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.
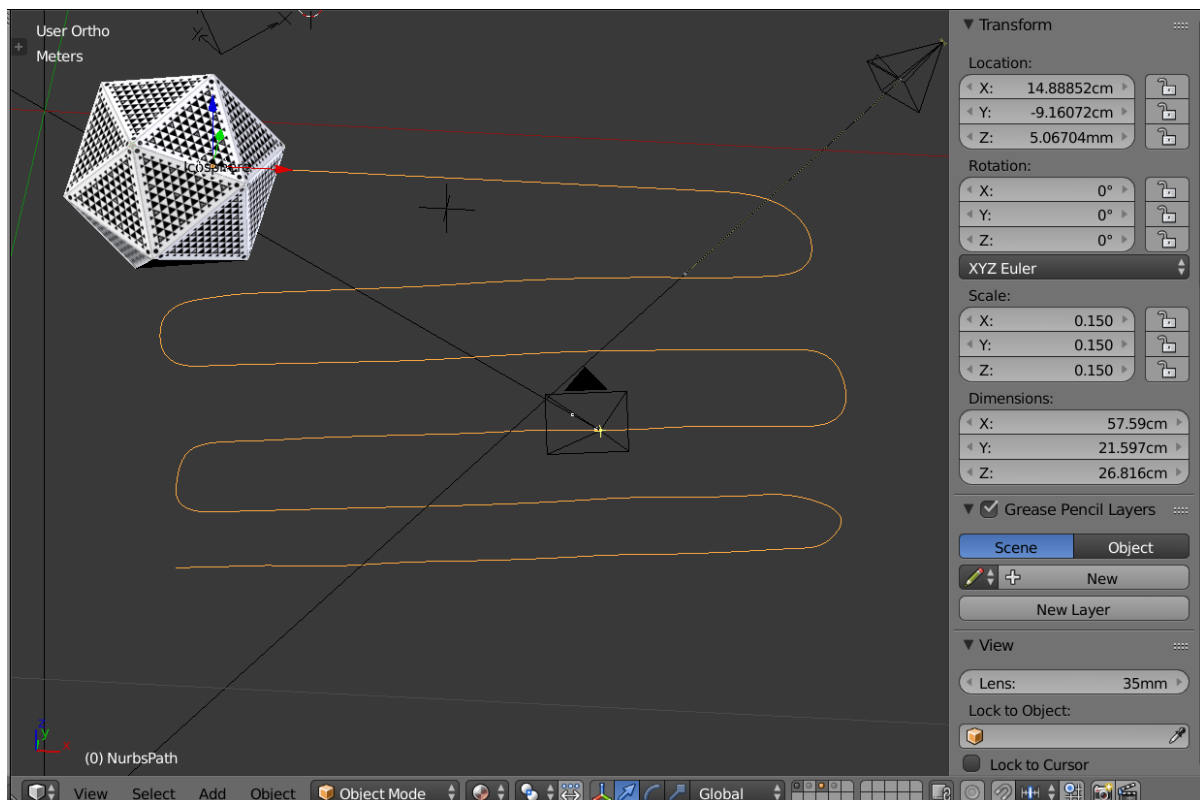
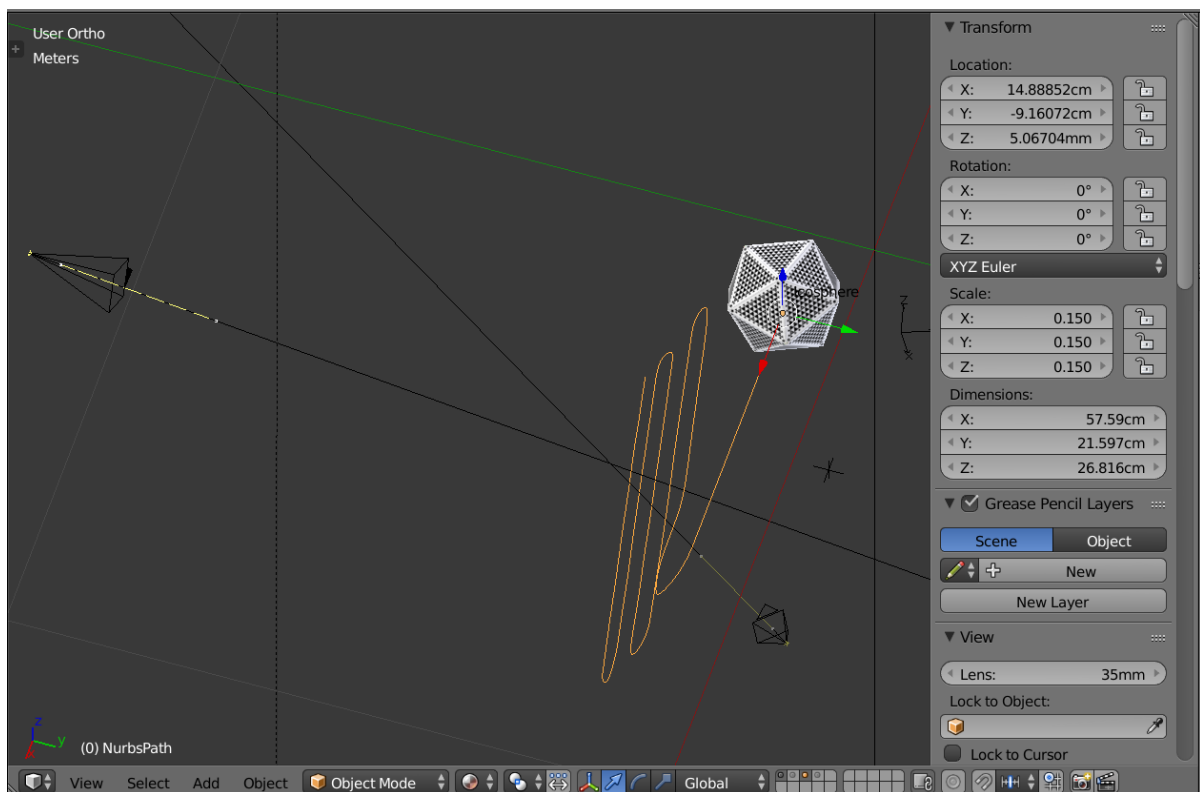Fig. 1.2.1, Trajectory of calibration target

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.
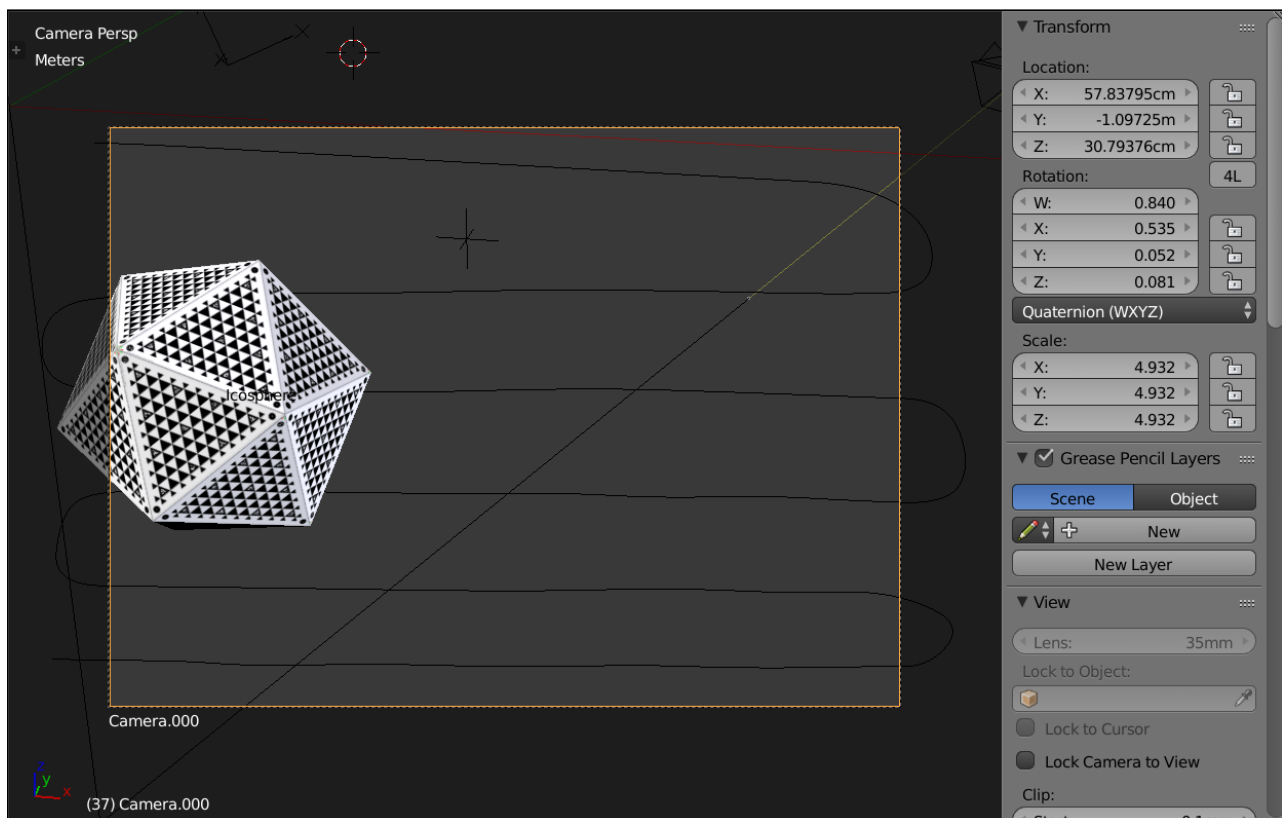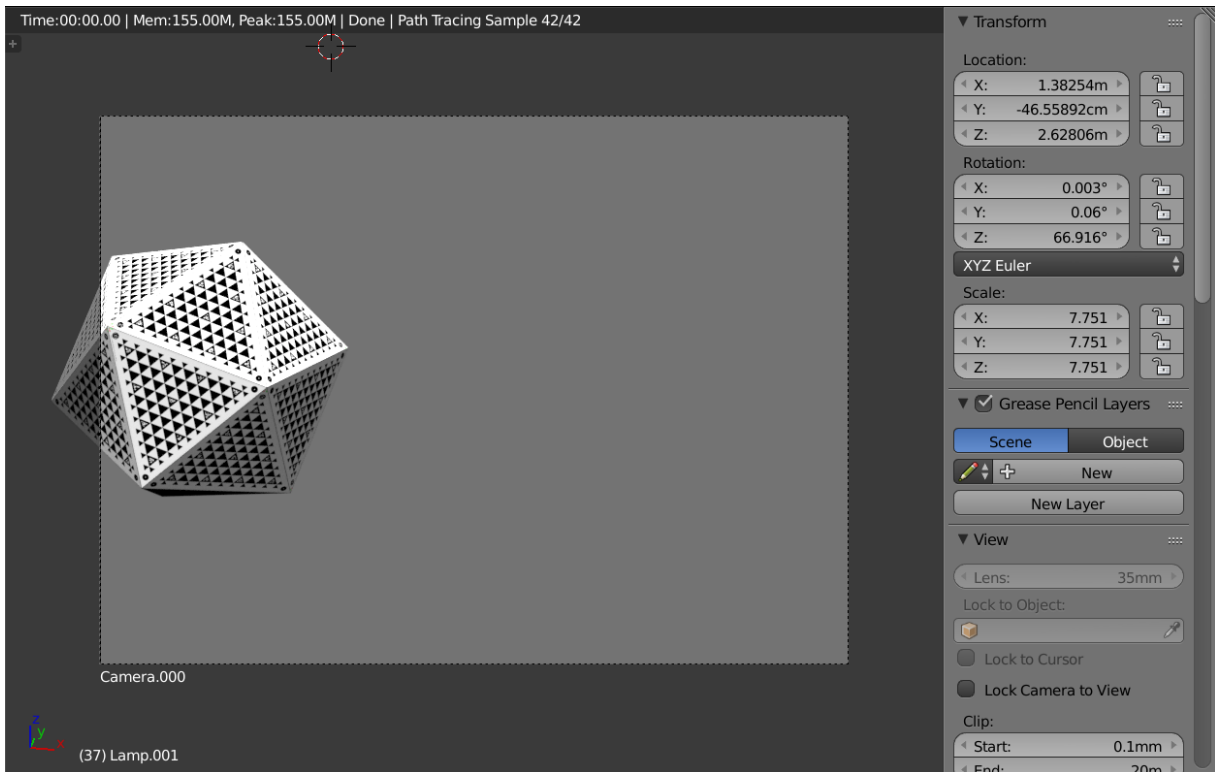

Fig. 1.3.1, FOV of Camera 1(Not Rendered)

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.

```
bpy.data.node_groups["Shader Nodetree"].nodes["Background"].inputs[1].default_value = 0.6
bpy.context.space_data.viewport_shade = 'WIREFRAME'
71 #--------------{EXPORT CAMERA DATA\start}--------------------------------
72
73 f = open('G:/Blender/camera_data.txt', 'w', encoding= 'utf-8')
74
75 #--------------------{CAMERA INTRINSICS\start}--------------------------
76
77 f.write('CAMERA INTRINSICS\n')
78
79 for cam in bpy.data.cameras:
80     f.write('\n%s\n focal length = %fmm\n sensor width = %fmm\n sensor height = %fmm\n Resolution (Width x
Height) = %fX%f px\n' %(cam.name, cam.lens, cam.sensor_width,
cam.sensor_height,bpy.data.scenes["Scene"].render.resolution_x,
bpy.data.scenes["Scene"].render.resolution_y))
81
82
83 #--------------------{CAMERA INTRINSICS\end}--------------------------
84
85
86 #--------------------{CAMERA EXTRINSICS\start}--------------------------
87
88 f.write('\n\nCAMERA EXTRINSICS\n')
89
90 num_camera = len(bpy.data.cameras)
91
92 for idx in range(0, num_camera):
93     cam = bpy.data.objects[idx]
94     #Location
95     f.write('\n%s is located at:\nCartesian\n x = %fcm\n y = %fcm\n z = %fcm\n' %(cam.name,
cam.location.x, cam.location.y, cam.location.z,))
96     #Orientation
97     f.write('Quaternion\n w = %f\n x = %f\n y = %f\n z = %f\n' %(cam.rotation_quaternion.w,
cam.rotation_quaternion.x, cam.rotation_quaternion.y, cam.rotation_quaternion.z))
98
99 #--------------------{CAMERA EXTRINSICS\end}--------------------------
100
101
102 f.close()
103 #--------------{EXPORT CAMERA DATA\end}--------------------------------
104
105
106 #------------------------------------------------------------
107 #------------------------------------------------------------
108
109
110 #--------------{EXPORT MESH DATA\start}--------------------------------
111
112
113 #Scene Information
114 scene_info = bpy.context.scene
115
116 #Mesh
117 ico_obj = bpy.data.meshes['Icosphere']
118
119
120
121 f = open('G:/Blender/target_data.txt', 'w', encoding= 'utf-8')
122 f.write('MESH DATA OF TARGET PER IMAGE')
123
124 for frame in range(scene_info.frame_start, scene_info.frame_end+1):
```

```
CAMERA INTRINSICS

Camera.000
  focal length = 30.000000mm
  sensor width = 14.131200mm
  sensor height = 10.350000mm
  Resolution (Width x Height) = 4096.000000X3000.000000 px

Camera.001
  focal length = 30.000000mm
  sensor width = 25.400000mm
  sensor height = 18.000000mm
  Resolution (Width x Height) = 4096.000000X3000.000000 px

Camera.002
  focal length = 30.000000mm
  sensor width = 25.400000mm
  sensor height = 18.000000mm
  Resolution (Width x Height) = 4096.000000X3000.000000 px

CAMERA EXTRINSICS

Camera.000 is located at:
Cartesian
  x = 57.837948cm
  y = -109.724846cm
  z = 30.793760cm
Quaternion
  w = 0.839586
  x = 0.534598
  y = 0.051800
  z = 0.081353

Camera.001 is located at:
Cartesian
  x = 73.238808cm
  y = 10.035370cm
  z = 5.332790cm
Quaternion
  w = 0.324192
  x = 0.206426
  y = 0.495849
  z = 0.778731

Camera.002 is located at:
```

target_data.txt - Notepad
File   Edit   Format   View   Help

```
Image Number 0
Normal of Face 0 is: <Vector (0.1876, -0.5774, -0.7947)>
Normal of Face 1 is: <Vector (0.6071, 0.0000, -0.7947)>
Normal of Face 2 is: <Vector (-0.4911, -0.3568, -0.7947)>
Normal of Face 3 is: <Vector (-0.4911, 0.3568, -0.7947)>
Normal of Face 4 is: <Vector (0.1876, 0.5774, -0.7947)>
Normal of Face 5 is: <Vector (0.9822, 0.0000, -0.1876)>
Normal of Face 6 is: <Vector (-0.7946, -0.5774, -0.1876)>
Normal of Face 7 is: <Vector (-0.7946, 0.5774, -0.1876)>
Normal of Face 8 is: <Vector (0.3035, 0.9342, -0.1876)>
Normal of Face 9 is: <Vector (0.7946, -0.5774, 0.1876)>
Normal of Face 10 is: <Vector (0.3035, 0.9342, 0.1876)>
```

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)