

## Procedural Reconstruction of Simulation Terrain Using Drones

**Ryan McAlinden, Evan Suma, Timofey Grechkin**

**USC Institute for Creative Technologies**

**Los Angeles California**

**{mcalinden, suma, grechkin}@ict.usc.edu**

**Michael Enloe**

**National Simulation Center**

**Ft. Leavenworth, Kansas**

**Michael.r.enloe.civ@mail.mil**

### ABSTRACT

Photogrammetric techniques for constructing 3D virtual environments have previously been plagued by expensive equipment, imprecise and visually unappealing results. However, with the introduction of low-cost, off-the-shelf (OTS) unmanned aerial systems (UAS), lighter and capable cameras, and more efficient software techniques for reconstruction, the modeling and simulation (M&S) community now has available to it new types of virtual assets that are suited for modern-day games and simulations. This paper presents an approach for fully autonomously collecting, processing, storing and rendering highly-detailed geo-specific terrain data using these OTS techniques and methods. We detail the types of equipment used, the flight parameters, the processing and reconstruction pipeline, and finally the results of using the dataset in a game/simulation engine. A key objective of the research is procedurally segmenting the terrain into usable features that the engine can interpret – i.e. distinguishing between roads, buildings, vegetation, etc. This allows the simulation core to assign attributes related to physics, lighting, collision cylinders and navigation meshes that not only support basic rendering of the model but introduce interaction with it. The results of this research are framed in the context of a new paradigm for geospatial collection, analysis and simulation. Specifically, the next generation of M&S systems will need to integrate environmental representations that have higher detail and richer metadata while ensuring a balance between performance and usability.

### ABOUT THE AUTHORS

**Ryan McAlinden** Ryan McAlinden is the Associate Director for Digital Training and Instruction at the University of Southern California's Institute for Creative Technologies (USC-ICT). He rejoined ICT in 2013 after a three year post as a senior scientist at the NATO Communications & Information Agency (NCIA) in The Hague, Netherlands. There he led the provision of operational analysis support to the International Security Assistance Force (ISAF) Headquarters in Kabul, Afghanistan. Prior to joining NCIA, Ryan worked as a computer scientist at USC-ICT from 2002 through 2009. He has a B.S. from Rutgers University and M.S. in computer science from USC.

**Evan A. Suma** is a Research Assistant Professor at ICT and the Computer Science Department at the USC. As one of the co-directors of the MxR Lab, his work focuses on techniques and technologies that enhance immersive virtual experiences and provide new opportunities to study human perception. Dr. Suma has written or co-authored over 50 published papers in the areas of virtual environments, 3D user interfaces, and human-computer interaction. He received his Ph.D. in computer science from the University of North Carolina at Charlotte in 2010.

**Timofey Grechkin** is a Postdoctoral Scholar – Research Associate at USC-ICT. He is a researcher in human-computer interaction and virtual reality. He is interested in design and evaluation of immersive 3D experiences and naturalistic full-body interactions in virtual worlds and enabling naturalistic locomotion in virtual reality. Dr. Grechkin earned his Ph.D. in Computer Science from the University of Iowa. He previously graduated from Moscow Engineering and Physics Institute (State University) in Moscow, Russia with a Specialist degree in Applied Mathematics and Informatics.

**Michael Enloe** is the Chief Engineer for the United States Army National Simulation Center (NSC) at Fort Leavenworth, KS. He has supported the NSC Futures Division since 2001. Mike serves as the technical lead in support of the Army's Future Holistic Training Environment-Live/Synthetic (FHTE-LS) conceptual vision which will help produce an evolved simulation training capability able to meet the complex training needs of our 21st century soldiers and leaders. Mike has a B.S. in Information Technology from the Phoenix University.

## Procedural Reconstruction of Simulation Terrain Using Drones

Ryan McAlinden, Evan Suma, Timofey Grechkin

USC Institute for Creative Technologies

Los Angeles California

{mcalinden, suma, grechkin}@ict.usc.edu

Michael Enloe

National Simulation Center

Ft. Leavenworth, Kansas

Michael.r.enloe.civ@mail.mil

### INTRODUCTION

This paper discusses how data collected from low-cost, commercial drone technology can be used to create usable, interactive terrain for use in simulation and virtual environments. The capability provides small-units (fire teams, squads) with an organic asset that allows them to autonomously fly an area of interest (AOI), capture imagery, and then use that imagery to produce a 3D reconstruction. This reconstruction (or the imagery from which it is derived) may be used for simulation, mission rehearsal, or enhanced situational awareness. The resulting datasets have very high level of fidelity (up to 1cm precision), and the benefits of providing the units with this embedded capability mean they do not have to solely rely on large collectors or other services to supply them with information about a particular area.

### Aircraft and Camera Selection

Unmanned aerial systems (UAS) today range significantly in size, cost, flight duration, lift capacity, and sensor capabilities. Platforms range on the low-end targeting novices, to mid-level hobbyists, to full-grade commercial equipment. Prices can range from \$25 toys to \$10Ks for professional photographic or cinematic equipment. Our research centers around using mid-grade platforms to take images of an AOI and then photogrammetrically reconstructing the environment for a game or simulation. The basis of photogrammetric reconstruction for simulations (first or third-person) is rooted in images that are consistent, of decent quality, and overlapping. With that as a requirement, we decided to test two primary platforms as part of this effort - 1) a DJI Phantom 2 with GoPro 3+ (Black); and 2) DJI S900 with Panasonic GH3 camera. Both offer decent flight times, automated flight controls (i.e. autopilot), quality gimbals, moderate lift capability and reasonable cost. Below is a table summarizing the comparison between the two:

**Table 1. Multi-Rotor Comparison**

Aircraft	Cost	Flight Controller	Flight time	Gimbal	Camera
DJI Phantom 2	\$2000	Naza-M V2	15 – 20 min	H3-3D	GoPro 3+
DJI S900	\$4000	A2	12 – 17 min	Zenmuse Z15	Pana GH3

Other suitable candidates would have included craft like a 3DRobotics Iris+ or X8. There was no underlying reason to use DJI-only equipment, though overall it tends to be more plug-and-play than its competitors, and peripheral equipment (GPS, on-screen display (OSD), ground station) could be shared between the aircraft. A multi-rotor (i.e. quad / hex-motor) system was selected over a fixed-wing for several reasons. Though fixed-wing would have given us extended range and battery time, the stability of fixed-wing gimbals is not as precise as rotary-wing which results in distortion of the images, nor is the lift capacity as high, which is required for the better imaging systems. Finally, when reconstructing certain specific features (e.g. buildings), and orbit pattern is desired which can be achieved more consistently with multi-rotors than fixed-wing. Note that FAA regulations currently limit hobbyist flying of any RC aircraft to 400 ft and below, away from airports and populated areas, and always remaining within line-of-sight of the operator. Commercial UASs (including research equipment) require either a Section 333 exemption or Certificate of Airworthiness (COA), which can range from several hundred to several thousands of dollars and can take months or years to acquire. This should be taken into consideration before purchasing or flying any type of aircraft for non-hobbyist purposes. For this particular effort, which was considered non-hobbyist (i.e. commercial), permission was sought from base installations, specifically Ft. Irwin. Military and other controlled installations are exempt from FAA flight regulations (including air worthiness certifications), but base permission and approval is still required.

Camera selection for aerial photogrammetry is highly dependent on the lift capacity of the aircraft. Full-range digital single-lens reflex camera (DSLRs) are preferred but nearly impossible to lift for any meaningful period of time. For example, the DJI S1000 can lift a DSLR with 18 – 105mm lens but flight time is restricted to 6 – 8 min. Even with a stock 18 – 55mm lens the flight time is not much more than about 10 minutes. Because we were interested in mapping as much land area in as few flights as possible, the weight-time balance was of particular interest.

Another consideration with the aerial platforms are batteries. Lithium-polymer (LiPo) batteries continue to incrementally improve, though no large-scale enhancements have been made for a few years. That means that even the best LiPo batteries will only give you minor improvements in either flight time or lift. The cost is of significant concern here as batteries can be levels of magnitude more expensive (and heavier) for only marginally improved output and/or capacity. The flights times for each of these batteries will vary significantly depending upon the type of motor, payload, and weather conditions. Examples are in Table 2 below.

**Table 2. LiPo Battery Comparison**

Capacity	Cell Count	Weight	Cost
2200 mAh	2S	250g	\$10.00
5000 mAh	4S	600g	\$25.00
5800 mAh	8S	1375g	\$90.00
5800 mAh	10S	1300g	\$100.00

By far the most important decision when determining the aerial platform is the type of camera to be supported. They can roughly be broken into three categories:

- 1) Compact action cameras – e.g. GoPro
- 2) Point-and-shoot – e.g. Nikon Coolpix
- 3) Mirrorless Point-and-shoot: e.g. Panasonic GH3/4
- 4) DSLR – e.g. Canon Rebel T5

The only one of the above immediately ruled out given the limitations of the flight hardware was DSLRs. Though they may be mounted and flown using commercially available equipment, they often suffer from poor flight times and lack of control. This will likely change as batteries and airframes improve, but unless shooting high-definition, live-action shots, there is no need to use such high-end equipment, certainly not for photogrammetry. The compact action cameras are the cheapest and offer the best flight times. However, they suffer from poorer image quality, particularly the GoPros which have a tremendously wide angle field of view (FOV), software shutter, and plastic lens. Performing software corrections for these three variables even with the best commercial photogrammetric software can be challenging (Balletti et. al, 2014). The point-and-shoot (both compact and mirrorless) offer the best balance between size and image quality. Assuming a suitable gimbal can be found, they offer the ability of high-quality images that can be software corrected and offer the truest form of imagery for cost/weight.

### **Flight Planning & Image Capture**

After the airframe, camera, and peripherals (ground station, OSD) are selected, flight planning may begin. For this effort, flights were conducted both manually and autonomously using the stock DJI RC Transmitter (manual) and the DJI Ground Station (autopilot). This ground station allows users to input waypoints and set altitude, heading and speed so the aircraft can fly autonomously from waypoint to waypoint. Images were captured every 2- 5 seconds depending upon the area flown, altitude and forward speed. Flight planning must account for sufficient overlap such that the photogrammetric software can create sufficient tie points to produce a point cloud. This is detailed further below.

The first task in the image capturing process is to determine what an acceptable level of detail and pixel resolution is for reconstruction. In this case we will state that we would like each pixel to represent at least a centimeter on the ground plane. Ideally this number should be derived from the resolution of the output display, angle of view of the viewer and the height of the viewer. However we face some practical limitations as well such as how close we can fly our craft and the camera package it is carrying. In the example below, we are working with the GoPro3+ Black that can capture a 4000x3000 pixel image while displaying a, ~122.6° horizontal angle of view (Downing, 2015).

By flying the craft in stripes (Figure 1) 15 meters off the ground we can achieve a .83 cm resolution. The camera is flown nadir – i.e. 90 degrees pointing straight down. In 18:49 minutes we can take 351 photos, 13 meters apart, shooting a photo every 2.5 seconds at 5 meters per second. This timing is convenient as it is very close to the 2 second interval timing of the GoPro. This should fit within the 25 minute flight time of the Phantom II allowing that the flight time will be reduced by the weight of the gimbal and GoPro. This would cover an area of ~225 meters by ~225 meters or 56,000 square meters.

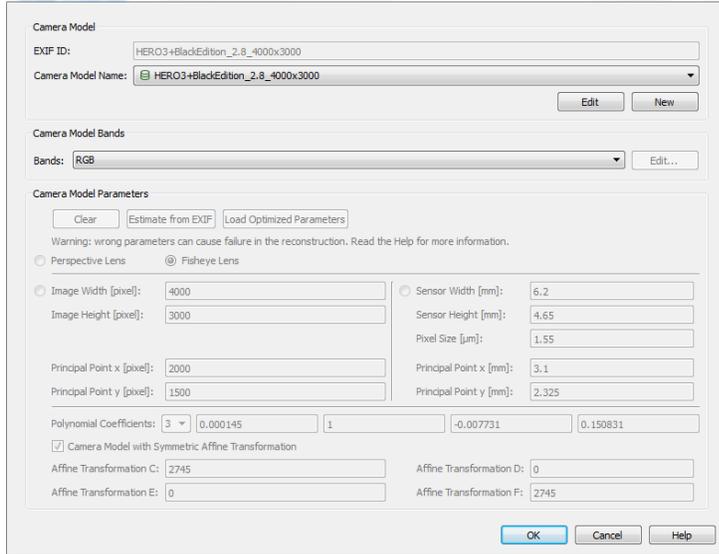


**Figure 1. Striping Flight Pattern**

#### High Flight

It is also recommended to fly the same area at a high altitude that will prevent accumulated error and ground curvature that can occur with a low flight. For this flight we recommend going to the full legal altitude, 121 meters. This can be flown very quickly and will only require a very small number of images. If your lower-altitude terrain flight requires multiple flights to cover a given area, the high altitude flight should cover this entire area and can be used to calibrate separate groups to one another without the aid of survey points. For a similarly sized area as above, it only requires 4 photos and 2.30 minutes to capture.

## PHOTOGRAMMETRIC RECONSTRUCTION



**Figure 2. Pix4D Mapper**

After the images were captured, they are then fed through commercial photogrammetric software to produce the initial point cloud. For this effort, we compared three different solutions – Pix4DMapper, Smart3DCapture and Photoscan. Each produce variations of point clouds, and depending upon the type of area reconstructed (urban vs rural, mountainous vs flat), will produce qualities of varying degrees. We did not find that any of the three packages were markedly better overall. Fundamentally, they are all rooted in taking each image and creating a tie point – these are corresponding points that are found in various images, and produce the foundation of the model. It allows the third dimension (depth) to be extracted by comparing the distance between points in various images. After creating the tie points, a dense three dimensional point cloud is produced. A triangle mesh was also generated to run basic algorithms to determine what polygons best matched the point cloud and to smooth out the model for use in different systems.

In addition to generating the point cloud, we must also insert ground control points (GCP's) to geo-locate the final model. At a minimum, 3 GCPs are required to geo-locate the entire surface area. In order to do this, one must take a point in an image and provide a latitude/longitude/elevation based on some existing survey or GIS data (such as Google Earth). After doing this for at least three (preferably more) points, the resulting point cloud will have coordinates attached at each point. Because the level of resolution of the model is higher than even the best elevation data (DTED-5, NASA ASTER), these geo-tags allow it to be draped over existing elevation models for inset areas requiring more detail.

### Tilt Test

In our some of the results from our initial flights we found some undesirable artifacts in the man-made structures of the scene. The walls of the reconstructed buildings would bulge out rather than reflect the flat construction. For example, the buildings in Figure 3 have clear right angles, but when reconstructed they appear with rounded walls seen below:



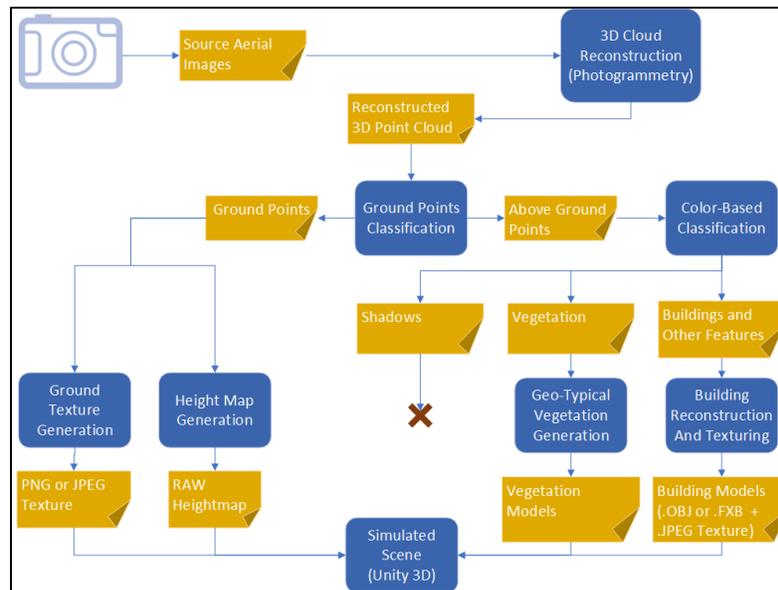
**Figure 3. Mesh Reconstruction of Buildings**

To correct for this, we propose tilting the camera to 45 degrees (+/- 10 degrees) to see if having more images of walls and having the walls in the center of the image provides us with better results. This would minimize linear distortion, particularly in intersections of the y and z axes.

## POINT CLOUD PROCESSING AND 3D SCENE GENERATION

Photogrammetry packages typically produce output that is not well-suited for direct import into a real-time simulation engine. The automatically generated point clouds and meshes are extremely dense, with a very large number of vertices and/or polygons, and often contain noise from errors during photogrammetric reconstruction. Furthermore, the point clouds are unstructured, and the features of the scene (e.g. terrain, buildings, vegetation, etc.) are not identified. In modern simulation engines, terrain is typically represented in a manner that is more efficient to load and display, such as height maps, where other geometry is rendered as 3D meshes that have been optimized with polygon count and texture sizes appropriate for real-time performance. Therefore, it is necessary to apply a post-reconstruction processing pipeline to prepare the dense and noisy data produced from photogrammetry for use in a game or simulation engine (in our case – Unity 3D).

Figure 4 illustrates the key stages of the processing pipeline. First, the unstructured point cloud is segmented into two separate clouds, representing ground and above-ground points. These segmented point clouds are then processed using two distinctly different pipelines. Ground points are transformed into height maps with corresponding textures, which can be directly imported into Unity's existing terrain system. Above-ground points are further analyzed and processed to generate either geo-specific or geo-typical 3D meshes suitable for real-time rendering. Each stage is discussed in further detail in the following sections.



**Figure 4. An Overview Of The 3D Scene Generation Process**

### Ground Point Classification

The first step in processing the unstructured point clouds generated from photogrammetry is to separate the ground points from above-ground points belonging to buildings, vegetation, and other features. To achieve this segmentation, we make use of a progressive morphological filter (Zhang et al., 2003), which is readily available in both the open-source Point Cloud Library (PCL) and commercial software packages such as Agisoft Photoscan. Although this algorithm was originally developed for airborne LIDAR data, we found it to be quite effective for photogrammetric point clouds as well. While this computational approach is automatic, we found that the quality of the output depends upon the selected algorithm parameters. Therefore, it was often necessary to run the algorithm multiple times, starting with more conservative parameters that would only identify larger structures, and then iteratively increasing sensitivity until the right balance was achieved. Figure 5 shows an example of a segmented point cloud after ground point classification.



**Figure 5. A Segmented Point Cloud After Processing With The Ground Classification Algorithm.**

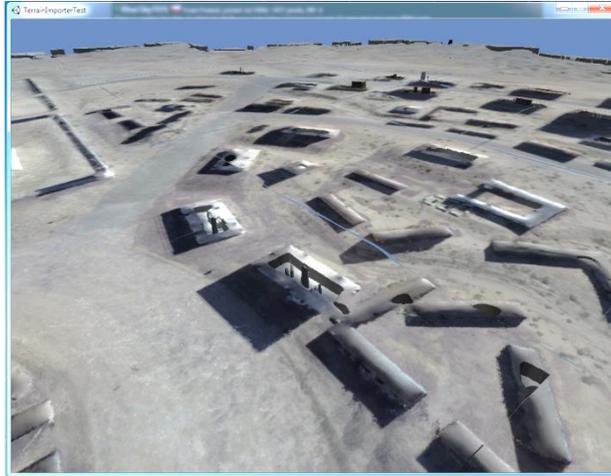
### **Ground Height Map and Texture Generation**

Game and simulation engines typically represent terrain in a more computationally efficient format than 3D meshes, since these surfaces need to be rendered over large areas. The Unity Game Engine generates terrain models from height maps, a compact image-based format where each pixel represents the elevation at a given coordinate. The engine can then leverage these terrain assets for real-time simulation, supporting functionality such as level-of-detail manipulation and automatic collision detection.

Processing the segmented ground point cloud begins with voxelization, which imposes a regular 3D grid onto the point cloud. The X and Y resolution of the voxel grid corresponds to the desired output resolution of the height map image. For each two-dimensional column, the highest occupied voxel (3D cell) is computed, and this height value is recorded in the corresponding pixel of the height map. Any holes left after removing the above ground structures and vegetation are filled in at the ground height level of the bordering pixels. Additionally, ground segmentation algorithms tend to generate some noise on edges of the removed above ground features, which produce “spikes” in the height maps. These artifacts are essentially salt-and-pepper type noise in the images, which we removed by applying a median filter. These height maps are stored in a binary RAW data format that can be directly imported into Unity.

Since the height maps only store the elevation of the terrain surface, it is also necessary to generate texture images that include the color information for each pixel in the height map. When constructing the voxel grid, the average color of all the points that fall within the boundaries of each 3D cell is computed. When a voxel is selected for the height map, the color for that cell is then copied to the corresponding pixel in the texture image. These images are saved in PNG format. The quality of the resulting terrain depends on the selected resolution of the height map and texture images. In our experience, visually appealing textures were achieved using approximately 50m x 50m terrain regions with height map and texture resolutions of 512 x 512 pixels. This corresponds to a voxel size of approximately 10cm.

Due to limitations in memory and computation, expanding the resolution of the texture map for areas much larger than 50m x 50m results in a significant framerate decrease. Therefore, to manage this process more effectively, larger areas are represented using tiles at the previously mentioned resolution. Our terrain processing algorithm generates an XML metadata file that identifies the size, global location, and the image assets for each tile. We developed a custom import script for Unity that processes the XML tilesets and generates the terrain for the entire region automatically. Figure 6 shows an example of terrain rendered in Unity after importing the height maps and corresponding textures.



**Figure 6. Terrain Rendered In Unity Engine. Artifacts From The Removed Above-Ground Features Will Be Covered By Geometry.**

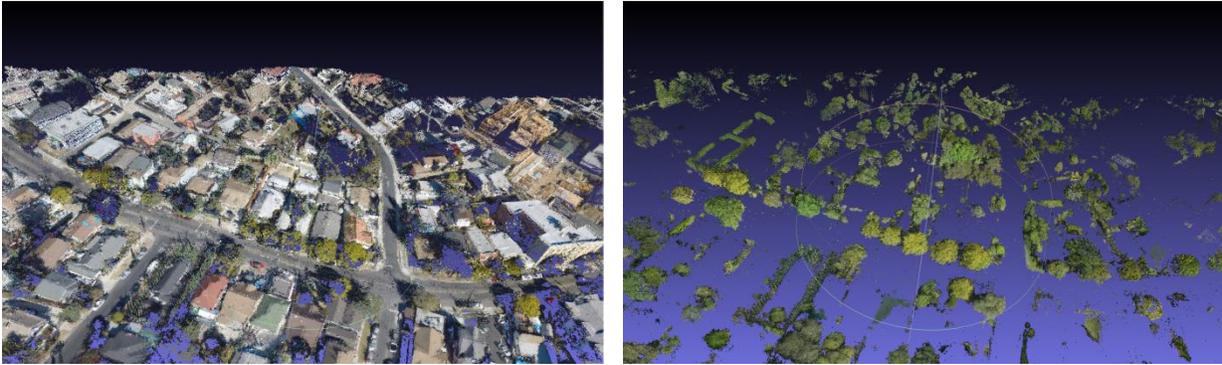
### **Above-Ground Feature Classification**

In order to generate the most appropriate 3D models, it is necessary to further classify above-ground features. Man-made structures, such as buildings, light poles, and other objects, have very different geometric qualities from organic objects, particularly vegetation. Because our pipeline needs to process these different object classes differently, we developed a method to extract vegetation from the above-ground point clouds.

When accurate depth information is available such as the case with dense aerial LiDAR data, vegetation can be effectively identified using geometric data (Zhou, Q. Y., & Neumann, 2013). In our case, however, the presence of irregular geometric forms and partially occluded or shady areas characteristic of vegetation results in a large amount of noise in photogrammetrically reconstructed point clouds. This limits the applicability of depth-based geometric approaches common for LiDAR data.

Alternatively, color information can be used to identify vegetation-covered areas. This approach is often used when working with a single aerial image. Typically, intensity value for each point is computed using a color-based invariant corresponding to vegetation or building roof, followed by automatic thresholding methods to classify points (Shorter and Kaspis, 2009; Sirmacek and Imsalan, 2008). However, in our experience applying these methods to photogrammetric point clouds, the proposed color invariants did not generate sufficiently high contrast between vegetation and non-vegetation areas to identify vegetation.

However, we found that vegetation can be identified using color-based classification based on Euclidean distance from red, green, and blue colors in RGB space. We begin a simple assumption that points with the colors to RGB green (0, 255, 0) are likely vegetation candidates. This method required a couple of further refinements to avoid misclassifying shadows, highlights, and aquamarine colored objects. First, we excluded from vegetation candidates any point with blue component value above 126, which includes many highlights, concrete, and water colors. Second, excessively dark point (all blue and red color channels values below 50 and green channel value below 25) were classified as shadows. Figure 7 shows an example of vegetation that was extracted from a complex and noisy point cloud using color-based classification. In general, the resulting filter was quite successful in identifying vegetation in a computationally efficient manner when applied to our test datasets.

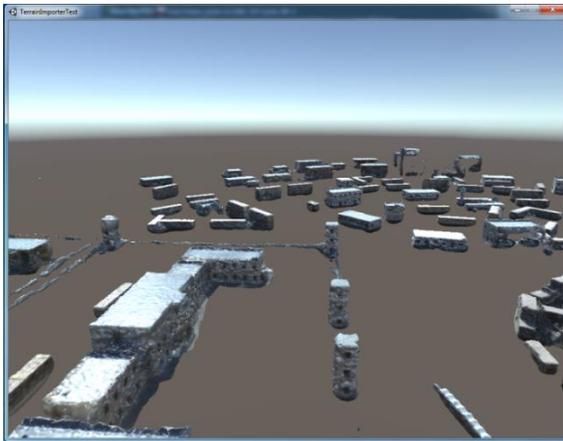


**Figure 7. A Point Cloud With Vegetation Extracted Using Color-Based Classification**

There are a number of ways in which this approach can be extended in the future. One limitation of is that structures with color properties similar to vegetation (e.g. a green roof in a residential area) may cause these points to be identified as vegetation. We believe that these false positives may be filtered out with an additional classification step that rejects vegetation candidates with planar geometry. Additionally, some types of vegetation may have permanent or seasonal yellow, orange, or red shades. Similarly, it may be possible to increase the robustness of the vegetation filter by considering off-color objects if they exhibit structural properties similar to other identified vegetation. In general, we suggest that a filter that considers both color and geometric properties in combination will likely yield the best results.

### **Building Reconstruction and Texturing**

To generate 3D models of buildings, processing the segmented point cloud using an automated meshing algorithm would seem a reasonable approach. The result of applying this procedure is typically a textured mesh representing all buildings in the scene in a standard format that can easily be imported into a game or simulation engine. Many commercial photogrammetry software packages include automated meshing functionality, although there are many other free implementations such as MeshLab or the open-source Point Cloud Library.



**Figure 8. Textured 3D Meshes Constructed Using The Automated Meshing Functionality In Photogrammetry Software.**

In our experience, 3D meshes generated from photogrammetric point clouds produce a reasonably good model when viewed at a distance (see Figure 8). While the number of polygons in the raw output is often much too large for real-time applications, the polygon count can be sufficiently reduced using automated mesh decimation algorithms with minimal noticeable impact to visual quality. However, while the generated models appear decent from a distant viewpoint, the visual quality degrades when viewed from a first-person perspective on the ground. This is due to the noisy nature of point clouds produced from photogrammetry; small errors during the reconstruction may produce visual artifacts that may not be visible from far away, but become more obvious at close range. Furthermore, the 3D meshes produced directly by photogrammetry software are unstructured: all above-ground features are combined into a single mesh. Despite these drawbacks, however, we believe this is a reasonable first step in generating building models for simulation.

We are currently building a more sophisticated processing pipeline that can further segment the point cloud into individual buildings and other man-made features (roads, light and electric poles, fences, etc.) and group these features into appropriate classes. There is currently intensive ongoing research focusing on reconstructing 3D models of buildings using data from various sources; for an overview see (Musialski et al., 2013) and (Haala and Kada, 2010). This process requires analyzing the geometry of each building to identify the footprint, the height of each of its sections, and the structure of the roof. Once the model of the roof is created, vertical walls can then be generated along the edges of the roof, according to the building footprint. We suggest that in many cases, this will produce better visual

representation of the structure, since it leverages architectural knowledge (e.g. most building walls are vertical) and approximates noisy surfaces from the photogrammetric point cloud with cleaner geometry. Textures would then need to be generated by reprojecting the color pixels from the source images onto the newly generated polygons.

### **Vegetation Models**

Generating 3D models of vegetation from photogrammetric point clouds is a challenging problem that we are currently exploring. Due to the extremely complex geometric properties of vegetation, the meshes generated from photogrammetry packages tend to be extremely poor – trees often appear as solid green “blobs.” In general, generation of geo-specific vegetation models, such as leaf-based modelling of trees by fitting a surface segment to every leaf (Zhou and Neumann, 2013), rely on low-noise high-density depth measurements. Such approaches are not possible when working with noisy data generated from aerial photogrammetry. Additionally, furthermore, even if it were possible generate an accurate detailed model, this would be computationally taxing for real-time simulation scenarios.

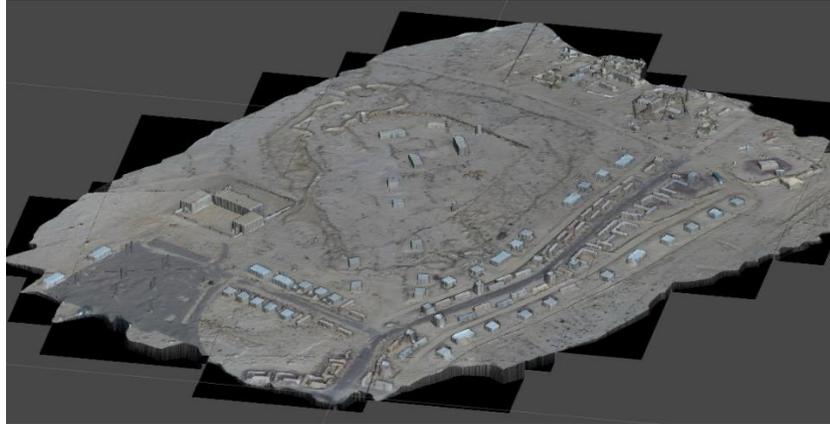
Fortunately, in most cases, it is not necessary to generate geo-specific models for trees and other common types of vegetation. Therefore, we suggest an alternative approach where a similar 3D model is substituted using a library of geo-typical vegetation assets. For this problem, the key challenges are twofold. First, the properties of the original object in the point cloud must be analyzed to determine key information (e.g. vegetation type, color, size, shape, etc.) Second, a sufficiently large library of assets must be accumulated with relevant metadata that can be used to identify the most suitable candidate to substitute.

Ultimately, we believe that geo-typical approaches will be most appropriate for representing vegetation in simulations generated from aerial photogrammetry. Unlike man-made structures, which often exhibit planar geometry, these highly complex organic objects are difficult or impossible to reconstruct accurately from noisy images. However, plants can be reasonably approximated using existing 3D assets that have been optimized for real-time rendering. Thus, research in analyzing and classifying different types of vegetation in point clouds is an important area for future work.

## **RESULTS AND DISCUSSION**

We have collected and processed two datasets through our reconstruction and processing pipeline, representing two distinctly different types of outdoor scenes: (1) a region of the Fort Irwin National Training Center (NTC), consisting of 1268 aerial images, and (2) a portion of the Echo Park neighborhood of Los Angeles, consisting of 505 images. Both were collected using a GoPro Hero3+ Black Edition, at a resolution of 4000 x 3000 pixels.

Figure 9 shows results of a 3D scene constructed from the NTC dataset, rendered in the Unity game engine. From a technical perspective, we found this dataset to be the more straightforward of the two, as it consisted of mostly flat desert terrain with rectangular buildings and sparse vegetation. In general, the ground segmentation and terrain generation pipelines produced reasonable quality results, even when viewed at a first-person perspective. However, the 3D meshes for above-ground structures are currently best viewed at a distance; there are noticeable visual artifacts when viewed at close range. Some of these issues may be mitigated by different image collection strategies. The NTC dataset was among the first we collected, and predominantly consisted of overhead images collected in a “zigzag” flight pattern. We believe that first-person 3D scenes generated from aerial photogrammetry will benefit from the inclusion of both overhead and oblique images, collected in a grid pattern with tighter overlap.



**Figure 9. A Complete 3D Scene Rendered In Unity Game Engine At Real-Time Framerates**

In contrast to NTC, the Echo Park dataset had many challenging features, including steep sloping hills, irregularly shaped buildings, tight clusters with shadows and occlusion, dense vegetation, and moving objects. This dataset presents an excellent test case for developing and testing new approaches, as the current state-of-the-art is not yet able to handle all of these features. However, despite these challenges, we found that the color-based classification method performed quite well at identifying most of the vegetation in the point cloud. However, more research and development is necessary before this dataset would be “game ready” without substantial manual intervention when automated algorithms fail to produce reliable results.

## CONCLUSION AND FUTURE WORK

Generating 3D scenes from aerial photogrammetry appropriate for real-time, first-person simulation poses unique challenges that have yet to be solved. While it is possible to leverage some of the approaches developed for LIDAR data, point clouds produced from photogrammetry are much more dense and noisy, and new methods are required to automatically process and reconstruct 3D scenes from drone aerial images. In particular, we have identified the following challenges for future work:

- 1) Best practices for drone-based image acquisition are not fully understood. First-person simulation requires 3D models with sufficient geometric and texture quality at ground level, yet photogrammetry methods require sufficient overlap between adjacent images. The ideal balance between distant/overhead images (to provide overlap) and close-range oblique images (to provide high quality textures) has not yet been determined.
- 2) 3D scenes generated from photogrammetry tend to be noisy and error-prone. When applied to these point clouds, automated mesh generation algorithms produce warped geometry and substantial visual artifacts. New 3D reconstruction approaches that can generate reliable 3D models from noisy point cloud data, possibly by leveraging architectural knowledge, would be extremely valuable.
- 3) Because models generated from photogrammetry tend to be dense and extremely complex, they require substantial optimization before it is possible to render at real-time framerates. However, different classes of objects (e.g. terrain, buildings, irregular structures, vegetation, etc) need to be processed differently in order to preserve visual quality. Therefore, new approaches for automatic object detection, segmentation, classification are needed, specifically targeted towards point clouds that are extremely vast, dense, and noisy.
- 4) While we generally prefer geo-specific for man-made structures, vegetation is a particular case where geo-typical methods have advantages for real-time simulation. However, more work is necessary to be able to reliably identify and analyze different types of foliage in point clouds. Additionally, this points to need for an extensive library of 3D vegetation assets with sufficient metadata to match the plant type and other visual/geometric properties.

- 5) Automated texturing approaches included in photogrammetric software packages appear to produce reasonable quality results for terrain, particularly for third-person perspectives. However, when viewed from a first-person angle, errors and visual artifacts appear very obvious, particularly for above-ground objects. We have observed that subtle texturing errors are much more noticeable than geometric errors; the structure of an object need not always be perfect, so long as the texture is photographically consistent. We therefore suggest that improved methods for generating composite textures from large numbers of source images (with roughly estimated camera poses) would greatly improve the perceived visual quality of 3D scenes produced from aerial photogrammetry.

## REFERENCES

- Balletti, C.; Guerra, F.; Tsioukas, V.; Vernier, P. (2014). Calibration of Action Cameras for Photogrammetric Purposes. *Sensors 2014*, 14, 17471-17490.
- Downing, G. (2015). Flight Characteristics. *ICT Point Paper*, January 2015.
- Haala, N., & Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6), 570-580.
- Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Gool, L. & Purgathofer, W. (2013). A survey of urban reconstruction. *Computer Graphics Forum*, 32(6) 6, 146-177.
- Rusu, B. R., Cousins, S. (2011, May 9-13). 3D is here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.
- Shorter, N., & Kasparis, T. (2009). Automatic vegetation identification and building detection from a single nadir aerial image. *Remote Sensing*, 1(4), 731-757.
- Sirmacek, B., & Unsalan, C. (2008, October). Building detection from aerial images using invariant color features and shadow information. *In Proceedings of 23rd IEEE International Symposium on Computer and Information Sciences (ISCIS'08)*, pp. 1-5.
- Zhang, K., Chen, S. C., Whitman, D., Shyu, M. L., Yan, J., & Zhang, C. (2003). A progressive morphological filter for removing nonground measurements from airborne LIDAR data. *Geoscience and Remote Sensing, IEEE Transactions on*, 41(4), 872-882.
- Zhou, Q. Y., & Neumann, U. (2013). Complete residential urban area reconstruction from dense aerial LiDAR point clouds. *Graphical Models*, 75(3), 118-125.