Sensor Placement Optimization in LVC Environments for Training, Analysis, and Operational Applications

Jennifer Lewis and Joyner Livingston
Science Applications International Corporation
Huntsville, AL
jennifer.e.lewis@saic.com, joyner.livingston@saic.com

ABSTRACT

Advances in specialized processor capabilities, such as Graphics Processing Units (GPUs), have contributed to the ability to efficiently process high density terrain. Using these technologies, the Aviation and Missile Research Development and Engineering Center (AMRDEC), System Simulation and Development Directorate (SSDD), Soldier Protection Laboratory (SPL) developed a physics-based sensor-terrain interaction model that accurately predicted and synthesized radio frequency (RF) coverage in dense foliage for the Army Expeditionary Warrior Experiment (AEWE) conducted at Ft. Benning in August 2013. Since that time, the development team has built upon the initial capability to include unique user features such as intuitive comparisons of the mathematically optimal placement for baseline and experimental sensor sets and the ability to respond on-the-fly to changes in the underlying terrain. This paper describes the capabilities of the Automated Sensor Placement Engine (ASPE) and potential operational applications, such as Intelligence Preparation of the Battlefield (IPB) and mission planning and rehearsal. It also describes the technical design of the tool and underlying models as well as its transparent middleware approach to integrating with existing toolsets and visualization options. During the past year, the SPL team has successfully integrated ASPE into a range of Live, Virtual and Constructive (LVC) environments, including networked tactical sensors, command and control (C2) nodes, constructive simulations such as OneSAF, and web-based interfaces such as Ozone Widget Framework (OWF). Integration leveraged the use of both tactical and simulation interoperability standards, including the Security Equipment Integration Working Group (SEIWG) Interface Control Document (ICD) series and Distributed Interactive Simulation (DIS). The paper will discuss lessons learned and document repeatable processes developed while integrating these multi-architecture environments.

ABOUT THE AUTHORS

Jennifer Lewis is a senior software engineer, who has designed and implemented network protocols for the defense and telecommunications industries. She has developed interoperability solutions for distributed training and analytic LVC environments for the past 12 years. She holds a Master of Science degree in Computer Science with an emphasis in Telecommunications and Networking from the University of Texas at Dallas and is a Certified Modeling and Simulation Professional.

Joyner Livingston is a senior Operations Research / Systems Analyst, who has an operational background in mounted maneuver in the US Army, as well as Special Operations Logistics and Procurement. He has conducted numerous studies and analyses using LVC environments for hardware in the loop, human in the loop, and closed form applications to inform system design, concepts of operation, system sustainment, optimization, and to conduct trade studies. He earned a Bachelor of Science degree in Mechanical Engineering from Auburn University and a Master of Science degree in Systems Engineering with concentrations in Operations Research and Modeling and Simulation from the University of Alabama in Huntsville.

Sensor Placement Optimization in LVC Environments for Training, Analysis, and Operational Applications

Jennifer Lewis and Joyner Livingston
Science Applications International Corporation
Huntsville, AL
jennifer.e.lewis@saic.com, joyner.livingston@saic.com

INTRODUCTION

The Automated Sensor Placement Engine (ASPE) originated out of a need to provide predictive modeling and analysis for the technology down-select of a radio solution for the operation of an Unmanned Ground Vehicle (UGV) in densely vegetated terrain. Selection of the radio solution through pure live experimentation was cost- and schedule-prohibitive, so it was important to understand the interaction between the radio (sensor) and terrain through physics-based modeling and simulation. This need quickly revealed similar needs for predicting the sensor-terrain interactions across multiple placement options for Line of Sight (LOS), seismic, acoustic, radar, and other sensor types, specifically in the base defense (static) operational use case. An important and continued Intelligence, Surveillance, and Reconnaissance (ISR) challenge is the operational units' ability to rapidly identify collection requirements based on tactical priorities and then to determine how to best meet these prioritized requirements based on effective and efficient use of the collection assets/means available.

The ability to forecast and provide adequate, reliable, persistent surveillance is critical to meeting challenges identified in pre-mission planning. Teams have access to a heterogeneous suite of optical, seismic, acoustic, radar, and other sensors that can be deployed within an Area of Operations (AO), Area of Interest (AI), and around Named Areas of Interest (NAIs). As such, understanding the interaction between collection assets and the operational environment is necessary in determining how many, what type(s), and in what locations these sensors must be placed to support the operational mission. These challenges cross two domains: the operational domain and the engineering/analytical domain. For operational units conducting Intelligence Preparation of the Battlefield (IPB), a rapid, dynamic and intuitive understanding of these sensor-terrain-foliage-obstruction interactions is necessary to predict optimal sensor coverage, to predict dead space, and to identify overlap and interference. Ideally, units can fuse ISR capabilities with the limitations and constraints determined through IPB and terrain analysis. Unfortunately, units lack the tools to provide automated fusion of IPB data with ISR asset characterization models. This results in less than optimal pre-mission collection management and ISR planning as well as greater operational risk due to a lack of timely geo-specific situational awareness and threat analysis and asset visibility. For engineers and analysts providing the materiel solution, this same understanding is necessary, in order to determine what types, sizes, and numbers of sensors may be developed to support surveillance and security kitting for various unit sizes expecting to encounter various threats and in a variety of dynamic environments.

Further examination of the operational use case yielded a need to include automatic, computationally-based optimization of sensor placement, in order to alleviate the time-intensive task of exploring user-defined sensor laydowns one at a time. In addition to supporting the terrain analysis steps during IPB, it is important to make these simulation and analysis capabilities and outputs available to the warfighter on existing visualization tools. Doing so mitigates negative impacts resultant from training and fielding requirements of yet another toolset for the warfighter to gain familiarity with and sustain. ASPE accomplishes this through a unique interoperability framework. Finally, in an effort to transition from mission planning to mission rehearsal and execution, the team assessed the importance of updating the sensor-terrain model with Position Location Information (PLI) directly from live sensors, which have already been emplaced in the AO. This transition from mission planning to rehearsal and execution further highlighted the need for a robust interoperability framework. This framework allows ASPE to ingest sensor and terrain models from external sources as well as PLI from both simulation protocols and live (tactical) protocols. The

framework can then export the resulting sensor coverage maps to a menu of existing, user preferred visualization options. In order to meet the objectives of both domains, there is a need for an integrated solution that accounts for a wide variety of sensors, terrains, weather, and other considerations and that is interoperable with existing user interfaces, databases, visualization tools, and analysis tools (e.g. constructive simulations).

INITIAL USE

The original physics-based simulation model used in assisting in the technology down-select for a radio solution for a UGV during a specific application was developed in Mathematica, with no user interface or ability for a "non-engineer" to use. Through the course of the development of this core physics model, however, the outputs were iteratively validated against live data collection processes at Redstone Arsenal, AL, which provides a wide variety of foliage types and densities, as well as manmade structures. Following this effort, an initial attempt to design a developmental user interface was initiated, in order to provide a means to elicit feedback from both technical and operational user communities. This led quickly to an opportunity to use this model in support of the Lead Technical Integrator (LTI) - *Live Experimentation Branch, Combat Development and Integration Division, Signal Center of Excellence at Ft. Gordon, GA* – for the wireless network for Army Expeditionary Warrior Experiment (AEWE) Spiral I. The core physics engine and propagation model was used during experiment planning during the summer of 2013 in hypothesizing an appropriate configuration and predicting the coverage of an RF network for a live experiment involving a company-sized BLUFOR and squad-sized OPFOR in a mobile and operationally relevant environment. During the course of that effort, live signal data was collected at Ft. Benning, GA across many of the frequency ranges of interest, and was once again compared to the model outputs (see Figure 1). Discrepancies were observed at points where the data was collected very close to transmission stations. During the time this data was taken, ASPE assumed all sensors emitted isotropically; therefore, values close to the tower would be affected due to lack of sensor pattern analysis. Accurate modeling at the transmission source is less operationally relevant since users are more concerned about signal strength at greater distances; however, ASPE has since then been updated to account for this.

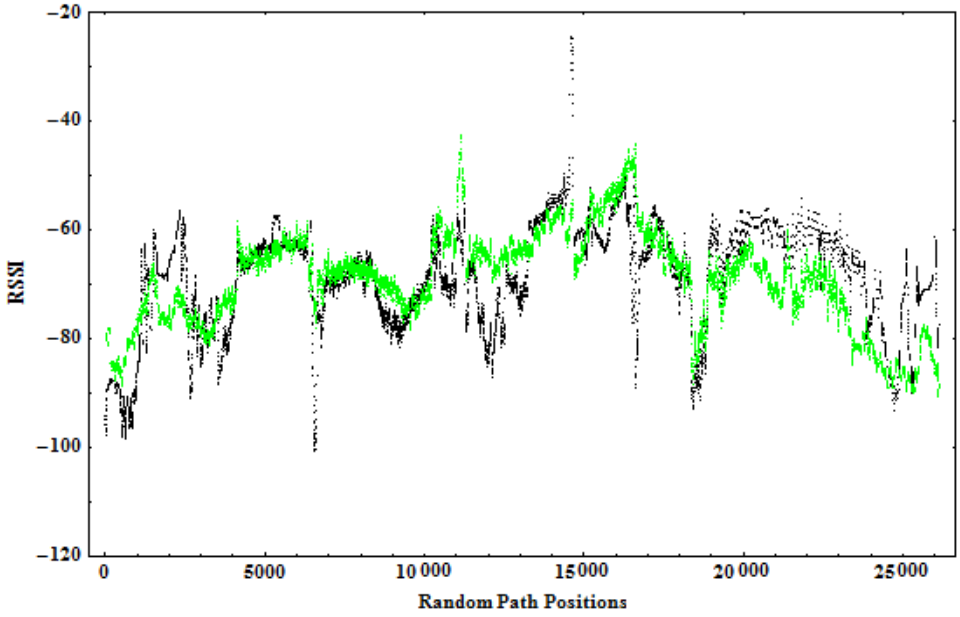

Figure 1. AEWE Results

The use of this RF propagation model in this context was a significant milestone, which informed the early development of the model into a deployable tool, with a suite of capability designed to meet the needs of a diverse user community. Additional uses include relying upon model output to plan for camera placement and LOS coverage and to conduct trade-off analysis for a large Forward Operating Base (FOB) in a Joint Defense Operations Center (JDOC) context.

UNIQUE FEATURES

Many tools exist that provide capabilities that can support IPB and mission planning, such as FalconView, RaptorX, C2PC, and CPOF. ASPE uses state of the art computational geometry, optimization algorithms, and composable modeling techniques to provide capabilities not possible in other solutions. For example, several Commercial Off the Shelf (COTS) and Government Off the Shelf (GOTS) tools provide trial-and-error hypothesis testing of sensor laydowns. The user must manually place sensors in what he believes to be optimal locations and subjectively compare visual representations of the coverage maps. ASPE provides automated optimization by calculating coverage areas for all possible combinations of sensor placements. As another example, Zones of Protection (ZoP), a GOTS product evaluated in conjunction with ASPE, provides visualization of water flow over terrain. However, it may show water flowing through buildings or walls added by users because it does not account for user-defined obstructions on the terrain, added after the original terrain file was loaded into the system. The following subsections detail the technical processes ASPE implements for achieving improved results.

Dynamic Terrain

ASPE generates a Triangulated Irregular Network (TIN) to model the underlying terrain of a specified area based on data from online map servers. ASPE then modifies its TIN to include terrain objects such as buildings, trees, roads and water. In this way, surface features become an inherent part of the terrain, allowing the model to interact with them correctly and producing valid results even in densely vegetated or urban areas.

As shown in Figure 2, the user can specify terrain objects in several ways. One, users can add, edit or delete objects via the ASPE user interface, which uses standard mapping platforms such as Google Earth, World Wind and Open Layers to manually draw geometric areas on the terrain to represent surface features. Two, the team has conducted proof of concept exercises to show ASPE features can be updated through photogrammetry, a process to render 3D geometries from a series of 2D images, typically from an aerial flyover. Modern cameras embed geo-location information into the metadata of every photo, meaning photogrammetry software can use photos from almost any source. Photogrammetry is particularly useful in areas where seasonal changes significantly affect the terrain, e.g. in dense foliage or near rolling icebergs, or in areas where significant construction is taking place because it provides a quick and simple way to keep the ASPE terrain model up-to-date.

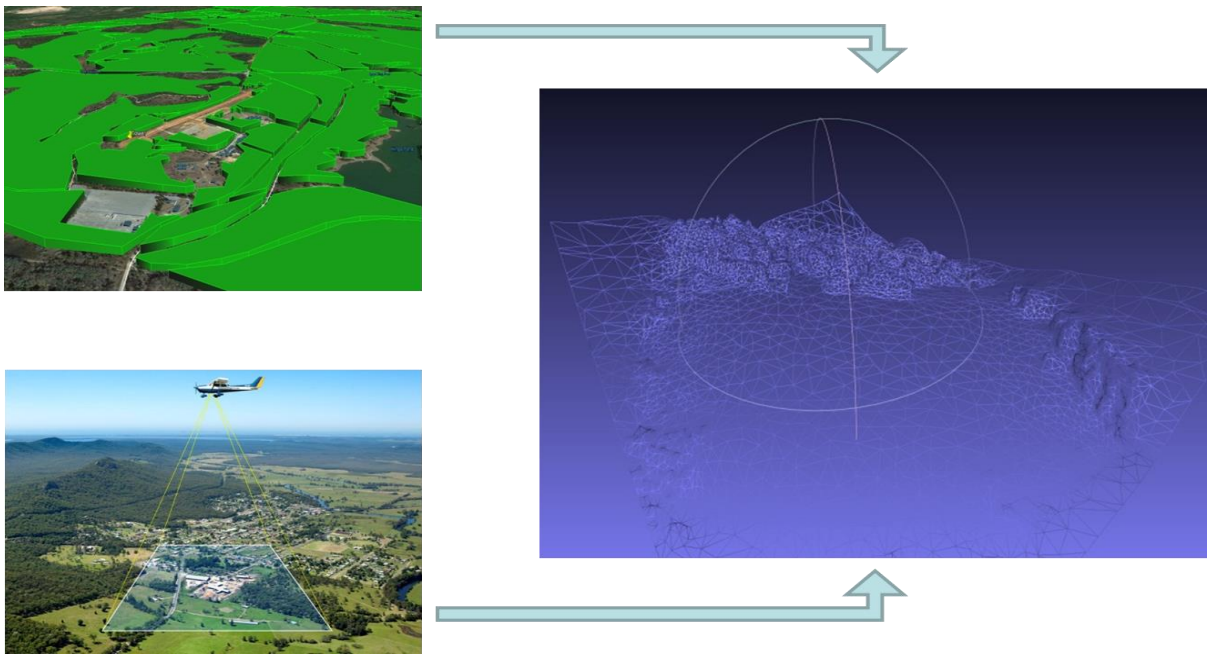

Figure 2. ASPE updates its Triangulated Irregular Network (right) using input from mapping platforms (upper left) or photogrammetry (lower left)

Lastly, features can be provided from external LVC simulations such as OneSAF Ultra High Resolution Buildings (UHRBs). While this option may not represent the true terrain, it provides the ability to model additional "what if" scenarios based on training and analytical simulation executions. More discussion on LVC interoperability is provided in the following section.

The size and complexity of the terrain and its terrain objects determine the number of terrain points and faces in the generated TIN. Generating the TIN is a computationally time consuming process. While the user may modify the terrain at any time, reducing the number of on-the-fly modifications made will dramatically improve ASPE's computation time.

Mathematical Optimization

ASPE's sensor model uses the generated TIN to calculate the coverage area for a collection of sensors. This allows ASPE to evaluate many combinations of the sensor laydown and directly compare the resulting coverage areas to find the mathematically optimal solution for placement of a set of sensors. ASPE is able to do this for several reasons.

One, ASPE executes visibility calculations quickly through the use of the open source Computational Geometry Algorithms Library (CGAL) software (Alliez, 2014). The development team has put a specific focus on the speed of ASPE's visibility calculations, which can be a significant bottleneck to optimization routines. ASPE's current runtime for a line of sight (LOS) calculation on one processor is approximately 0.03 seconds per sensor, and its runtime for a radio frequency calculation is on the order of two seconds. The increase in computation time is due to additional calculations for free space path loss (FSPL), reflection, diffraction and the Radiative Energy Transfer (RET) through materials such as foliage.

Future work will take advantage of the matrix-oriented structure of Graphics Processing Units (GPUs) to efficiently execute calculations against the TIN to calculate coverage area. GPUs were designed with a matrix structure in order to efficiently render pixel data, which is inherently matrixed with each pixel incorporating a red, green, blue and alpha component. Game engines can quickly render three-dimensional shading in highly detailed scenes because they take advantage of the structure of the underlying GPU for visual effects. Many sensor optimization tools also use shading to display sensor coverage areas. However, these visual-oriented tools do not typically calculate geographically representative data associated with the visual representation, such as calculating a mathematical value for coverage area. Doing so would involve mapping the existing depth buffer to the triangular faces of the terrain, followed by a series of other calculations to determine the coverage area. This type of mapping would incur an additional cost to the GPU shader, resulting in decreased frame rates. By using the GPU architecture purely for calculation, ASPE will further speed up its calculations (Bell, 2008). In addition, the team expects to produce LOS calculation times on par with typical shading-only methods by implementing parallel processing over multiple GPUs. Because the calculations for each sensor are mutually independent (ie. the calculation of one sensor has nothing to do with the calculation of another sensor), parallel processing is an ideal approach for this type of optimization. However, parallel processing has not been implemented beyond the proof of concept level to date in order to focus on increasing the speed of the fundamental sensor calculations.

Finally, ASPE uses a genetic algorithm (GA) wrapped around a sequential least squares programming (SLSQP) algorithm to optimize for both variable and non-variable parameters within a collection of sensors. For example, if the user has a camera already installed at a specific location and another camera that has not been installed, ASPE can optimize the pan and tilt angles for the "fixed" camera while optimizing position, height, pan and tilt angles for cameras to be installed. Earlier versions of ASPE used GAs exclusively but were limited to optimization of a single parameter, namely position. The SLSQP algorithm made possible the ability to optimize multiple sensor parameters.

In addition, users have the option to define two types of sensor boundary constraints on the terrain. The sensor location boundary (SLB) is used to tell ASPE where sensors can or cannot be placed. For example, users may not want ASPE to place sensors on physically unreachable locations, such as on the side of a steep mountain slope, over water, or in the middle of a forest. A SLB can be used to exclude these regions from the feasible set of possible sensor locations. The second type of constraint, the sensor detection boundary (SDB), tells ASPE where sensors

should or should not be able to see. This added flexibility allows users to prioritize which regions on the terrain ASPE should cover.

Composable Models

Another unique feature of ASPE is its ability to model experimental sensor types. ASPE's engine is designed in a modular fashion consisting of two components called generators and manipulators. Generators contain the fundamental attributes of the sensor, such as a radio's waveform, while manipulators modify the environment around the sensors, such as amplifying a sensor's signal. ASPE's sensor models are based entirely on these two components, allowing it to perform the same validated calculations on any existing or future concept sensor or sensor configuration. This allows the user to formulate any number of "what if" scenarios ranging from "what if our cameras had a slightly larger focal length" to "what if we mounted the camera on a tower on top of a UAV"? While many configurations may not be feasible or physically-realistic, ASPE provides the ability to explore the theoretical limits of sensor utility.

INTEROPERABILITY

Interoperability is a fundamental concept to ASPE. It is intentionally developed as a capability to be integrated into existing tools, rather than as a tool itself. Rather than offering training and support to learn yet another tool and accompanying interface, ASPE's developers have designed an interoperability framework, a set of standardized APIs and modular interoperability capabilities, designed to simplify the integration of ASPE into any of the numerous live, virtual and constructive tools that already exist within the simulation community. This section will describe the design of the interoperability framework as well as how it has been used thus far to enhance LVC environments.

Technical Approach

According to Blair (2011), the interoperability framework uses a transparent middleware approach to provide efficient data translation between any number of interface specifications. As shown in Figure 3, protocol specific messages, behavior and data are captured by the interoperability framework and then translated to an internal representation, known as the core data model. The interoperability framework then translates from the core data model into another protocol's specific messages, behavior and data. The transparent middleware approach means that one protocol can be mapped to any other without developing a direct mapping between each protocol.

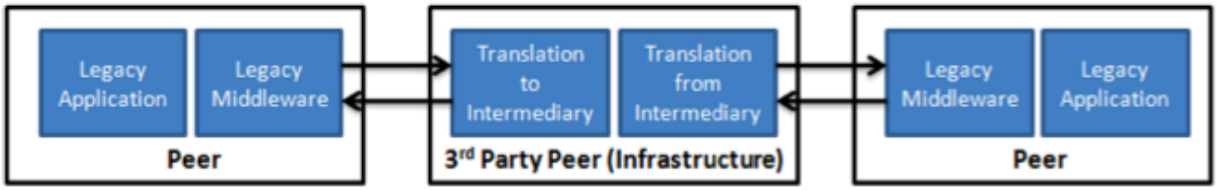

Figure 3. Transparent Middleware Approach

The primary challenge with the use of transparent interoperability solutions is that the developer must identify a subset of functionality between all interoperating protocols. In a general interoperability use case, this can be impractical and sometimes impossible. However, within the domain of defense-oriented modeling and simulation, it becomes more achievable because the types of information that need to be communicated are generally well known based on operational use case, e.g. SPOT reports or fire engagements. Because ASPE has an even more specific subset of operational use cases, this limitation becomes an asset, allowing the developers to focus on the required subset of behaviors ASPE needs to support rather than attempting to create a general mapping function that supports all features of all protocols.

ASPE's interoperability framework takes the transparent middleware approach a step further by removing the concept of a legacy application in favor of a standard "behavior". This creates a simplistic mapping function that is easy to conceptualize, code and maintain. For example, rather than creating a mapping function to interact with OneSAF, the interoperability framework has a mapping function for a sensor "behavior" that allows a sensor to 1) exist at a location, 2) sense a specified distance, and 3) detect objects. This simplified mapping function allows developers to design data driven functionality without creating overly complex code. The mapping function, itself, acts as an interface to a database storing message structures and message population routines for the applicable protocol. Therefore, the ASPE interoperability framework does not require a sensor behavior mapping function for each sensor protocol but instead uses a single, simplified data-driven routine to reduce coding, testing and maintenance. This approach also allows the simulation environment to change the owner of each model without any update to the interoperability framework. If OneSAF models a radar in one execution and a live component models it in a subsequent execution, the mapping function simply will be "driven" by a different protocol's data.

Currently, ASPE's interoperability framework supports subsets of seven industry interoperability standards, including IEEE 1278, IEEE 1516, Security Equipment Integration Working Group (SEIWG) ICD 010x series, and Pelco-D. It also supports OneSAF's Representational State Transfer (REST)-ful User Data Gateway (UDG) and the Ozone Widget Framework (OWF) inter-widget communication system.

Implementation

The interoperability framework uses the ASPE internal data structure as the basis for its core data model. Its external API is implemented as a minimal interface with only four methods – connect, send, receive, disconnect. This works well for ASPE's current design of creating simple, behavior-oriented mapping functions and helps to hide the complexity of supporting multiple architectures within the implementation of the framework. However, it also provides the greatest flexibility in connecting a variety of LVC protocols and keeps ASPE flexible so that it can quickly support use cases or applications beyond that of its original design and intent. Each mapping module is required to implement the minimal interface to communicate with other modules in the framework. However, more complex mapping modules can extend internal APIs to support more sophisticated connect, send, receive or disconnect methods without breaking the design of the overall framework.

As input, ASPE receives information about terrain objects (buildings, water, trees, roads) and sensors (lifeforms, cameras, radar and radios). The interoperability framework receives this information in the form of DIS PDUs, HLA objects, SEIWG position reports and stores relevant data internally in a format similar to that expected by ASPE. As output, ASPE produces JSON-formatted text describing the optimized locations of the sensors as well as binary data representing PNG-formatted images of the relevant coverage map.

Integration Events

The first event to integrate ASPE into an LVC environment was an evaluation of ZoP, a GOTS terrain visualization tool used for IPB. The event featured ZoP as the visualization system for an LVC environment. The live portion of the environment included the Rapid Deployment Integrated Surveillance System (RDISS), the Man-portable Surveillance and Target Acquisition Radar (MSTAR), and a Command and Control Suite known as the Force Protection Architecture (FPA). The constructive portion of the environment included OneSAF as the scenario driver and ASPE as the sensor coverage model. In addition, ZoP could model its own sensors. Visualization of the scenario was possible from ZoP, FPA and the OneSAF Plan View Display (PVD) as well as from an OWF Google Maps widget running from a standard web browser. Although the event was designed to use ZoP as a visualization system, these other options demonstrated the consistency of the LVC environment from all perspectives.

The interoperability framework used three mapping behaviors to input data from the simulation environment. Live sensor assets provided their PLI data via FPA using SEIWG ICD 0100B. OneSAF provided PLI data for soldiers and ground vehicles using the XML-based UDG schema. For ASPE purposes, soldiers represented Line of Sight (LOS) sensors and ground vehicle models containing radios represented RF sensors. Moving OneSAF entities update their position through the UDG at a much higher frequency than ASPE can handle. The interoperability framework monitored updates from OneSAF but limited the amount of data translated to ASPE by monitoring whether ASPE

was currently performing a calculation. In this way, ASPE provided output as often as possible without being overloaded by the constructive environment.

The interoperability framework informed ZoP that a new coverage map was available via the SEIWG Geometry Report. The Event Report message contained a link to a RESTful web interface hosted by the interoperability framework from which ZoP could request the most recent image. This process served two purposes. One, although SEIWG ICD-0100B does not bound the length of a Geometry Report, the command and control system in use truncated the message, making it impossible to send the binary data within the message itself. Two, providing an indication that an updated image is available allowed ZoP to request the image data at a convenient time, resulting in fewer hangs and crashes than when ZoP was forced to process too much information at one time. As an additional feature, the interoperability framework also provided ASPE output to an OWF Google Maps widget. The widget allowed the user to request an updated coverage map via the same RESTful interface used by ZoP.

Lessons Learned

A major lesson learned in this integration, assessment and demonstration is that system architectures are dependent not on the protocol's standard but on the implementation of the standard in the integrated system. In a large distributed environment, no team can be familiar with every facet of every tool. The integration team and other event planners must be open to the idea of modifying the system architecture to conform to system constraints. This is especially true when creating a multi-architecture system with more than two interoperating protocols. A multi-architecture environment is more prone to latency issues since live, virtual and constructive simulations and protocols have different expectations on the frequency and amount of external data to be processed (Henninger, 2010). This point was exacerbated by ASPE's computationally intensive optimization methods and data intensive output. Terrain object and sensor updates arrived much more frequently than ASPE could process the updates, and the images generated by ASPE were difficult for ZoP to ingest while maintaining real time modeling of its internal sensors. The development team used techniques from the Distributed Simulation Engineering and Execution Process (DSEEP) and its Multi-Architecture Overlay (DMAO), IEEE 1730-2013, to address common LVC design and development issues. These standards highlight typical steps and issues that should be addressed when designing and developing a LVC environment from deriving technical requirements to defining heartbeat rates and selecting translation software. These standards document the LVC community's experiences and provide a wealth of information to new integration teams. Most critically, these standards define an iterative process where each step provides input to the following step and feedback to the previous step. This type of process is fundamental to a successful LVC environment.

CONCLUSIONS

The team's primary focus continues to be the speed and accuracy of performing the optimization calculations on dense terrain. This is the fundamental capability ASPE provides. However, future research and development will include a greater emphasis on how to augment existing tools to include ASPE's emerging capability into existing tools and future mobile applications. The path forward includes multiple aspects as described below.

Interoperability

Future development within the interoperability framework will include the ability to interact with web applications using HTML5 techniques, such as WebGL and WebSockets, as well as the WebLVC standard currently under development (SISO, 2013). Figure 4 shows a prototype three-dimensional visualization of ASPE defined terrain developed using three.js, a simplified interface to WebGL written in Javascript which abstracts away some of the more complex 3D modeling concepts. This simplified interface makes data translation between LVC environments and a 3D terrain model practical and efficient, allowing developers without in-depth 3D modeling experience to create an intuitive user interface for general operators and technicians. This type of 3D visualization and manual terrain update capability could be used in conjunction with advancements in mobile technology to provide an immersive environment to ensure proper location of terrain features and sensors for site surveyors, installers, technicians, and soldiers in the field.

Figure 4. 3D Model of ASPE Dynamic Terrain

Mobile Applications

Currently, the computationally intensive processes for conducting LOS analysis, predicting and modeling RF signal propagation, and yielding resultant sensor coverage across varying and user-defined terrain types must be accomplished using commodity desktop hardware. User-defined constraints, operational considerations, sensor options, performance characteristics, and other inputs, however, may be initialized by and exported from mobile platforms (tablets) to the ASPE physics and optimization engine through a network. Conversely, the visual output of optimized sensor laydowns may also be used by a mobile platform. In order to facilitate the use of mobile platforms, while leveraging the capacity of less mobile hardware, mobile applications and interfaces are being developed beginning with iOS. This development will assist in executing additional use cases, such as pre-deployment virtual site surveys for network, surveillance, and detection equipment. The interoperability framework will continue to handle data exchange requirements between various user interfaces (mobile, web-based, and desktop), the ASPE physics and optimization engine, and external sources such as live systems and virtual and constructive simulations. Therefore, as mobile technology becomes more computationally capable, the overall ASPE capability may likewise migrate more of the computational burden out to network-degraded or network-denied environments at the tactical edge.

Sensor Modeling

ASPE will continue to develop additional predictive sensor-terrain interaction modeling and optimization, to accommodate seismic, acoustic, and other unattended sensor technologies. Use cases for threat analysis are also being developed, which will include the ability to predict, through sensor-terrain interaction modeling and optimization, where the threat is most likely to employ signal jammers and electronic warfare devices in order to degrade or deny the effectiveness of friendly operations. This capability will further enhance the relevance of ASPE in IPB and mission planning and rehearsal.

Generalizing Optimization

All too often, applications are pigeon-holed into a discrete set of capabilities. The ASPE technical team has taken this into consideration, and has designed the optimization engine to be as flexible as possible. The goal is to have a multi-dimensional matrix of possible operations, given a distinct set of requirements. Take into consideration the following list of possible optimization runs, where each builds on the one before:

1. Optimize fastest route from Point A to Point B.
2. Do Step 1, with a tank.
3. Do Step 2, while maintaining communications with known radio towers.
4. Do Step 3, while Point B is moving on an un-anticipated path.

It's easy to see how these scenarios compound over multiple use cases. However, the goal of ASPE is to utilize this multi-dimensional decision matrix to arrive at the correct result. This is best described by taking the inverse of Steps 1 – 4. Instead of dynamic route planning, ASPE would instead optimize a sensor network around user-defined routes. This time, add known areas of possible enemy fire and optimize the safest routes for evacuation, while supplying over-watch from Marine snipers. Where do the snipers go? What's the best route? Where should assets be placed? The matrix is designed to start and end from any given point.

Improving the API

As organizations continue to fill technical gaps, new user interfaces seem to be created daily. While ASPE utilizes a developmental user interface for testing, the technical team has concentrated on an embeddable environment. Meaning, the future of ASPE entails widening the flexibility of ASPE's API. Currently, the project is broken into two APIs. The first is a direct API to ASPE, which gives the developer unencumbered access to its capabilities. However, as a caveat, the user is required to display everything through his or her own map client and data visualization tools. The second (visualization and access to ASPE) API is really a wrapper for multiple existing, open-source APIs, with embedded calls to ASPE. For a user wishing to be up and running, right out of the box, this API allows them to instantly embed Open Layers, WebGL (Figure 4) or a combination of the two. For example, if a user wished to draw an object in Open Layers and then display it in the visual appealing WebGL, this now takes two lines of code. However, the required code behind the API is obviously much more extensive making all the necessary calls to the original APIs. This is beneficial to a user, because the majority of the error checking is already done for them. As it stands, a complete environment drawn in Open Layers and rendered in WebGL can be accomplished in seven lines of code if a user were to start with nothing else.

ACKNOWLEDGEMENTS

Technical contributions provided by Willie B. Maddox IV, PhD were instrumental in documenting the achievements in optimization described in this paper. Special thanks to Mr. Brett M. McWilliams, sole developer of the original physics engine upon which ASPE was built, for contributions made during the development of this paper.

REFERENCES

Alliez, P. et al. (2014). CGAL Computational Geometry Algorithms Library (Version 4.4) [Software]. Available from http://www.cgal.org.

Bell, N. & Garland, M. (2008). Efficient Sparse Matrix-Vector Multiplication on CUDA. *NVIDIA Technical Report NVR-2008-004*.

Blair, G., Paolucci, M., Grace, P., & Georgantas, N.. (2011). Interoperability in Complex Distributed Systems. *11th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Connectors for Eternal Networked Software Systems*.

Henninger, A. et al. (2010). The Live Virtual Constructive (LVC) Architecture Roadmap: Foundations from the Past and Windows to the Future. *Proceedings of the 2010 Interservice/Industry Training, Simulation and Education Conference.*

Simulation Interoperability Standards Organization (2013). WebLVC Study Group Final Report. Available from http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=41272.