# Augmented reality integration of fused LiDAR and spatial mapping

Matthew B. Selleck, David Burke, Chase Johnston, Varun Nambiar

Matthew B. Selleck, David Burke, Chase Johnston, Varun Nambiar, "Augmented reality integration of fused LiDAR and spatial mapping," Proc. SPIE 10666, Three-Dimensional Imaging, Visualization, and Display 2018, 106660S (16 May 2018); doi: 10.1117/12.2304977

Event: SPIE Commercial + Scientific Sensing and Imaging, 2018, Orlando, Florida, United States

# Augmented Reality Integration of Fused LiDAR and Spatial Mapping

Matthew B. Selleck, David Burke, Chase Johnston, Varun Nambiar
Electro-Optical Systems Laboratory, Georgia Tech Research Institute, Atlanta, GA 30332 USA

## ABSTRACT

Fusing 3-D generated scenes from multiple, spatially distributed sensors produces a higher quality data product with fewer shadows or islands in the data. As an example, while airborne LiDAR systems scan the exterior of a structure, a spatial mapping system generates a high resolution scan of the interior. Fusing the exterior and interior scanned data streams allows the construction of a fully realized 3D representation of the environment by asserting an absolute reference frame. The implementation of this fused system allows simultaneous real-time streaming of point clouds from multiple assets, tracking of personnel and assets in that fused 3D space, and visualizing it on a mixed-reality device. Several challenges that were solved: 1) the tracking and synchronization of multiple independent assets; 2) identification of the network throughput for large data sets; 3) the coordinate transformation of collected point cloud data to a common reference; and 4) the fused representation of all collected data. We leveraged our advancements in real-time point cloud processing to allow a user to view the singular fused 3D image on a HoloLens. The user is also able to show or hide the fused features of the image as well as alter it in six degrees of freedom and scaling. This fused 3D image allows a user to see a virtual representation of their immediate surroundings or allow remote users to gain knowledge of a distant location.

**Keywords:** LiDAR, AR, augmented reality, point cloud, spatial mesh

## 1. INTRODUCTION

The risk of loss of personnel and aircraft increase substantially once a degraded visual environment (DVE) is entered. A DVE condition exists when a pilot loses spatial awareness between the orientation of the aircraft and the physical world around it. Military aviation has experienced catastrophic losses of life and equipment due to DVEs which has led to a substantial effort to aid the pilot's ability to maintain situational awareness of the aircraft in relation to an obscured world around them [1]. DVEs affect both military and first responders alike such that the speed and manner in which they operate must change according to the level of degradation they experience. Proposed solutions to DVE include light detection and ranging (LiDAR), radar, and optical sensors. The biggest challenge for any sensor is to be able to penetrate the obscurant and resolve a usable return signal [2]. Alternatively, a forward-looking LiDAR scanner can collect point cloud data of the landing zone or flight path on ingress before a DVE condition presents itself. During the DVE condition, the pilot would then monitor the aircraft's position within the point cloud created on ingress. This point cloud captured on ingress, and up until a DVE condition, would be handed off to all operational personnel and aircraft within the same area of operations. This would permit the personnel deploying from the rotary aircraft to be able to effectively move through the DVE to the objective and also allow other aircraft to use the data to operate safely within the obscured area. This is feasible by creating a common operating picture (COP) by utilizing a combination of various types of optical sensors that perform simultaneous localization and mapping (SLAM) of the area of operation by creating point clouds. The combining of SLAM data from multiple sensors allows for a COP to be generated from various perspectives in order to aid users in a DVE. If done in real time, the ability of the user to operate in a DVE with a synthetic representation of the world around the user would undoubtedly mitigate risk and increase operational tempo. This paper addresses this issue directly by providing users a synthetic representation of the world around them as well as a method to track personnel and assets in 3D space.

In general terms, this is facilitated by a suite of sensors that pass collected data to a server that aggregates the sensory data, processes it into fused scene, and disseminates the fused scene to clients who request it. This allows for the creation of a much higher detailed scene for both the direct action and command elements. This fused scene is largely attainable through SLAM processing which will also provide the precise position of the personnel or assets that is using the SLAM sensor. This presents 3D awareness of where personnel are within the area of operation through optical means, which is invaluable when operating in GPS denied or urban environments.

Three-Dimensional Imaging, Visualization, and Display 2018, edited by Bahram Javidi, Jung-Young Son,
Osamu Matoba, Manuel Martínez-Corral, Adrian Stern, Proc. of SPIE Vol. 10666, 106660S
© 2018 SPIE · CCC code: 0277-786X/18/$18 · doi: 10.1117/12.2304977

## 2. SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)

### 2.1 SLAM Implementation

SLAM can be performed by using visual odometry with passive electro-optical (EO) cameras, depth sensing EO active cameras, and LiDAR scanners. The two methods focused on are depth sensing EO cameras and LiDAR scanners such as the Microsoft Kinect and Velodyne VLP-16 as seen in Figure 1. Since passive EO sensors, such as RGB cameras used in visual odometry, perform poorly with low ambient light, active sensors were chosen because of their ability to create higher fidelity data in a wider range of lighting conditions [3]. The Microsoft HoloLens, shown in Figure 2, functions similarly to the Kinect where it uses coded infrared light pulses to generate a depth map of the scene [4]. The LiDAR scanner that was used is a proprietary bathyometric LiDAR scanner that was developed by the Electro-Optical Systems Laboratory (EOSL) at the Georgia Tech Research Institute (GTRI) [5]. It had previously performed a scan on the Georgia Tech campus which will be used for the data fusion process discussed later in this paper.

Figure 1. Active scanners designed to create point clouds of their viewable area. Velodyne VLP-16 360° LiDAR scanner (left) and Microsoft Kinect structured light sensor (right).

Figure 2. Microsoft HoloLens which operates by using structured light.

### 2.2 HoloLens Standard Operation and Practices

Instead of starting from the ground up and creating a stand-alone SLAM system, the HoloLens was utilized because of its ability to perform SLAM using structured light as well as being able to act as both a scanning and visualization device. The HoloLens constantly captures surrounding spatial information using an array of photoelectric sensors. Using spatial information captured by the photoelectric sensors, the HoloLens can introduce realistic interactions between virtual objects and the user's physical environment. Some common interactions include placing objects, occlusion, and physics simulation. Using the two sets of depth cameras and the IMU, the HoloLens performs SLAM at run-time and obtains the orientation and position of the device with respect to its environment. This is inherent to the normal operation of the

HoloLens and occurs the moment it is powered on. A user's application accesses this spatial information and allows the user to interact with holograms that behave as if they were physically in the same space as the user. This is the manner in which the HoloLens is typically used by consumers where holograms are positioned relative to the user and the physical boundaries of the user's surroundings. This method of visualization works for enclosed spaces such as homes or offices, but the HoloLens' control loops begin to experience challenges in other environments.

## 2.3 HoloLens SLAM Capabilities and Limitations

The HoloLens' front-end SLAM process is a fixed function that the user does not have access to. The spatial mesh construct that the HoloLens generates is a product of this fixed function and is accessible to the user. This allows a user to create applications that interact with the environment. Furthermore, the spatial mesh can also be displayed as a hologram to the user as seen in Figure 3. When viewing the special mesh, existing programs in the Unity IDE allow the user to modify certain parameters. The Spatial Mapping Observer script allows the user to alter the triangle density, limits of extent, and update rate of the virtual mesh [6].

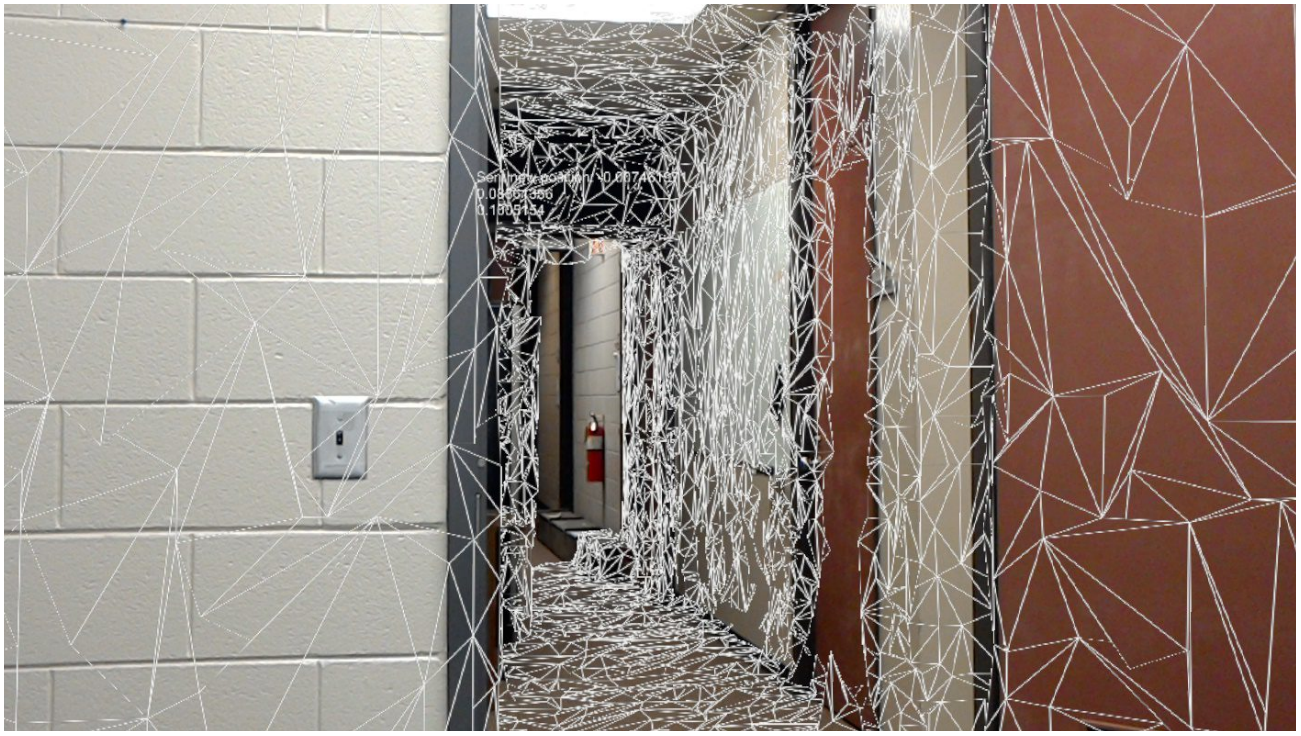

Figure 3. Virtual representation of stored spatial mesh.

Once the HoloLens is powered on, it will scan its surroundings and attempt to match it to any spatial meshes that it has previously stored into memory. Instead of completely re-scanning a room that it has scanned in the past, it can recall a stored mesh and updated any physical changes to the room between a past scan and the current one while also correcting for a dynamically changing scene. These changes constantly refine the spatial mesh that the HoloLens is building for the room by making it extremely detailed. If the HoloLens encounters a room it has not been in before, it will begin making an initial scan of the room and commit it to memory. The HoloLens SLAM performance is outstanding when used in a small setting, but it can misbehave when it counters some seemingly trivial situations.

The first situation where the SLAM processing fails if differential movement is found between the on-board IMU and SLAM process. For instance, if the user is entering a moving elevator or vehicle. Once the vehicle or elevator begins to move, the scanned surroundings that appear to be stationary relative to the HoloLens optical sensors does not agree with the IMU's detected of movement relative to the world. When this occurs, the SLAM processing fails and re-starts. Because of this, the HoloLens is not suitable to be used on any moving platform where it is not able to track any features external to the platform.

The second situation is where the user moves too quickly through a scene. If the HoloLens isn't given enough time to correlate where it is now to where it was, the SLAM process will fail and it will have to re-start scanning process. This limitation occurs as it is no longer able to perform localization and track itself in space or add new mesh data to its stored spatial mesh. The implication is that the HoloLens is not suited for anything more than casual office or home use.

The third situation is where and how it will attempt to perform loop closure. The HoloLens is exceptional in how it performs loop closure by combining at least two processes. The first process is what appears to be a standard iterative closest point method. The second process is where it will attempt to recall a stored mesh if it appears sufficiently similar to its current surroundings. For instance, a stairwell that connects multiple floors will often look the same from the first step from the basement, to the last step at the top floor. Because the stairwells lack defining features on each floor, once the HoloLens climbs to the second flight of stairs, it will look relatively exactly like the first flight of stairs and cause the HoloLens to perform loop closure. As a result, the HoloLens will begin injecting spatial mesh data and corrupt the model as seen in Figure 4. With these limitations in mind, it was decided to access the spatial mesh data in a different manner to avoid some of these pitfalls.

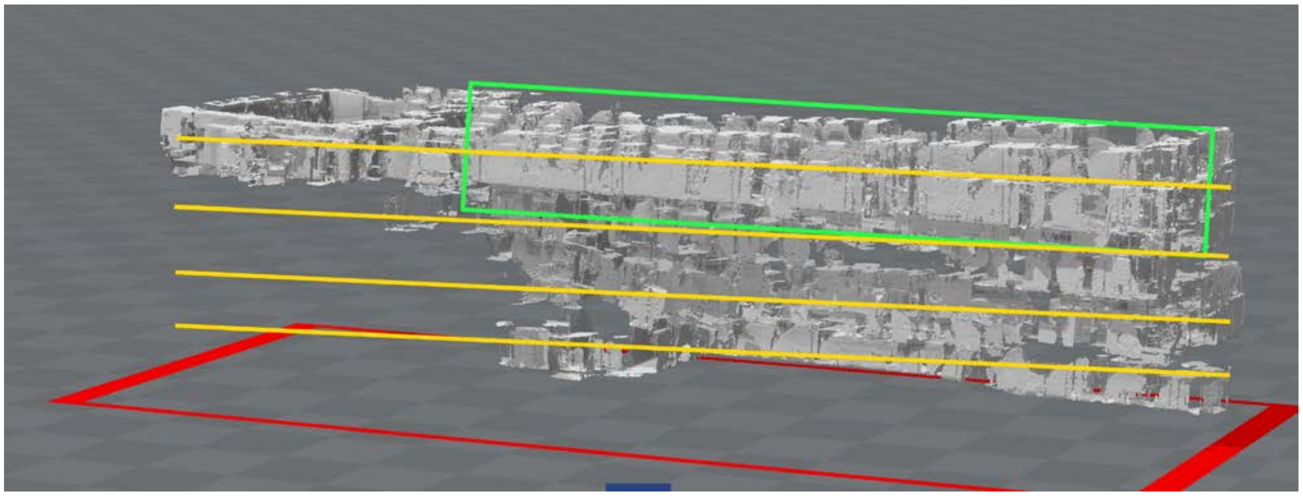

Figure 4. Loop closure problem experienced in stairwells. Notice how a four story structure has five to seven stories, some of which appear between stories and some superimposed on the same level. The orange lines indicate where the four stories are approximately supposed to be. The green box indicates an example of superposition of the same story over itself.

### 2.4 HoloLens New Implementation

While the HoloLens can perform SLAM and visualize the mesh in real-time, we've chose to decouple the two processes and have one HoloLens scan while another visualizes. The two users are described as the scanner and the observer who both are utilizing HoloLens devices. The scanner will scan their environment in order to create the spatial map of it and transmit it to a server along with the scanner's position in 6 degrees of freedom (6DOF). The observer, who is at a remote location, can then pull that data from the server, display a hologram that renders the spatial map that the scanner created, and display the scanner's orientation and position within the hologram. This implementation is the manner in which the HoloLens is to be utilized for this paper.

## 3. MESH & 6DOF STREAMING

### 3.1 Mesh Streaming

A Unity application was developed to tap into the spatial mapping and 6 degrees of freedom (x,y,z position, and x,y,z rotation) information built-in to the HoloLens to build a mapping dataset. As the HoloLens creates new meshes and updates its position in the world, the data is serialized and packetized so it can be transmitted in real time to a central server. Within the Unity application, a function runs every frame to check for any updates to the spatial map generated by the HoloLens. If a new mesh update exists, the HoloLens attaches a 'Mesh ID' to each new mesh and performs updates to previously existing spatial mesh constructs. All new and updated meshes are provided by a SpatialMapping class in each frame [7].

When a mesh is updated or newly generated, the application serializes the data and sends it to a central server via an HTTP Post Request. The data is serialized as follows: a header, list of vertices, and a list of indices for generating the triangles for each mesh. The header has the mesh id, number of vertices, number of triangle faces, and the 7 floats associated with position and orientation data of the HoloLens user. The XYZ positional information is with respect to the initial location of the HoloLens at power-up and is streamed as long as the application runs. In addition, information about the meshes is streamed to the server by using every 3 indices in the index list. Every 3 indices point to locations in the list of vertices that correspond to a face of the mesh. A reconstructed mesh is seen in Figure 5.

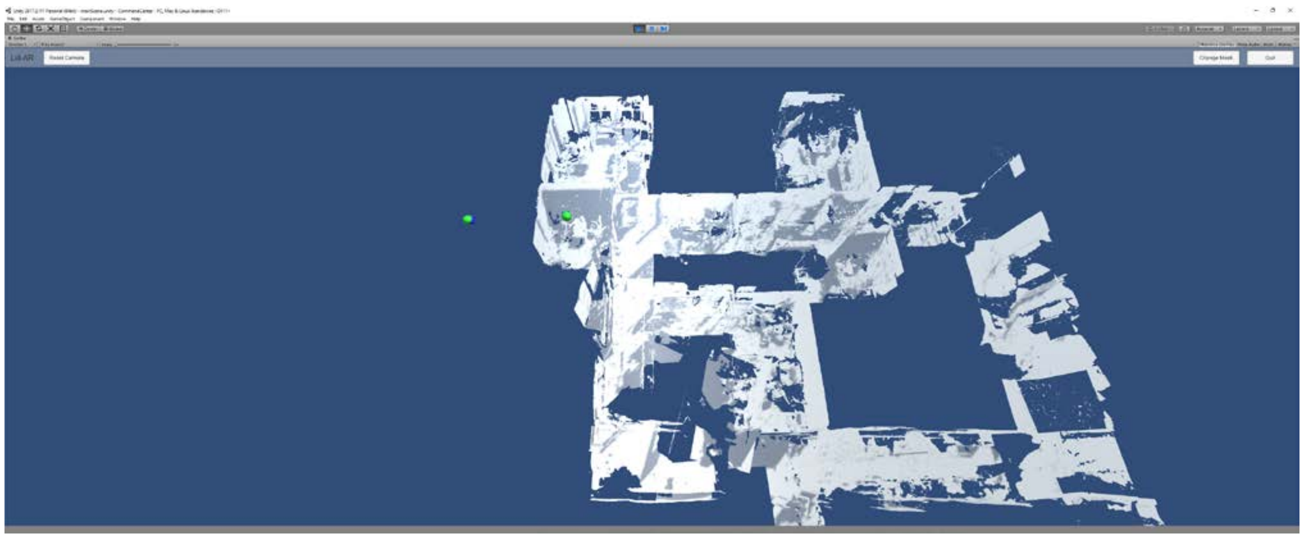

Figure 5. Top down view of a reconstruction of a streamed mesh of offices and hallways at GTRI. Two green spheres indicate a user's position in the spatial mesh.

## 3.2 Tracking of the User in 3D Space

The HoloLens tracks its position and orientation in space by using SLAM and an onboard IMU with positional and pose accuracy at centimeter resolution. The process is abstracted from the application by the HoloLens API, which simply provides the already-computed mesh data. Lower-level access to the mapping information is not accessible, so we must rely on the built-in processing. A user’s position and 6DOF orientation is tracked and displayed in Figure 5.

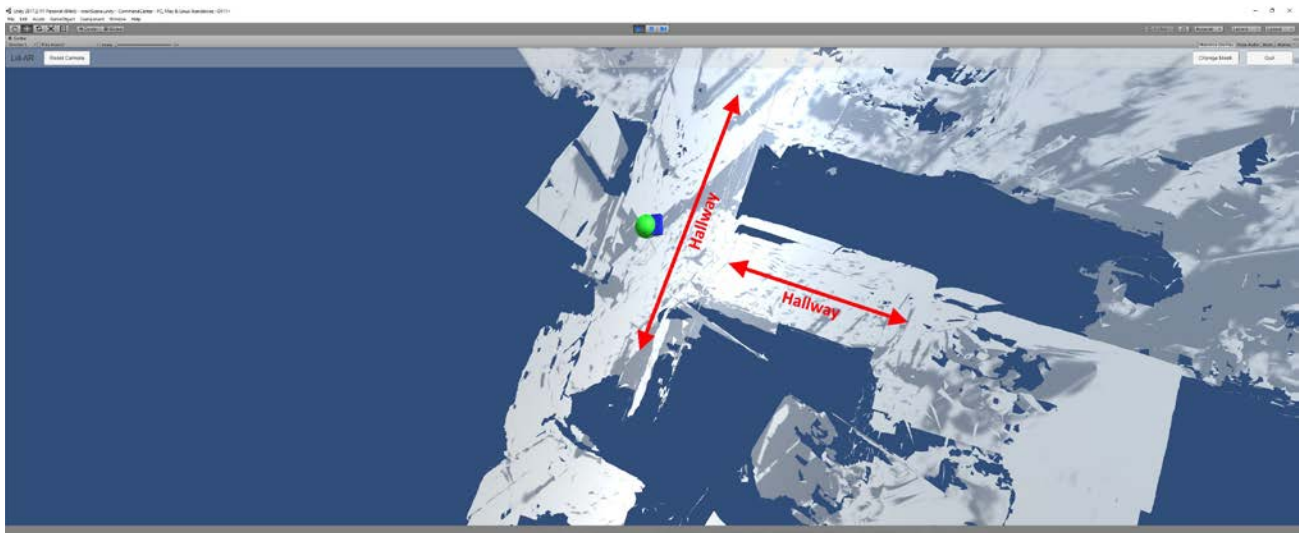

Figure 6. HoloLens user (green sphere with blue visor) position in 6DOF shown in real time in the spatial mesh they just created.

### 3.3 Timing and Throughput

The HoloLens posts these newly generated meshes to the central server every three seconds and 6DOF positional information is posted every 500 milliseconds. The client can then request the newly generated serialized meshes from the central server at any interval desired. The client has the ability to pull all mesh and positional data from the server, a subset thereof, or only the latest additions since the last request. In this manner the throughput is minimal such that the average mesh file size is less than one megabyte.

## 4. POINT CLOUD STREAMING

In addition to HoloLens-generated spatial meshes, the server can also store generic point cloud data, for example, point clouds produced by a LiDAR system. The point values may be produced in any of a number of formats: local coordinates, Universal Transverse Mercator (UTM), geodetic latitude, longitude, altitude, or geocentric coordinates to name a few, so the format must be provided to the server as metadata. Currently the server only accepts coordinates in geocentric x,y,z format, however there are a number of transformation algorithms being implemented to convert between the various formats. Once converted to the default geocentric format, the data is stored and available for client systems to download. When a client requests stored data, the server could use the same transformation algorithms to convert the geocentric coordinates to one of the other formats required by the client's viewing device.

### 4.1 Receiving New Point Clouds

Similarly to HoloLens mesh data, the server provides an HTTP endpoint for submission of new point cloud data. The data must be in JSON [8] format, come from a registered device, and provide metadata that includes a pre-defined value indicating the data format. Once received, the server converts from the provided format to geocentric coordinates if necessary and stores it with a unique key.

### 4.2 Coordinate Conversion

Well-known conversions exist between the various earth-based coordinate transforms, and the server will house implementations of a number of these formulas. Currently our LiDAR data is transformed to geocentric coordinates before being sent to the server, so there is no need for further transformations. HoloLens coordinates are stored on the server unchanged from their local coordinate system, so the transformation to geocentric coordinates is performed on the viewing application. Since the HoloLens is not outfitted with a GPS tracker this gave us a convenient location to manually enter origin coordinates for the HoloLens, and then use this origin to render the meshes in their proper location relative to the point-cloud data. For a HoloLens or other SLAM sensor equipped with a GPS, the sensor would provide a GPS location and an orientation to the server along with local-coordinate mesh or point data. Given this information, plus the provided data format, the server can convert the data to geocentric coordinates, eliminating the need to do it on the client observer.

### 4.3 Point Cloud Retrieval

Clients that wish to display or otherwise use point cloud data can then request a list of datasets currently stored on the server and use the provided list of keys to request the actual data. Along with the data request, the client can include a desired data-format. The server converts the point cloud to the requested format before sending it to the client, making it flexible and convenient for clients that need to consume the data in a particular format. In the future, a location based query system will be implemented so that a client may retrieve all data near a particular area instead of pulling all files from the server or having to sort through a list of point clouds and spatial meshes.

## 5. REAL TIME SCENE FUSION

While the server component of our system provides a means to aggregate, normalize, and store spatial information from multiple sensor types, the usefulness of that data only becomes apparent when combined together. To do so, a client program that GTRI built in Unity pulls both interior mesh data and exterior point cloud data from the server, adjusts them based on the current coordinate space, and displays them as a fused scene. By polling the server for updates to these data sets, the client can also update the scene in near real-time as sensors actively explore an area. The 6DOF information is updated at a rate of 2Hz and renders the scanner as a green sphere and a black visor among the mesh data. The mesh and point cloud data is updated every three seconds, with the client intelligently managing which meshes it already has vs meshes that need to be updated. This scheme handles updates from any number of sensors, allowing new data to stream in from multiple devices at once. Figure 6 shows how the data flow between sensor and clients.

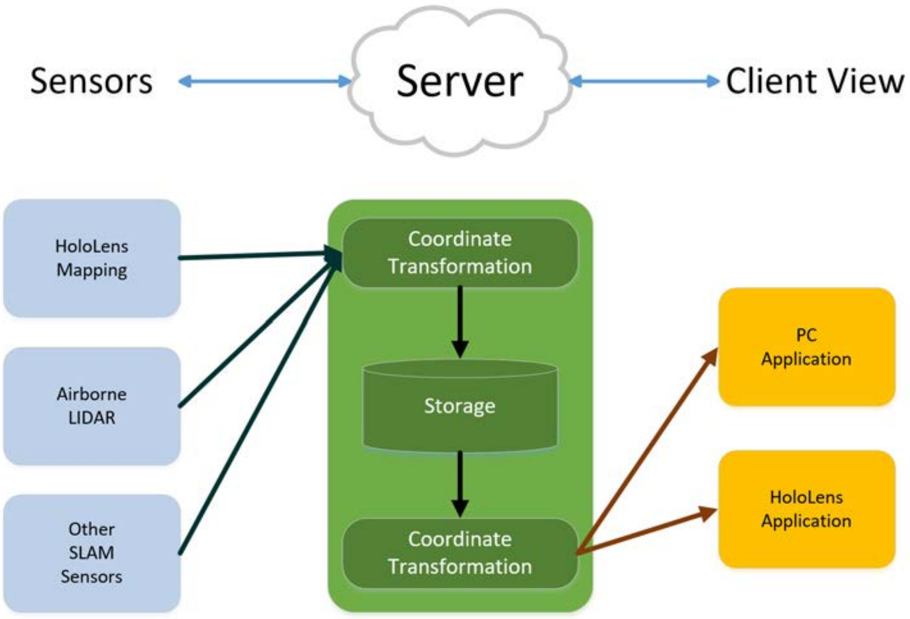

Figure 7. Data fusion flow chart

### 5.1 Data Fusion

To reduce the client application's workload, the server is planned to do most of the heavy lifting with regards to data fusion. However, currently the client program is responsible for transforming the HoloLens spatial mesh data to the same coordinate space as the point cloud data. To do so, the client needs origin location and orientation information for the HoloLens mesh data. Unfortunately, there is no GPS onboard the HoloLens and the IMU is not accessible to the user. This information is currently entered manually in the client, however it could easily come from GPS and magnetometer sensors (or similar) attached to the HoloLens itself and connected to it via Bluetooth. The client is capable of displaying mesh information from multiple HoloLens devices, but unique location and orientation information would be required for each device to render them properly in the same scene. HoloLens location data is provided in the same coordinate space as the meshes, so the client transforms it in the same way to display sensor location in the scene. Finally, point cloud data is already in global coordinate space, so the client simply shifts the point cloud such that it is centered in Unity's coordinate space, stores the adjustment and applies it to any new data that comes in.

## 6. REAL TIME FUSED SCENE REPRESENTATION

In its simplest form, all of the spatial data collected by the server are just points in space. As long as they are transformed to the same coordinate space, they can all be plotted together and form a cohesive image as seen in Figure 7. In practice, the HoloLens provides some additional information (triangles and surface normals) which, when implemented, gives the advantage to show a more detailed interior scene. The main client application runs on a desktop PC and allows the user to 'fly through' the scene in real time, like a first person camera. Although the PC application is the most practical way to view the fused scene, it can also be represented as a hologram via the HoloLens. Augmented reality (AR) provides a more immersive view of the scene which can be helpful in getting a better idea of distance, scale, and relative positioning compared to a computer monitor.

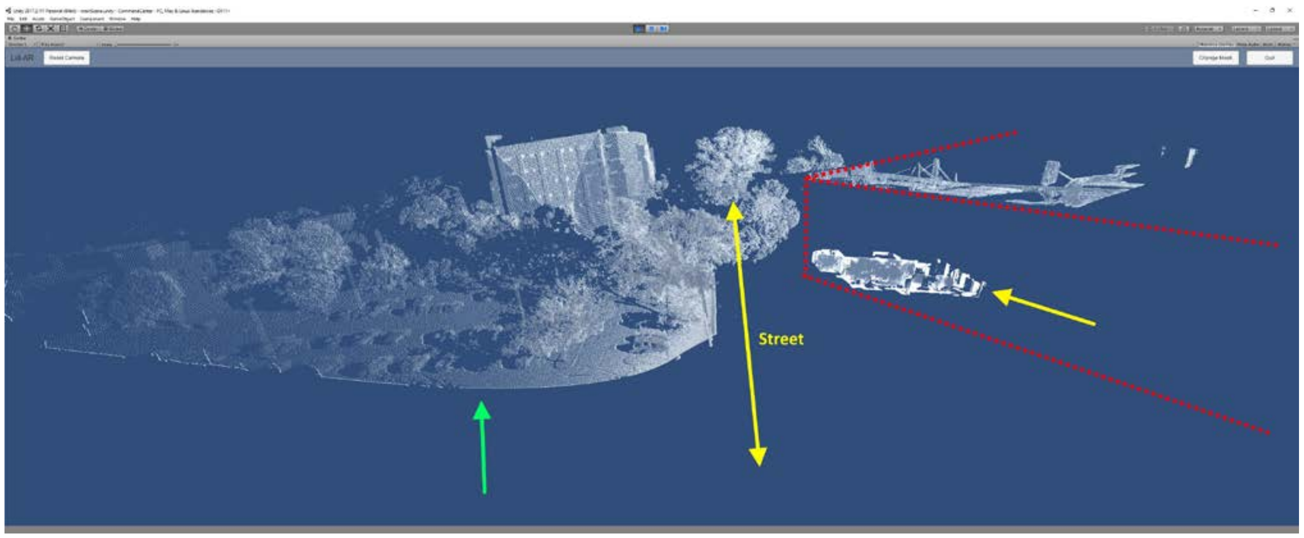

Figure 8. HoloLens spatial mesh (yellow arrow) and fused LiDAR point cloud (green arrow). External point cloud and internal scanned mesh are positioned relative to global coordinates to generate a common operating picture. Since these are two separate buildings, superimposed red dotted lines are drawn to help visualize the fused scene.

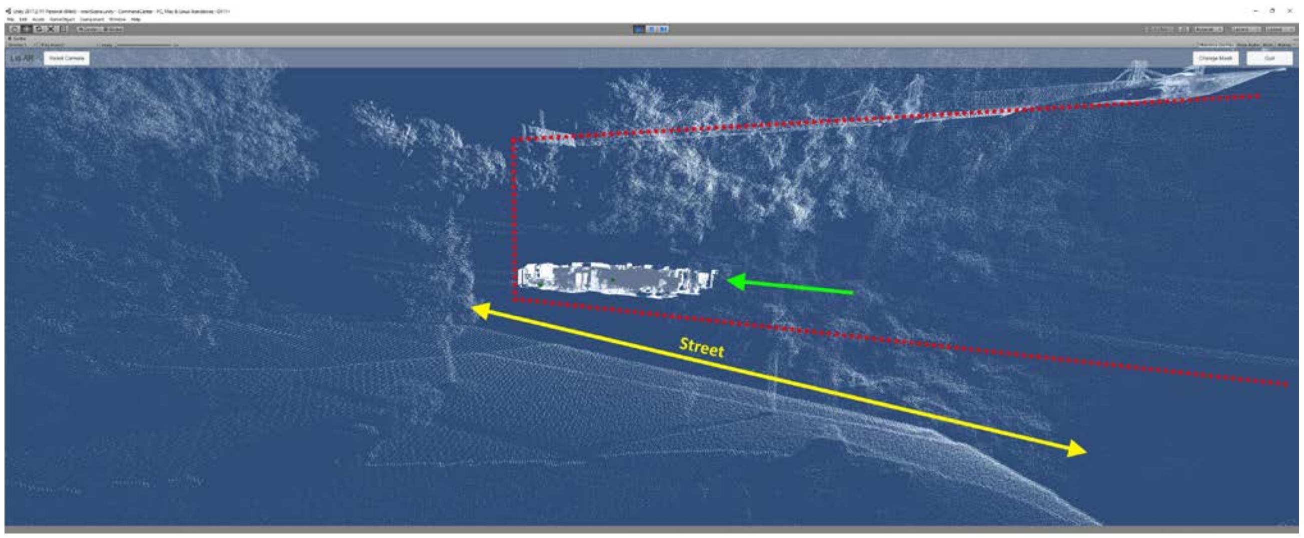

Figure 9. View of the reconstructed mesh (green arrow) through the LiDAR point cloud. Red dotted lines indicate the approximate structure of the building and Figure 8 can be used as a reference as to the orientation of the point cloud and mesh.

## 6.1 Data Display

The Unity client application on a PC dynamically constructs mesh objects for both the point cloud and HoloLens mesh data. The point cloud data is rendered as points-only while the HoloLens' generated spatial meshes include triangles which form basic surfaces. The client periodically queries the server for any updated information and updates the scene accordingly in real time. Finally, the client also displays the last known position and orientation of each HoloLens device that is sending spatial meshes to the server, allowing an observer to actively monitor progress and track assets as they map out a scene. All objects in the scene are correctly localized and the user has a free-look camera control they can use to get an overview of the entire scene or explore specific areas in more detail. The same scene could also be viewed in AR by using the HoloLens as a client. If the client HoloLens's position is known, the hologram could be rendered at full scale, allowing the user to be immersed in the holograms and point clouds, as well as, see through walls and obstructions to areas that have been mapped. Another useful display method the HoloLens can utilize is fiducial markers. Using preexisting libraries and toolkits, the HoloLens is able to recognize predefined fiducial markers to display unique holograms [2]. Using the integrated RBG camera, it is able to identify the orientation of the marker and therefore rotate the hologram accordingly. The advantage to using fiducial markers invites the use of other AR capable devices to view the same hologram amongst other clients who are viewing the same fiducial marker.

## 7. CONCLUSION AND PATH FORWARD

While the HoloLens is a very capable AR device, it is not suited to be used in any sort of a tactical situation. Instead, a suitable SLAM device can be created that would be oriented to this tactical application. We are planning on developing a lightweight SLAM system using a combination of passive and active EO sensors. These sensors will be used on airborne and ground assets to perform rapid development of fused scenes to be used in austere tactical situations. We will also integrate a Velodyne VLP-16 that will move dynamically through a scene to generate dense point clouds at a faster rate than RGB-D sensors or visual odometry can create.

The server implementation as a proof of concept worked well and we will be working towards integrating additional sensors and clients into it. Additional effort needs to be placed in the client's display of the fused scene and to find the appropriate manner in which to render relevant information to the user given the display type.

Overall, this moderate effort into the fusion of point clouds and spatial meshes is considered promising success. The implication of this real time spatial mapping and point cloud fusion is wide reaching. The ability to perform SLAM and display the output in real time is a significant feat considering most SLAM analysis and modeling isn't fully realized in real time and typically not transmitted to a central location that can be distributed to any number of clients. The implications of this real time capability to military and first responder operations in DVE conditions will change tactics, techniques, and procedures (TTPs) currently being used and allow for a higher operation tempo and even using the DVE as cover and concealment instead of treating it a risk factor.

## 8. ACKNOWLEDGEMENTS

We would like to thank Justin Kalloor who performed the feasibility analysis of the HoloLens and Greg Bishop for setting up the server framework that facilitates data transfer for this project.

## 9. REFERENCES

[1] D. Vergun, "Army Aims 'to Own the Weather'," 4 May 2016. [Online]. Available: https://www.army.mil/article/167293/army_aims_to_own_the_weather.

[2] A. Oran and et al, "Three-dimensional imaging in degraded visual field," *Journal of Physics : Conference Series,* vol. 707, 2016.

[3] P. Henry and et al, "RGB-D mapping: Using Kinect-style," *The International Journal of Robotics Research,* vol. 31, no. 5, 2012.

[4] PrimeSense, "PrimeSense 3D Sensors," [Online]. Available: http://www.i3du.gr/pdf/primesense.pdf. [Accessed 21 March 2018].

[5] A. Harvey, "Georgia Tech Research Institute Lidar Project," Military Embedded Systems, 27 January 2015. [Online]. Available: http://mil-embedded.com/articles/georgia-institute-lidar-project/. [Accessed 18 March 2018].

[6] Microsoft, "Holograms 101E," Microsoft, [Online]. Available: https://developer.microsoft.com/en-us/windows/mixed-reality/holograms_101e. [Accessed 1 September 2017].

[7] Quian256, "HoloLens AR Toolkit," GitHub, 16 July 2017. [Online]. Available: https://github.com/qian256/HoloLensARToolKit/blob/master/README.md. [Accessed 1 September 2017].

[8] "Introducing JSON," JSON.org, [Online]. Available: https://www.json.org/. [Accessed 21 March 2018].