TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Per-Pixel Calibration and 3D Reconstruction on GPU

In this chapter, a simple parallel RGB-D camera calibration method with a simple undistorted 3D reconstruction on GPU is introduced. A KinectV2 camera, as one famous consumer RGB-D camera, is utilized to during the calibration and 3D reconstruction. We will finally show an undistorted 3D reconstruction with color (RGB) on GPU. The calibration method can be applied universally on all kinds of RGB-D cameras.

show the piled frames

1.1 Calibration System

In this section, we will find a best-fit calibration system for a KinectV2 camera's easy parallel calibration and reconstruction, which is able the handle both of lens distortion and *depth distortion*(as introduced in section **??**). The KinectV2 camera has one depth sensor and one RGB sensor. The RGB sensor offers RGB streams, whose color infos will be aligned to pixels according to their corresponding world space coordinates after the 3D reconstruction. The depth sensor offers NearIR and Depth streams, both of which share the same size (*row* and *column*) and lens distortions of the depth sensor, and can both be utilized for lens distortion correction. We will use the Depth stream to recover world space 3D coordinates $X^W Y^W Z^W$ for reconstruction, during which the NearIR stream will be used for lens distortion correction.

To easily show 3D reconstruction in a parallel way on the GPU, we would like to find a per-pixel mapping model that can generate per-pixel world coordinates $X^W Y^W Z^W$ from per-pixel Depth(D) value. Kai [1] did a good job on structed light 3D scanner parallel calibration on GPU. He derived the per-pixel beam equation (1.1), the linear relationship that map to X^W/Y^W from Z^W, directly from pinhole camera matrix M, while the per-pixel Z^W comes from features of structured light. Got inspired by Kai, we would like our calibration system to be able to determine a per-pixel D to Z^W mapping model. In this way, not only eqn. 1.1 can be employed during the parallel reconstruction on GPU, but the *depth distortion* could also be corrected during the per-pixel D to Z^W mapping.

$$\begin{aligned} X^W_{[row,col]} &= c_{[row,col]} Z^W_{[row,col]} + d_{[row,col]} \\ Y^W_{[row,col]} &= e_{[row,col]} Z^W_{[row,col]} + f_{[row,col]} \end{aligned} \tag{1.1}$$

Of all different kinds of calibration systems, a camera on rail system with a planar

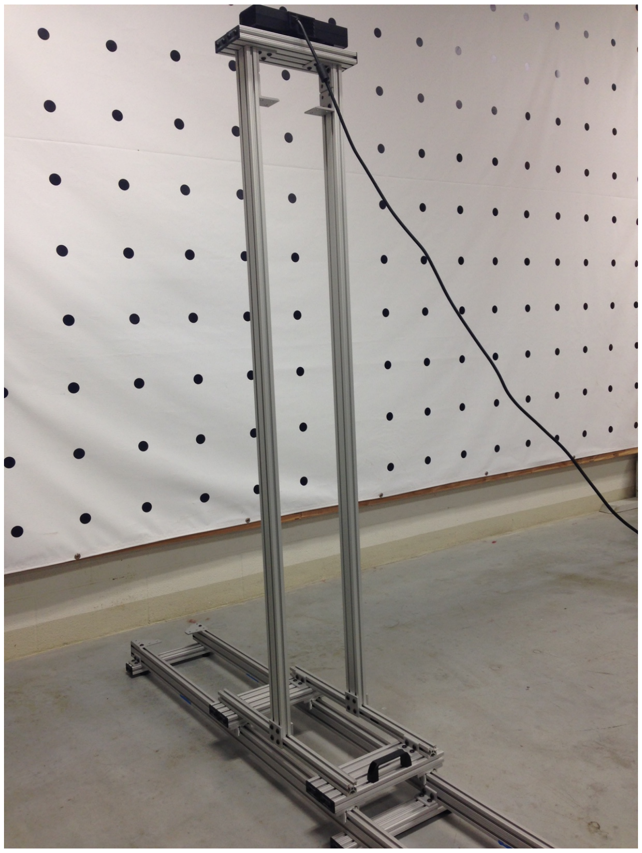

Figure 1.1: KinectV2 Calibration System

pattern on wall is finally decided, which offers a moving plane with respect to the camera when the camera moves along the rail. As fig. 1.1 shows, a canvas on which printed an uniform grid dots pattern is hung on the wall, and the rail is required to be perpendicular to the wall. The RGB-D camera waiting to calibrate is mounted on the slider. Note that, in this calibration system, the only unit that needs to be perpendicular to the wall is the rail, whereas the RGB-D camera has no need to require its observation orientation. Because the per-pixel calibration requires only accurate world space coordinates which will be decided by the rail and the wall, whereas the camera's space is not considered at all. We will assign the pattern plane as the $X^W Y^W$ plane in world space, and the rail to be along with (not exactly on) Z^W-axis. The world coordinate is static with the camera on the slider.

Figure 1.2 shows one frame of NearIR $X^W Y^W Z^W$ 3D reconstruction, which can help a lot explaining how the world space origin is assigned. Inside the figure, both of the origin and Z-axis are high-lighted in blue, and the origin of world space is on the left end of the blue line. On the pattern plane with dot-clusters marked in circles, we can see one dot-cluster is high-lighted inside a thick circle, and its center point is where $X^W/Y^W = 0$. The dot-cluster which will be sitting on the Z^W-axis is the one whose center point is closest

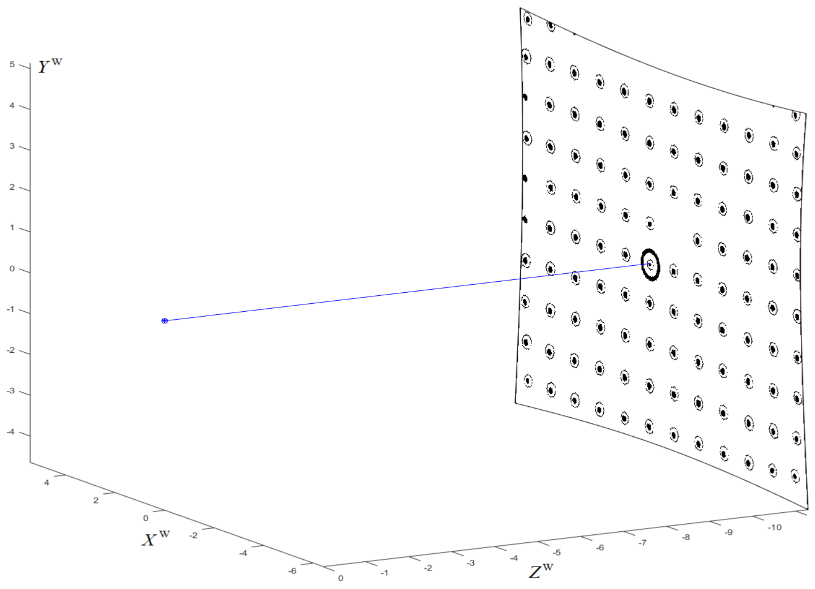

Figure 1.2: NearIR $X^W Y^W Z^W$ 3D Reconstruction

to the center pixel of the sensor. All pixels in this frame share the exact same Z^W, which is also why we require the rail to be perpendicular to the pattern. And the value of Z^W is measured by a laser distance measurer that static with the camera. The final origin of the world space will be decided by both of the camera's observing orientation and the laser distance measurer's position. This kind of world coordinate assignment is totally for simplifying image processing during calibration. Practically, we do not even care where exactly the origin is, as long as the rail is perpendicular to the pattern and the distance measurer is static with the camera.

With this rail calibration system, infinite number of frames with infinite number of calibration points could be utilized for training a calibration model. Besides, as the slider moves along the rail, the amount and distribution of the grid dots captured by the camera will change, which means a dynamic pattern for calibration instead of static. With more and more dots walking into the camera's field of view under a certain rhythm as the slider moves further from the pattern plane, the dots (which will be extracted as calibration points) are able to cover all pixels of a sensor. What's more, a moving-plane system (multiple frames calibration instead of one frame calibration) makes it possible to do dense D to Z^W mapping, which will handle *depth distortion*.

1.2 Calibration Procedures

With the calibration system built up and world coordinate assigned, we are now ready to calibrate. Z^W values for all pixels of every frame will be supported from external laser distance measurer. To simplify potential calculation during image processing, we assign the world coordinate *Unit One* based on the uniform grid dots patter, to be same with the side of pattern's unit-square. Concretely, the distance between every two adjacent dots' centers in real-world is 228mm. Therefore, Z^W = -Z(mm) / 228(mm), where Z is the vertical distance to the pattern plane in reality measured by the laser distance measurer. Note that, Z^W values are always negative, based on the assignment of Cartesian world coordinate. The outline of calibration procedures is listed below.

1. Mount both of the camera and laser distance measurer onto the slider.

2. Move the slider to the nearest position to the pattern plane.

3. Record one frame of RGB data and one frame of NearIR data at this position.

   a) measure $|Z^W|$ using the laser distance measurer.

   b) grab RGB, NearIR and Depth streams from KinectV2 camera.

   c) extract center points (R/C) of dot-clusters from RGB and NearIR streams respectively.

   d) assign X^W/Y^W values to the extracted points, RGB and NearIR respectively.

   e) train and determine the best-fit high order polynomial model that map from RC to X^W/Y^W, RGB and NearIR respectively.

   f) generate dense X^W/Y^W for all pixels using the model, for NearIR and RGB streams respectively.

   g) save 2 frames of RGB and NearIR all pixels' data respectively: $X^W Y^W Z^W RGBD$ for RGB stream, and $X^W Y^W Z^W ID$ for NearIR stream, where the channel I in NearIR frame denotes *Intensity*.

4. Move the slider to the next position, and repeat step 3.

5. Check all X^W/Y^W values of all frames, unify the staggered origin by adding or subtracting a corresponding integer (make sure the point $X^W Y^W = 0$ is at the same dot-cluster for all frames).

6. Determine the per-pixel mapping from D to Z^W and from Z^W to X^W/Y^W for all pixels using the NearIR frames of data, and generate a *row*-by-*column*-by-6 look-up table.

7. Determine a pinhole-camera matrix that can map $X^W/Y^W/Z^W$ to R/C using RGB frames of data (for the RGB values' alignment to pixels after 3D reconstruction).

Bibliography

[1] Kai Liu. *Real-time 3-D Reconstruction by Means of Structured Light Illumination*. PhD thesis, University of Kentucky, 2010.