Draft

Version 0.1

A Simulated Imagery Rendering Workflow using Blender for Photogrammetric 3D Reconstruction Accuracy Assessments

Prepared By:
Richard Slocum
Dr. Chris Parrish

December 5, 2016

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

Three dimensional Geospatial pointcloud data with an accuracy <5cm and with a density greater than 10 points/m^2 has traditionally been acquired using either terrestrial or airborne lidar, but recent advances in Structure from Motion and MultiView Stereo algorithms have increased in popularity within the past 10 years. SFM and MVS algorithms began development 30+ years ago (site) but have only begun to be utilized for commercial surveying applications recently due to the advances in camera hardware, Unmanned Aerial Systems (UAS), computer processing power, and commercial SFM-MVS software. Research into SFM and MVS in the geomatics community is currently focused on both the accuracy and potential applications of the commercial SFM and MVS software packages such as Agisoft Photoscan and Pix4d . The accuracy of SFM-MVS can vary greatly depending on a variety of factors , and the geomatics community is currently focused on determining potential use cases where SFM-MVS derived pointclouds can begin to replace lidar as an alternative surveying tool, without sacrificing accuracy. . The most common methodology for assessing the use cases and accuracy of SFM-MVS algorithms is to collect data in the field using a UAS and compare the data to lidar data. It is difficult to constrain many of the independent variables using this methodology, and the data acquisition can be expensive and time consuming. We propose a computer graphics based workflow to simulate various scenes and maintain full control over the ground-truth and the camera parameters. This workflow will allow for more robust experiments to be performed by geomatics engineers to assess the feasibility and accuracy of SFM-MVS in various applications.

cite: OLD SFM-MVS

cite: Geomprph sfm paper

cite: accuracy vs factors

Should I cite specific use examples?

The 3D reconstruction methods used in most commercial software consist of an SFM algorithm first, then an MVS algorithm. Unordered photographs are input into the software, and a keypoint detection algorithm such as SIFTis used to detect keypoints and correspondences between images. The SFM algorithm calculates camera interior orientation, exterior orientation, and a "sparse" pointcloud. A bundle adjustment is performed to optimize the matches. Without any additional information, the coordinate system is arbitrary. Either the camera position for each camera, or ground control points are used to transform the coordinate system. This transformation can either be done with a helmert transform, or using a nonlinear optimization. The interior and exterior orientation for each image is used as the input to the MVS algorithm, which generates a more dense pointcloud. The MVS algorithm can generate more correspondences because it utilizes a 1D search along the epipolar line between two images due to the known interior and exterior orientation. For this reason, the accuracy of the MVS algorithm is highly dependent on the accuracy of the parameters calculated with the SFM algorithm. A detailed explanation of the various MVS algorithms can be found in paper. Each of these algorithms also assumes that the scene is rigid with constant lambertian surfaces, and deviations from these assumptions will drastically effect the accuracy.

cite sift

cite: helmert uas

cite MVS algorithms

Numerous studies have been performed to quantify the accuracy of the SFM-MVS algorithms in a variety of environments. The most common and robust method has been to compare the SFM-MVS derived pointcloud to a groundtruth terrestrial lidar survey . Independent GPS control points have also been used, but result in fewer correspondences to compare with . The use of independent control points also can exhibit a bias when the points used are easily photo identifiable targets (eg. checkerboards). The highly textured independent control points will demonstrate a much better accuracy than a homogeneous surface, due to the lack of matching features, and therefore are not necessarily representative of the accuracy of the entire scene. The accuracy of SFM is adversely affected by: homogeneous scene texture, poor image overlapping, lens distortion not modeled by the nonlinear lens distortion equation, poor GCP distribution, inaccurate GCP or Camera positions, poor image resolution, blurry imagery, noisy imagery, varying sun shadows, moving objects in the scene, user error in manually selecting image coordinates of GCPs, low number of images, or a low number of GCPs. With a computer graphics workflow, each of these variables can be constrained and varied in order to assess the effect on the overall accuracy of the scene. The accuracy of the resultant SFM-MVS pointcloud is also assessed by comparing it to a scene with no uncertainty because it is simulated.

cite lidar accuracy studies

cite: sfm vs lidar

cite: sfm vs independent gcps

2 Computer Graphics for Remote Sensing Analysis

The field of computer graphics has been developed through the years, with many advances coming predominantly for more realistic video games and movies. There are numerous settings when rendering a scene which can drastically alter the resultant rendering. Camera parameters are input with interior and exterior orientations, but the determination of the intensity for each color value can be calculated a a few ways.

The discussion below will encompass the settings used by the Blender Internal Renderer. More detailed discussion of the Rendering settings can be found in citation .

cite: detailed rendering overview paper

Need to expand these thoughts

Development of the scene. Adding objects, declaring diffuse (perfectly lambertian), specular, alpha, color, texture, adding light sources, shading/shadeless.

Raytracing rendering of scene, anitaliasing, issue with blender antialiasing.

Utilizing computer generated imagery as a methodology to test theoretical concepts has been used to test someone and something. .

cite dirsig

cite other CGI for remote sensing

3 Renderer Accuracy 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. In this paper, we present an assessment and validation using Blender Internal Renderer, but this validation methodology could be applied to any renderer. 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 recognize the renderer could also be validated by rigorously analyzing or developing the rendering source code.

3.1 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 1000m3 with 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 1000m3 cube 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.

3.2 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 âĂIJafsdafdsâĂİ filter which âĂIJensures that a certain amount of the sample color gets distributed over the other pixels as well.âĂİ 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.

3.3 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.

4 Proof of Concept

Describe example experiment and briefly discuss results.

5 Discussion

Discuss the implications of the methodology. Discuss potential issues.

6 Conclusion

recap. potential future work.