

**Individual Report |February 17, 2017**

**SAMBUDDHA SARKAR**

**eXcalibR**

**1. INDIVIDUAL PROGRESS**

1.1 Overview

In this report I would describe about trajectory generation, multiple image capture and exporting data from the virtual scene. The virtual environment is being modeled in an open source platform: Blender 3D 7.68a. It is a Maya based platform and is programmable by Python 3.

Topics covered have been listed below for a quick overview.

1.2 Test Trajectory

1.3 Image Capture (100 samples)

1.4 Exporting Camera Extrinsics & Intrinsics

1.2 Calibration Target Mesh Model: Icosahedron

Using the model editor, a trajectory was created in the virtual scene with something called a NurbsPath (it is basically an editable vector with user desired subdivisions or ‘control points’). This trajectory is shown in Fig. 1.2.1.

The Icosahedron was then constrained to travel along the path specified by the NurbsPath. The increments in its motion were not synchronized with the camera shutter as to simulate real world motion blur. This helped us to pin down the window of rate of change of position of the calibration target so as to avoid motion blur, the results were desirable. The multi camera view of the path is shown in Fig. 1.2.2.

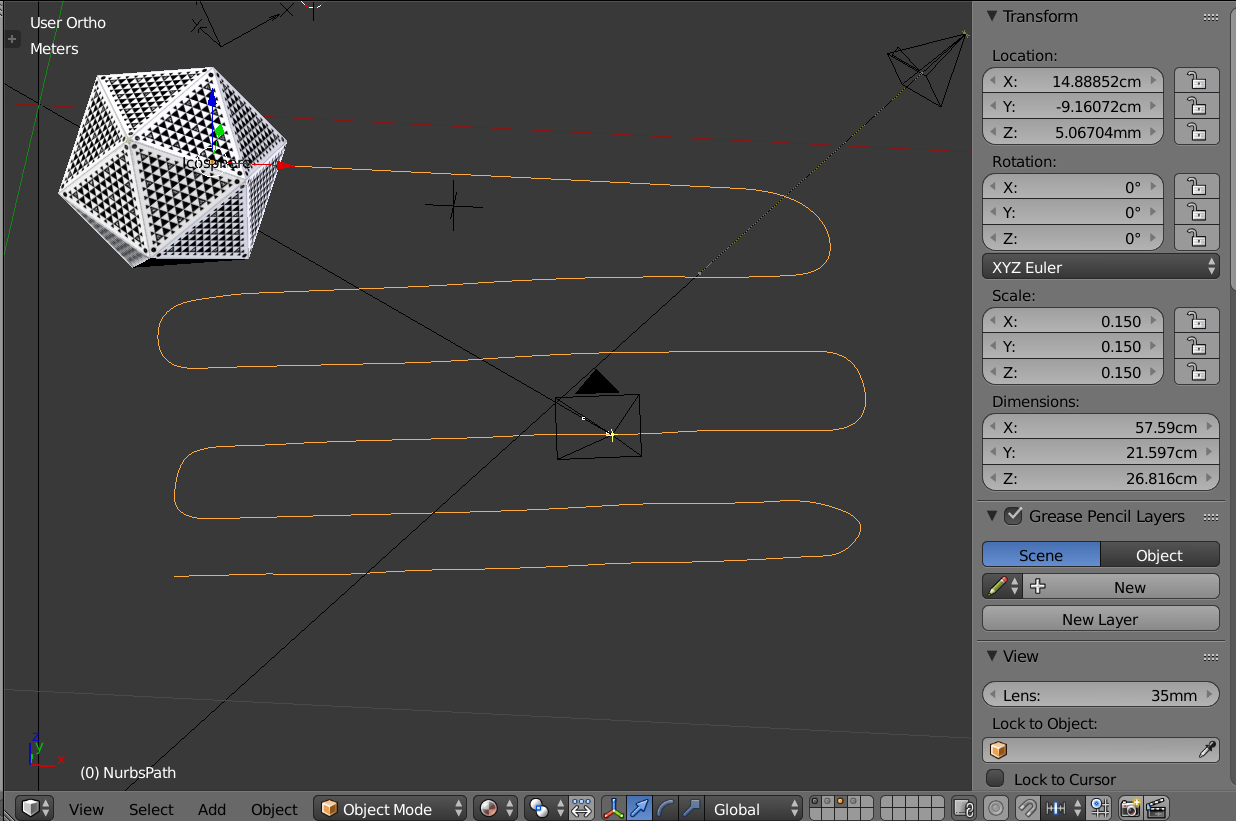


Fig. 1.2.1, Trajectory of calibration target

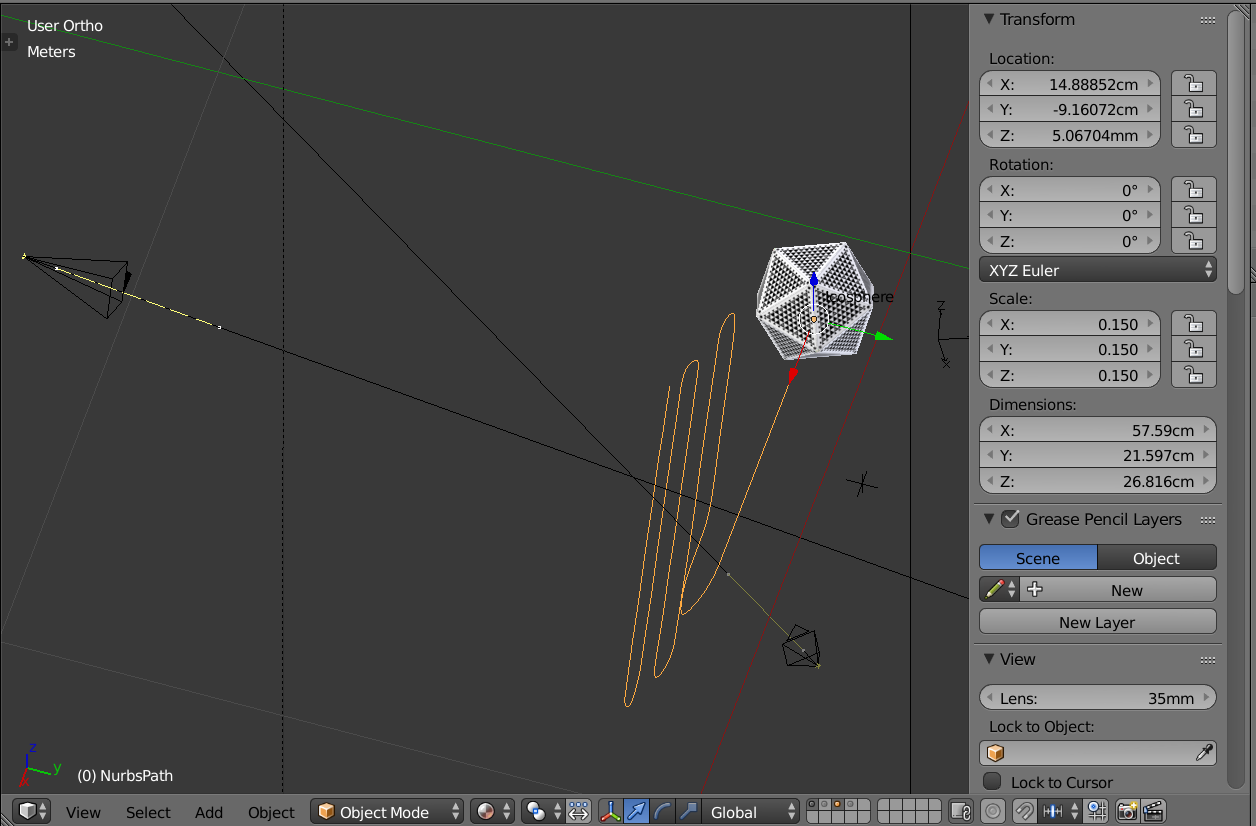


Fig. 1.2.2, Multi-camera positioning for Trajectory

1.3 Image Capture (100 samples)

The multi-camera locations were locked and the calibration target was animated to move along the NurbsPath and cameras were scripted to capture 100 images. The path was chosen such that the calibration target covered the whole FOV(field of view) of the respective camera. The FOV of camera 1 is shown in Fig 1.3.1. The rendered FOV of camera 1 is shown in Fig. 1.3.2. The orientation of the faces on the calibration target is very specific and this has to mapped exactly to the designated vertices of the Icosahedron. This mapping has to be exploited so that we know the geometric calibration is working. So the path was designed in such a manner that the target changed its orientation in 3D space. Fig. 1.3.3 shows 100 rendered images from the FOV of camera 1 and you can see how the face angles change throughout.

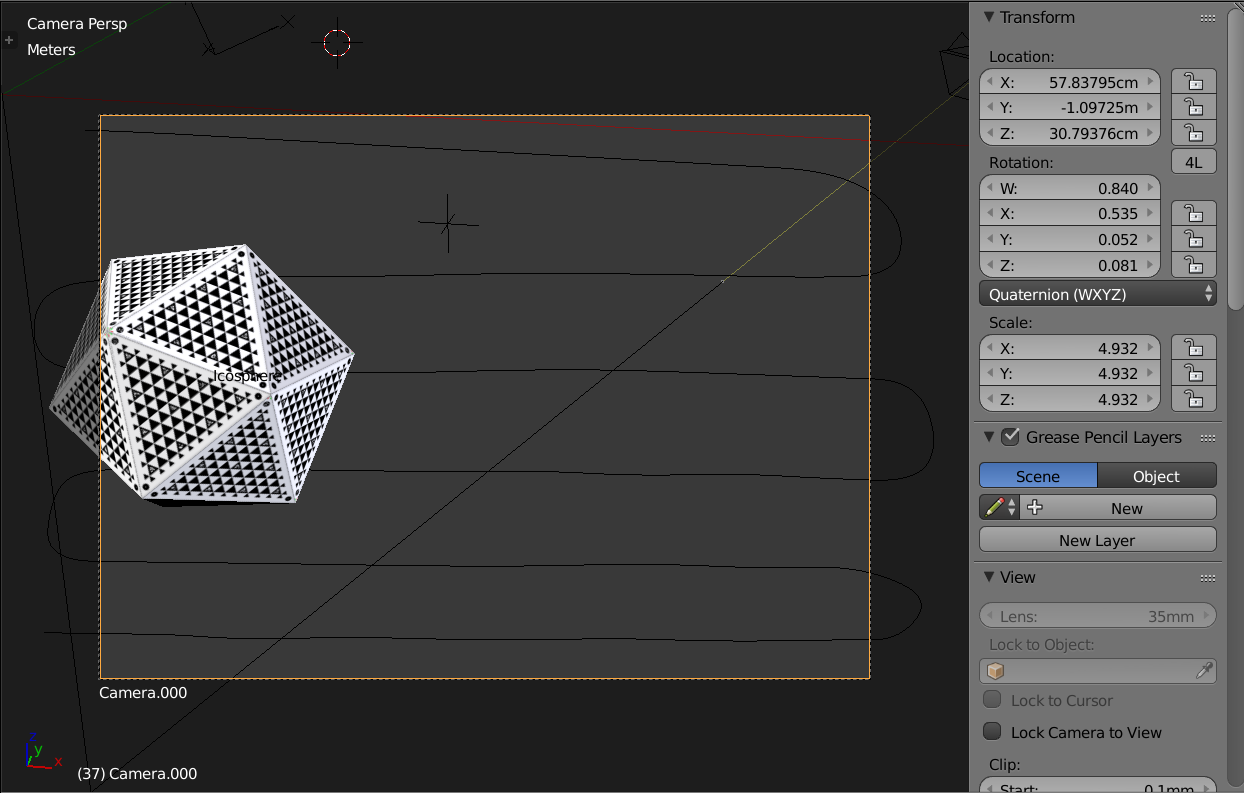


Fig. 1.3.1, FOV of Camera 1(Not Rendered)

Fig. 1.3.2, FOV of Camera 1 (Rendered)



Fig. 1.3.3, 100 Images (Post Rendering)

1.4 Exporting Camera Instrinsics & Extrinsics

The camera Intrinsics and extrinsics were exported from blender using python. The script is shown on the left and the output on the left in Fig. 1.4.

The mesh data of the calibration target is exported in form of surface normals and centre of the faces.

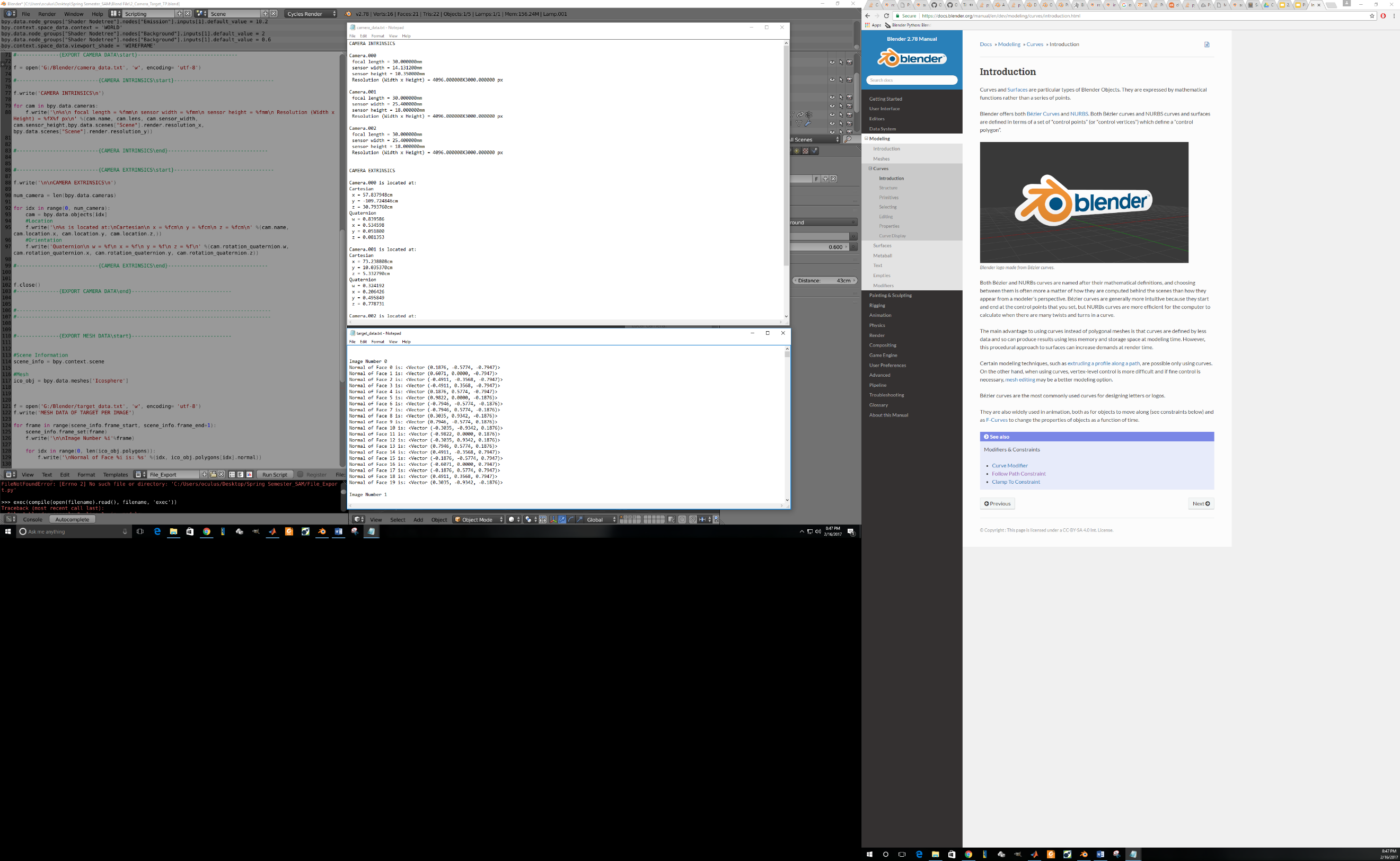


Fig. 1.4, Exporting blender data

**2. FUTURE PLANS**

My future plans until the next progress report is to come up with a pipeline for image generation using blender. Also exporting the face pattern geometric data to validate the geometric calibration algorithm. The development and deployment of this pipeline will take about 90 days.

**3. BONUS: IMAGE RENDER SAMPLES**

**Some image samples from the 100 sample range. (showing the drastic changes in the orientation of the target to test the robustness of the geometric calibration algorithm: this is from camera 2)**

