1. **Abstract**

Structure from Motion (SfM) and MultiView Stereo (MVS) algorithms are increasingly being used to generate pointcloud data for various surveying applications, however the accuracy and sources of error in the resultant pointcloud across various use cases are difficult to realize without thorough experimentation. The acquisition of imagery and rigorous ground control data at field sites required for this experimentation is a time consuming and sometimes expensive endeavor. These experiments are also almost always unable to be perfectly replicated due to the numerous uncontrollable independent variables, such as solar radiation and angle, cloud cover, wind, objects in the scene moving, exterior orientation of cameras, and camera dark noise to name a few. The large number of independent variables creates a scenario where robust, repeatable experiments are cost prohibitive and the results are frequently site specific. Here, we present a workflow to render computer generated imagery using a virtual environment which can mimic all the independent variables that would be experienced in a real-world data acquisition scenario. The resultant modular workflow utilizes the open source software Blender for the generation of photogrammetrically accurate imagery suitable for SfM processing, with tight control on camera interior orientation, exterior orientation, texture of objects in the scene, placement of objects in the scene, and Ground Control Point (GCP) accuracy. The challenges and steps required to validate the photogrammetric accuracy of computer generated imagery are discussed, and an example experiment assessing accuracy of an SFM derived pointcloud from imagery rendered using a computer graphics workflow is presented.

1. **Introduction**

Structure from Motion and

1. **Photogrammetry/ Surveying Background**

Photogrammetry was initially used in \_\_\_\_. 3D measurements were first made in \_\_\_\_. The computer vision community developed numerous automated keypoint detector and descriptors, most notably SIFT in \_\_\_\_\_. These automated keypoint detectors and descriptors were utilized to generate Structure from Motion in \_\_\_. Patch based surface reconstruction (MVS stuff) evolved as a second step in SFM processing to generate dense pointclouds of scenes with known camera interior and exterior orientations, as described by \_\_\_\_. A summary of SfM use cases for geomorphometry by \_\_\_\_ show the many case studies and applications of SFM-MVS that have been tested in the past few years. Most the studies by the geomatics community are currently focused on exploring potential applications and examining the accuracy of the resultant pointclouds.

1. **Computer Graphics Background**

Computer Graphics began in \_\_\_, and has been mostly developed for video games and animated movies. There are numerous resources describing the computer graphics pipeline, such as \_\_\_, and \_\_\_, and \_\_\_. Raytracing vs Radiosity. Global vs Local Illumination. Texturing (Texels vs Pixels). Antialiasing Methodology.

1. **History of Computer Graphics for Remote Sensing**

Using computer graphics rendering as a methodology to test theoretical photogrammetric concepts has been used by \_\_\_. Dirsig has a lot of literature. Cite some papers doing similar things. Currently multiple datasets for testing MVS algorithms. None use GCPs and assess accuracy of SFM pose estimation. Restate why I’m doing what I’m doing.

1. **Goals and Methods again**

**6. Validation**

There are many different renderers available to generate rendered imagery of a simulated scene. Before using a renderer to analyze surface reconstructions a series of validation experiments should be performed to ensure that the renderer is generating imagery as expected. These validation experiments are performed to ensure that any resultant error in an uncertainty analysis is due to SFM algorithm, not inaccurate rendering. Note that there is no experiment presented to validate that accuracy of the lighting as the radiometric accuracy of the lighting is not the focus of this experiment. The authors also recognize the renderer can also be validated by rigorously assessing the rendering source code, or developing a custom rendering algorithm.

**6.1 Photogrammetric Projection Accuracy**

The first validation experiment ensures that the camera interior and exterior orientation are set accurately using a pinhole camera model. This experiment is performed by creating a simple scene consisting of a 1000m3with a 10x10 black and white checkerboard on each wall. The black and white corner of each checkerboard corner is at a known world coordinates. A series of images are rendered using a various camera rotation, translation, focal length, sensor size, and principal point coordinates. To ensure that the images are rendered correctly, the coordinates of the checkerboard corners are calculated from the rendered imagery using a Corner Feature detector and compared to the expected coordinates of the targets using the photogrammetric projection equation (need to reword with actual term). The difference between the image derived coordinates and the photogrammetric equation derived coordinates should have a mean of 0, and a subpixel variance on the order of the accuracy of the image corner feature detector. There should also be no correlation between the accuracy of the coordinate and the location of the coordinate in the image

To validate the photogrammetric projection accuracy of the Blender Internal Renderer, a 1000m3cube was placed with the centroid at the origin. Five hundred images were rendered using five different interior orientations and random exterior orientations throughout the inside of the cube. The parameters used are shown in TABLE X. The accuracy of the imagery was observed qualitatively by plotting the photogrammetric equation calculated points on the imagery in Matlab to ensure a rough accuracy. Once the rough accuracy is confirmed, a nearest neighbor is used to develop correspondences between the Harris corner coordinates and the photogrammetric equation derived coordinates. The mean and variance of the differences between the correspondences in each experiment are shown in Table X. The correlation between the difference in x, y, and radius versus several parameters shows no statistically significant correlation. The correlation results are summarized in Table X.

To ensure that the variance is not an artifact of the rendering, an experiment was performed to determine the expected accuracy of the Harris Corner detector. 1000 Simulated checkerboard patterns were generated with random rotations, translations, and skew to create a synthetic image dataset. The known coordinates of the corners were compared to the coordinates calculated with the Harris Corner feature detector, and the results are shown in in Table X. From these results, the hypothesis that the variance of the rendered image coordinate error is statistically different than the variance of the simulated image coordinate error is false. Therefore, all the variance can be statistically attributed to the Harris Feature Corner detection algorithm, rather than the renderer.

**6.2 Point Spread Function**

The second validation experiment ensures that there is no blurring applied to the rendered image. Specifically, this test determines that the point spread function is a unit impulse. This test is performed by creating a white sphere placed at a distance and size such that it exists in only one pixel. The rendered image should therefore only contain white in the one pixel and not be blurred into any other pixels. This test is particularly important when antialiasing is performed, as the sampling and filter to combine the samples can sometimes create a blurring effect. For example, the default antialiasing in blender uses a “afsdafds” filter which “ensures that a certain amount of the sample color gets distributed over the other pixels as well.” This effect can be seen below, where the intensity of the white sphere is evident in four of the neighboring pixels, even though the sphere should only be visible in one pixel.

To validate the point spread function of the Blender Internal Renderer, a sensor and scene are set up, as depicted in Figure X, such that the geometry of the sphere is only captured with one pixel in the render. This experiment ensures that any other pixels that contain white are an artifact of the rendering. Rendered imagery is shown with and without antialiasing. The antialiasing used is the default settings for the Blender Internal Renderer (8 Samples, Mitchell-Netrevali filter). The rendered image with no antialiasing contains no blurring of the image, while the antialiased image contains a slight amount of blurring. The antialiased imagery renders a smoother, more photorealisitic imagery, and is deemed to be suitable for experimentation.

**6.3 Texture Resolution**

The final validation experiment ensures that any textures applied to the objects in the scene are applied in a manner which maintains the resolution of the imagery. This validation experiment is performed by rendering a texture on a flat plane and rendering an image that contains a small number of the texture pixels. By qualitatively looking at the image, it should be clear that the desired number of pixels are in the frame, and no smoothing is being applied. When rendering textures in computer graphics there is an option to perform interpolation of the texture, which yields a smoother texture. An example of a texture with and without interpolation is shown in Figure X.

To validate the texture resolution of the Blender Internal Renderer a black and white checkerboard pattern where each checkerboard square is 1x1 texel is applied to a flat plane such that each texel represents a 10cm x 10cm square. An image is rendered using a focal length and sensor size such that each texel is captured by 100 x 100 pixels. The rendered image is qualitatively observed to ensure that the checkerboard is rendered for each pixel.

\*honestly this one is highly unlikely to fail… but worth checking I guess.

**7. Automated Workflow using Blender**

Blender Internal Renderer was chosen because it is open source and contains a python API to aid with automated for scene generation and image rendering. The Blender Internal Renderer uses raytracing with local illumination by default and enables the ability to add image textures to objects.

An XML schema was developed using photogrammetric terms to simplify the generation of a scene, camera interior orientation, and camera exterior orientations (trajectory). The design of the XML schema is focused on utilizing photogrammetric terminology that is easy for a geomatics engineer or end user of SfM to understand. A python script using the Blender API automatically converts the XML files into a 3D scene and rendered imagery using a pinhole camera model. The Python script serves as a methodology to convert the photogrammetric settings into rendered imagery without the user needed to understand the various computer graphics terminologies. This data is then processed using Matlab to add nonlinear distortion, noise, blurring, and intensity vignetting to the imagery. In cases where fisheye distortion is desired, the image is rendered as a larger image with the same focal length so that the fisheye distortion has data to query on the edges of the image before being cropped to the desired image size.

**8. Proof of Concept/Demo**

An example experiment is generated as a proof of concept to demonstrate a potential workflow for testing the effect of various independent variables on Sfm Accuracy. A 100m x 100m hilly topography is generated manually using blender with elevations ranging from -10 to 10m. A MxN texel image of \_\_\_ from \_\_\_ source is used as an image texture on the topography. This image corresponds to a texel GSD of \_\_\_ cm. To improve the GSD, a MxN texel image of grass from \_\_\_\_\_ (source) is placed in a 5x5 pattern on the texture. The image is “seamless,” which ensures that the edges between the repeating pattern is not visible. The texel GSD of this image is \_\_\_ cm. (NUMBER) Ground Control Point (GCP) Objects, shown in Figure \_, are distributed across the topography to simulate a common field experiment layout. A trajectory is generated with camera poses on a uniform grid with random Gaussian noise added to both the translation and rotation parameters, as defined in equation X. The images are rendered using the camera parameters shown in Table X.

The imagery is processed in Agisoft Photoscan (version) using the setting shown in Table X. The camera poses, interior orientation, and GCP locations in XYZ and pixel coordinates are all known, and are used as inputs to photoscan. This unrealistic case is used to demonstrate what the potential ideal pointcloud accuracy. The pixel coordinates of the GCPs are input to the milli-pixel, which is near impossible in a real scenario.

The accuracy of each control point in Photoscan is shown in Table X. And the

The dense RGB pointcloud is compared to the actual mesh of the scene using Cloud Compare Point-Mesh distance calculation. The results are output as a CSV file and gridded in Matlab using a nearest neighbor gridding algorithm. The accuracy and standard deviation of each point are shown in Figure X.