Augmented Reality for Force Protection Systems

Sean S. Cullen, Richard Schaffer
Lockheed Martin RMS
Burlington, MA
sean.cullen@lmco.com
richard.l.schaffer@lmco.com

Susan Harkrider, Christopher May
NVESD
Ft. Belvoir, VA
info@nvl.army.mil

ABSTRACT

The Night Vision and Electronic Sensors Directorate (NVESD) develops sophisticated, trailer-based, deployable surveillance sensors and operator stations used to monitor areas near sensitive and hostile work locations. Effective training for system operators is limited due to the expense of providing training targets in the live environment. In collaboration with NVESD, Lockheed Martin has developed a prototype embedded Augmented Reality training capability that enables the injection of realistic virtual training targets into the sensor displays. The system intercepts the sensor data prior to the sensor display computer, inserts virtual entities and effects, and forwards the results to the sensor display. The pros and cons of various insertion points and the challenges encountered when trying to synchronize video images with the sensor meta-data are examined. The system integrates the Night Vision Image Generator (NVIG) technology developed at NVESD into the rendering system to ensure virtual renderings match multiple sensor types, and sensor data along with instructor controls are used to blend entities with different environmental conditions. Finally, an innovative approach was developed to detect and mask video overlays that allowed virtual insertions to appear to be behind the overlays. The resulting prototype provides robust training capability with dynamic virtual insertions at a fraction of the cost of live training scenarios.

ABOUT THE AUTHORS

Mr. Sean S. Cullen is a Senior Staff Software Engineer at Lockheed Martin with 20 years of experience in military simulation and a focus in augmented reality for the past 8 years. His work includes R&D projects at the Office of Naval Research, an Augmented Reality Joint Capability Technology Demonstration, and Virtual Reality R&D at Naval Research Labs. He received his Bachelor's Degree in Computer Science from Middle Tennessee State University.

Mr. Richard Schaffer is a Lockheed Martin Fellow and Principal Investigator at Lockheed Martin Rotary and Mission Systems (RMS). He leads the Human Immersive Simulation Lab at RMS's Advanced Simulation Centers. Richard received his S.B. degree from the Massachusetts Institute of Technology and has over 33 years of experience in modeling and simulation research and development. His areas of research have included distributed simulation, environment modeling, immersive simulation and augmented reality. In 2010 he received the NTSA's lifetime achievement award.

Ms. Susan Harkrider is the Director (A) of the Modeling and Simulation Division (MSD) at the CERDEC Night Vision and Electronic Sensors Directorate (NVESD). She has over 25 years of experience supporting M&S, training and systems engineering efforts for the DoD. She has served on the Simulation Interoperability Standards Organization (SISO) Executive Committee and Standards Activity Committee. Ms. Harkrider is currently the chairperson of the RDECOM M&S Senior Working Group (SWG), which facilitates collaboration of M&S efforts between RDECOM laboratories in support of PEOs and PMs. She holds a BSE and MSE from the University of Central Florida.

Mr. Christopher May is the Simulation Applications Branch Chief at the CERDEC Night Vision and Electronic Sensors Directorate. He has over 15 years of experience supporting electro-optic and infrared modeling, simulation, and training efforts for the DoD. He holds a BSE from Penn State University.

Augmented Reality for Force Protection Systems

Sean S. Cullen, Richard Schaffer
Lockheed Martin RMS
Burlington, MA
sean.cullen@lmco.com
richard.l.schaffer@lmco.com

Susan Harkrider, Christopher May
NVESD
Ft. Belvoir, VA
info@nvl.army.mil

INTRODUCTION

The Night Vision and Electronic Sensors Directorate (NVESD) funded research to develop a cost effective operator training capability for one of its tower based force protection systems. These force protection systems provide an elevated view of the area by placing the sensor on a deployable mast that can be placed near sensitive or hostile areas (Figure 1). These systems allow the use of multiple cameras that can see in varying wavelengths, enabling the operators to monitor activities. Imagery from these sensors is sent to remote operators who can control the sensor to locate threats or suspicious activity. The goal of this research program was to design and prototype an Augmented Reality (AR) capability for the sensor that could insert virtual vehicles, people, and weapon effects into the sensor's views. This capability could then be used to provide enhanced operator training or to enable the operators to participate in combined live and virtual training by networking the system larger scale training simulations.

Figure 1 Force Protection System

HISTORY

The program leveraged code and experience gained while developing an AR capability for the Office of Naval Research's (ONR) Augmented Immersive Team Training (AITT) program (Kumar, et al., 2013 and Schaffer, et al., 2015). The AITT program was focused on providing forward observer training with a two-in-one system that mounted to the trainee's helmet, and provided a camera image that was augmented and then displayed to the user with a helmet mounted display (Figure 2). The user was also able to raise an instrumented prop of Vector 21 binoculars that enabled them to see at 7x magnification. The two-in-one system could be mounted on a vest or attached to I-TESS equipment to allow enhanced force-on-force training (Schaffer, et al., 2015).

This prior AR work has some distinct differences from the force protection system. The sensor for the AITT program was designed specifically for AR and, as such, provided accurate position and orientation of the sensor that was synchronized with the sensor imagery. The data came from high frequency inertial measurements, GPS location, and video imagery that was combined using a Kalman filter. This provided the basis for tracking sensor position and orientation closely enough that the virtual insertions would appear stationary within the augmented video. Adding

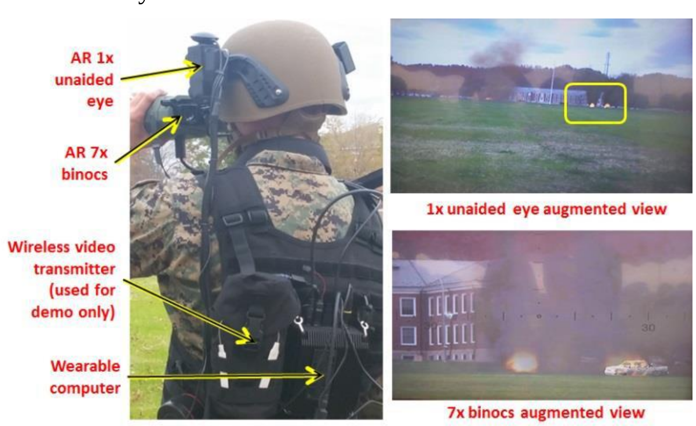

Figure 2 AITT Two-in-One System

custom sensors to the hardware was not possible with the force protection system due to the design of the sensor. Anything attached to the sensor would have interfered with its ability to properly rotate. Instead, a system was designed to use only the video and meta-data available and determine if it provides enough information to allow for stable insertions.

TRAINING

In the context of military training, AR promises to bring the best of virtual and live together (Defense Science Board, 2013, pp. 60-65). Training for this type of equipment might mean learning how to control the sensor, or how to adjust the camera's focus and zoom levels to properly identify nearby troops or equipment. Augmented reality insertions can be used to facilitate this type of training; an instructor places multiple threats in the area and tests to see how long it takes the trainee to locate and identify the threats for instance. This represents one level of training but it seems that, while useful, this type of training alone doesn't take full advantage of the advanced AR capability. In some situations it would likely not be too difficult to locate and identity real objects nearby to provide a similar level of training. There is some added benefit, as the instructor has the ability to create custom entities that are exhibiting a specific behavior that would otherwise require supporting personnel to replicate. In other situations AR would allow for placing entities in locations that training personnel would not be able to enter because of security concerns.

Augmented reality becomes even more valuable in combined arms training scenarios. When you can have multiple teams working together to engage and overwhelm an active threat, training value is increased. Augmented reality is currently being developed for both mounted and dismounted infantry units and has been experimented with for adding virtual insertions into the sensors for Stryker vehicles. The near future could see the system being developed here used as a combined training capability to drive a larger scale training scenario where the force protection operators spot a threat and communicate information to a quick reaction force. A squad using a helmet-worn AR system can also call in support from an AR enabled support vehicle. All participants are reacting to what they observe instead of being given a scenario by an instructor or verbal description of activities around a fixed dummy target. Additionally, with the use of AR, inserted vehicles can be programed to move and exhibit behaviors that can be used to enhance training value.

OVERALL APPROACH

The goal of augmented reality is to insert a virtual entity or weapons effect into the video stream with as much realism as possible, such that the resulting video would appear to the operator as if the vehicle was really there. Accomplishing this requires the creation of a virtual environment that matches the real world with as much accuracy as possible and a virtual camera that is synchronized in position, azimuth, elevation, and field-of-view with the real sensor. Creation of the virtual environment requires a 3d model of the terrain. This model was generated using elevation data downloaded from the USGS website. With augmented reality, the virtual terrain is not seen because the camera looking at the real world provides this imagery. The main difference between this program and the previous AR work was that the force protection system sensor had a larger viewing area compared to the sensors used by infantry. The force protection sensor is placed high on a tower, and with the zoom capability of the sensor it could easily focus in on an area 5km or more away. This meant the generated virtual environment was much larger than was needed for infantry standing at ground level. The generated terrain mesh is used by the system to conceal 3d insertions that would otherwise be obscured by the natural environment; i.e., a vehicle driving up to your position from behind a hill needs to be obscured from sight. The ideal system would generate this "depth" map with range sensing equipment in real-time from the camera's point of view, but that is not currently feasible. The downside of using a pre-computed terrain is that it does not contain moving objects such as live vehicles and people, or it may contain objects that have been removed.

Beyond occlusion, the virtual terrain is used as a platform for placing virtual objects. Elevation accuracy is important for virtual objects because any errors make virtual entities appear too high or too low in the scene. If a vehicle is placed 10 meters above the actual terrain it appears to be floating and an operator could quickly adjust to correct, but if a vehicle is placed 1 or 2 meters too high then they can appear farther away than they should. With the elevation error the rendering system is still using the correct distance for its perspective and the vehicle seems farther away but appears larger than it should. The same holds true for elevations that are too low where vehicles appear too small because they seem closer than they should.

In addition to the terrain this research focused on developing new technologies to provide AR in a way that would enable the operator to be unaware of the training computer to the point that if no virtual insertions were in the field of view then the screen would be identical to the system when not in training mode. These areas of focus included the ideal insertion point for the technology, the video capture and augmentation process, video overlay detection and masking, and research into how to provide detailed sensor tracking.

INSERTION POINTS

In order to provide the desired AR capability, the system needs access to the data coming from the sensor and, ideally, it also needs access to the operator's commands going to the sensor. The easiest way to do this is to insert the AR computer in the chain between the operator's station and the sensor itself. With the AR computer acting as a relay, it is able to intercept the data going back and forth and substitute modified versions of this data to provide the AR insertions. This system was unique in that it provided two different options for the insertion point. The first insertion point considered was to place the AR computer near the sensor and intercept video and meta-data in its raw form (Figure 3). The second insertion point is to intercept the data streams after they have been converted to network data for transmission to the operator's station (Figure 4). The near sensor insertion seemed like a much more reliable and predictable way to receive the data. With a video capture board, the incoming video frames could be sent directly to the video card's GPU by using modern video processing equipment designed for this purpose. It also means that the video signal has not been degraded by the conversion process to send the video over the network. The Network insertion point provides access to the same data, but with the possibility for packet loss and data bundling. Because of this the data is no longer going to be received with the same level of precision as the near sensor solution. This adds another layer of variability between the video data stream and the meta-data that contains the current sensor state like azimuth and elevation. Another side effect of this configuration is that the video data will be slightly degraded by the MPEG compression process. This degradation might have an adverse effect on computer vision techniques.

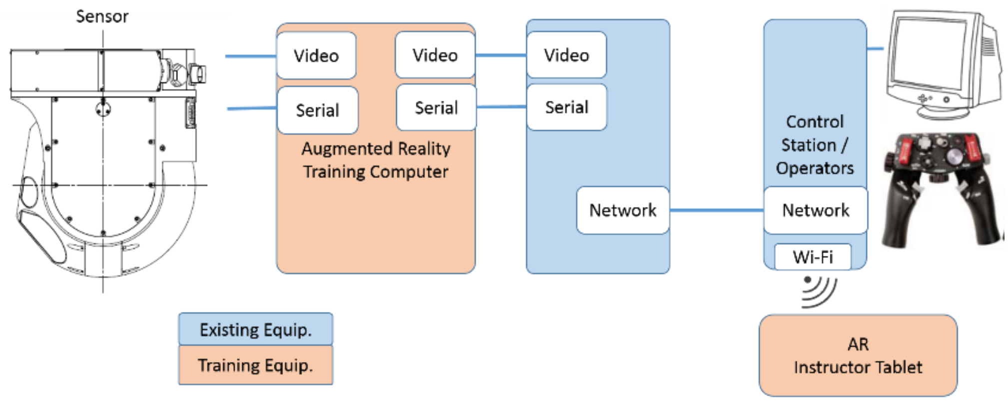

Figure 3 Near Sensor Insertion Point

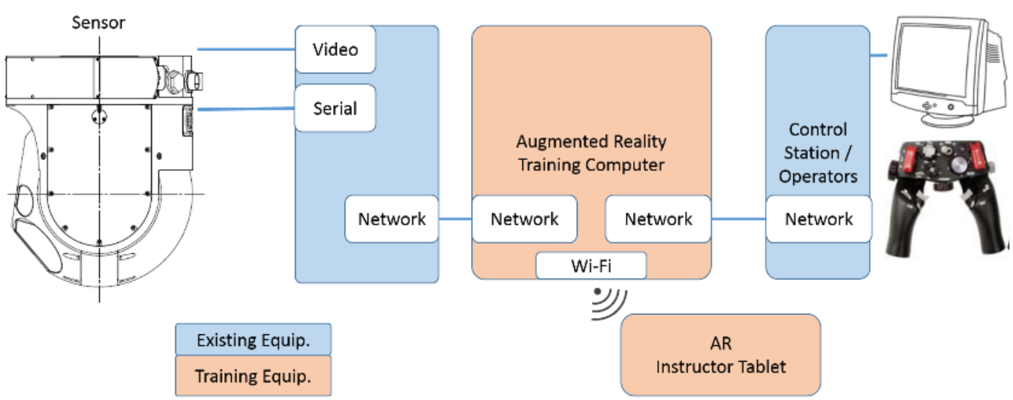

Figure 4 Network Insertion Point

When starting a research program it is understandable to want to take the approach that has the greatest likelihood for success. Here that would seem to be the near sensor solution, but in this situation there was one major drawback with that approach. The conversion of the sensor data into network form happens within a few feet of the sensor itself. Since this system is mounted on a mast and in the sun and rain, it meant that any equipment developed needed to operate in these conditions and it would also impact the physical size and weight of the equipment. Another consideration was the portability of the solution developed. By placing the system near the sensor you are limiting the adaptability of any developed technology to other systems with similar video and data connections. It would seem very unlikely that a small UAV would have the capacity to add the necessary hardware for training purposes, but with the network approach the solution could be adapted to modify the signal on the ground near the operator. For these reasons, it was determined that the research would focus on the network insertion approach until and unless it was determined that this approach would not provide an adequate AR capability.

VIDEO AUGMENTATION

The intercepted video is transmitted from the tower as an MPEG encoded stream. This means that the computer needs to receive the data, convert it to individual image frames, pass these frames to the rendering engine, augment the images with the virtual objects, convert the augmented images back into MPEG, and then send the video out over the network encoded into a transport stream (Figure 5). This needs to be accomplished while adding as little overall latency as possible. Since the sensor uses transmitted HD video, it also means that this is done with a 1920x1280 resolution image at 30 frames per second. The solution to solve this problem leveraged the computer's GPU to both decode and encode the MPEG video stream. This was implemented using NVidia's Video Codec SDK. This API has two primary components, called NVDEC and NVENC. NVDEC is an NVidia API that uses GPU based hardware to decode several popular video codecs including the H.264 codec used by this system (NVidia Corporation, 2017). The real advantage is that the data transfer between the CPU and GPU is done with the compressed data. This significantly reduced the amount of time to transfer the data. Additionally, by using this API the decoded frames are in GPU memory and can easily be mapped onto a texture to be used by the rendering engine. Once rendering is complete, the system uses NVENC encode the video image back into MPEG. NVENC is a hardware based video compression engine that can take GPU texture data and encode back into MPEG. The NVDEC and NVENC APIs only provided conversion to and from MPEG format. The sensor video data on the network is sent in a network

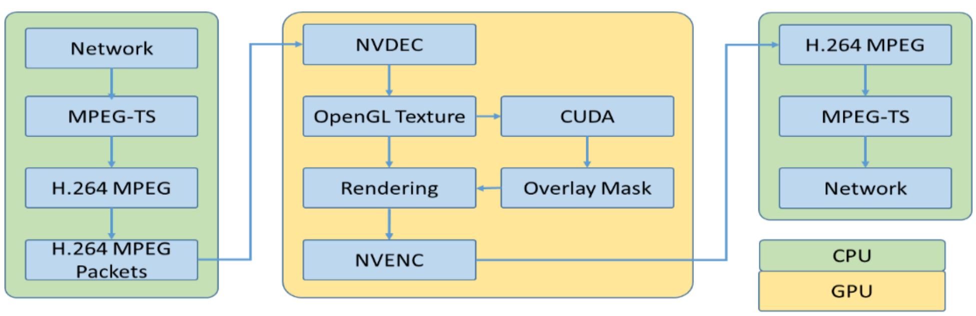

Figure 5 Video Augmentation Process

friendly MPEG-TS format. This means that there are actually two decoding steps. Initially, an approach that used open source video decoding software was leveraged to extract the MPEG packets from the MPEG-TS data stream. A separate process was started that decoded the MPEG-TS and piped the data to the main executable. This appeared to work but it was determined that the separate process introduced too much latency. Another side effect of this approach was that the buffer between the two processes could easily overflow causing data loss and subsequent image corruption. In order to address these issues, code was added to the main process to decode the MPEG-TS directly instead of using a separate application. This was a much simpler algorithm that just extracted the MPEG. It was still crucial to ensure the network buffers were big enough and that the functions to extract the data were called frequently enough to prevent buffer overflow.

The sensor has multiple cameras that the operator can choose from for viewing. This means that the rendering system must be able to render the insertions as they would look if viewed by that camera. This sensor contains cameras that see into the infra-red spectrum. The rendering system used Night Vision Image Generator (NVIG) software provided by NVESD to produce appropriate sensor visualization (Figure 6 and Figure 7). The NVIG rendering software can adjust settings to control the simulated ambient air temperature and other environmental controls that enable the operator to adjust the rendering so that the vehicle blends with the camera imagery. This is equivalent to an operator adjusting lighting levels for a daylight camera so that the rendering matches with the amount of direct sunlight.

Figure 6 Augmented Reality Insertion

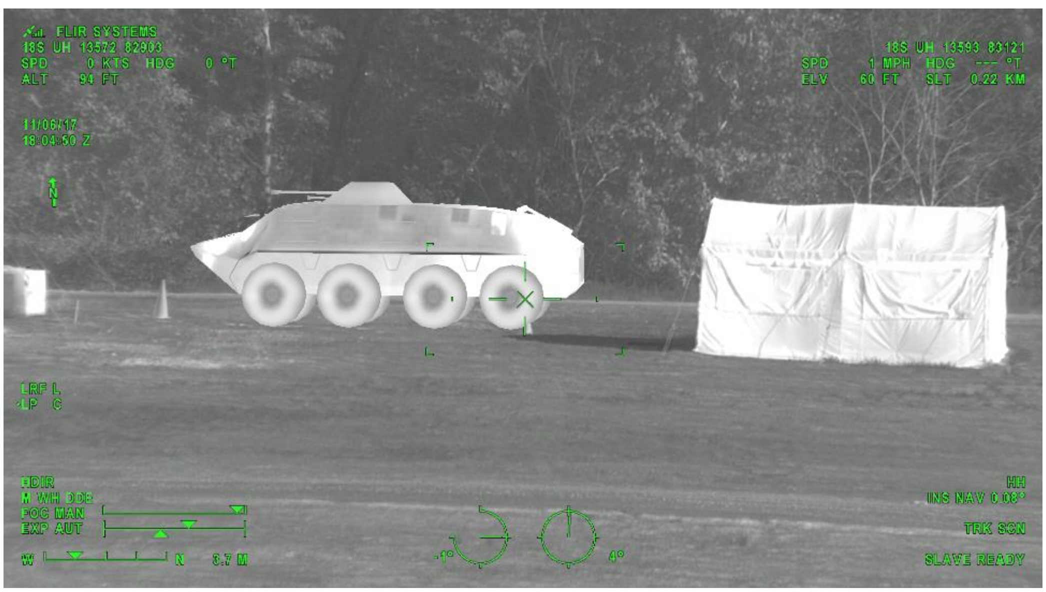

Figure 7 IR Camera Insertion

VIDEO OVERLAY DETECTION AND MASKING

One of the technical challenges of augmenting this sensor dealt with the video overlay added to the imagery by the sensor. The video signal sent from the sensor is typically configured to add an overlay, providing the operator with state data such as azimuth and elevation. One way to deal with this problem would be to configure the sensor to send the signal without the added overlay and to then recreate the overlay in the rendering computer. This approach seemed time consuming and error prone. Any deviation from the actual overlay could lead to negative training and the time needed to match it would be non-trivial. Another approach considered was to configure a second video stream from the sensor that matched the first, but without the overlay. The two data streams could be compared to determine the areas where the overlay was added. This would mean that the renderer would have decode two data streams and even then, because of the MPEG compression, there would be differences near the overlays that would incorrectly be seen as an area to mask.. The ideal solution was an approach that didn't require any configuration changes to the sensor, so an approach was developed that identified the pixels on the screen that contained overlay data and then generated a mask. The mask is then used to prevent any virtual renderings from covering those areas of the screen (Figure 8).

Figure 8 Overlay Masking

The algorithm used to detect the areas to mask is based on the same algorithms used for Chroma key compositing or "green screen" techniques. While the color used for the overlay can be configured, it is typically configured to pure green making this approach ideal. The GPU program to identify pixels that should be masked evaluates each pixel of the incoming data frame independently. For each pixel, the algorithm starts by converting the RGB pixels into the blue-difference (Cb) and red-difference (Cr) Chroma components (Equation 1). Once the Cb and Cr values are calculated, the distance between this pixel's color and the Chroma key color is calculated using the Pythagorean Theorem that is then compared to a threshold value to determine if it is the color of the mask. This by itself works well to determine some mask pixels, but the system surrounds the overlay features with black pixels so that they stand out against the camera imagery. This complicates the algorithm in that the system must not only identify a specific color but it must also identify pixels that are near other pixels of a specific color. To accomplish this, the search for matching pixels was expanded by 1 pixel in each direction. This means that 9 texture sample lookups are conducted for each pixel that is tested, and if any of the pixels pass the Cb Cr threshold then that pixel is added to the mask being generated. After the mask is generated the rendering system uses the mask to composite the final image. The mask bit pulled from the texture is used by a custom shader to decide if the final image for a pixel comes from the original image from the sensor, or the augmented image produced by the renderer where the overlay has been obscured.

$$\text{Cb} = 0.5 + 0.168736 * RED - 0.331264 * GREEN + 0.5 * BLUE$$
$$\text{Cr} = 0.5 + 0.5 * RED - 0.418688 * GREEN - 0.081312 * BLUE$$

Equation 1 Blue difference and Red Difference

The performance and correctness of the algorithm was tested by rendering a full screen white background for the augmentation that resulted in a display of only the overlay. This allowed any masked areas to be easily identified and any incorrect results noticeable.

The resulting image generated with this technique worked for most cases. The IR camera appears in greyscale imagery and worked quite effectively as expected. For the daylight camera, no anomalies were apparent during typical usage, but it was found that it was possible to adjust the camera's level and gain adjustments to degrade the video to the point that colors from the camera matched the overlay incorrectly.

A future optimization of this technique comes from the fact that areas of the screen that are not being covered by a virtual insertion do not need to be checked for overlay data. The camera imagery is already being copied to the background of the rendered image with the augmentation placed on top. If no augmentation is placed in the area then the original camera image and the rendered image are identical, making the choice to pull from one or the other irrelevant. This means that errors in the mask generation are only noticeable under situations where the area is also covered by an insertion. This also means large areas of the screen can be ignored by the detection process a large portion of the time. The system does need to be able to handle the worst case though, since once an insertion is detected it is normal for the operator to zoom in on it for identification and this would cause the virtual entity to fill a larger portion of the screen.

The overlay is limited to certain areas of the screen as well. Many areas contain text that is continuously changing but is still limited to small areas of the screen. Other areas contain fixed text or imagery that can be computed once and assumed present. Further optimizations to this system would allow a precomputed mask to be generated that denotes the subset of the screen that needs to be checked for overlays and reduce the overall time to process.

TRACKING

In earlier augmented reality work, additional sensors were used to get real-time data on movement. This data, when matched to the video frames from the camera, allowed accurate position and orientation changes to be tracked. An inertial measurement unit (IMU) tracked velocity changes at high update rates to provide the tracking calculations with reliable frame to frame motion. This sensor input was combined with computer vision techniques that can match features in a sequence of images to provide more data to drive an accurate tracking model. In AR, we are rendering a virtual image onto video from the camera. If we want the insertion to look realistic then the position of the insertion must remain fixed to the scene as the camera moves. In order to accomplish this, the AR rendering system must calculate the orientation of the camera for each frame at less than the field-of-view of one pixel on the display.

One benefit of the force protection sensor is that it publishes meta-data of the sensor state that can be used for tracking. The accuracy of the sensors that drive this data is stated to be ±0.045° in azimuth and ±0.037° in elevation (FLIR Systems Inc., 2013). The data is sent at 5 hertz and is not synchronized to a specific video frame. This by itself is not good enough to provide stable virtual insertions. The error in placing insertions based on this data would range from

Range(M)	Azimuth Error	Elev. Error
100m	3.09"	2.54"
200m	6.18"	5.08"
500m	15.46"	12.71"
1km	30.92"	25.42"
2km	61.84"	50.85"
3km	92.76"	76.27"
4km	123.69"	101.70"
5km	154.61"	127.12"
6km	185.53"	152.54"
7km	216.45"	177.97"
8km	247.37"	203.39"
9km	278.29"	228.82"
10km	309.21"	254.24"

Figure 9 Azimuth and Elevation Error Data

a few inches close in and up to 20 feet at further ranges (Figure 9). It would be ideal to have the error in azimuth and elevation to be less than a pixel, with an update rate greater than the video update rate. At HD resolutions a pixel is quite small in terms of the area of ground covered and errors of 3 or 4 pixels would likely go unnoticed. The error in accuracy in terms of the number pixels on the screen is dependent on the current selected FOV. With this sensor at HD resolution, the error ranges from about 3 pixels for 30° FOV up to 345 pixels for 0.25° FOV. The physical error that this represents on the ground remains similar for both zoom levels, but is much more noticeable at higher magnification as each pixel represents a smaller area.

The azimuth and elevation sensors are helpful, but as the sensor is zoomed in, the amount of error becomes quite noticeable. Noise in the sensor also results in significant jumps and jitter. To compensate for this the tracking system will need to use the sensor images and track the frame to frame motion by identifying unique features in each frame. These unique features can precisely describe the angular changes by locating the same feature in each image and using the location change to define the sensor movement. The tracking algorithm is made slightly easier for this platform by the fact that the sensor head remains stationary during operation and only angular motion needs to be calculated.

In prior AR work the system used a camera that was only used to aid the tracking system. The force protection system doesn't allow for the addition of special hardware. This leads to challenges with using the real sensors for tracking. The user can manually change settings that will affect the image quality. An example would be if the user manually adjusts the focus so that the entire scene is blurry. The tracking algorithms have no way to find image features to track movement with the degraded imagery Also, zooming in and out presents a challenge to the tracking system as the system screen goes blank momentarily during these changes. Conversely, these limitations are helped by the fairly precise azimuth and elevation sensors that allow the visual tracking system to reset after the cameras are back in focus, or after the zooming operation is finished.

The operator controls send signals to the sensor that tell it to turn and how fast it should turn. This input provides a significant advantage over other AR systems in that this data gives a good estimate of the sensor's future position and velocity that can be fed into tracking algorithms. The renderer also knows about the expected movement before the sensor does since the commands that control the sensor can be viewed as they are on the way to the sensor. Also, the commands that control the sensor behave in a very predictable manner. The sensor accelerates to the commanded speed, adjusts speed as new messages come in, and then stops by decelerating. This is very predictable and means that the position estimate, even without visual tracking, can be accurate. This is a big advantage to a helmet mounted system that is only able to react to sudden and unpredictable head motion. The downside is that all of this data is coming in and going out on a network connection so the exact time the sensor starts and stops moving will still have some variability.

The tracking capability implemented for the NVESD program is not currently utilizing any features or techniques described here beyond the meta-data coming from the sensor at 5Hz. The task of providing precision tracking is anticipated to be started in a future phase.

NEXT STEPS

The research described here is still in progress. The system has been focused on adapting our previous work to this system in the areas described, but to become a quality training aid, other areas still need work. The tracking, as mentioned already, is the main area for research. Beyond tracking, the system benefits from algorithms to adapt to changes in lighting conditions. For IR sensors, the colors used by the system for hot to cool are scaled based on the range the sensor is detecting. This means a method to detect these changes and apply rendering changes to the NVIG rendering model would be desirable to ensure the virtual insertions blend with the sensor view.

Many other areas, such as camera changes, can be better synchronized to the video stream by starting simple GPU programs to find the video frames when data changes instead of waiting on meta-data to notify the renderer. One example would be to detect the operator changing from IR to daylight cameras and, upon detecting the operator message going to the sensor, to start a program that looks for the color changes in the scene. This would allow exact frame synchronization between the sensor and the rendering engine.

Upon completion of the next development phase of the Augmented Reality project, two (2) prototype systems will start to be incorporated into NVESD testing and training activities. The first system will be set up for testing and evaluation in NVESD's Sensor Integration Lab (SIL). The SIL will initially use the system to provide realistic targets as part of their day-to-day testing activities. Once fully integrated, the SIL will use the system as part of their Government Acceptance Testing activities. The second system will be added to one of the existing Mobile Training Facilities to augment the Soldier training on the live tower system. Once proven valuable to the Soldier Training, additional systems will be added to the other training facilities and NVESD will look at adding the AR capability to other EO/IR sensor systems.

REFERENCES

Defense Science Board, (2013). *Report on Technology and Innovation Enablers for Superiority in 2030.* Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C..

Schaffer, R., Cullen, S., Cerritelli, L., Kumar, R., Samarasekera, S., Sizintsev, M., Oskiper, T., Branzoi, V. (2015). Mobile Augmented Reality for Force-on-Force Training, in Proceeding of the *Interservice/Industry Training, Simulation & Education Conference 2015*. Arlington, VA: National Training and Simulation Association.

Kumar, R., Samarasekera,S., Sizintsev, M., Oskiper,T., Branzoi,V., Schaffer, R., Cullen, S. , Krishnaswamy, N. (2013). Augmented Reality Training for Forward Observers, in Proceeding of the *Interservice/Industry Training, Simulation & Education Conference 2013*. Arlington, VA: National Training and Simulation Association.

FLIR Systems Inc. (2013, May 22). STAR SAFIRE TYPE II CEU / 380-HD - ICD – Serial Type "A" Interface Protocol.

NVidia Corporation (2017, Feb). NVIDIA VIDEO CODEC SDK APPLICATION NOTE - DECODER