CRITICAL DESIGN REVIEW

Submitted: 12/15/2016

TEAM G

EXCALIBR

Huan-Yang Chang (Peter)
Man-Ning Chen (Mandy)
Sambuddha Sarkar (Sam)
Siddharth Raina (Sid)
Yiqing Cai (Cece)

Abstract

This report is a comprehensive summary of the work completed by Team G (Excalibr) on their project. The report is begins with a description of the problem being tackled, then it moves on to the system level requirements specific to the problem being solved. This is followed by the system architecture where the nuances of the system are highlighted. The successive pages discuss about the current status of the work and the project management techniques being used for this project. Then the report concludes with a summary and set of inferences with appropriate references as applicable.

This project is being supervised and sponsored by the ORP (Oculus Research Pittsburgh) and the requirements generated adhere to the demands of our sponsor.

Table of Contents

1. Project description

Oculus Research, Pittsburgh is constructing a multi-sensor capture system consisting of a multitude of cameras and microphones to perform motion tracking and 3D reconstruction of objects with unprecedented precision. The capture system consists of a dome, 11 ft. in diameter, mounted with multiple sensors required for capturing images and sound. The first step for achieving this goal of motion tracking and 3D reconstruction involves accurate calibration of the sensors.

The scope of the project involves calibrating these sensors using a robotic arm and an engineered calibration target. The calibration target would be attached to the end effector of the robotic arm, which would move around in the capture system and calibrate the sensors attached to the capture space. The calibration pipeline would involve three methods, namely: geometric calibration of the cameras, estimation of the illumination: light field calibration, photometric calibration of the cameras and acoustic calibration of microphones.

The sensors must be calibrated with a very high accuracy and precision. The results must be repeatable and reproducible and the entire pipeline must be fast and efficient.

2. Project Goals

To design a turnkey solution for accurate sensor calibration including geometric calibration, photometric calibration and light field calibration.The total calibration process will be finished in one night. For geometric calibration, the reprojection error will less than 0.1 pixels.

3. Use case

3.1 Narrative

After the meeting with the customer, Peter was very depressed because of the complaints about the quality of the 3D reconstruction which were haunting his mind. Peter was a hard worker and usually stayed up late for calibrating the system in order to maintain the system's accuracy for the next day's work. However, using the manual checkerboard method could only give him a coarse result. Besides, the huge number of cameras and microphones to be calibrated left him totally drained out of energy(Figure 3.1). In the midnight, he started to think if there was a system which could help him to autonomously calibrate this multi-sensor system.

One-day Peter found out a new machine had been installed in the office. This new calibration system, called 'Excalibr' could calibrate cameras and microphones autonomously and had an amazing accuracy. At last, Peter didn't have to stay up all night and would just push a button to calibrate the sensors. The calibration system would finish its job autonomously throughout the night with excellent quality.

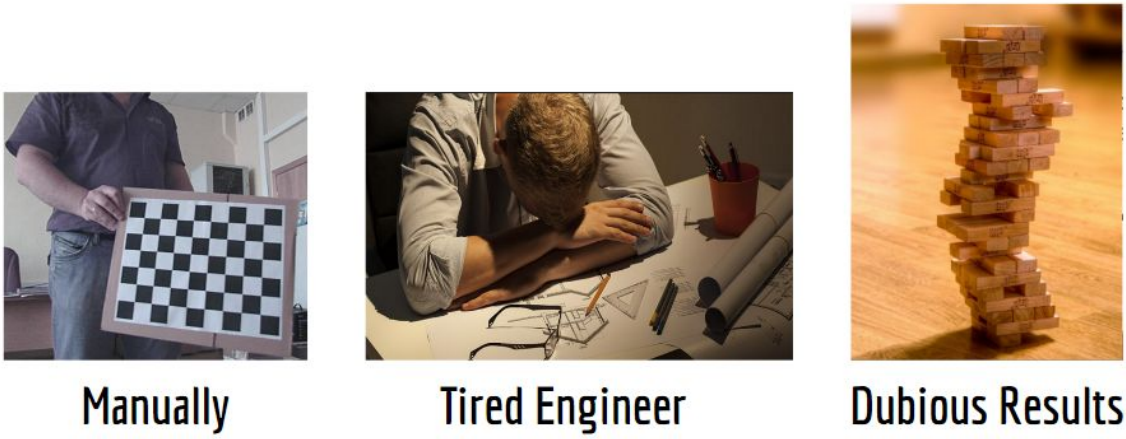

Figure 3.1 Problems in calibrating multi-sensor system

3.2 About Excalibr

Excalibr(Figure 3.2) was a multi-sensor calibration system which was designed to automatically calibrate the sensors in one night. Its robot arm could move the calibration target with high precision. The camera trigger system could take the high quality image with no motion blur. The sensor noise removal could ensure the high quality images. With the high quality image, the Excalibr could provide the repeatable camera calibration results. Besides, the Excalibr could conduct the light-field detection with the robot arm to get accurate light-field information in the environment. This efficient and precise machine could provide reliable results every time.

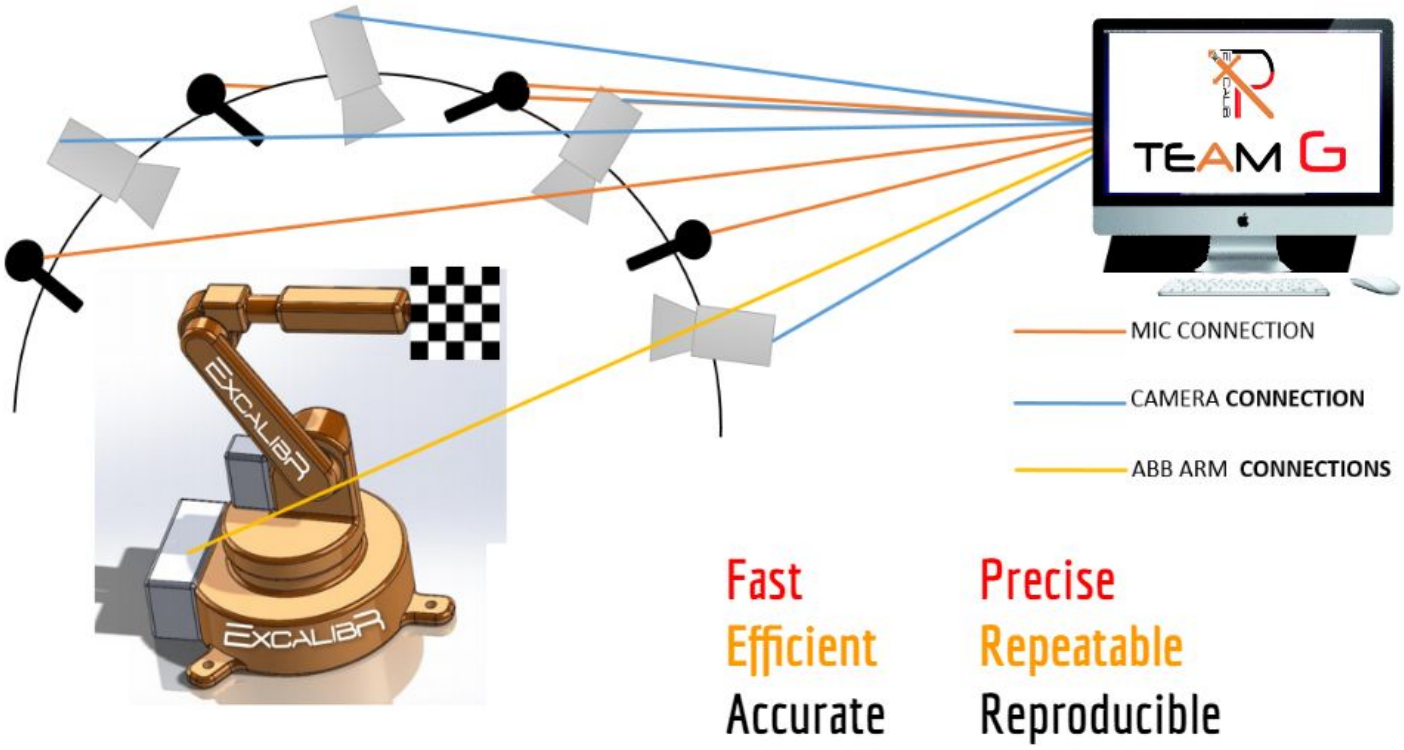

Figure 3.2 Concept of Excalibr

4. System requirements

4.1 Functional requirement

4.1.1 Functional requirements in Fall

M.F.1: Operate Autonomously
M.F.2: Fabricate calibration target
M.F.3: Control and Manipulate the calibration target by robot arm
M.F.4: Take high-resolution, stable and clear pictures of calibration target
M.F.5: Implement geometry camera calibration algorithms on RGB cameras
M.F.6: Calibrate the camera on end-effector for light-field calibration
M.F.7: Implement photometric calibration and generate camera response function curve for GRB cameras
M.F.8: Implement sensor noise correction on RGB cameras
M.F.9: Build the calibration pipeline for multiple cameras

4.1.2 Functional requirements in Spring

M.F.1: Operate Autonomously
M.F.2: Fabricate calibration target
M.F.3: Control and Manipulate the calibration target by robot arm
M.F.4: Take high-resolution, stable and clear pictures of calibration target
M.F.6: Calibrate the camera on end-effector for light-field calibration
M.F.9: Build the calibration pipeline for multi cameras

4.2 Performance requirements

4.2.1 Performance requirements in Fall

M.P.3: Manipulate the robot with 100 micrometers accuracy
M.P.4: Take pictures with multiple RGB cameras more than 10MP at 30fps
M.P.5: Complete one geometry calibration in at most 8 hours
M.P.6: The sensor noise correction algorithm must reduce the variance of the flat-field image for 90% or more.
M.P.9: The reprojection error of the geometry calibration result should be less than 1 pixel.

4.2.2 Performance requirements in Spring

M.P.1: One-click Operation
M.P.2: Fabricate the target with 50 micrometers tolerance
M.P.5: Complete one geometry calibration in at most 8 hours
M.P.7: Avoid collisions - keep a distance 0.3m away from the dome extremities and sensors
M.P.8: Build calibration pipeline for 20 GRB cameras
M.P.10: Complete light field calibration in 4 hours.

4.2.3 Desired Performance requirements

D.P.1: Build calibration pipeline for 100 RGB cameras

D.P.2: The reprojection error of the geometry calibration result should be less than 1 pixels.

4.3 Non-Functional requirements

4.3.1 Mandatory non-functional requirements

M.N.1: Complete the project by May 2017.

M.N.2: Keep budget within $5,000

M.N.3: Make the system user-friendly.

4.3.2 Desired non-functional requirements

D.N.1: Create a user-friendly GUI or physical button

5. Functional Architecture

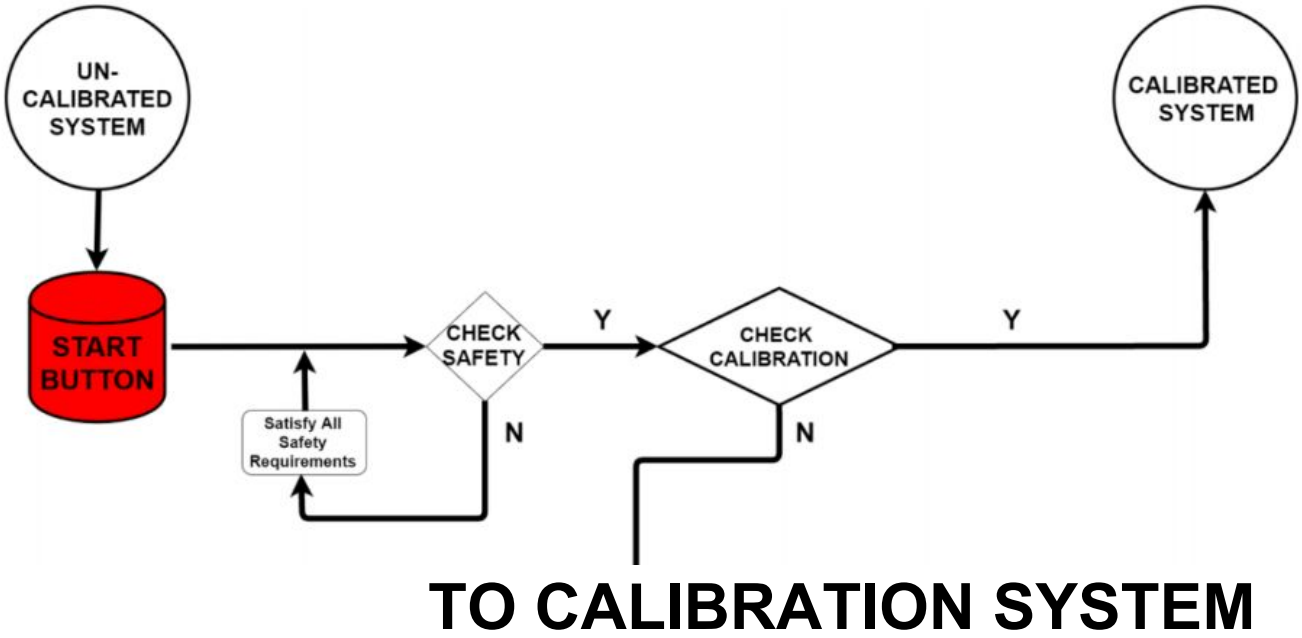

Figure 5.1 Functional Architecture- Decision Block

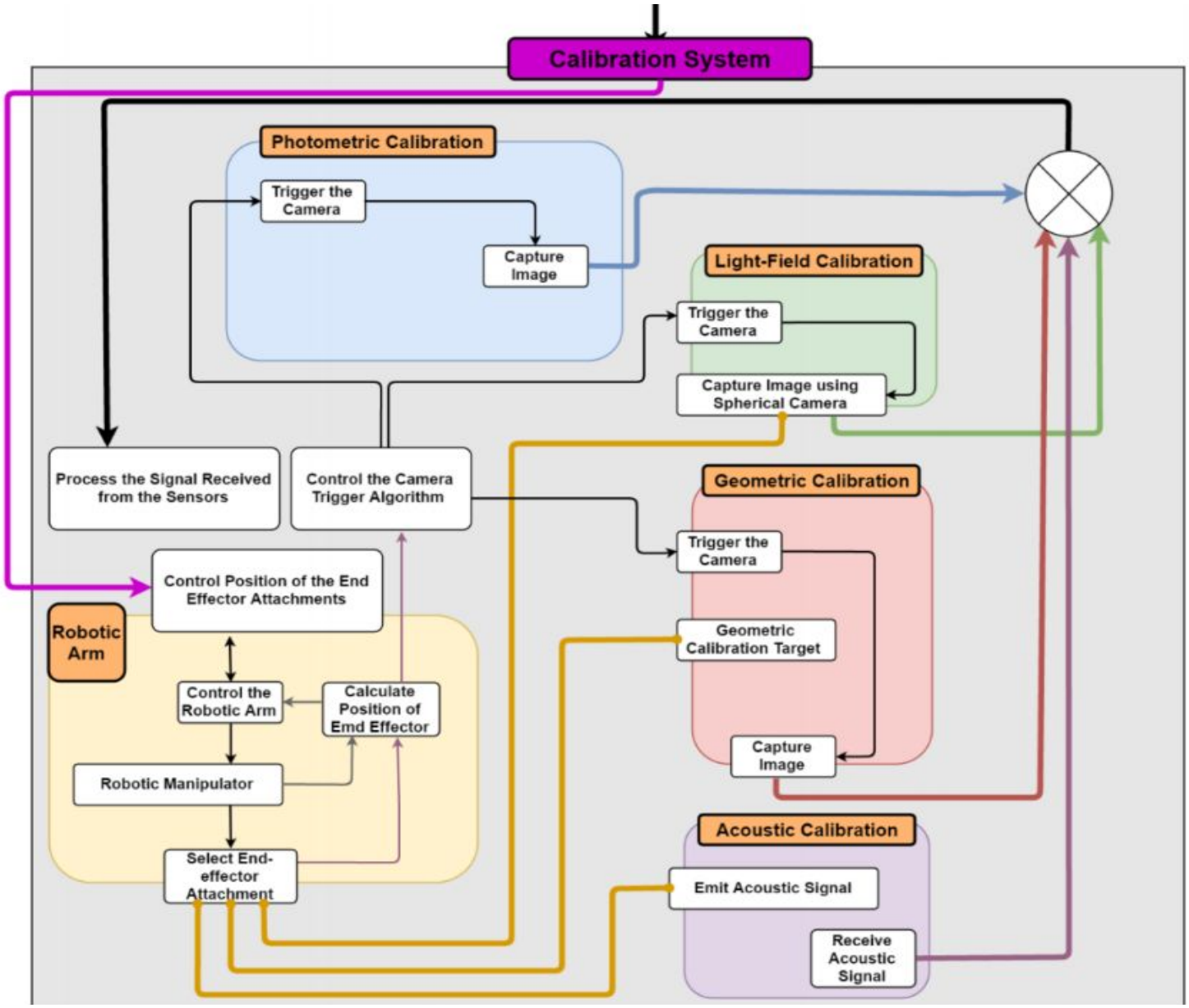

Figure 5.2 Functional Architecture - Complete system

The functional architecture (Figure 5.1 & 5.2 can be divided into the following sub-systems broadly:

5.1 Geometric Calibration

For geometric calibration, a target would be fastened to the end effector of the robotic arm. The position of the calibration target would be calculated using the position encoders mounted on the robotic arm and would be sent back to the controller (for feedback control) and the computer/ access terminal. The access terminal would trigger the cameras based on the position of the calibration target. The camera would capture the image of the target and feed it

into the calibration algorithm which would then compute the parameters required for camera calibration. The target can be programmed to move according to a particular trajectory. The target would move to a position and stop. Then a pulse would be sent to the computer which would trigger the cameras to capture images simultaneously. After the image capture, the computer would send the signal to the controller to move the robotic arm end effector to the next position. More details of geometric calibration and validation methods for geometric calibration can be found in the subsystem description.

5.2 Light-field Calibration

For light field calibration, we would determine the illumination of the desired space. The first step would involve calibration of a spherical camera mounted on the robotic arm and then this spherical camera would be used to calculate the required illumination. We would be working on light field calibration during the Spring.

5.3 Photometric Calibration

For photometric calibration, we would map the intensity of light (in lumens) to the corresponding pixel values. For this, we require the images to be corrected for the fixed pattern noise. After this correction, separate graphs for the red, blue and green channels can be obtained. Further details of photometric calibration and validation methods can be found in the subsystem description.

5.4 Acoustic Calibration

This method involves using a speaker to emit a multi-frequency variable amplitude sound signal. The microphones situated on the capture space would receive this signal, and the two signals (one emitted from the speaker and one received by the microphones) can be compared to compute the position of the microphones. This subsystem maybe descoped from our project depending on the demands from the sponsor over the winter break.

5.5 Robotic Arm

This is the corner-stone of the autonomous calibration technique. The robot arm plays a crucial role in executing trajectories in 3D space with the calibration target attached to its end effector. The calibration target target can be switch according to the respective calibration procedure being performed. The robot arm is hooked up with the cameras via a position synchronised output channel which helps us trigger the cameras at specific set points.

6. Cyber-physical Architecture

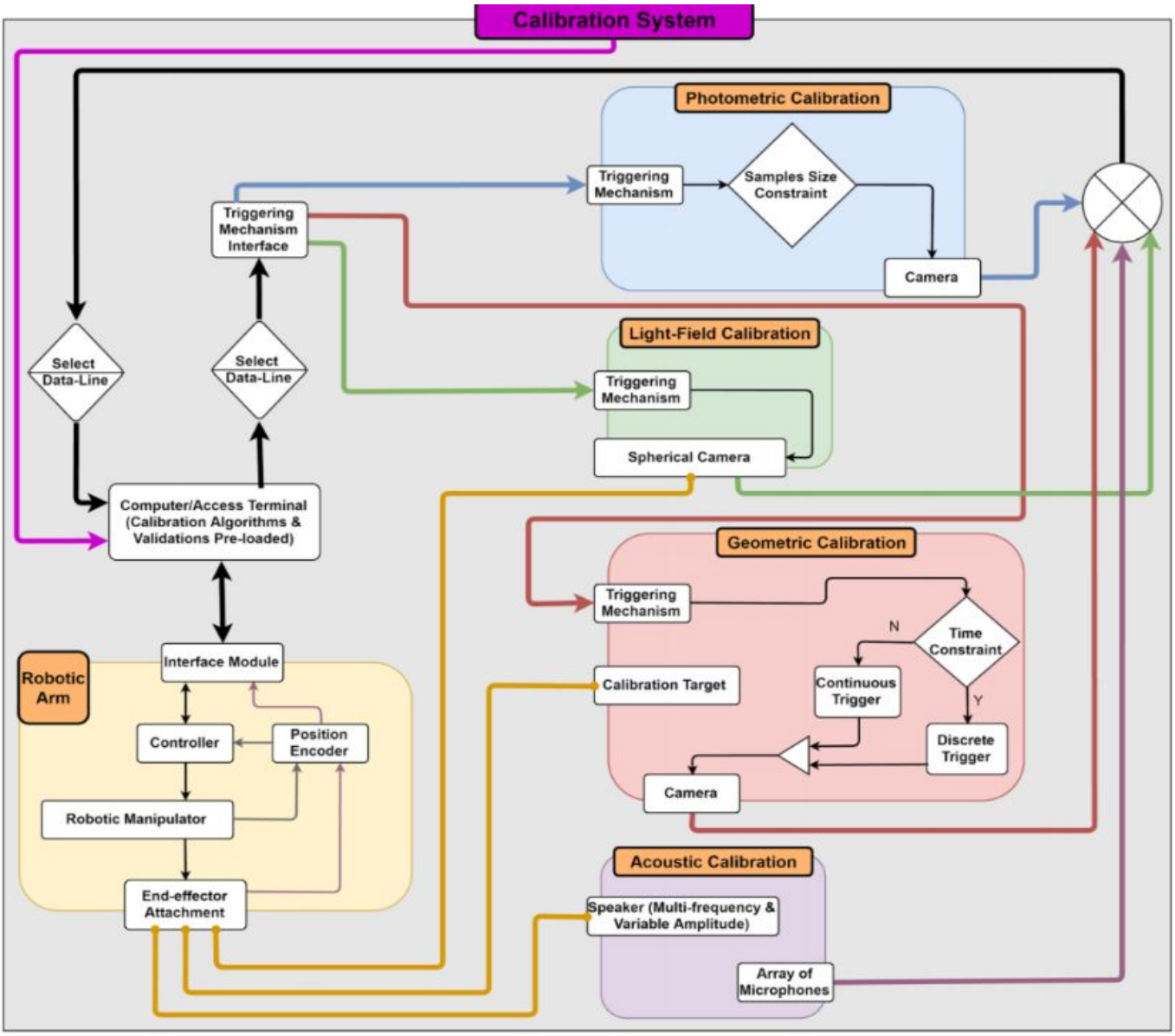

Figure 6.1 Cyber-physical Architecture - Complete system

The subsystem level implementation of the cyber-physical architecture (Figure 6) has been elaborated below.

6.1 Robotic Arm

The robotic arm is one of the major components to be used in this system. The robotic arm would have a controller and would be equipped with position sensors which would send the position of the end effector to the access terminal/computer. The robotic arm would also have end effector attachments for all types of calibration of the capture system.

6.2 Calibration Target

This target would be used for the geometric calibration. The motion of this target would be controlled by the robotic arm, which would send the position of the target back to the computer to trigger the cameras to capture images.

6.3 Spherical Camera

A spherical camera would be used to compute the illumination of the entire capture system. This camera needs to be calibrated first and then used to capture images which would help in calculating the illumination of the capture system.

6.4 Speaker

A speaker would emit multi frequency and variable amplitude signals, which would be received by a 3 -directional differential and pressure microphone. The difference from these two signals can be used to calibrate the microphones. This part maybe descoped from our project.

6.5 Common Access Terminal

All the signals would be received by the access terminal which would compute the calibration parameters for the various components of the system.

7. Current System Status

For our current system, we have the Aerotech Robot Arm fixed on the ground with the 2D calibration target mounted on it. We are now able to connect 2 RGB cameras to the Aerotech controller and trigger them at the same time. The images will be stored real-time in the computer and once the image capturing process is done, the computer can execute the calibration process and generate the camera parameters. Our current system dependency is shown in Figure 7.1.

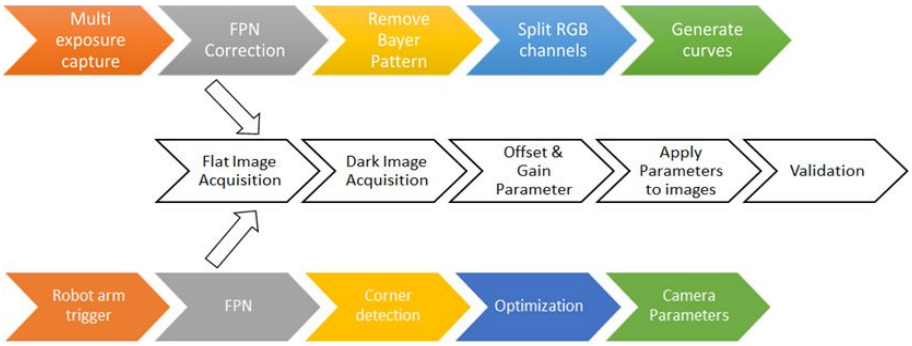

Figure 7.1 Subsystem level dependency

Our current system is consist of two main pipeline, the photometric pipeline and the geometric pipeline. The sensor noise (FPN) Correction is a subsystem which is required in both pipelines.

The photometric calibration should be done only once for each camera, and the geometric calibration should be done each time before the system operates.

The sensor noise correction is the dependency of both pipelines, we are now done with this procedure and also the validation part so that we can build the photometric pipeline and geometric pipeline based on the removal of FPN.

7.1 Fall-semester targeted system requirements

Table 7.1 clearly illustrated the targeted system requirements.

Table 7.1 Fall Target System Requirements

Subsystem	Functional requirements	Performance requirements
Calibration target	M.F.2: Fabricate calibration target	
Aerotech robot arm	M.F.1: Operate Autonomously M.F.3: Control and Manipulate the calibration target by robot arm	M.P.3: Manipulate the robot with 100 micrometers accuracy
Photometric calibration	M.F.7: Implement photometric calibration and generate camera response function curve for GRB cameras M.F.8: Implement sensor noise correction on RGB cameras	M.P.4: Take pictures with multiple RGB cameras more than 10MP at 30fps M.P.6: The sensor noise correction algorithm must reduce the variance of the flat-field image for 90% or more.
Geometric calibration	M.F.4: Take high-resolution, stable and clear pictures of calibration target M.F.5: Implement geometry camera calibration algorithms on RGB cameras M.F.9: Build the calibration pipeline for multi cameras	M.P.4: Take pictures with multiple RGB cameras more than 10MP at 30fps M.P.5: Complete one geometry calibration in at most 8 hours M.P.9: The reprojection error of the geometry calibration result should be less than 1 pixel.

For the calibration target, we are supposed to fabricate both 2D and 3D precise calibration target and mount the checkerboard and april tag pattern on that.

For the aerotech robot arm, we should be able to generate the intended trajectory by matlab, and be capable of controlling and manipulating the robot arm and the calibration target mounted on it with the accuracy of within 100 micrometers. The robot arm is supposed to operate autonomously, pause at certain positions and trigger the connected cameras to take images of the calibration target.

For the photometric calibration pipeline, we should be able to capture multi-exposure images and implement the calibration process including sensor noise correction, bayer pattern

removal, RGB channel splitting and response function curve generation for each channel. The sensor noise correction must reduce the variance of the flat-field image for more than 90%.

For geometric calibration pipeline, we are supposed to take high-resolution, stable, and clear pictures of the calibration target more than 10MP at 30fps, then the computer would execute the geometric calibration algorithm automatically and the whole process for more than 100 cameras should be able to complete in less than 8 hours. The reprojection error would be used to evaluate the calibration result and the targeted requirement is less than 1 pixels.

7.2 Subsystem - Aerotech robot arm

7.2.1 Aerotech robot arm

We made a trajectory code generator for Aerotech programming environment implemented in MATLAB which enables us to just enter the trajectory type and required parameters and it will generate a code which can be directly run by Motion Composer Suite of Aerotech. We could also use the multiple position synchronized outputs in the Aerotech robot to trigger the camera. Finally we could easily to generate the different trajectory we need and triggered the camera to get images at where we need(Figure 7.2).

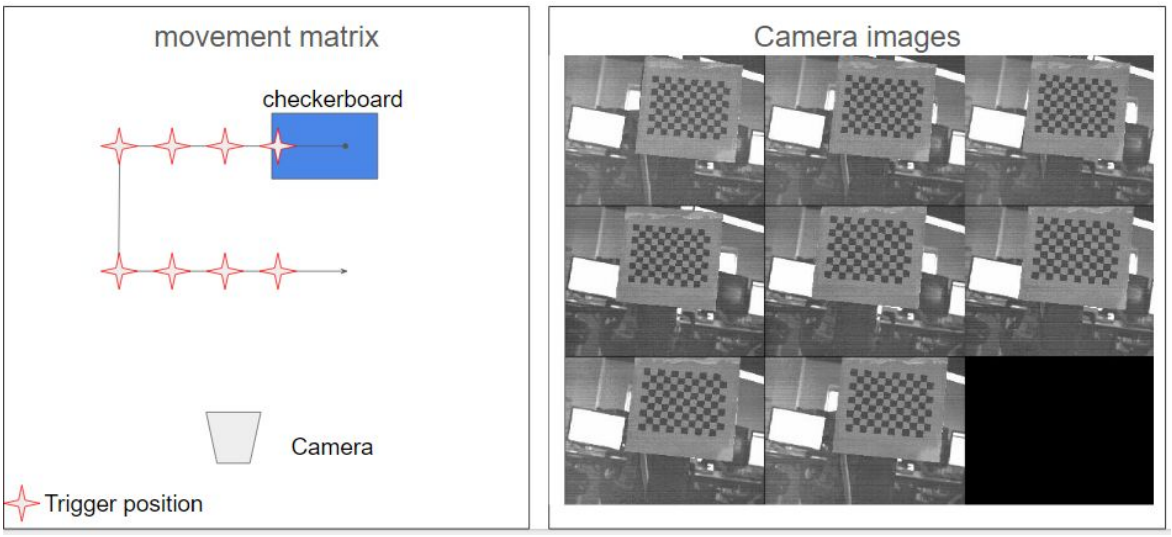

Figure 7.2 Movement matrix and corresponding images

7.2.2 Multiple axis PSO triggering

The PSO can be programmed to generate an output synchronized to the encoder position, typically used to fire a laser or sequence an external device (here, camera). Trigger signals may be derived from the standard encoder channel, auxiliary encoder channel, or a software trigger. The synchronized output pulse is generated using high-speed hardware, allowing minimal latency (200 nanoseconds) between the trigger condition and the output.

The algorithm used for triple axis tracking is stated below:

1. Enable and home the axes.
2. Reset the PSO. This resets the internal state of the PSO hardware, including the distance tracking counters and the window counters.
3. Configure the PSO hardware to send the PSO signal out the auxiliary marker. If using the dedicated PSO output, this command should not be used.
4. Configure the PSO for three axis tracking using the standard encoder input, SSI Net Port1 and SSI Net Port 2. This configuration is only valid when using the Ndrive HPe or HLe.
5. Configure the PSO to generate a firing event at a fixed distance.
6. Configure the pulse generator by setting the total and off times and the number of cycles.
7. Specify the output of the PSO to come from the pulse generator.
8. Arm the PSO to start the distance tracking.
9. Move the axis so as to cause the PSO to fire.
10. Delay slightly to allow the PSO firing to finish.
11. Disable the PSO pulse generator.

7.2.3 Velocity Profiling

Velocity profiling is nothing but ensuring that the velocity transitions between successive positions of the linear actuator is smooth and without any jerks or sudden motions which would be better for removing the motion blur by jerks.

The concept of velocity profiling has been shown in Figure 7.3 and Figure 7.4, which shows the difference between sudden velocity drops when there is a transition. This is overcome by velocity profiling.

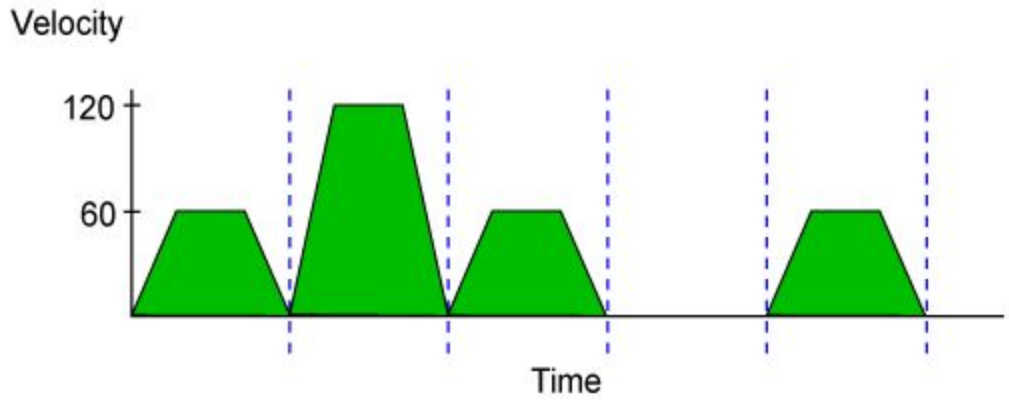

Figure 7.3 Without velocity profiling

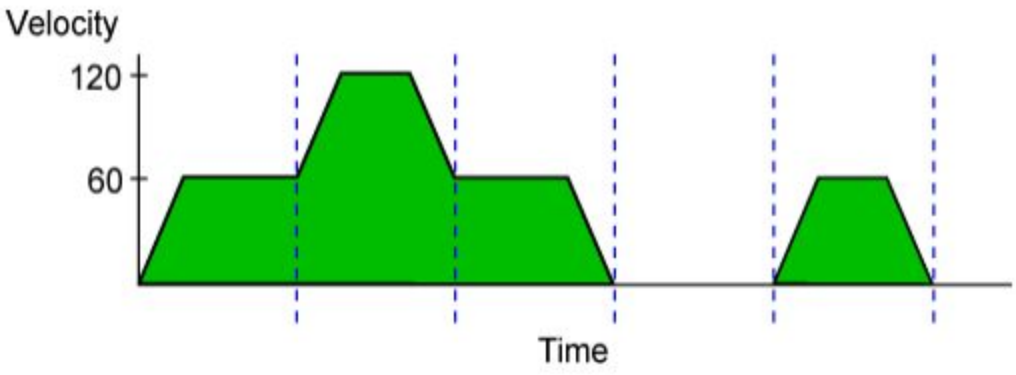

Figure 7.4 With velocity profiling

7.3 Fixed Pattern Noise (FPN)

7.3.1 Description

Since our project aims for highest calibration accuracy, in addition to general geometric calibration, we begin with addressing more rudimentary problems caused by sensors or lens of the camera. Fixed pattern noise is composed of dark current and pixel response non-uniformity (PRNU). Dark current results from electric current in CMOS sensors. Theoretically, if we provide no light to a camera, all pixel values in the image it takes should be zero. However, because of the dark current, even the camera is covered in darkness, we can not get all-zero images from it. If we scale a dark image it takes, we will see stripes in the image.

PRNU is, as its name indicates, is because of that each pixel responses differently to the same light input. Furthermore, since the camera has lens, vignetting also becomes a problem. Let a camera take images of a surface with the same brightness. We can see that the result flat-field image does not have uniform brightness.

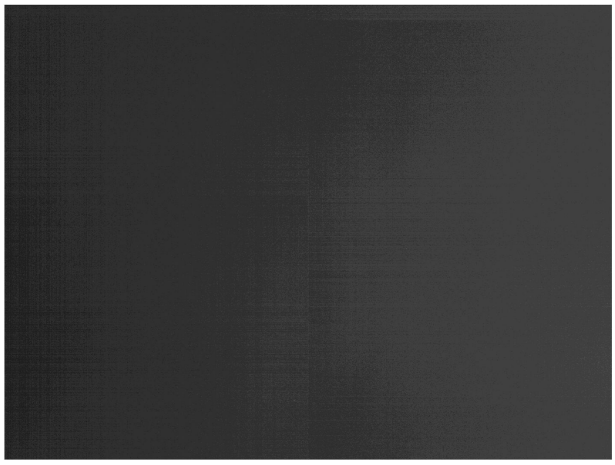

Figure 7.5.1 A dark image

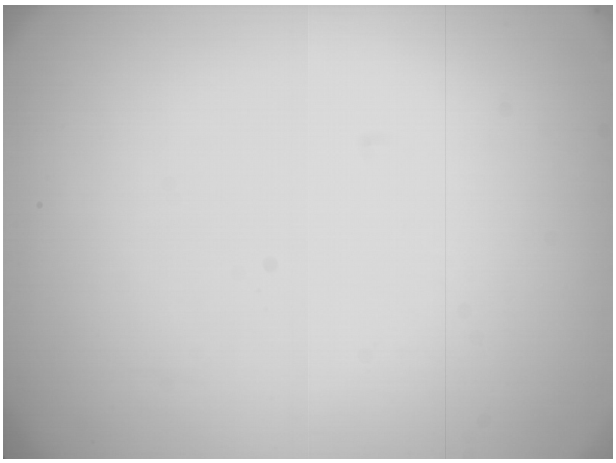

Figure 7.5.2 A flat-field image

Dark current and PRNU are also called offset and gain. Offset parameters can be obtained simply by taking images of darkness(Figure 7.5.1). The gain parameters can be obtained from dividing the intensity true value by the pixel value. Since we want to make every pixel responses equally to the same brightness, we can set the median of flat-field image(Figure 7.5.2) pixel values without offset as our true value.

Our hardware setup composites a light box(Figure 7.6), a camera and a lambertian surface. The lambertian surface is fixated inside the box. The camera can be put outside of the box hole or fixated in the box as well. The lambertian surface ensures the same brightness. Although the formulas only uses one dark image and one flat-field image, in practical

Figure 7.6 Light box setup

The system pipeline is shown in Figure 7.7,

Figure 7.7 System pipeline

Our system can automatically acquire images, calculate parameters and correct images by these parameters. Then, we validate the result images with variances and histograms.

7.3.2 Validating Results

The first test is flat-field image test. We have promised in our FVE performance requirement that we can reduce the variance of the flat-field image by 90% or more. The testing image here was taken with a different exposure from the exposure used when calculating the parameters. This can prove that even without the same exposure, we can still get a reliable result with almost 97% improvement. The results are shown in Figure 7.8.

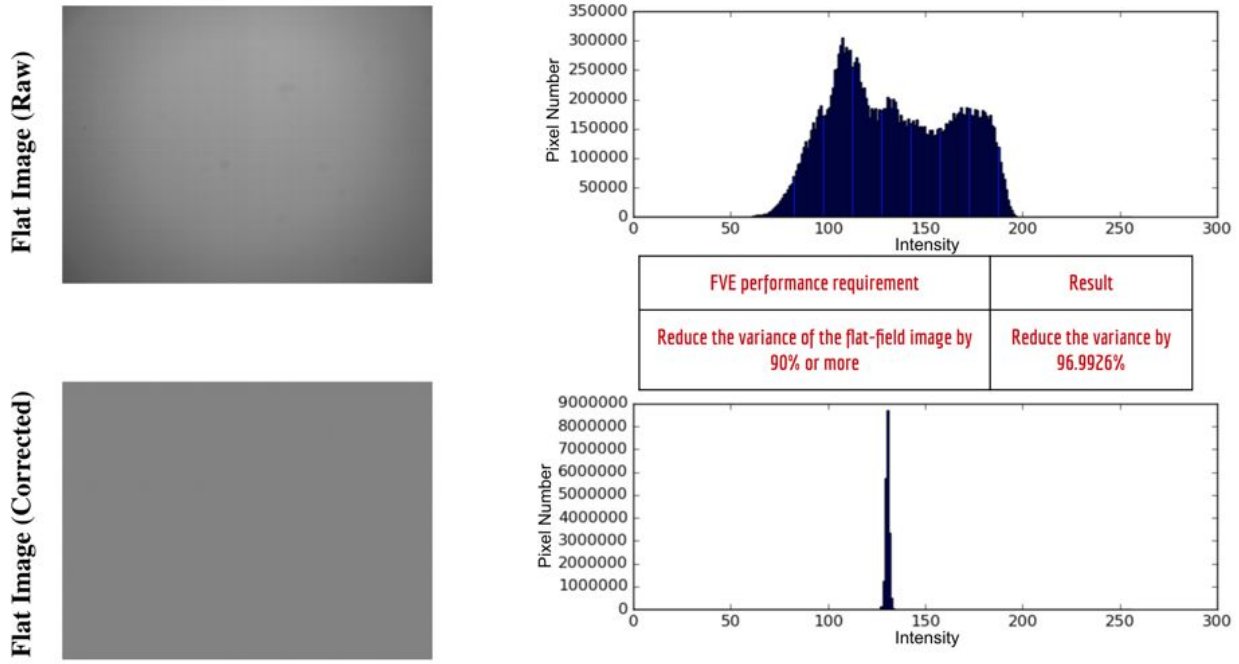

Figure 7.8 Flat-field image result

The second test is dark image test (Figure 7.9). Because of the dark current, dark image pixel values are not all zeros. After our correction, most of the pixel values become zero.

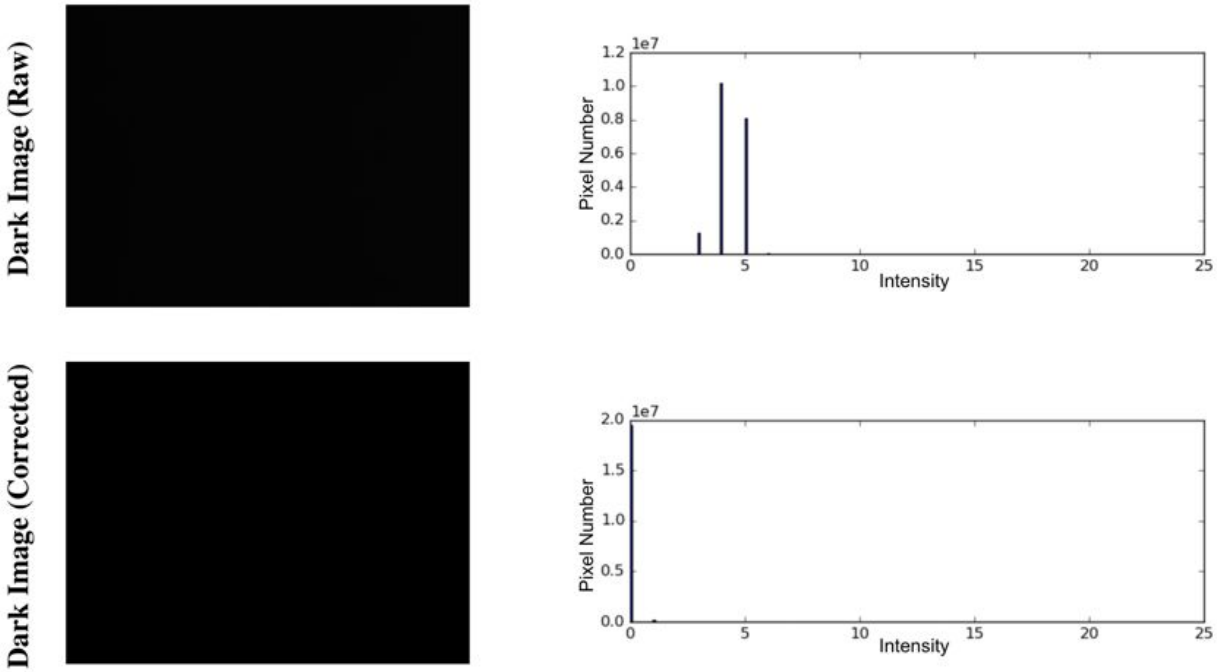

Figure 7.9 Dark image result

For the third test, we took pictures of a color board (figure 7.10). We can see there are a lot of noises in the raw image histogram. After our correction, the pixel values concentrate in specific intensity values according to the color we have in the image(Figure 7.11).

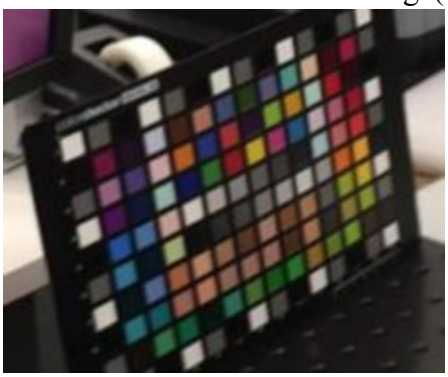

Figure 7.10 A color board

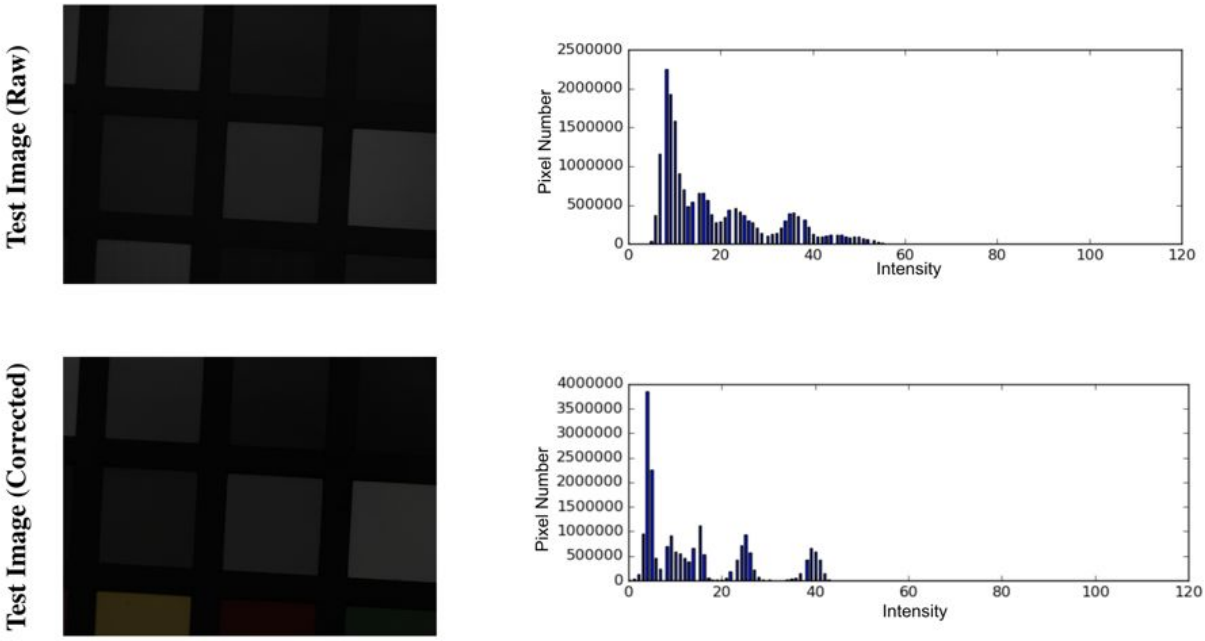

Figure 7.11 Test image result

7.4 Subsystem - Photometric Calibration

The pixel value is a function of the product of the radiance and the exposure time, which describe the relationship between the true pixel value and the pixel value we get through cameras. Photometric calibration is aimed at generating the camera inverse response function curve in order to implement the color correction of images.

7.4.1 Subsystem description

Figure 7.12 General process of photometric calibration

The general process of photometric calibration is shown in Figure 7.12. First we use the software trigger to take multi exposure images and implement sensor noise correction on all the images. For images taken by RGB cameras, we need to remove the bayer pattern and then split the RGB images into three channels, and each channel is a grayscale image. After that, we can execute the photometric algorithm and generate the curve for each channel.

Figure 7.13 shows the setup of multi exposure image capturing, captured images will be displayed on the computer screen real time, and the whole process is automatically triggered by computer.

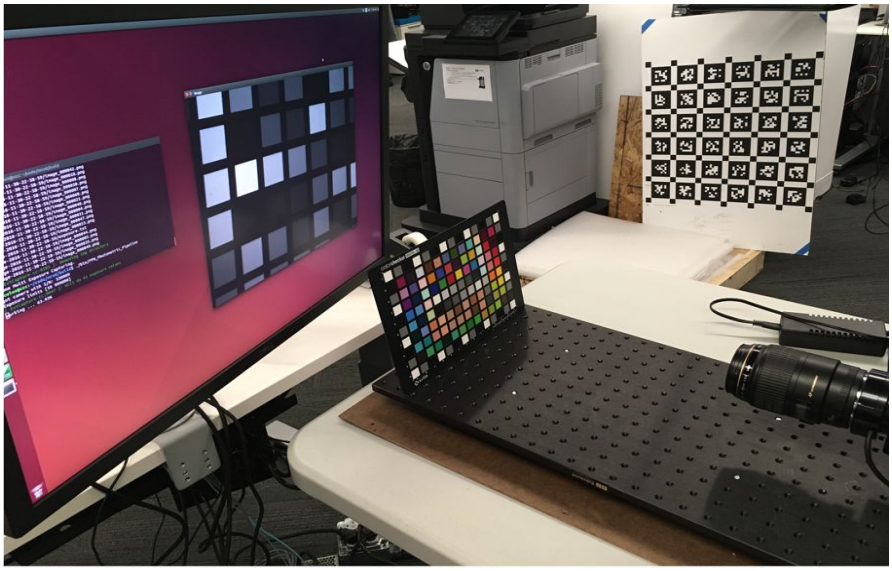

Figure 7.13 Multi exposure image capturing setting

Figure 7.14 and Figure 7.15 shows the result of bayer pattern processing and the general process of captured images in photometric calibration.

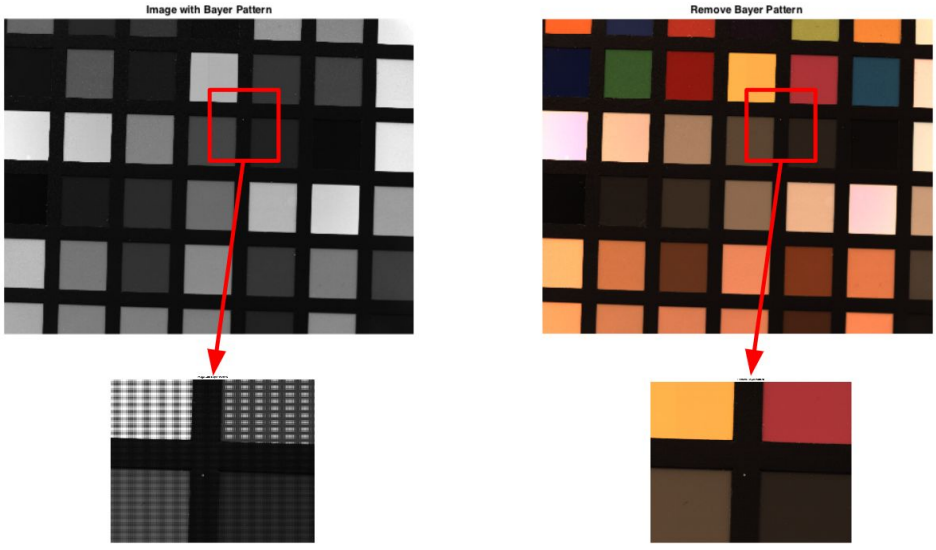

Figure 7.14 RGB image with bayer pattern (left) and without bayer pattern (right)

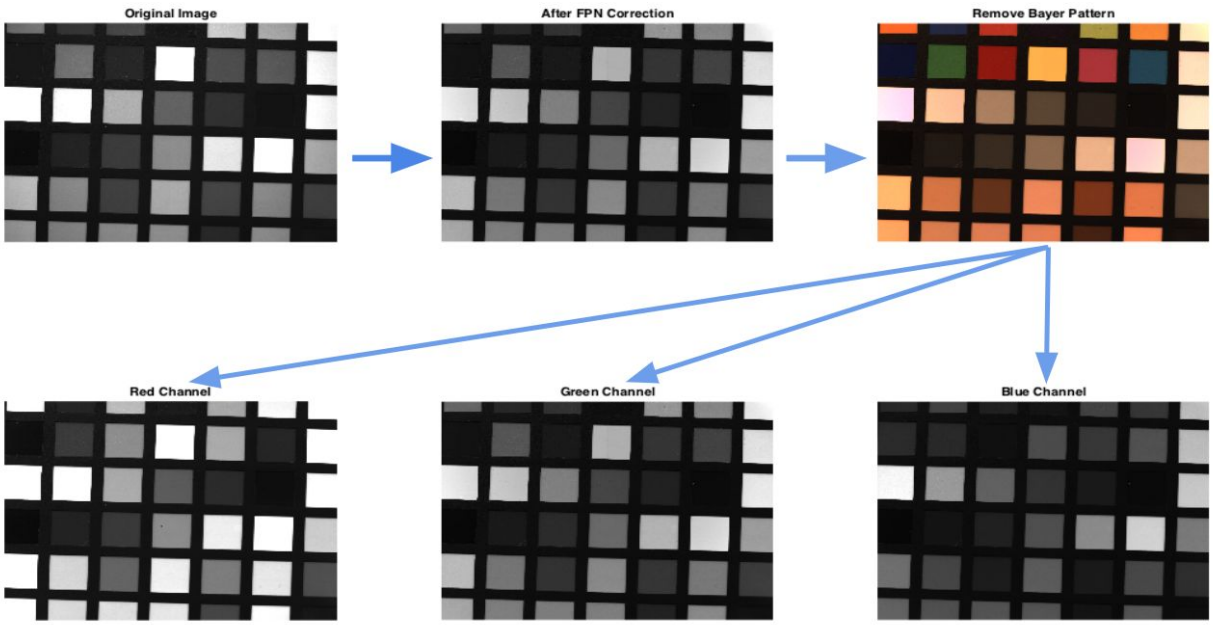

Figure 7.15 Image processing process of photometric calibration

7.4.2 Results

Figure 7.16 shows the result of the photometric calibration. The colored curves each represents a specific channel. The blue and the red channel curve is a line of slope 1, which means the pixel value we get is exactly as the true value. For the green channel, we would implement a one-to-one mapping on the pixel value we get in order to correct the true color of images. However, we still need ground truth to validate whether the curve is as accurate as we expected, and we are considering buying a device which is able to measure or control the true intensity of irradiance to provide validation support for the resulted curves.

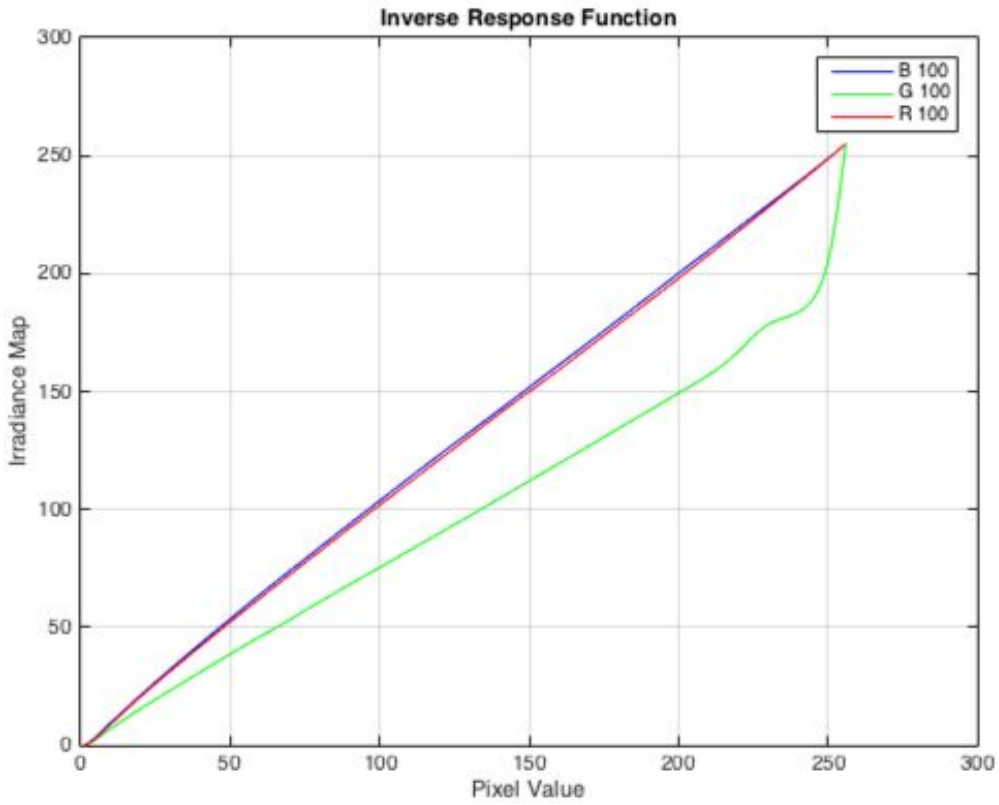

Figure 7.16 Camera inverse response function curve for RGB channels

7.5 Subsystem - Geometric calibration

Geometric calibration was using the sequence of images about the calibration target to get the camera parameters including intrinsic and extrinsic parameters. To evaluate the result of the camera calibration, the reprojection error was introduced.

7.5.1 Calibration target

In the first stage, we planned to create a quick framework to test each subsystem. In calibration target part, we choose using the 3D printer to make a prototype and using the checkerboard printed by Xerox printer. In the processing of design, we choose 3D target because we can get the multi-face checkerboard information at a single image, and if the faces' geometric relationships were known, we can use that information to do camera calibration.

We could take that the 3D target(Figure 7.17) would be the assembling of multi 2D checkerboards, Therefore, the angle between the face wouldn't be too large to reduce the distortion. The Rhombicuboctahedron shape has 45 degrees in each square which could get a better image than cubic shape(90 degrees). For deciding the size of the target, because our future application was doing human face reconstruction, our camera's image field of view was almost as same as the human's face.

However, due to the consideration of the code maturity problem from the sponsor and we want to make a minimum viable product first, we decided to use 2D calibration target (Figure 7.17) rather than 3D checkerboard for this semester.

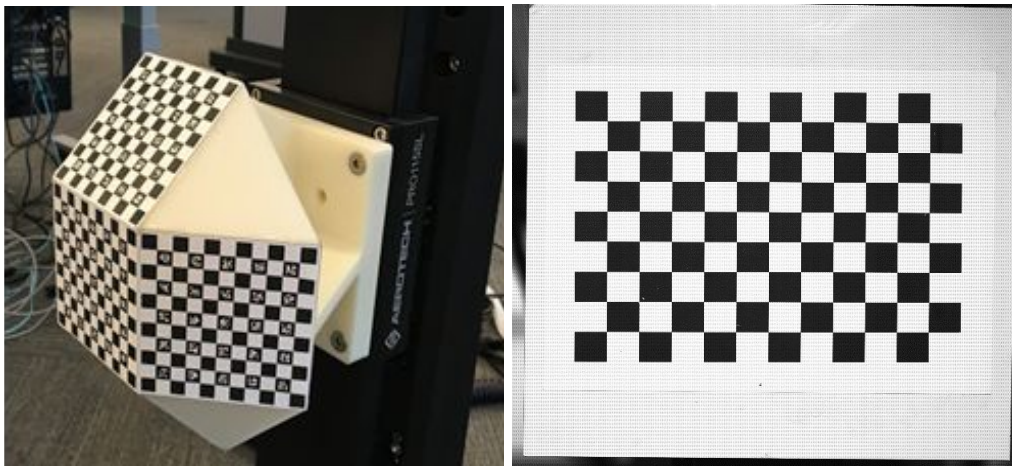

Figure 7.17 3D calibration target and 2D calibration target

7.5.2 Calibration algorithm

We first integrated the geometric calibration pipeline basing on the OpenCV library.

Basing on the structure of calibration pipeline we designed, we combined the FPN in the geometric pipeline (Figure 7.18). we first read the raw image and using the FPN parameter to reduce the dark noise to get the correct images. After we finished the image processed, we used that filtered image to detect corner and conduct the original camera calibration.

Figure 7.18 Combined FPN in geometric calibration pipeline

In the corner detection, we first detected the quadrangle's corners and used the subpixel refinement to locate the corner in subpixel level (Figure 7.19).

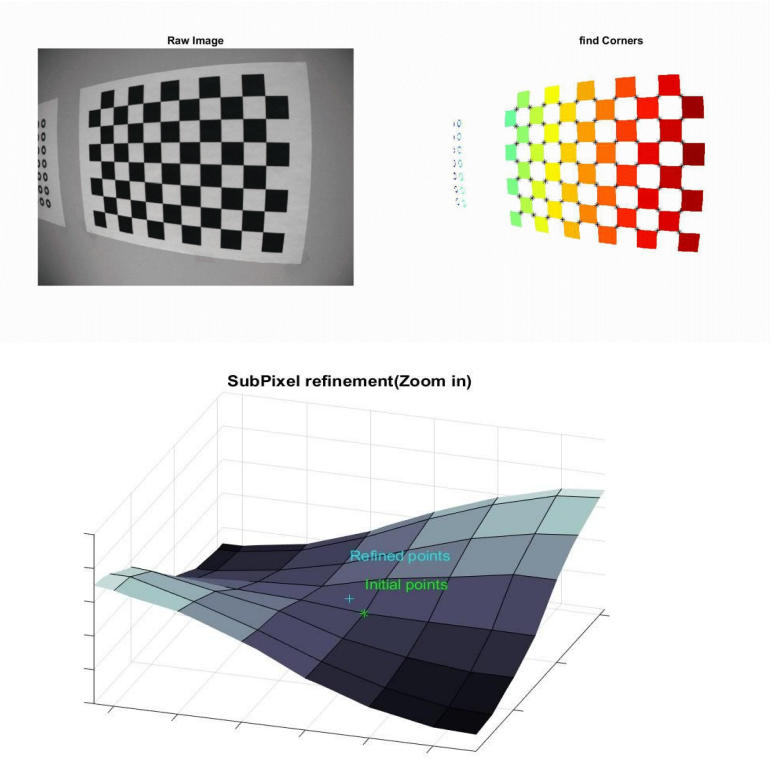

Figure 7.19 Corner detection and Subpixel refinement

7.5.3 Geometric calibration result

In validation of geometric calibration result, we used a different number of images to conduct the camera calibration to validate the process of camera calibration. The results(Figure 7.20) showed that the overall reprojection error was less than 0.4 pixels which satisfied the requirement we set. However, the trend in the reprojection was not expected by us. Because when the number of images increased, the gaussian noises in the corner detection should be eliminated. The reprojection error should reduce. In our experiments, the reprojection error was about 0.39 and there is no obvious trends to reduce the reprojection error with the number of images. In order to understand the reason why the reprojection error was keeping same. We investigated the distribution of the reprojection error with the different corners in the checkerboard(Figure 7.21). We checked the images of checkerboard in the different position and found out that the reprojection error around the corner was higher than the center of the checkerboard. We think it was because of the defect in the checkerboard which might because of bending in the surface or the imperfect

square size.

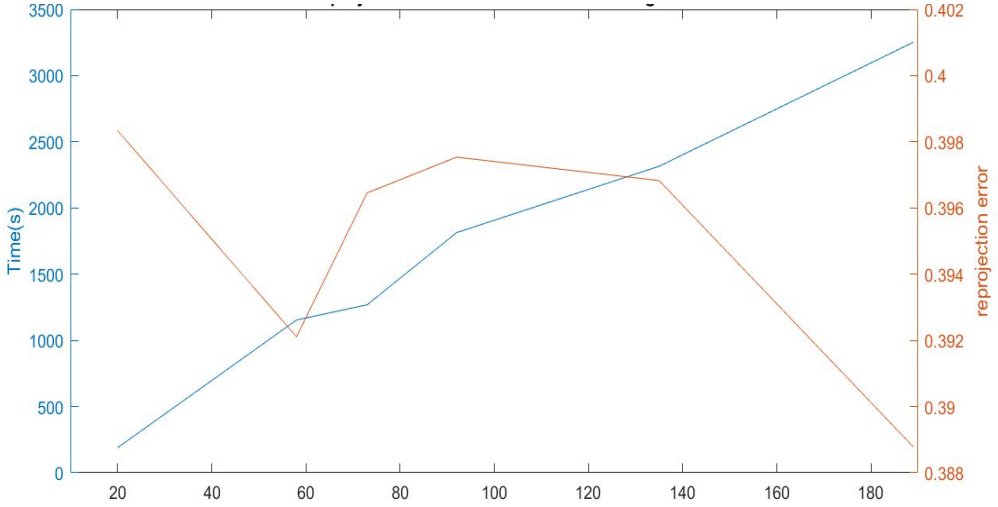

Figure 7.20 Reprojection error and Time cost with different number of images

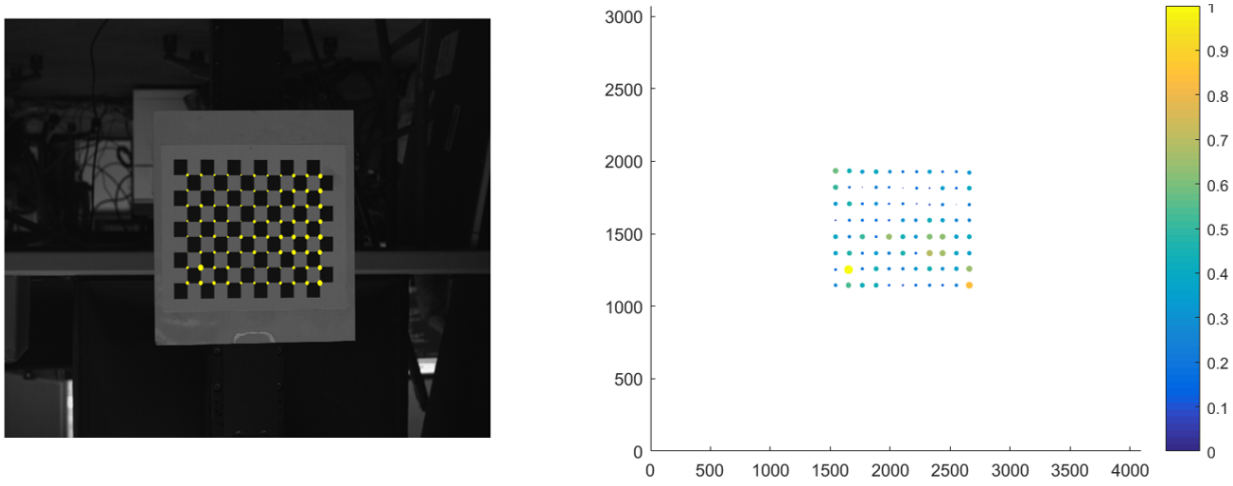

Figure 7.21 Reprojection error on the checkerboard in the field of view

7.5.3 Time cost

In the same experiments, we also recorded the time cost in the single camera calibration. Using that information, we could estimate the time cost for 140 cameras in the Dome. We just simply multiple the number of cameras and the time cost with the single camera and divide by the 8 hours. In our estimation, we assume that the computer has the same processing speed as our current machine. We found out that for 144 images we will need 20 computer power to complete the 140 cameras calibration within 8 hours. For spring, we thought one solution is using servers instead of a desktop computer. The other way is trying to speed up the algorithm. In this winter break, we will also investigate the possible method to improve the efficiency of the algorithm.

7.6 Conclusion

Strengths: We are now able be reduce the variance of flat-field images for 96.7% after sensor noise correction, the reprojection error after geometric calibration is 0.4 pixels and the robot arm positional accuracy is 10 micrometers.

Weakness: We are also able to generate the camera inverse response function curve as the result for photometric calibration, but we are still working on the validation part as the device which can provide ground truth is not ready yet. The efficiency of the geometry calibration is also a problem as the algorithm is a bit too slow to meet the time requirement. The calibration target is also not precise enough due to the manufacturing defect.

8. Project Management

8.1 Work Breakdown Structure

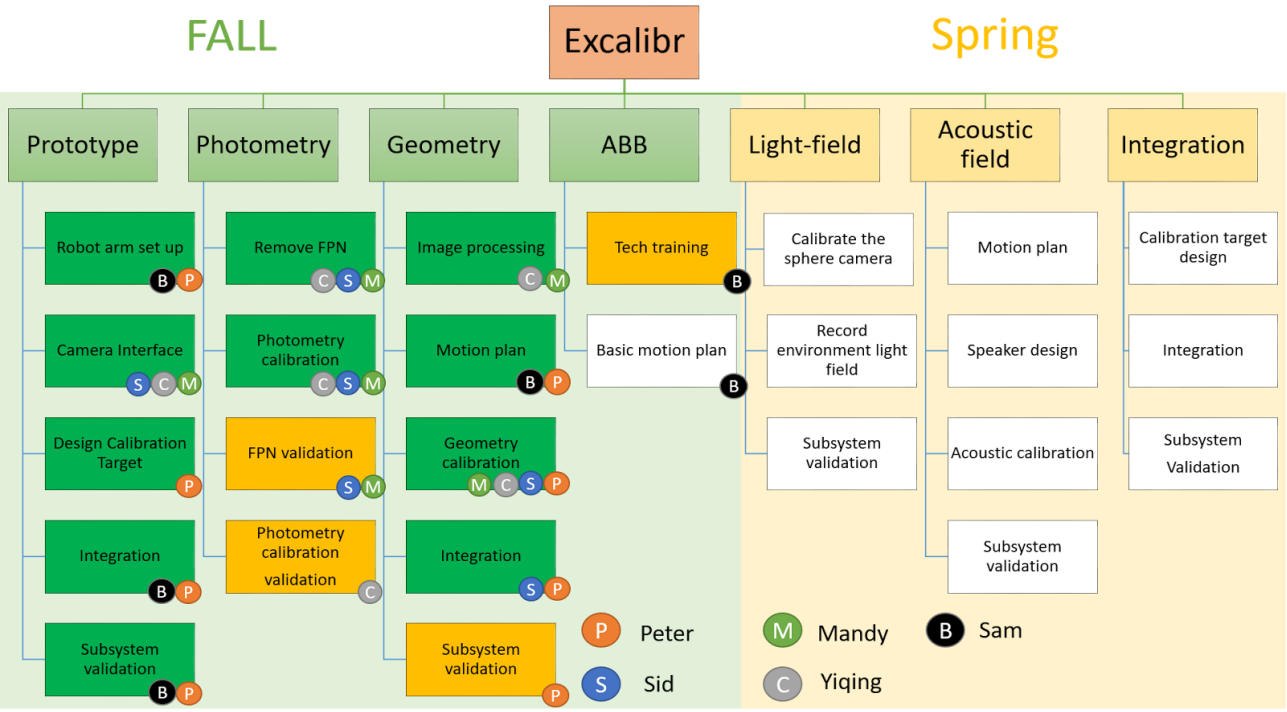

Figure 8.1 Work Breakdown Structure

Our project can be divided into seven parts: prototype, photometric calibration, geometric calibration, robot arm manipulation, light-field calibration, acoustic field and integration. We

have finished the first four parts in this semester. The light-field calibration, acoustic field calibration and integration will be addressed in the next semester. The green color indicates tasks that we have completed; yellow indicates what is in progress and white indicates what has not yet started.

Since the dome is still under construction, we are now building a prototype with an Aerotech robot arm, two cameras prototype in the Oculus office. Sid, Cece and Mandy are responsible for image processing, including camera setup, image acquisition, sensor noise calibration and photometric calibration. Peter is responsible for calibration target design and geometric calibration. Sam is responsible for hardware setup and robot arm control. Although the ABB robot arm has not been available to us during fall semester, Sam is working on document study for future use in spring semester. The work breakdown structure has been shown in Figure 8.1.

8.2 Schedule

Figure 8.2.1 shows the schedule overview of the spring semester, Table 8.2.2 shows the biweekly schedule

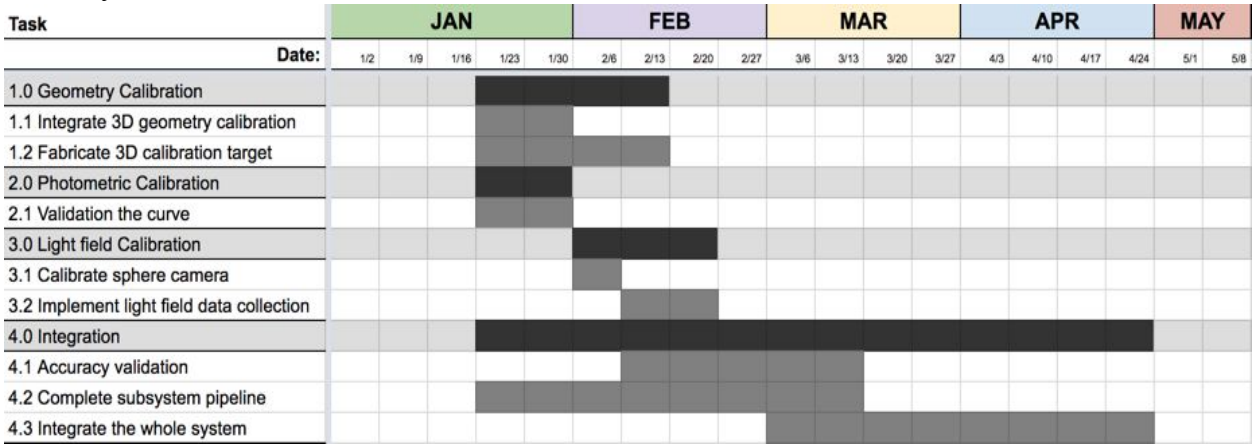

Figure 8.2.1 Spring semester schedule

Table 8.2.2 biweekly schedule of major system development milestones

Date	Biweekly Schedule
1.18 – 1.31	Integrate 3D geometry calibration algorithm into geometry pipeline
2.01 – 2.14	Fabricate precise 3D calibration target and validate photometric calibration results
2.15 – 2.28	Calibrate sphere camera for light-field data collection
3.01 – 3.14	Implement light-filed calibration and build the pipeline
3.15 – 3.28	Accuracy validation of each subsystem
3.29 – 4.11	Complete integration of all the subsystems
4.12 – 4.25	Complete integration of the whole system
4.26 – 5.09	Testing and revising functions and performances of the whole system

The major system development milestones (shown in Table 8.2.2. & 8.3.1) includes geometry calibration for 3D calibration target, the validation for the photometric calibration, the light-field calibration and the integration of the whole system.

We are now a bit behind our fall semester schedule, as we need to change our calibration target from 2D to 3D, and we are not able to validate the photometric calibration result due to lack of device which could provide ground truth data. We might descope our final goal and try to focus on improving the accuracy and efficiency of our current system.

8.3 Test Plan

Table 8.3.1 Milestones for spring-semester progress reviews

Date	Capability Milestones
PR 7	Integrate 3D geometry calibration algorithm & Photometric calibration validation
PR 8	Fabricate 3D calibration target
PR 9	Implement light field data collection
PR 10	Accuracy validation
PR 11	Complete the integration of the subsystem pipeline
PR 12	Integrate the whole system and apply to more than 20 RGB cameras

Table 8.3.2 Spring Validation Experiment

Location: Oculus Research, indoors place within a 7ft x 7ft x 5ft space.	**Equipment:** ABB Robotic Arm, A3200 N drive controllers, 3D Calibration target, Computer Terminal, RGB EVT Cameras.

Test Steps:

Step 1 - Light field calibration

1. Implement calibration process on the sphere camera.
2. Mount the sphere camera on the robot arm and move it according to a pre-designated trajectory to collect illumination data of the environment.
3. Show the processing time and light field data.

Step 2 - Geometry calibration for more than 20 cameras

1. Move the 3D calibration target according to a pre-designated trajectory and set points for triggering more than 20 RGB cameras and store the images.
2. Remove the FPN of the images and apply the response function to the images.
3. Use the filtered image for geometric calibration of the cameras.

4. Show the processing time and the camera calibration result.

Performance Metrics:

1. Light field Calibration: Successfully generate the light field data.
2. Geometry Calibration: Reprojection error is less than 0.5 pixel & complete the whole process within 12 hours.

8.4 Budget

Table 8.4 Budget List

MRSD Total Budget: (USD)5000

Budget List						
#	Item	Quantity	Unit(s)	Cost per unit (USD)	Cost (USD)	Purchaser
1	AEROTECH PRO225SL	1	set	20,000	20,000	Oculus
2	AEROTECH PRO115SL	2	set	15,000	30,000	Oculus
3	AEROTECH A3200 Controller	3	set	1,000	3000	Oculus
4	Emergent:HR-12000 with lens	3	set	5,400	16,200	Oculus
5	Desktop PC	2	set	1,500	3,000	Oculus
6	3D printing material(PLA)	1.5	kg	47.47	71.2	Oculus
7	Cable carriers	12	ft	19.63	236	Oculus
				Total cost (USD)	**72506.8**	
				Amount spent from TeamG budget (USD)	**0**	

The budget list of our project is shown in Table 8.4. We have 5000 dollars budget in total, and the big-ticket items includes Aerotech controllers, Robot arm and Emergent cameras. For now, our sponsor Oculus is paying for all the items so our total cost is 0.

8.5 Risk Management

We list out 6 major risks in our project this semester:

1. Robot arm malfunction
2. Camera malfunction
3. Integration failure
4. PSO trigger problem
5. Memory deficiency
6. Validation Difficulty

To assess the risks during the life of our project, we develop the rubrics that define our risk likelihood and consequences as shown in Table 8.5 and Table 8.6.

Table 8.5 Risk likelihood legend

1	Highly unlikely
2	Possible to occur
3	Likely to occur
4	Expected to occur
5	Estimated to occur

Huge time delay may also results in the failure in satisfying our requirements.

Table 8.6 Risk consequences legend

1	Time delay < 3 days
2	3 days <Time delay < 1 week
3	1 week <Time delay < 2 weeks
4	2 weeks < Time delay < 1 month
5	Time delay > a month

Table 8.7 Total risk level definition

Low risks	1-5	Acceptable risks
Medium risks	6-10	Risk tracking is required
High risks	10+	Mitigation strategies are mandatory

Through this semester, we mitigate three risks. Red numbers in the risk matrix (shown in Figure 8.10) indicate the original risk statuses; white indicates the new risk statuses and the black numbers are risks remained same status.

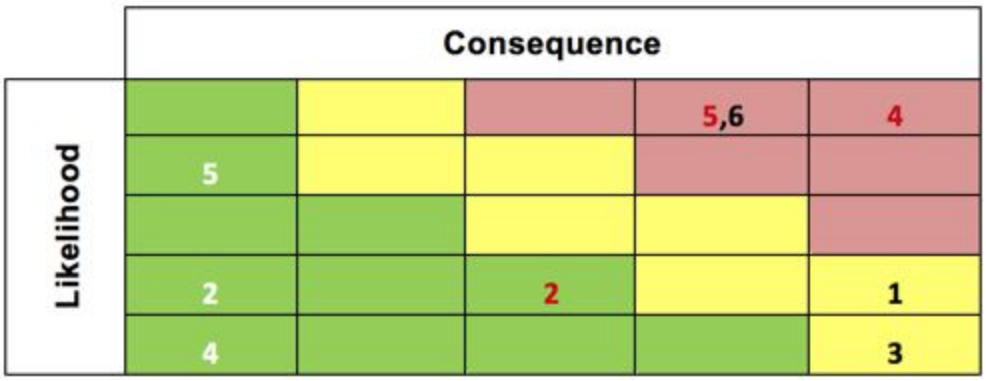

Figure 8.10 Risk matrix

In the risk mitigation shown in Table 8.11, H means hardware and S represents as software.

Table 8.11. Risk mitigation

Type	Description	Mitigation	Likelihood change	Severity change
H/S	Unable to use cameras to take pictures	We have prepared spare cameras in case any camera breaks	2 (0)	3 → 1
H/S	Unable to Trigger camera to collect data when the robot arm moves along X , Y or Z axis	We have found out the root cause of our PSO trigger problem. Therefore, we can prevent it from happening in the future.	5 → 1	5 → 1
S	Memory deficiency problem	Downsample the input images and increase hardware equipment to address this problem in the future	5 → 4	4 → 1

9. Conclusion

Take-away from Fall 2016.

9.1 Defining Scope

The MRSD project gave us an opportunity to put into action many of the tools learned in the Systems Engineering course. We started out by listing out our objectives and requirements and then classified them into functional, performance and non-functional requirements. We then moved on to plot out our system's design using system architectures. This helped us to clearly state our goals and create a systematic plan for our project.

9.2 Tracking Progress

We used an issue log to track our results and prepared a Gantt chart to schedule each of our tasks. This was helpful to coordinate tasks with each other and ensure effective team communications. We held weekly team meetings regularly and planned out each of our tasks. This made it easier to track our progress and reach our goals.

9.3 Work Breakdown structure

We used an explicit work breakdown structure to divide tasks between each other and helped us in coordinating between each other and delivering the required results. The System Engineering tools were very useful in coordinating the team and working efficiently. We learned the value of team building and understood the importance of mutual respect among team members.