Simulation of LIDAR Systems for Aerial Intelligence, Surveillance, and Reconnaissance

Mr. Steven Webster, President
KINEX INC.
United States of America

Dr. Todd Du Bosq, Physicist
US. Army CERDEC, RDECOM, Night Vision and Electronic Sensors Directorate
United States of America

Mr. Vinh Tran, Principal
Simeon Technology
United States of America

Mr. Kirby Thomas, Chief Army Intelligence Modeling and Simulation
Headquarters Department of the Army, Office of the Deputy Chief of Staff, G-2
United States of America

Mr. Christopher May, Branch Chief - Simulation Applications
US. Army CERDEC, RDECOM, Night Vision and Electronic Sensors Directorate
United States of America

ABSTRACT

Airborne Light Imaging, Detection, And Ranging (LIDAR) systems generate high resolution point cloud models of the surveyed terrain. The collected data supports the Intelligence processes of terrain surface reconstruction, material classification, and feature extraction. When composed with other sensor models observing a shared virtual environment, the simulated aerial LIDAR improves the Intelligence stimulus of Command and Control (C2) systems. Given the cost, availability, control and variability advantages over live alternatives, simulated aerial LIDAR LASer file format (LAS) point clouds also broaden the interoperability testing of consuming systems.

Our method applies LIDAR physics and performance parameters to 3D rendering to generate standard LAS data. The related application associates the LIDAR simulation with a maneuver flight path, orients the collection perspective as commanded, and simulates each scan with physics based propagation, attenuation, noise and voids. Successive scans are composited to append or refine the generated point cloud, and finally encoded into LAS files. We address the resolution mismatch between LIDAR capability and simulation databases and compare our simulated product to live collections. Mitigations of terrain fidelity and verification of simulation LIDAR products are discussed and solutions proposed.

1.0 LIDAR OVERVIEW

Light Imaging, Detection, And Ranging (LIDAR) systems measure range to surfaces by generating laser pulses and measuring the associated return time and intensity. Each pulse return indicates a relative position by range from the originating source, and hence an absolute point location can be derived from the LIDAR sensor's location, orientation, and relative range of each pulse return. LIDAR systems generate these points at rates on the order of 150,000 per second,[1] and the set of points in a collection is referred to as a "point cloud." Point clouds are post

processed into a variety of products such as Digital Elevation Models (DEM), terrain features (buildings, foliage …), and mensuration information. Recently given its capability to provide relative range data in real time, more compact LIDAR sensors have been applied to autonomous vehicles. A sample point cloud from our LIDAR simulation, generated with standard deviations of 0.3 milliradians and 0.82 meters in angular and range error respectively, is shown in Figure 1. As the parameters in the system improve so will the quality of the LIDAR simulation.

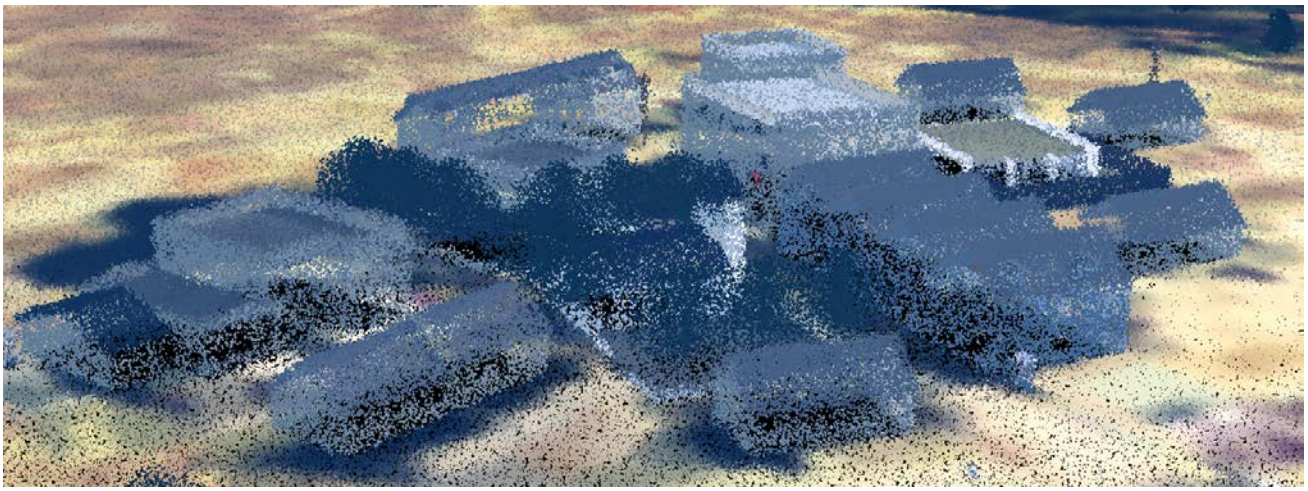

Figure 1: Simulated LIDAR point cloud

On the aerial side, LIDAR point clouds are used in general surveying, where a Digital Elevation Model (DEM) results from linear flight paths that accumulate LIDAR swaths over the area of interest. When high resolution of point data is desired, multiple passes from different observation angles for a given location are collected to increase the point cloud's points per meter squared (ppm^2). Multiple points in near positions are composited to decrease location uncertainty and improve accuracy. Multiple observation angles may also reveal features not apparent from direct overflight and nadir sensor angles. The technique brings "angular diversity," and it is the initial basis for our simulation implementation.

2.0 OPERATIONAL RELEVANCE

LIDAR provides the commander with an ability to find concealed/obscured targets, detect the presence of structures, and uncover enemy attempts at camouflage, concealment and deception. LIDAR also provides highly accurate elevation data and geospatially accurate terrain information supporting tactical commanders' planning, training, and operations cycle. Derivative data products from LIDAR collections are Bare Earth (BE), Human Activity Layer (HAL), and Feature Extractions (FE) which include DEMs, buildings, trails, power line, planar segment and voids. Benefits to the Soldier include detailed terrain information for determining mobility, obstructions, Helicopter Landing Zones (HLZ), FOliage PENetration (FOPEN) and 3D visualization and Line-of-Sight (LOS) analysis of operational environments. Beyond analysis, LIDAR collections support targeting with accurate mensuration and associated collateral damage estimation and reduction. Within the planning cycle, LIDAR products support detailed mission planning (raid, strike) in engagements from dense urban areas or across wide geographic areas. Given the integral role of LIDAR collection in military intelligence and maneuver execution, combined with the utility of modelling and simulation in mission training and rehearsal, LIDAR simulation that directly correlates with other component simulations enables the force to virtually operate within a single cohesive environment.

3.0 SIMULATION APPLICATIONS

Beyond supporting mission functions, modelling and simulation tools are also being used throughout a sensor's acquisition lifecycle from virtual prototyping and concept exploration to new equipment and sustainment training. Sensor areas such as Electro-Optic/Infrared (EO/IR) and Synthetic Aperture Radar (SAR) have very mature simulation tools to support these activities. The recent conflicts throughout the world have pushed other sensor technologies such as LIDAR to the field, and simulation tools are now needed to support these technologies.

Multiple use cases for LIDAR simulation are immediate needs in the community. With the advent of deep learning and sensor analytics, simulated LIDAR point clouds can ease the training and verification stages of Artificial Intelligence (AI) algorithm development. Simulation has advantages over live content, since ground truth is simply the parametrization of the simulation rather than manually correlated observations on live content. Consequently, the number of permutations is much greater with simulation than can be derived from live-only training. This technique is currently used to train analytics for visible and Infrared (IR) full motion video, and the LIDAR content is expected to follow.

Beyond the AI training use case, LIDAR simulation can be used as alternative stimulus to Command and Control (C2) applications and systems. Through integration with C2 protocols, the LIDAR simulation could be virtually employed over a constructive environment, and thereby generate prerequisite intelligence products for large scale simulation experimentation. On a smaller scale, the same C2 – simulation integration could support LIDAR collection training, where the operator is virtually emplaced on an aircraft and has identical control over the simulated LIDAR sensor as in live operations. From individual through collective training, accurate representations of LIDAR sensor performance can be used to refine Tactics, Techniques, and Procedures (TTPs) collection activities and stimulate the Intelligence analysts.

All use cases of sensor simulation, including LIDAR, are dependent on being sufficiently "representative" for purpose. The LIDAR simulation consists of a layered approach where first principal physics is executed in a rendering pipeline, point clouds with intensity values are generated, and are subsequently written to file. By starting with physics rather than artistic approximates, the level of representation is well defined.

4.0 SIMULATION PHYSICS

The general equation to calculate the power received, P_r, in Watts from a LIDAR system is the laser radar range equation[2] given by

$$P_r = \frac{4\,K\,P_s\,\tau_{(R1)}\eta_t}{\pi\,\beta^2\,R_1^2}\,\Gamma\,\frac{\tau_{(R2)}}{4\,\pi\,R_2^2}\,\frac{\pi\,D^2\,\eta_r}{4}, \tag{1}$$

where P_s is the source laser power in watts, K is the laser beam profile function, $\tau_{(R1)}$ is the atmospheric transmission from source to target, η_t is the optical efficiency of the transmitter, β is the beam divergence, R_1 is the range from the transmitter to the target, Γ is the laser cross section of the target in square meters, $\tau_{(R2)}$ is the atmospheric transmission from the target to the receiver, R_2 is the range from the target to the receiver in meters, D is the diameter of the receiver aperture, and η_r is the optical efficiency of the receiver. The laser range equation can be separated into four sections. The first term represents the propagation of the laser light to the target. The second term represents the reflection of the laser light by the target. The third term represents the propagation of the scattered laser light from the target to the receiver. The fourth term represents the collection of the scattered laser light by the receiver.

The basic equations used in the LIDAR simulation are similar to equation 1 but are defined radiometrically. A schematic of the LIDAR geometry is shown in Figure 2. The LIDAR system is composed of a laser and camera, or single detector for a scanning LIDAR. Radiation from the laser and other sources elastically scatters from the scene to the detector with a given reflectivity and range. The laser and detector can be either bistatic, where the laser and receiver are separated by a distance, or monostatic, where the laser and receiver are collocated. For the examples presented in this paper, the monostatic configuration is used for simplicity with $R_1=R_2$.

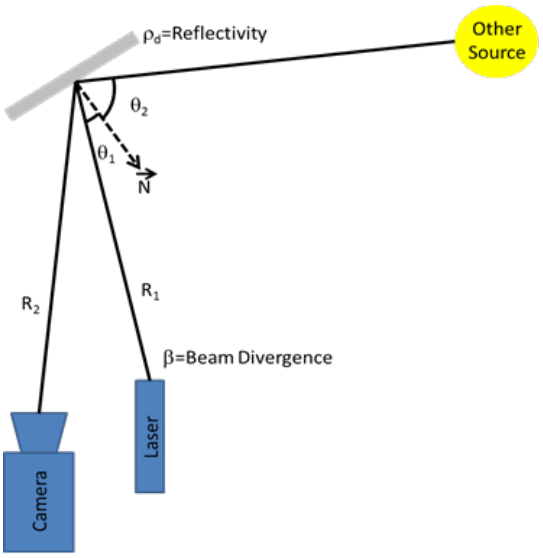

Figure 2: Schematic of LIDAR Geometry

The LIDAR simulation accounts for all of the photons and their interaction within the scene. The primary source of photons is from the laser. The laser photons propagate through the atmosphere to the objects in the scene. The spectral irradiance illuminating an instantaneous field of view of the scene from the laser, $E_{e\lambda,laser}$, is given as,

$$E_{e\lambda,laser}(R_1) = \frac{\varphi_{e\lambda}(R=0)\cdot\tau(R_1)}{\pi\cdot R_1^2\cdot tan\left(\frac{\beta}{2}\right)^2}\left[\frac{watt}{cm^2\cdot\mu m}\right], \quad (2)$$

where $\phi_{e\lambda}(R=0)$ is the spectral power of the laser in watts per μm at zero range, R_1 is the range from the laser to the target in cm, τ is the atmospheric transmission, and β is the beam divergence.

The photons from the laser will then reflect off of the scene. The spectral radiance from each IFOV of the scene, assuming Lambertian reflectance by the laser, $L_{e\lambda,laser}$, is given as

$$L_{e\lambda,laser}(R_1,\lambda) = \frac{\rho_d(\lambda)}{\pi}\cdot E_{e\lambda,laser}(R_1)\cdot\cos(\theta_1)\left[\frac{watt}{cm^2\cdot sr\cdot\mu m}\right], \quad (3)$$

where ρ_d is the reflectivity of the scene, θ_1 is the angle between the laser and normal vector of the scene, and λ is wavelength in μm.

The photons from the scene then propagate back through the atmosphere to the camera. The spectral power received by the detector from reflected scene illuminated by the laser, $\phi_{e\lambda,laser}$, is given as

$$\varphi_{e\lambda,laser}(R_2,\lambda) = \rho_d(\lambda)\cdot\frac{E_{e\lambda,laser}(R_1)}{4\cdot\left(F/_{\#}\right)^2}\cdot\cos(\theta_1)\cdot A_{pix}\cdot\tau(R_2)\left[\frac{watt}{\mu m\cdot pixel}\right], \quad (4)$$

where R_2 is the range from the target to the camera in cm, $(F/\#)$ is the f-number of the camera, A_{pix} is the area of the detector pixel in cm^2, and the laser spot is larger than the IFOV of the scene.

The photons are then detected by the sensor and converted to electrons. The number of electrons per pixel per frame at the detector from the laser, N_{laser}, is given as

$$N_{laser} = \tau_{int}\int_{\Delta\lambda_{laser}}\varphi_{e\lambda,laser}(R_2,\lambda)\cdot\frac{\lambda\cdot\eta(\lambda)}{h\cdot c}d\lambda\left[\frac{electrons}{pixel\cdot frame}\right], \quad (5)$$

where τ_{int} is the integration time in sec per frame, $\Delta\lambda_{laser}$ is the bandwidth of the laser in μm, η is the quantum efficiency of the detector in electrons per photon, h is the planck constant in joule·sec, c is the speed of light in μm per sec, and $\Delta\lambda_{det}$ is the bandwidth of the detector in μm.

Additional sources of radiation may be present in the scene, such as the sun or moon, and must be accounted for in the simulation. The spectral irradiance on the scene from an "other" source, $E_{e\lambda,other}$, is defined as

$$E_{e\lambda,other}(\lambda) \equiv E_{e\lambda,source}(\lambda)\left[\frac{watt}{cm^2\cdot\mu m}\right], \tag{6}$$

where $E_{e\lambda,source}$ is the spectral irradiance from the other source in watts per cm^2 per μm.

The photons from the additional source will then reflect off of the scene. The spectral radiance from each IFOV of the scene, assuming Lambertian reflectance from the additional source, $L_{e\lambda,other}$, is given as

$$L_{e\lambda,other}(\lambda) = \frac{\rho_d(\lambda)}{\pi}\cdot E_{e\lambda,other}(\lambda)\cdot\cos(\theta_2)\left[\frac{watt}{cm^2\cdot sr\cdot\mu m}\right], \tag{7}$$

where θ_2 is the angle between the other source and the normal vector of the target.

The photons from the additional source then propagate through the atmosphere to the camera. The spectral power received by the detector from the reflected scene illuminated by the other source is given as

$$\varphi_{e\lambda,other}(R_2,\lambda) = \rho_d(\lambda)\cdot\frac{E_{e\lambda,other}(\lambda)}{4\cdot\left(F/_{\#}\right)^2}\cdot\cos(\theta_2)\cdot A_{pix}\cdot\tau(R_2)\left[\frac{watt}{\mu m\cdot pixel}\right], \tag{8}$$

In addition to "other" sources, such as the sun or moon, path radiance will also contribute to the overall detected signal. The spectral power received by the detector from spectral path radiance, $\phi_{e\lambda,path}$, is given as

$$\varphi_{e\lambda,path}(R_2,\lambda) = \frac{L_{e\lambda,path}(R_2,\lambda)}{4\cdot\left(F/_{\#}\right)^2}\cdot\pi\cdot A_{pix}\left[\frac{watt}{\mu m\cdot pixel}\right], \tag{9}$$

where $L_{e\lambda,path}$ is the spectral path radiance in watts per cm^2 per sr per μm.

The number of electrons per pixel per frame at the detector for the other sources, N_{other}, is given as

$$N_{other} = \tau_{int}\int_{\Delta\lambda_{det}}\left(\varphi_{e\lambda,other}(R_2,\lambda)+\varphi_{e\lambda,path}(R_2,\lambda)\right)\cdot\frac{\lambda\cdot\eta(\lambda)}{h\cdot c}d\lambda\left[\frac{electrons}{pixel\cdot frame}\right]. \tag{10}$$

The total number of electrons per pixel per frame at the detector from all sources, N_{total}, is given as

$$N_{total} = N_{laser} + N_{other}. \tag{11}$$

As with all detectors, noise must also be incorporated into the simulation. Detector noise, ΔN, is applied to the simulation by adding a stochastic irradiance δE to each pixel and converting to electron per pixel per frame given as

$$\Delta N = \sqrt{\left(\frac{\tau_{int}\cdot\delta E\cdot\lambda_{mid}\cdot\Delta\lambda_{det}\cdot A_{pix}\cdot\eta_{avg}}{h\cdot c}\right)^2 + N_{laser} + N_{other}} = \sqrt{\left(\frac{\tau_{int}\cdot\delta E\cdot\lambda_{mid}\cdot\eta_{avg}}{h\cdot c}\right)^2 + N_{total}}, \tag{12}$$

where λ_{mid} is the middle wavelength of the detector in μm and η_{avg} is the average quantum efficiency of the detector. The term δE is assumed to be a random variable governed by a Gaussian probability distribution,

$$\delta E = \sigma_1 \cdot randn(x, y) \tag{13}$$

where σ_1 is the standard deviation of the stochastic spectral irradiance (σ_{tvh}).

In LIDAR systems, the range information is derived only when sufficient laser light reflected from the scene is detected by the sensor. Only pixels with an overall detected signal above a noise floor will form the final simulated range image. If $N_{laser} < \gamma \Delta N$, then a void will appear, where γ is a threshold value.

Noise will also be associated with the range measurement. Range noise, caused by contamination due to small additive error to the range value, must be taken into account by the simulation by adding a system defined random error δR to each pixel's range value. Once again, δR is assumed to be a random variable governed by a Gaussian probability distribution,

$$\delta R = \sigma_2 \cdot randn(x, y) \tag{14}$$

where σ_2 is the standard deviation of the range error.

Small beam divergences over long ranges can lead to uncertainties in position and attitude, especially in the presence of platform vibration or jitter. The acquisition uncertainty due to platform jitter is taken into account by adding a system defined jitter deviation δxy at each pixel. The jitter error is projected as a range dependent angle allowing the proper geometry to be maintained. The δxy is assumed to be a random variable governed by a Gaussian probability distribution,

$$\delta xy = \sigma_3 \cdot randn(x, y) \tag{15}$$

where σ_3 is the standard deviation of the platform jitter error.

5.0 SIMULATION IMPLEMENTATION

5.1 Night Vision Image Generator (NVIG)

The Night Vision Image Generator (NVIG) is the scene generation component of the NVToolset software suite. NVIG is used to create simulated multispectral imagery, including EO and IR of virtual targets and terrains in real-time. It is used in research and development as well as training and war-gaming applications. It is developed using the OpenGL 3D rendering API, and the shading language used is the OpenGL Shading Language (GLSL). This allows NVIG the flexibility to support various operating systems and hardware platforms.

NVIG has traditionally been used for simulation of passive sensor systems providing customers with sensor imagery from visible and thermal wavebands. LIDAR is a new challenge for NVIG, requiring higher precision frame buffers and updated GPU (graphics processing unit) read-back behaviours. However, the highly customizable architecture of NVIG allows it to easily accommodate new rendering methodologies and interface with user-created plugins for diverse approaches to data exploitation.

5.2 Deferred Rendering Pipeline

The NVIG rendering pipeline supports both a traditional forward renderer, where geometry and lighting calculations all occur in a single draw pass, as well as a deferred shading path. In the deferred shading path, most lighting computation is delayed until a final full-screen quad draw. An in-depth explanation of forward vs deferred rendering is beyond the scope of this paper but details can be found in the reference section.[3] NVIG begins with vertex data from target and terrain models, and associated material containers with parameters and textures for the applicable wavebands to be rendered.[4] During the initial forward draws for meshes, the NVIG stores geometry buffers (G-buffers)[3] of frame data that include eye-space positions and normals, as well as data such as albedo (diffuse color) and parameters for thermal and other wavebands. Once all the meshes in the scene have been rendered, full-screen quad draws are done per light source affecting the scene, creating a light G-Buffer. Then, a final compositing full-screen quad draw combines all the previously filled G-buffers into the final lit image. This rendering process is depicted in Figure 3.

5.3 LIDAR Shader Implementation

LIDAR rendering in NVIG takes advantage of the deferred rendering pipeline. Since the eye-space position is already supplied in the G-buffers, the compositing pass can simply write this position to the output off-screen frame buffer (Figure 3). This camera local position will make up three of the four values that will be written. The remaining channel is filled with the LIDAR return intensity value, which is calculated using Equation 11 above in Section 4.0 describing the Simulation Physics. The equation accounts for laser and detector characteristics, atmospheric attenuation, as well as energy from directional and non-directional solar effects. These LIDAR parameters are passed into NVIG shaders as uniform parameters. The shader uses these parameters to compute a LIDAR intensity value that is normalized before being written into an off-screen frame buffer that will be used as input to NVIG's sensor effects pipeline.

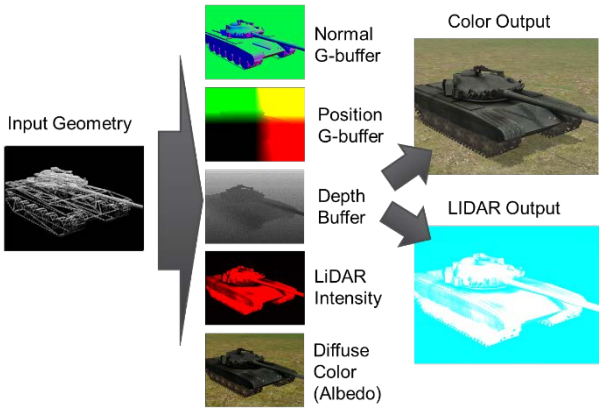

Figure 3: Deferred Rendering for LIDAR Simulation

5.4 LIDAR Noise and Voids

NVIG has a full-screen sensor effects pipeline in which the simulated sensor image quality can be degraded based on models developed at Night Vision Electronic Sensors Directorate (NVESD). NVIG sensors are built with configuration files that specify waveband, pixel resolution, fields of view and other crucial information for simulating imagery from that sensor. Based on this configuration, NVIG applies full-screen shader effects like gain, bias, blur, noise, polarity, and much more. Additions made to the configuration files in support of LIDAR simulation include noise standard deviations for controlling the amount of noise to apply to the output positions and intensities. The x, y, noise will be represented in radians and the z noise will be in meters. After applying positional noise to the coordinates, we multiply the eye-space position by the inverse of the camera local view matrix to obtain a camera local position. This camera local position and LIDAR intensity value are stored in the final output buffers. The camera local position is the position of the rendered point relative to what NVIG calls the camera anchor position. NVIG renders all its geometry relative to this camera anchor position. This relative rendering is needed to alleviate precision issues when rendering large terrain databases in OpenGL. In order for proper coordinates to be written to the output files, the positions read back from the GPU will be offset by this camera anchor position before export.

Voids occur when no return is detected from the laser or when a return that is so small that it is below the noise floor set from solar effects and the system. NVIG simulates voids caused by small returns in the LIDAR shader. If the computed energy from the laser return is less than the energy from the direct and indirect solar contributions along with the noise from the system, then a void is created and no LIDAR intensity or range information is returned. NVIG also simulates voids during the export process using a Bloom filter before writing the points to a file. This is discussed in more detail in following sections describing the LAS file creation.

5.5 High Dynamic Range Precision

NVIG has supported high dynamic range (HDR) rendering by providing off screen frame buffers with 16-bit per channel half precision float. The pixel format was GL_RGBA16F using the half precision float documented in the ARB_half_float_pixel extension specification.[5] This half float representation provided better precision than previous 8-bit per channel pixel formats, but in order for NVIG to store eye-space positions at expected ranges from simulated aerial vehicle mounted LIDAR systems with minimal error, support for 32-bit per channel frame buffer object (FBO) off screen rendering was added. The OpenGL pixel format associated with the new FBO is GL_RGBA32F. With this added feature, the NVIG is able to read full single-precision floating point camera local positions from specific points in the rendering pipeline.

5.6 Read Points and PBOs

NVIG supports frame buffer capture through the use of asynchronous pixel buffer object (PBO) reads from the GPU. This provides fast read-back from the GPU to CPU through DMA (Direct Memory Access).[6] The NVIG LIDAR image capture plugin uses this asynchronous PBO read to grab the pixel data from the GPU to the CPU. NVIG supports multiple read points where frame buffers can be recorded at different stages in the rendering pipeline. Once on the CPU, the positions are translated using the NVIG camera anchor position for the corresponding frame at the time of image capture request. NVIG rendered the scene relative to this camera anchor position. Thus, translation by the camera anchor position puts the positions computed out in terrain relative coordinates using a double precision floating point representation.

The PBO read-back here includes read-back of LIDAR intensity and pixel world locations. NVIG's read point system also allows for frame capture from multiple wavebands during a single render pass. This allows the LIDAR image capture plugin to grab the simulated LIDAR frame as well as pixel data for RGB (red, green, blue) color from a simulated visible color camera. This RGB data is also incorporated into the simulated output file.

6.0 POINT CLOUD AGGREGATION AND FILE GENERATION

6.1 Point Cloud File Output

LASer File Format (LAS)[7] content and the related compressed version, LAZ, was chosen to persist the rendered point cloud data since it's a widely accepted and open standard supporting LIDAR data exchange. However, writing LAS files to disk while rendering point clouds is a challenge given the mismatch in data rates between the two processes. File Input / Output (I/O) is the constraining component of the pipeline, since modern Serial ATA (SATA) drives (v3) have a theoretical maximum of 6MB per second. Our LAS point data conforms to LAS Format Standard Record Format 3 which includes location, intensity, color, GPS time, and other parameters, yielding a total record size of 34 bytes. Applying SATA rates to this data type, the capacity to store points is 176,470 points per second. A public domain article on defence applications of LIDAR[1] suggests rates up to 150,000 points per second, which is 85% of theoretical SATA data rates. While the required data rates are within theoretical

capabilities, bus and controller contention, while hosted on pre-emptive multitasking Operating Systems (OS), results in lower disk throughput. Consequently, LIDAR point persistence may not reach 150,000 points per second, but the real time persistence of representative LIDAR points to disk looks achievable.

Assuming an upper bound of 150,000 points per second, we can calculate the LIDAR sensor resolution that will result in this value. A common frame rate for rendering is 30 frames per second (fps), which would result in 5000 pixels per frame or a nominally square 70 pixel frame dimension. This frame dimension is quite low, and well within standard definition (640x480) specifications. Given that the LIDAR simulation is associated with a moving platform, larger frame dimensions at a lower frame rate deliver sufficient points per second while reducing the likelihood of duplicative points from small changes in viewpoint. The data depicted in this paper was synthesized using a 320x240 sensor dimension at 10 fps or 76,800 raw points per second, which is well within SATA rates. This sizing was mostly out of convenience, as larger files become more cumbersome to move and process. Sensor sizing is a configuration parameter, which will be adjusted when concrete sensor concepts or hardware are considered.

Our LIDAR simulation overlaps file writes to improve system performance. The file persistence logic was serviced by a thread pool that was uncoupled from the rendering engine to reduce impact on frame generation. Rather than continuously writing to one file, the simulation configuration contains a “files per frame” parameter which drives aggregation of multiple frames into a single file. The writing of points to file is a serial operation, but as one set of frames is written to disk, the next set is being aggregated. As a function of thread pool size, it was then possible to have multiple file writes in process while new points were being aggregated. While this overlapped behavior improved performance, the raw point set was already reduced to eliminate duplicates prior to file persistence.

6.2 Point Cloud Filtering

Prior to writing to LAS, raw point sets were processed through Bloom filter[8] to remove duplicate locations. In the configuration described above, the aggregated 10 frames point count was reduced to approximately 65,000 points per file. This filtering efficacy is highly use case dependent where simulated LIDAR on faster moving platforms would contain fewer duplicate points to remove. The filtering reduction benefits point cloud production rates, but at the cost of introducing voids when the filter incorrectly indicates that the set already contains point (false positives). A conventional hash set approach would not have false positives, but its greater memory consumption would sooner constrain total points processed as compared to the Bloom filter. An explanation of Bloom filters is beyond this paper, but an illustrative graph of false positive probabilities (p) as a function of filter size (m) and samples (n) is shown in Figure 4, where the number of hash functions (k) is assumed optimal. The graph depicts that the first point tested has a lower probably than the last point attempted for a given filter size. In our LIDAR simulation, the probability of false detections is an input parameter, and the filter is sized to meet that performance on the last point tested.

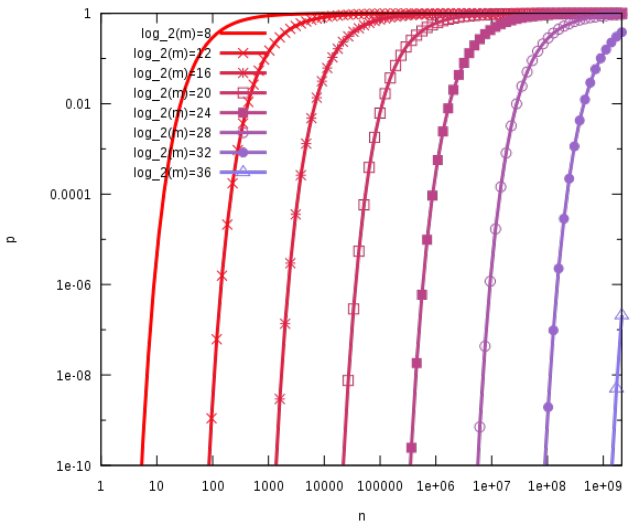

Figure 4: Bloom filter probability false detection[8]

However, the false detection for a given point is dependent on insertion order. Consequently, for a given frame points are drawn at random insertion so that the voids associated with false positives are randomly distributed

within the frame. Although the filter is sized and used for all points in a LAS file, the random insertion is not used between aggregated frames to reduce memory consumption by avoiding holding all associated frames. The simulation user can specify any number of frames to aggregate, and if that number times point count per frame exceeds available memory, the simulation would fail before writing LAS output.

Accordingly, there's a hybrid approach in generating final LAZ files, where many small LAS files are created during run time and subsequently post processed into final form. From the example described above, LAS files encompassing 10 frames, or one second per file, were generated during run time, and the final product merges 50 of these files into one compressed LAZ file. The post run time merge utility has similar logic to the run time file generation. This utility, "merge_las," uses a much larger Bloom Filter, currently sized at 512MB, to reduce the number of voids while reducing the final point count. The same randomized approach is taken while processing input LAS files for merge to prevent read order bias from the increasing false positive rate. Each run time LAS file is approximately 22MB, 650,000 points and the combined and compressed final LAZ product is 342M, 30,345,000 points with a final false positive rate of 0.7%. Figure 1 shows a close up on the final product.

7.0 VERIFICATION

At this point, verification of simulated LIDAR has come through implementation inspection and comparison with known live LIDAR sets. The mathematics described above has been implemented in the rendering logic, and this implementation has been inspected by the authors. Random point samples have also been verified by multiple parties to conform to the algorithm through the course of simulation development. Finally, the resultant total point cloud was compared with a similar live collection for a qualitative verification. Figure 5 is a comparison of similar scenes between live and simulated collections. Both images show peaked rooflines and trees, and appear to have similar density and voids.

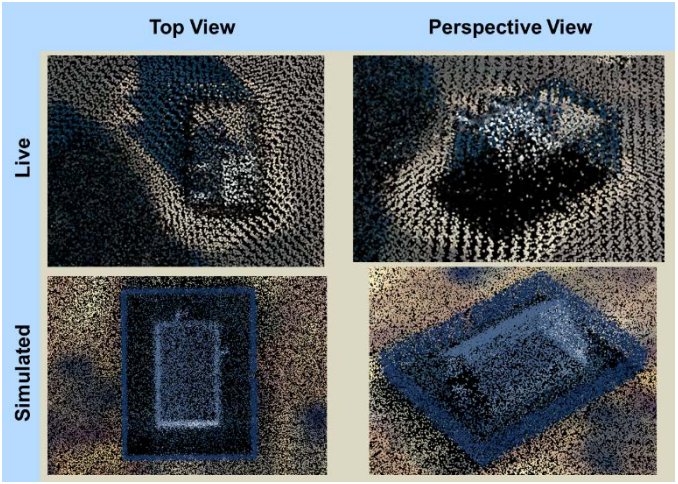

Figure 5: Qualitative comparison between live and simulated LIDAR

Objectively, the simulation will be verified and validated through a series of measurements. A LIDAR system with measured parameters in the lab will be flown over a calibrated scene. The results of the simulation will be compared to the measured 3-D point cloud using an error analysis for position, intensity, and range. The fidelity of the simulation will be determined as well as potential improvements.

8.0 CONCLUSIONS AND FUTURE WORK

The current physics based simulation generates representative point clouds from simulation standard environments, and persists them in accepted standard formats. Adapting the NVIG rendering pipeline to LIDAR physics and applying statistical point filtering at runtime, this LIDAR simulation supports operational training and exercise on live timelines. Prior art preferred ray tracing to rendering, and while more accurate, live activities could not be supported. Also, by using NVIG larger distributed simulation through Distributed Interaction Simulation (DIS) and the High Level Architecture (HLA) is already supported.

This fundamental capability is a good cornerstone for training level activities, however no integration with sensor control applications has been performed. The point clouds generated for this paper leveraged standard simulation sensor association with simulated entities, but parameter, relative orientation, and other potential sensor controls were static throughout the collection. The goal would be to implement STANAG 4586[9] or similar, and provide an interface between the Core Unmanned Aircraft (UA) Control System (CUCS) and an appropriate LIDAR Vehicle Specific Module (VSM). The likely development path will be to integrate with fielded LIDAR controllers in response to customer trainer requirements.

Beyond training, greater fidelity may be required for more intensive experimental and analytical activities. However, the simulation presented in this paper has some limitations compared to a fielded system. First, the simulation currently assumes a uniform atmosphere and does not take altitude dependent atmospheric transmission, atmospheric scintillation, or atmospheric turbulence into account. The atmospheric backscatter from the laser is also not addressed. Also, the laser beam is currently assumed to be uniform with no laser speckle and the geometrical probability factor, or overlap factor, is assumed to be unity. Finally, the objects in the scene are assumed to be Lambertian with no specular reflections. Many of these shortfalls will be addressed, as the LIDAR simulation is applied to experiments or analyses that require greater fidelity.

The developers of the LIDAR simulation are familiar with the encoding of live video, and the LIDAR point cloud processing has many similarities. Given that both video and point clouds have been implemented using the rendering pipeline, leverage the same capabilities for capturing frames, it follows that the point content could be encoded and streamed. Streaming would allow point transformations and file operations to be off loaded from the rendering engine, and given that rendering is significantly faster than file I/O, a means of concurrent and load balancing LAS production seems possible. Any selected codec would have to be lossless, and provide the numerical resolution required to express the total point cloud in a georeferenced basis. Since the initial LIDAR simulation requires 32 bit position components, video codecs did not appear to be viable. Commonly, video codecs required a color space transformation with loss, fixed precision values of up to 10 bits, and subsequent compression. If a technique develops where point location and intensity data can fit within these capabilities, or if video codec implementations grow to greater capacity, this approach may warrant further investigation.

[1] “Warfighters reap benefits of LIDAR mapping technology,” Defense Systems, 26 JUL 2011, https://defensesystems.com/articles/2011/07/18/tech-watch-geoint-lidar.aspx

[2] Fox, C. S., [The Infrared and Electro-Optical Systems Handbook Volume 6], SPIE Optical Engineering Press, Bellingham, Chapter 1 (1993).

[3] Michal Valient, Senior Programmer, Guerrilla Games. Deferred Rendering in Killzone 2 Develop Conference, July 2007.

[4] May, Christopher; Harkrider, Susan; Paul, Douglas; Osborn, Walter; Tran, Vinh, "Target IR Signature Creation Used in NVIG Sensor Simulation" Proc. SimTecT 31 May - 2 June (2011).

[5] Kirkland, Dale. ARB_half_float_pixel OpenGL ARB Extension #40 October 22, 2004, https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_half_float_pixel.txt

[6] Ahn, Song Ho. OpenGL Pixel Buffer Object (PBO), 2014, http://www.songho.ca/opengl/gl_pbo.html

[7] LAS Specification, Version 1.4, Revision 13. 15 July 2013, http://www.asprs.org/a/society/committees/standards/LAS_1_4_r13.pdf

[8] Bloom Filter, Wikipedia, https://en.wikipedia.org/wiki/Bloom_filter

[9] STANAG 4586 Ed.3 Nov 2012, Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability, NATO Standardization Agency (NSA), 2012.