rates are shown in Figure 13. The results shown suggest that (depending on network quality) the use of MetaTracker only

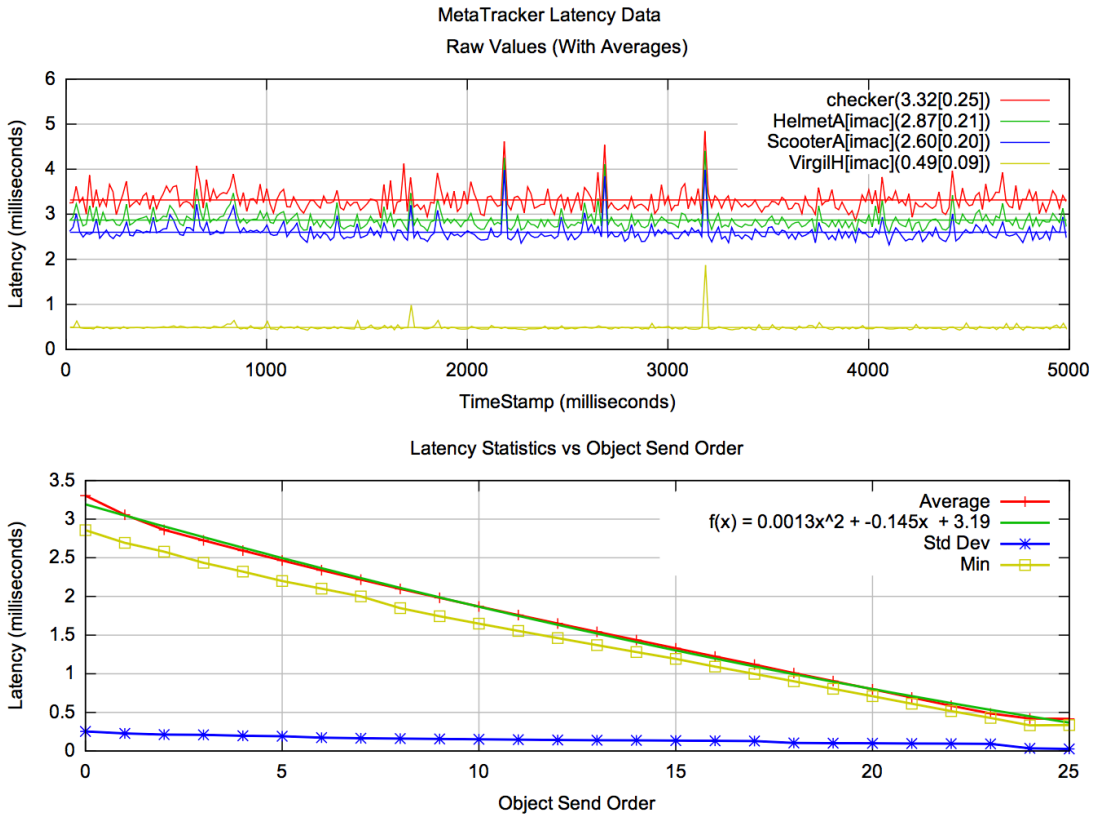

Figure 12a-b. Round trip times for data sent to and received back from MetaTracker. For this particular trial, 25 objects were simulated in StressTester, sending data at a rate of 60 Hz over a UDP connection. In part (b), the relation between the order object data was sent in is shown compared to its latency time.

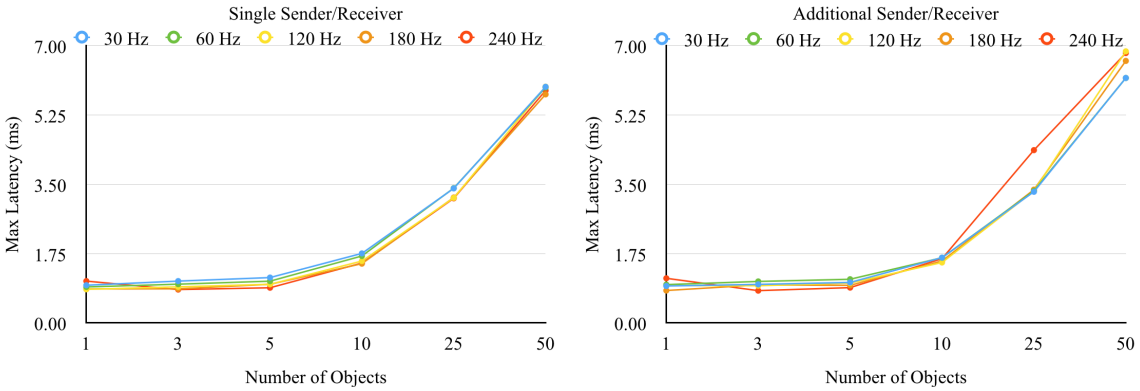

Figure 13. Average round-trip latency of a virtual object, by number of objects in the simulation and update frequency. Points shown are for the objects in each test with the highest average latency. Note: The second sender/receiver (right-hand graph) always ran with 5 objects at 60 Hz. The data shown is for the primary instance of StressTester.

adds 1-3 ms of tracking data delay for most use cases, a small value compared to the overall system delays of 50-60 ms found by He, et al [25].

Real-World Latency Testing

Using the setup described in Figure 7, several slow-motion videos were shot. To prevent the results from being influenced by whether tracking data was sent to MetaTracker first or to the immersive application first, one or the other of these cases were chosen randomly each time tracking data was to be sent. In the end, a dozen center-crossings for each video were recorded, and the results are shown below in Table I. The average end-to-end latency of the MIRAGE system was calculated to be between 71-83 ms. Unlike the round-trip network time determined with StressTester, this number includes time for the tracking system to determine object position and send data to the master computer, information to be propagated over the cluster, and for image generation and display. Tracking data sent via MetaTracker actually arrived slightly sooner on average than data sent directly from the tracking system, although not by a significant amount. So, these observed latency times, when using Metatracker, did not affect the running application at all.

Additional Benefits of Virtual Testing

As previously described, much of the development and testing of MetaTracker involved simulated data sources sending information for virtual objects rather than real tracking systems measuring the motion of physical objects. As these virtual testing methods were refined, a number of benefits to them were discovered in addition to their primary purpose of removing the need for a person to walk about, carrying a

tracked object. One major benefit was scalability testing: It became possible to test MetaTracker's behavior when working with several dozens objects from many different sources, even though only a handful of trackable objects and a maximum of three real tracking systems were available for real-life testing. Another benefit of virtual testing was repeatability: Moving virtual objects along a fixed path and sending position updates to the server at fixed points in time allowed for MetaTracker's data filtering to be tested, visualized, modified, and re-tested very quickly and reliably. This greatly sped up the process of developing and tuning the methods used. Finally, virtual testing with the StressTester application was able to help uncover several "Heisenbugs." These types of software bugs are so named because they seem to vanish when one attempts to investigate them [26]. MetaTracker suffered from several bugs that only manifested themselves after several days of ordinary operation. However, when StressTester was used to multiply MetaTracker's workload by a factor of one hundred, the bugs appeared within seconds, and fixing them became much easier. For example, one MetaTracker server bug resulted in the program crashing due to a memory error. This bug was infrequent, often occurring only if the program was left running for several hours. Using StressTester to simulate hundreds of tracked objects caused the crash to appear within seconds of launching the server, allowing much more rapid testing to track down the cause. It was quickly determined that the cause was a threading issue, and fixing it greatly increased the stability of the server.

Table I. Overall system delay The raw data is in frames (120 frames per second) and is converted to milliseconds for average and standard deviation values. The "MT"' header denotes data sent from MetaTracker.

	Trial 1			**Trial 2**			**Trial 3**			**Trial**		
	Direct	**MT**	**Delta**	**Direct**	**MT**	**Delta**	**Direct**	**MT**	**Delta**	**Direct**	**MT**	**Delta**
	10	10	0	9	8	1	9	9	0	9	9	0
	11	11	0	11	10	1	9	9	0	10	10	0
	9	9	0	11	11	0	10	10	0	10	10	0
	13	13	0	9	8	1	10	10	0	9	9	0
	10	10	0	10	10	0	8	7	1	7	7	0
	9	9	0	8	8	0	11	11	0	9	9	0
	12	12	0	10	9	1	9	9	0	10	10	0
	10	10	0	9	8	1	8	8	0	8	8	0
	9	9	0	8	8	0	9	9	0	9	9	0
	13	13	0	7	7	0	10	10	0	9	9	0
	10	10	0	8	8	0	8	8	0	8	7	1
	9	8	1	8	8	0	10	10	0	9	9	0
	9	9	0	8	8	0	7	7	0	7	7	0
	7	7	0	8	7	1	10	10	0	8	8	0
	8	7	1	8	8	0	11	11	0	7	7	0
Avg (ms)	82.78	81.67	1.11	73.33	70.00	3.33	77.22	76.67	0.56	71.67	71.11	0.56
StdDev (ms)	13.77	14.97	2.83	9.72	9.03	4.08	9.36	10.18	2.08	8.50	9.06	2.08

DISCUSSION

The results of the MetaTracker evaluation, both qualitative and quantitative, were very promising. From a quantitative standpoint, MetaTracker was able to handle a large amount of motion tracking data without

introducing significant latency, 1-3 ms in most cases. Although its current implementation of data filtering for multiple tracker situations isn't yet ideal for all cases, it is numerically and, more importantly, visually superior to the alternative: an object rapidly jumping back and forth between two positions. Even in the worst case, illustrated in Figure 11e, the use of data filtering cut down visible jumping behavior, with the summed delta error value going from 10.39 feet without filtering to 3.13 feet with filtering: a reduction of 69.9%.

Qualitatively, MetaTracker has opened up new possibilities for virtual and mixed reality applications and scenarios by enabling new, larger, and more complex areas to be 3D-tracked with available tracking hardware. Its hardware-agnostic client approach lets an application collect data from different tracking systems without any changes to code. However, there are still difficulties in the use of MetaTracker. One of these is the need to write a tracker controller application for each type of tracker that will collect the tracking data and send it to the MetaTracker server. Fortunately, many tracking hardware vendors include an SDK with a sample application for collecting data, which can be modified to act as a MetaTracker source application.

Overall, MetaTracker can bring increased flexibility to existing 3D tracking systems and allow easier access to the data these systems provide. The server application provides a place for easy inspection of current tracking data, and its options for data manipulation—clamp-to-ground and persistent objects—can help improve the quality of 3D training applications.

References

[1] P. Milgram, and F. Kishino, "A Taxonomy of Mixed Reality Visual Displays," *IEICE Transactions on Information Systems,* vol. E77-D, no. 12, December 1994, 1994.

[2] D. A. Bowman, and R. P. McMahan, "Virtual Reality: How Much Immersion is Enough?," *Computer*, July 2007, 2007.

[3] Universal_Studios. "Amazing Adventures of Spider-Man Ride in 3-D at Islands of Adventure," September 10, 2012; http://www.universalorlando.com/rides/islands-of-adventure/adventures-of-spider-man-ride.aspx.

[4] VRSim. "SimSpray," September 9, 2012; http://www.vrsim.net/simspray.

[5] J. Lee, M. Cha, B. Choi *et al.*, "A Team-Based Firefighter Training Platform Using the Virtual Environment," in The 9th International Conference on Virtual Reality Continuum and Its Applications in Industry, 2010.

[6] B. Pollock, E. Winer, S. Gilbert *et al.*, "LVC interaction within a mixed-reality training system." pp. 82890K-82890K-10.

[7] P. Zamora. "Virtual training puts the 'real' in realistic environment," August 30, 2014; http://www.army.mil/article/97582/Virtual_training_puts_the__real__in_realistic_environment/.

[8] M. Biagini, P. Trotta, and B. Joy, "Immersive Technology Supporting Individual and Collective Training," 2013.

[9] C. Schwetje, "Integrating Intelligence and Building Teams within the Infantry Immersion Trainer," Naval Postgraduate School, 2009.

[10] USMC. "Training Resources Online (Combat Convoy Simulator)," September 6, 2012; http://www.marines.mil/unit/basecamppendleton/Pages/TrainingResources/ccs.html.

[11] J. W. Shufelt Jr., “A Vision for Future Virtual Training,” in Virtual Media for Military Applications, Neuilly-sur-Seine, France, 2006.
[12] L. Yi-bo, K. Shao-peng, Q. Zhi-hua *et al.*, "Development actuality and application of registration technology in augmented reality." pp. 69-74.
[13] V. Lepetit, and P. Fua, “Monocular model-based 3d tracking of rigid objects: A survey,” *Foundations and trends in computer graphics and vision,* vol. 1, no. CVLAB-ARTICLE-2005-002, pp. 1-89, 2005.
[14] J. Grimes, "Global Positioning System Standard Positioning Service Performance Standard," D. o. Defense, ed., 2008.
[15] ART. "ART Advanced Realtime Tracking," May 17, 2013; http://www.ar-tracking.com/home/.
[16] InterSense. "Precision Motion Tracking Solutions," June 5, 2013; http://www.intersense.com/pages/20/14.
[17] J. P. Rolland, L. Davis, and Y. Baillot, “A survey of tracking technology for virtual environments,” *Fundamentals of wearable computers and augmented reality,* vol. 1, pp. 67-112, 2001.
[18] VRJuggler_Team. "The VR Juggler Suite," 3-13, 2013; http://vrjuggler.org/documentation.php.
[19] VRPN_Team. "VRPN," May 28, 2013; http://www.cs.unc.edu/Research/vrpn/.
[20] S. Gilbert, A. Civitate, J. Kelly *et al.*, "Comparing Training Performance With Vibrotactile Hit Alerts vs. Audio Alerts."
[21] ART. "SMARTTRACK - Tracking Systems," http://www.ar-tracking.com/products/tracking-systems/smarttrack/.
[22] M. Mine, “Characterization of end-to-end delays in head-mounted display systems,” *The University of North Carolina at Chapel Hill, TR93-001*, 1993.
[23] A. Steed, "A simple method for estimating the latency of interactive, real-time graphics simulations." pp. 123-129.
[24] M. Ribo, A. Pinz, and A. L. Fuhrmann, "A new optical tracking system for virtual and augmented reality applications." pp. 1932-1936.
[25] D. He, F. Liu, D. Pape *et al.*, "Video-based measurement of system latency."
[26] catb.org. "Heisenbug,"

APPENDIX: SAMPLE METATRACKER CLIENT CODE

The following C++ code shows how to connect to a MetaTracker server and print its tracking information to the console.

```
#include "TrackerClient.h"

int main(int argc, char** argv)
{
  float dt = 0.5;  //update rate, in seconds
  TrackerClient client;
  client.connectToServer("tracking.vrac.iastate.edu");

  while(1)
  {
    //update the client.  Handles incoming tracker data
    client.update(dt);

    //just print out the position
    printf("\n\n*********Tracking Data********\n");
    std::vector<TrackedObject*> objects = client.getObjects();

    for(size_t i = 0; i < objects.size(); i++)
    {
      TrackedObject o = *objects[i];
      QMath::Vec3 pos = o.transform.pos();
      printf("%-30s          %+04.2f   %+04.2f   %+04.2f\n",
        o.name.c_str(), pos.x, pos.y, pos.z);
    }
    printf("************************************\n");

    sleep(dt); //wait a bit before we do this all again
  }
  return 0;
}
```

CHAPTER 6: SUMMARY AND FUTURE WORK

This dissertation has explored simulated training using virtual and mixed reality, and has gone in depth into MRVC (mixed reality, virtual, constructive) simulations and the challenges associated with increasingly complicated simulations. In particular, focus was placed on the following issues: Rapid detection of hardware, software, and configuration problems; initialization, reconfiguration, and launching of simulations; and 3D tracking of relatively large areas subject to occlusion from moveable objects. The four software tools developed as part of this research, collectively referred to as the Mixed Reality Toolbox, or MRT, each addressed one or more of these.

The first issue, rapid detection of problems in a running simulation, is the only one addressed by all four MRT components. Each component of MRT provided an "inspection portal" to the state of the simulation and input devices at different points along the data path. Launcher's display of cluster node ping times and application running status verified basic network connectivity and provided a way to know if a particular node of a clustered application had crashed. Next, MetaTracker's server application displayed the state of all tracking systems connected to it, and the positions of each object they were tracking. GadgetProbe provided access to configuration and input device data received by the simulation itself. By displayed button states and object positions and associating them with the names used to identify each device in the VRJuggler configuration files, GadgetProbe could simultaneously verify that a given device was working and that it was properly "hooked up" within the configuration file. Finally, Overview received position data from MetaTracker and GadgetProbe and

attached it to 3D models representing tracked objects. These objects were then shown amongst other 3D models representing real and virtual elements of the simulation. The 3D models provided context to the tracking data and gave a more informative view of the simulation than could be provided with the abstract views of MetaTracker and GadgetProbe. Comparing data provided by the different components of MRT helped simulation operators to identify where the flow of data was interrupted, decreasing the time needed to identify and fix problems.

The next issue covered was initialization, reconfiguration, and launching of simulations. There were two major aspects to this problem: software configuration and launching, and synchronization of the real and virtual worlds. The Launcher component of MRT addressed the first of these aspects by providing a graphical user interface with which to choose the specific hardware on which to run the simulation. Launcher was able to generate shell scripts and VRJuggler configuration files, eliminating the need for the operator to create different scripts and configuration files for each specific use of the application.

The other aspect, synchronization of the real and virtual worlds, was addressed by Overview and MetaTracker. MetaTracker introduced the idea of persistent tracking data; that is, position data that is accessible even after the tracking system has stopped tracking the object in question. This persistent data allows for moveable objects, such as mobile walls and other props, to be repositioned and then tracked briefly, such as with active infrared markers. Then these positions can be later used for positioning the

corresponding elements in the virtual world. Overview allowed a user to define preset positions for the same sorts of moveable objects, as dictated by a particular training scenario. While these objects were being tracked, Overview showed both the user-selected preset positions and current tracked positions. This could be used to as a guide for placing the real objects in the required positions. A precise placement of the real objects wasn't necessary, as objects in the virtual world would be repositioned based on the real, tracked position.

The final issue addressed by this research was 3D object tracking in large, dynamic environments subject to occlusion. MetaTracker addressed these challenges by providing a mechanism through which multiple tracking systems could provide tracking coverage of a larger area from more angles than could a single system. In addition to this, self-contained systems, such as the SMARTTRACK, could be repositioned in response to changing tracking needs without recalibrating the tracking system or even interrupting the simulation. MetaTracker was evaluated for introduction of tracking system latency as well as for effectiveness of multiple tracker error resolution. It was found to introduce under 2 ms of latency in most circumstances, a very small number compared to the measured overall system latency of 71-88 ms in the MIRAGE. Additionally, its error correction methods reduced the jitter caused by naïve acceptance of data from different tracking systems by 73-91%.

Future Work

MRT has shown itself to be very effective at the tasks for which it was designed, but there is still some room for improvement. First, incorporating DIS packet decoding and display could greatly increase its usefulness for complicated scenarios by providing more context to its data view by displaying virtual and constructive entities. DIS decoding would also benefit the system operator by providing a direct, visual comparison between the input data from tracking systems and the results of its conversion to and from the DIS format. Next, porting Launcher and GadgetProbe to other VR and MR frameworks could extend its use to other VR facilities. Additionally, the basic concepts on which GadgetProbe is based could be integrated into other software. For example, an application designed for the Oculus Rift and Razer Hydra could transmit the state of its input devices (the Oculus' position and orientation, and the position, orientation, and button states of the Hydra) to a separate computer for viewing during user testing.

Taken together, the tools presented in this research help to simplify the use of complex mixed reality systems. They have each been used in a variety of situations and have been shown to be effective at their individual tasks. In the future, mixed reality will likely continue to become more important for training and other immersive experiences. Larger and more complex simulations will be introduced, and tools like MRT that serve to temper this complexity will become an important part of mixed reality. Clearly, the existing components of MRT are not a full solution to the difficulties associated with mixed reality, but they embody concepts, such as input data visualization, that will help to guide complex MR systems in the future.

ACKNOWLEDGEMENTS

This dissertation has been far too long in the making, and it wouldn't have happened without a lot of people. First, naturally, I have to thank my fiancé, Laura Funk. She's stuck by me for these past few years and encouraged me when I was a little discouraged, which anyone would do, really. More importantly, she's been willing to correctly point out that I was being an idiot during those times I wanted to quit and just be a woodsman or some other ridiculous occupation. She's also helped me to see a future that is awesome Not just some awful, nebulous real world, but something to be excited about!

Next is Emmy, for never judging me when I was frustrated, and for reminding me every day that there's a great outdoors, with adventures to be had, just in case I needed to take a break for an hour or six.

And Ted. Ted F. Martens. A lot of work that we do in grad school disappears into the void of unpublished research. That can be depressing. Working with Ted on Hexels, a product which has helped many people find an inner artist they had once lost, made me feel like I was doing some good in the world during my "what's the point?" moments.

And Eliot. There have been a few times when he, by all rights, should have given up on me, but he never did. Of course, I greatly appreciate that. But what had the greatest impact on me was standing in his office in 2008, telling him that a close family member in the Chicago area was unexpectedly in the hospital. I told him I wanted to drive out there as soon as possible, and that I was planning on leaving as soon as some fairly important task was finished. He looked at me and said "I'll take care of it. Go be with

your family. ” He didn’t say “you can leave, but get that done by the end of the week.”, he freed me from worrying about anything other than my family at the moment I needed it most.

Next is Grandmaster Pak. He taught me to always finish what I started, whether it was a earning Ph.D. or a black belt. As he’s fond of saying, “When you go to the bathroom, do you leave without wiping? No! You FINISH!”

I’ve worked with many fellow grad students over the years, and many of them have helped me get to where I am. In particular, Marisol Martinez-Escobar was a great co-worker for years on several projects, helping to make my time at VRAC much more pleasant. The work for this dissertation probably wouldn’t have been possible without the help of Anthony Civitate. He spent weeks drilling holes in ping pong balls and soldering infrared LEDs for the active markers in the MIRAGE, and helped me countless other times with data collection and demo setup. Finally, Joe Holub has been a great sounding board for my ideas and has been kind enough to proofread many of papers.

Finally, I want to thank my parents. Their unwavering support for over 30 years has inspired me and helped me become the man I am today.

BIBLIOGRAPHY

B. Pollock, K. K., S. Gilbert, J. de la Cruz, H. Gonzalez, and E. Winer (2011). "Putting it Together: A System for Real-time, Reconfigurable, and Distributed Live, Virtual

and Constructive Training."

Baar, J. v., T. Willwacher, et al. (2003). Seamless multi-projector display on curved screens. Proceedings of the workshop on Virtual environments 2003. Zurich, Switzerland, ACM**:** 281-286.

Benedetti, M. (2008). 102nd Security Forces: MOUT Training. Seagull. Otis ANG Base, 102nd Intelligence Wing, Massachusetts Air National Guard. **23:** 6-7.

Boeing (2007). Apache Longbow Crew Trainer(Product Card).

BohemiaInteractive. (2013). "VBS2 | Bohemia Interactive Simulations." from http://products.bisimulations.com/products/vbs2/overview.

Boothe, D. (2008). "Pendleton hosts newest DOD Combat Convoy Simulator." Retrieved September 10, 2012, from http://www.marines.mil/unit/basecamppendleton/Pages/News/2008/PendletonhostsnewestDODCombatConvoySimulator.aspx.

Brooks, F. (1999). "What's Real about Virtual Reality?" IEEE Computer Graphics and Applications **19**(6): 16-27.

CAE (2011). KC-135 Aircraft Training System (ATS).

Calytrix. (2013). "LVC Game--Introduction." Retrieved 3-6-13, 2013, from http://www.calytrix.com/products/lvcgame/introduction/.

Choi, B. (2010). "Flying the 787 Flight Simulator." Retrieved April 16, 2012, from http://www.boeing.com/Features/2010/08/bca_787_flight_sim_08_26_10.html.

Corporation, M. (2013). "trackd the Device Driver Software for Immersive Displays." Retrieved May 28, 2013, from http://www.mechdyne.com/trackd.aspx.

Costanza, E., A. Kunz, et al. (2009). Mixed reality: A survey, Springer.

Cruz-Neira, C., D. J. Sandin, et al. (1993). Surround-screen projection-based virtual reality: the design and implementation of the CAVE. Proceedings of the 20th annual conference on Computer graphics and interactive techniques. Anaheim, CA, ACM**:** 135-142.

CRYTEK. (2013). "CryENGINE." Retrieved May 28, 2013, from http://www.crytek.com/cryengine.

Currie, A. (2008). "McConnell now top KC-135 aircrew, follow-on training base." from http://www.amc.af.mil/news/story.asp?id=123123815.

CVL, U. (2012). "Cave2: Next-Generation Virtual-Reality and Visualization Hybrid Environment for Immersive Simulation and Information Analysis." Retrieved Sept 10, 2012, 2012, from http://www.evl.uic.edu/core.php?mod=4&type=1&indi=424.

Dahmann, J., R. M. Fujimoto, et al. (1997). The Department of Defense High Level Architecture. Winter Simulation Conference.

Dean, F., P. Garrity, et al. (2004). Mixed reality: A Tool for Integrating Live, Virtual, & Constructive Domains to Support Training Transformation. Interservice/Industry Training, Simulation, and Education Conference. Orlando, FL.

Deering, M. F. (1998). The Limits of Human Vision. 2nd International Immersive Projection Technology Workshop. Ames, iowa.

Delta3D. Retrieved 2-17, 2013, from http://delta3d.org.

DOD (2012). The Federal Budget. D. o. Defense. **2012**.

Filkins, D. and J. F. Burns (2006) "Mock Iraqi Villages in Mojave Prepare Troops for Battle." The New York Times.

Fu, D., B. Wu, et al. (2010). "Virtual Reality Visualization of CFD Simulation for Iron/ Steelmaking Processes." ASME Conference Proceedings **2010**(49392): 761-768.

General_Motors. "Cadillac News." Retrieved September 10, 2012, from http:// media.gm.com/media/us/en/cadillac/vehicles/xts/2013.html.

Google. (2013). "Google Glass - What It Does." Retrieved 2-27, 2013, from http:// www.google.com/glass/start/what-it-does/.

Gustavsson, P. M. and J. Wemmergård (2009). LVC Aspects and Integration of Live Simulatio. Fall Interoperability Workshop 2009. Orlando, FL.

Havok. (2013). "Havok." Retrieved May 28, 2013, from www.havok.com.

Henderson, S. J. and S. Feiner (2009). Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on, IEEE.

Hodge, R. W. (2003). Back To Basics: Development Of Firefighter Training Burn Building Guidelines For The Vermont Fire Academy.

Hodge, R. W. (2003). Back To Basics: Development Of Firefighter Training Burn Building Guidelines For The Vermont Fire Academy.

Huard, T. (2010). Angular Resolution: ASTR 288C: Lecture 6 presentation.

Jean, G. (2008) "Success of Simulation-Based Training is Tough to Measure " National Device.

LeapMotion. (2014). "Leap Motion For Virtual Reality." Retrieved 10-15-2014, 2014, from https://www.leapmotion.com/product/vr.

Lechner, R. and C. Huether (2008). Integrated Live Virtual Constructive Technologies Applied to Tactical Aviation Training. The Interservice/Industry Training, Simulation, and Education Conference. Orlando, FL.

Lechner, R. and W. Phelps (2002). CIGI - A Common Image Generator Interface. Interservice/Industry Training, Simulation and Education Conference. Orlando, FL.

Lee, J., M. Cha, et al. (2010). A Team-Based Firefighter Training Platform Using the Virtual Environment. The 9th International Conference on Virtual Reality Continuum and Its Applications in Industry.

MacLeod, M. J. (2013). Virtual Convoy Simulator Maximizes Training Time. Military.com.

Martin, G. (2013). "Laser Shot bringing space-age firearm training to Africa." Retrieved March 4, 2013, from http://www.defenceweb.co.za/index.php?option=com_content&task=view&id=29079&Itemid=109.

McCall, M. (2010). IEEE 1278 Distributed Interactive Simulation (DIS).

MetaVR. (2013). "MetaVR Virtual Reality Scene Generator (VRSG)™ Features." Retrieved May 28, 2013, from http://www.metavr.com/products/vrsg/vrsg-feature-details.html.

Microsoft. (2014). "Kinect for Windows." Retrieved 10-15-2014, from http://www.microsoft.com/en-us/kinectforwindows/.

Milgram, P. and F. Kishino (1994). "A Taxonomy of Mixed Reality Visual Displays." IEICE Transactions on Information Systems **E77-D**(12).

Moore, K. (2006). "A Potted History of Flight Simulation." from http://www.simuser.com/joomla/index.php?option=com_content&view=article&id=72.

Noon, C. J. (2012). A volume rendering engine for desktops, laptops, mobile devices and immersive virtual reality systems using gpu-based volume raycasting Dissertation, Iowa State University.

OculusVR. (2014). "The All New Oculus Rift Development Kit 2." October 20, 2014, from http://www.oculus.com/dk2/.

Orlansky, J. and J. String (1977). Cost-Effectiveness of Flight Simulators for Military Training. Institute for Defense Analyses--Science and Technology Division.

Pair, J., U. Neumann, et al. (2003). "FlatWorld: Combining Hollywood set-design techniques with VR." Ieee Computer Graphics and Applications **23**(1): 12-15.

Parviz, B., S. Lee, et al. (2012). "Project Glass." Project Glass https://plus.google.com/+projectglass/posts.

Powell, E. T. and J. R. Noseworthy (2012). The Test and Training Enabling Architecture (TENA), United State Department of Defence Test Resource Management Center.

Project, T. O. "Home." from www.openscenegraph.org.

Razer. (2014). "Razer Hydra Portal 2 Bundle." Retrieved 10-15-2014, 2014, from http://www.razerzone.com/gaming-controllers/razer-hydra-portal-2-bundle.

Robert Lechner, W. P. (2002). CIGI - A Common Image Generator Interface. Interservice/Industry Training, Simulation and Education Conference. Orlando, FL.

Schall, G., E. Mendez, et al. (2008). Virtual redlining for civil engineering in real environments. Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on, IEEE.

Schwetje, C. (2009). Integrating Intelligence and Building Teams within the Infantry Immersion Trainer, Naval Postgraduate School.

Sensics, I. (2012). "zSight™ Integrated SXGA HMD." Retrieved 9-10, 2012, from http://sensics.com/products/head-mounted-displays/zsight-integrated-sxga-hmd/.

Sheridan, A. (2010). "BOEING DELIVERS RECORD FIVE APACHE LONGBOW CREW TRAINERS IN 2009." Retrieved September 10, 2012, from http://www.boeing.com/apachenews/2009/issue_01/news_s20_p2.html.

Shufelt Jr., J. W. (2006). A Vision for Future Virtual Training. Virtual Media for Military Applications. Neuilly-sur-Seine, France.

SIM_Industries. "Sim Industries | Visual-system." Retrieved 1-29, 2013, from http://www.sim-industries.com/visual-system/.

Sixense. (2014). "STEM System." Retrieved 10-15-2014, from http://sixense.com/wireless.

Sony. (2014). "Sony Computer Entertainment Announces "Project Morpheus" - A Virtual Reality System That Expands The World Of Playstation®4 (PS4™)." Retrieved 10-15-2014, 2014, from http://www.sony.com/SCA/company-news/press-releases/sony-

computer-entertainment-america-inc/2014/sony-computer-entertainment-announces-project-morp.shtml.

Spitzer, C. R. (2001). The Avionics Handbook. United State of America, CRC Press.

Storms, E. (2012). "KC-135 Simulator to Help with Budget Cuts." Retrieved Sepember 10, 2012, from http://www.milpages.com/blog/1748584.

Summers, J. E. (2012). "Simulation-based military training: An engineering approach to better addressing competing environmental, fiscal, and security concerns." J. Wash. Acad. Sci., Spring.

Team, V. (2013). "VRPN." Retrieved May 28, 2013, from http://www.cs.unc.edu/Research/vrpn/.

Ternion. (2013). "FLAMES Overview." Retrieved 3-6-13, 2013, from http://www.ternion.com/flames-overview/.

Universal_Studios. (2012). "Amazing Adventures of Spider-Man Ride in 3-D at Islands of Adventure." Retrieved September 10, 2012, 2012, from http://www.universalorlando.com/rides/islands-of-adventure/adventures-of-spider-man-ride.aspx.

USAF. (2011, December 29, 2011). "Factsheets : KC-135 Stratotanker." Retrieved September 10, 2012, from http://www.af.mil/information/factsheets/factsheet.asp?id=110.

USMC. "Training Resources Online (Combat Convoy Simulator)." Retrieved September 6, 2012, from http://www.marines.mil/unit/basecamppendleton/Pages/TrainingResources/ccs.html.

USMC. "Training Resources Online (Indoor Simulated Marksmanship Trainer)." Retrieved September 10, 2012, from http://www.marines.mil/unit/basecamppendleton/Pages/TrainingResources/ismt.html.

USMC. (2008). "Marine Corps Concepts and Programs." 2012, from http://www.marines.mil/unit/pandr/Documents/Concepts/2008/CHPT3PRT7.htm.

USMC. (2012). "Training Resources Online (Indoor Simulated Marksmanship Trainer)." Retrieved September 10, 2012, from http://www.marines.mil/unit/basecamppendleton/Pages/TrainingResources/ismt.html.

Van Krevelen, D. and R. Poelman (2010). "A survey of augmented reality technologies, applications and limitations." International Journal of Virtual Reality **9**(2): 1.

Visser, D. S. (2007). MOUT--Urban Training Complex Standard Operating Procedures. I. A. N. Guard.

VRAC. (2012). "Virtual Reality Applications Center." Retrieved May 2012, 2012, from http://www.vrac.iastate.edu/c6.php.

VRJuggler_Team. "The VR Juggler Suite." Retrieved 3-13, 2013, from http://vrjuggler.org/documentation.php.

VRSim. (2012). "SimSpray." Retrieved September 9, 2012.

Wacker, F. K., S. Vogt, et al. (2006). "An Augmented Reality System for MR Image–guided Needle Biopsy: Initial Results in a Swine Model." Radiology **238**(2): 497-504.

White, C., J. Carson, et al. (1991). "Training and Effectiveness of an M-16 Rifle Simulator." Military Psychology **3**(3): 177-184.

Wikitude_GmbH. (2012). "Wikitude - World's Leading Augmented reality SDK." Retrieved September 10, 2012, 2012, from http://www.wikitude.com.

X-Plane. (2013). "X-Plane 10 Global." Retrieved May 27, 2013, from http://www.x-plane.com/desktop/home/.