Yiqing Cai

Team G: The ExcalibR

Teammates:
Huan-Yang Chang
Man-Ning Chen
Siddharth Raina
Sambuddha Sarka

ILR07
Feb.16, 2017

Individual Progress

Overview

For this stage of project, I was primarily responsible for generating projections for all the cameras, and visualizing projections on the camera FOV. This is useful for evaluating the necessary points the ABB robot arm needs to reach and is useful for the following path calculation process. We planned to do the simulation in RobotStudio for the whole dome setup and calibration process, so the generated points will be the input to the RobotStudio.

Implementation

The camera parameters consist of intrinsic matrix (focal length and principal point), and extrinsic matrix(rotational matrix and translation vector). During last stage, I have successfully generated 200 camera parameter models, and the calculation for projections is based on the parameter models.

The ABB robot arm has working constraints, so my original path is generated based on it. Figure 1 shows the motion range for the robot arm (left) and the original path generated based on it (right). Then I uniformly sampled points on the path and calculate projections.

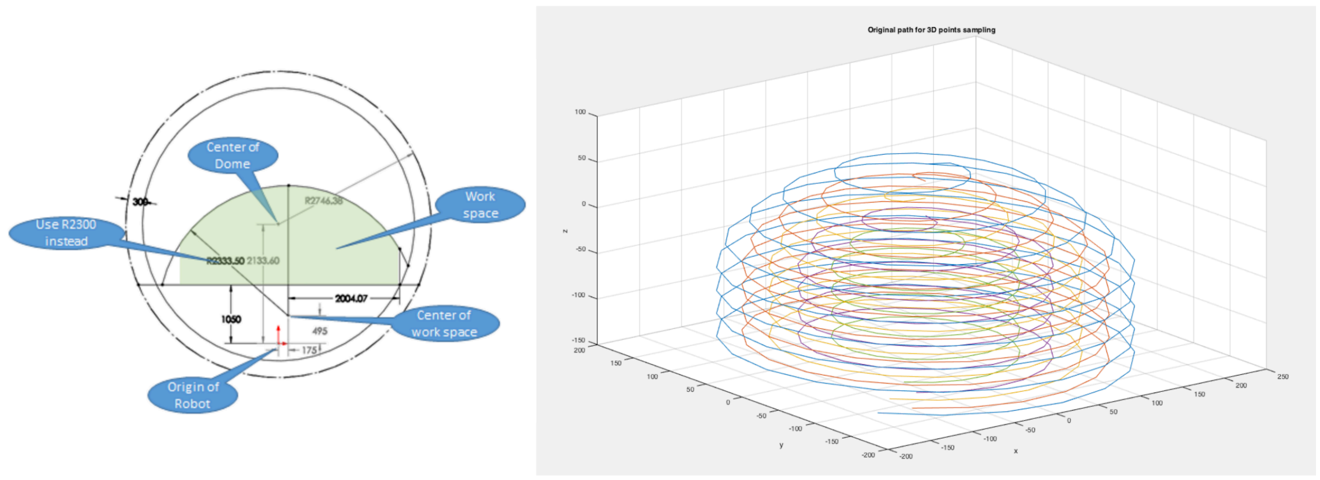

Figure 1. Motion range of ABB Robot Arm(left) and generated original path(right)

Initially, I sampled 695 points uniformly along the original path, which is able to cover all the possible positions for the calibration target which is mounted on the ABB Robot Arm. In order to record projections at each 3D position for all the cameras, I created a cell array which has 200 structs and each of them contains a matrix which is the same size of image plane. The matrix is initially all zeros. Once got a projection, all the pixels in the covered area plus one. When all the projections are generated, for each matrix, the larger the pixel value, the more overlaps the area has.

As generating projections at 695 points for 200 cameras would take a lot of time, I randomly selected 4 cameras just for the test stage. Figure 2 shows the generated projections for 4 camera FOVs, we can see clearly from the visualization that many points are unnecessary because they are covering repeated areas. I approximated the calibration target as a sphere, and its projection as a circle.

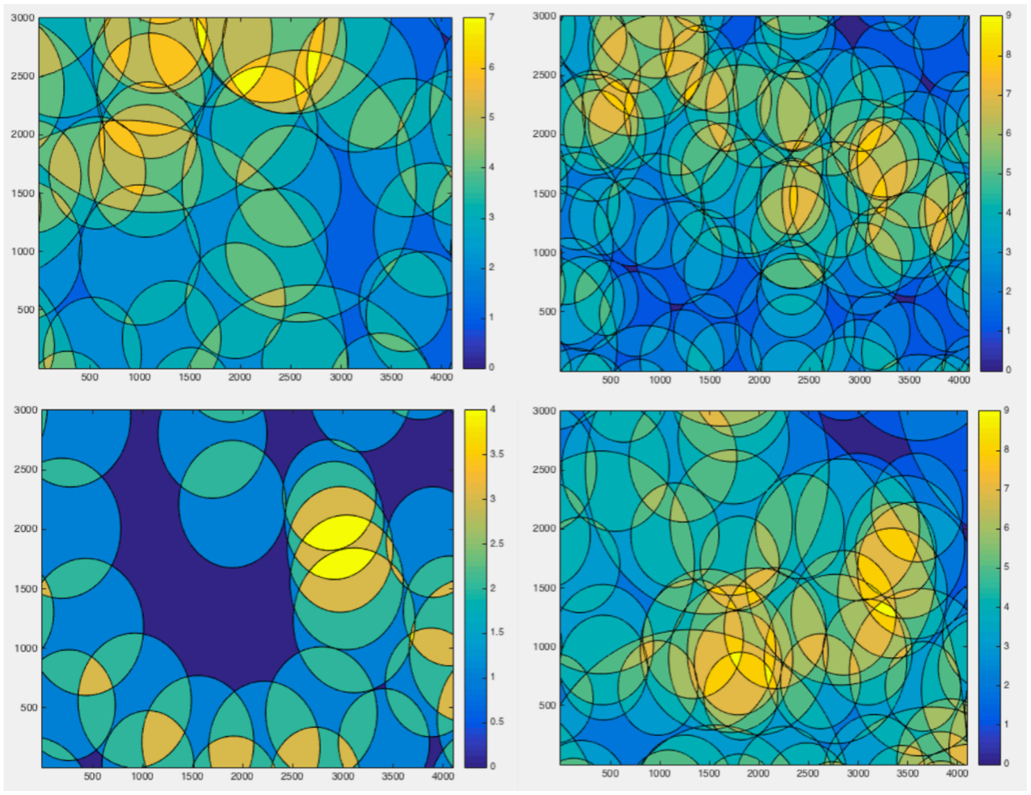

Figure 2. Visualization of projections on 4 camera FOVs

The percentage of projection coverage of the 4 camera FOVs are 99.9%, 99.3%, 76.6% and 98.3% respectively.

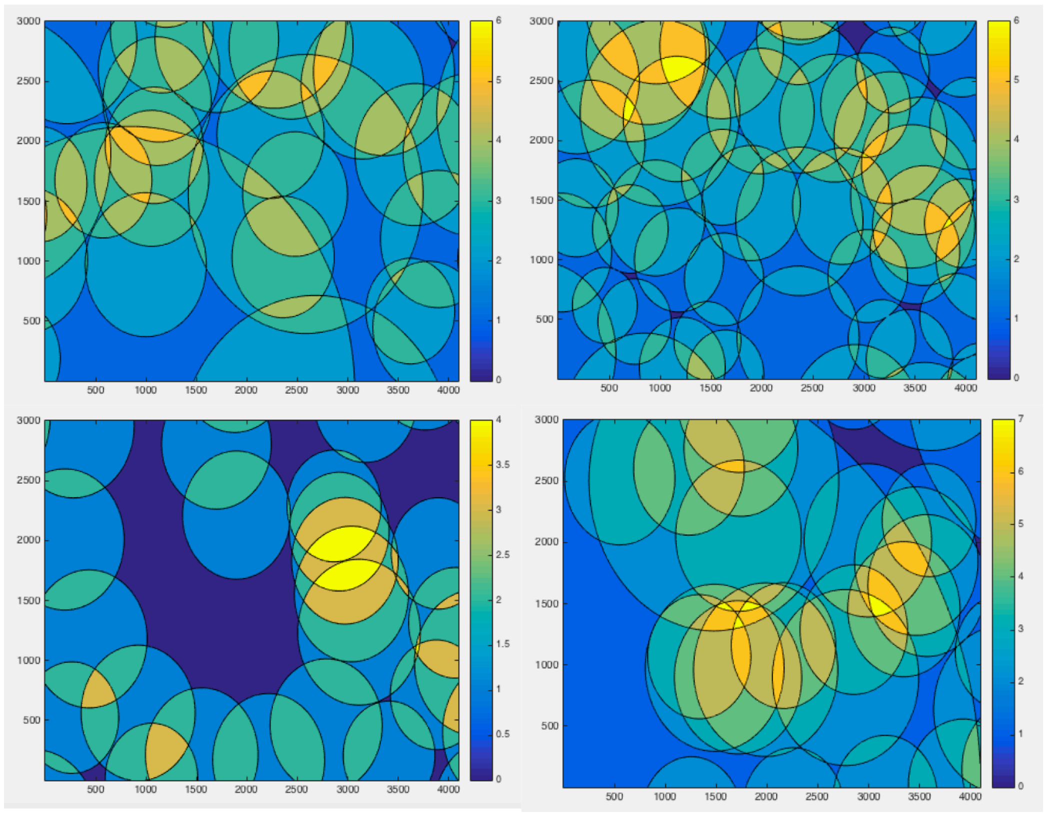

Figure 3. Visualization of projections on 4 camera FOVs after optimization

Then I went through all the 3D positions again, and check if the projections for a specific point on all the camera FOVs are totally overlapped by other projections (which means whether the pixel values on the projection areas are all larger than 1). If it is, the points will be moved. So all the remaining points are necessary based on the objective that the projections should cover as much camera FOVs as possible.

After optimization, I got 86 points left out of 695 points, and the percentage of projection coverage of the 4 camera FOVs remain the same. Figure 3 shows the visualization of projections on 4 camera FOVs after optimization, and Figure 4 shows the difference between

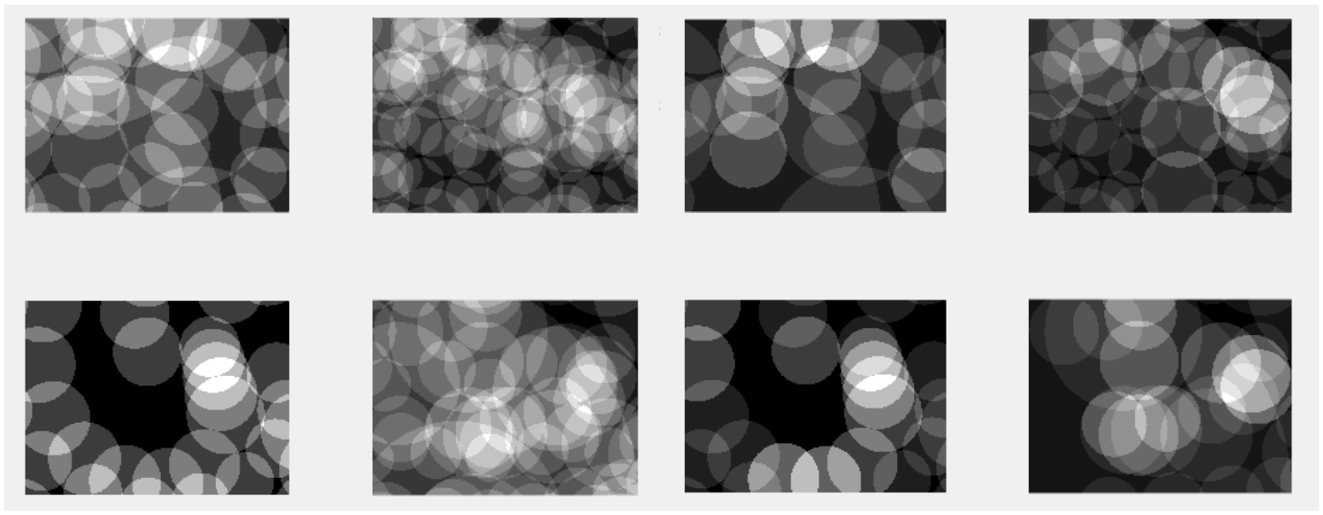

Figure 4. Comparison between before(left) and after(right) optimization

before and after optimization. We can see clearly that a lot of unnecessary points have been moved. However, if I applied this method on all cameras, there would still be several hundreds of points left. So we might need a better way to evaluate the objective and to decide what kinds of 3D positions for the calibration target is the best fit for the calibration process.

Challenges

The main challenges I faced during this task were:

Although I can successfully generate projections, and optimizing for 4 randomly selected cameras seems feasible. However, if I optimize for 200 cameras based on the same criteria —— trying to cover as much camera FOV as possible, it would take a lot of time and there will still remain several hundreds of points. An effective evaluation function is needed to give a better evaluation on whether a specific 3D point is necessary or not.

Teamwork

Work undertaken by each team member is as follows (see Table 1):

Member	Tasks
Huan-Yang Chang	Simulate the robot arm motion in RobotStudio based on the input 3D points
Man-Ning Chen	Detection of each color box and get the RGB value of them
Yiqing Cai	Generate projections of calibration target on camera FOVs
Sambuddha Sardar	Rendering of the environment and generate virtual images
Siddharth Raina	Other Sensor noise Correction

Table 1. Team co-work

Our team worked with great coordination during execution of the second stage of this project. We communicated during the entire task and solved problems together. Sam was working on the rendering of the environment and generating virtual images in Blender. Peter was working on simulating the ABB Robot Arm motion in the Robot Studio according to the 3D points for the motion path. Mandy was working on detection of each color box and get the RGB values of them for color correction. I was working on generating projections of calibration target on all camera FOVs and storing them. Sid is trying to work on other kinds of sensor noise except for the FPN and PRNU we dealt with last semester. We faced many difficulties but we worked them out eventually as a team.

Future Plans

From now on, my task will be focused on the designing the effective evaluation function for projections of calibration target. In order to decide what kinds of 3D points are valuable, we would need a good evaluation function to help with the decision, which is critical to the path planning optimization process for ABB Robot Arm motion. I am considering taking the size and distribution of calibration target as the standards, but I still need to decide the form of the evaluation function, like how to give the weight, scores and structure for recording information.