Final Report

High Resolution Motion Capture

Team IRIS
Namrata Bhakta, Kalyani Nirmal, Pan Tan, Zhuo Xu

Masters in Robotic Systems Development
Robotics Institute, Carnegie Mellon University

ABSTRACT

Multi-camera networks are being used for several applications in today's world like animation, video surveillance etc. These networks require accurate calibration and good data management capability.

In our MRSD project, we present a system for efficient autonomous calibration and consequent image capture and 3D reconstruction. We have implemented this system for the 'Virtualization Studio' present in B510 of Newell Simon Hall, Carnegie Mellon University which is a geodesic dome with 480 synchronized VGA cameras arranged on 20 panels, along with HD cameras and projectors, facilitating high resolution motion capture and analysis.

Our system uses a quadrotor with an LED mounted on it as a 'virtual calibration object'. The images of this LED captured by all the cameras are processed by the calibration software to estimate the intrinsic and extrinsic parameters of the cameras and display the position and orientation of each camera with a sub-pixel reprojection error within a specified time-constraint. In order to facilitate the use of this system by various people, an intuitive graphical user interface has been developed and the current system has been updated. A real-time viewer is embedded into the GUI to display data live during the image-capture process. Furthermore, once the calibration of the cameras is complete, a 3D reconstruction of the objects in the dome is performed using the 'Visual Hull Algorithm' and displayed on the GUI.

We aim to make this system autonomous and user-friendly to aid future work in 3D reconstruction and dynamic motion capture and analysis for high-resolution motion capture.

TABLE OF CONTENTS

1. INTRODUCTION

Camera Networks have been in use for various applications like motion capture and 3D reconstruction which are used in industries such as animation, military simulations, gaming, medical applications and many more. The scale of camera networks used for the aforementioned applications so far has been comparatively small. However, what prevents people from using larger camera networks is the requirement of technical knowledge in fields such as that of computer vision.

2. PROJECT DESCRIPTION

Multi-camera technology is increasingly find applications in a large number of fields from security surveillance systems to film-making industries. With multiple cameras working together, we can build a detailed 3D model as well as capture dynamic motion. However, this new technique requires a deep understanding of computer systems and in-depth knowledge of computer vision.

For our MRSD project we aim at using these techniques and fine-tuning them so that anybody can use the technology without any hassle, in the camera network present in the 'Virtualization Studio' (NSH-B510 in Carnegie Mellon University). The aim of our system is to ease the use of the various features of multi-camera technology for people with no technical knowledge about its construction.

So far, with the limitation of the current hardware and software designs, the camera network systems face some problems:

- The network system cannot be calibrated rapidly. Usually it takes an entire day just to complete the calibration. This is time-consuming and very inefficient.
- The current user interface is not friendly at all. Only students with specific knowledge of the workspace and computer vision are able to operate it.

Our main objective is to accelerate the calibration by developing an automated quadrotor system and develop an intuitive graphical user interface which would serve to be a manual to any user who wishes to use the setup for an image capture, calibration or 3D reconstruction.

3. USE CASE

Anne, an art student at the CMU College of Fine Arts, has developed a cartoon character of herself which she wants to animate to form a 3 minute video in 3D. Unfortunately, she has none of the technical skills required to do this. She doesn't want to outsource it because she has very specific ideas that she wants to implement. Hearing about her predicament, someone tells her about the easy-to-use system present in the 'Virtualization Studio' in RI which can help her accomplish her final goal.

She heads over to the lab in NSH B510 after getting permission to use it by herself. The principle of the capture system in the lab is shown in Figure 1. She only needs to work on the GUI to follow the instructions for the set-up, send the calibration command and click on the start button to switch on the system and start the capture.

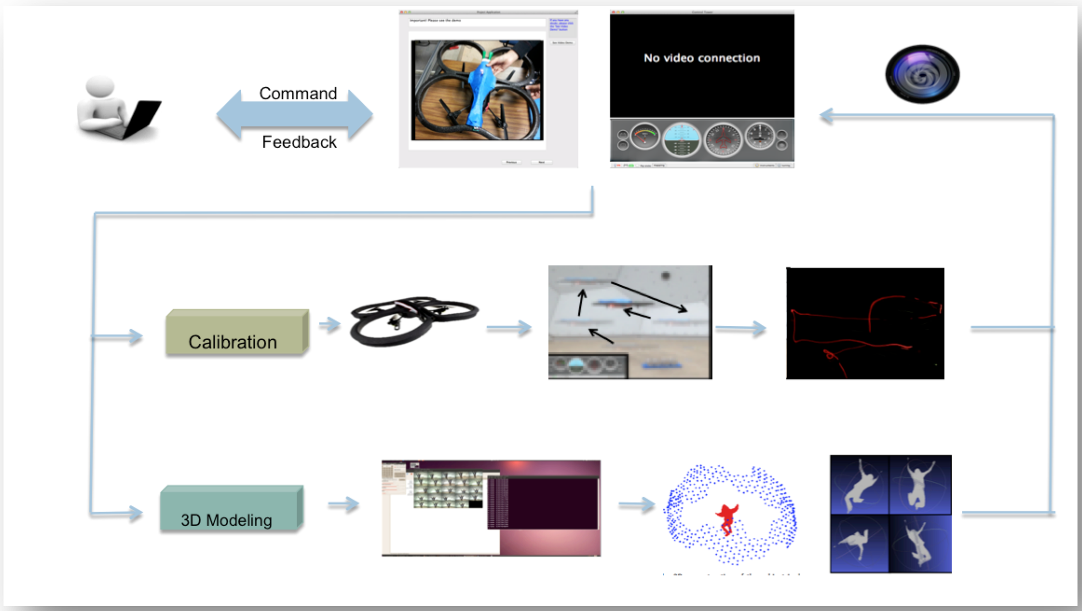


Figure 1: Graphical representation of capture system

She powers on the system by flipping the power switch. The computer screen in front of her powers on to reveal an intuitive UI which guides her through the set-up procedure for the quadrotor. The GUI displays specific instructions for tethering the quadrotor inside the dome and switching off all external lights. It also includes videos in case any of the steps is unclear. Following the simple instructions, Anne tethers the quadrotor inside the dome and clicks on 'Ready To Go'. The quadrotor autonomously takes off and flies in a pre-programmed path inside the dome while the system does an image capture. Once this is done, Anne unhooks the tether and returns the quadrotor to its storage area as the system starts the extraction of the images.

She returns the following morning before class and starts up the calibration using the calibration software provided by clicking the 'Calibrate' button on the GUI and inputting the file path. When she gets back in the evening, the machine has calibrated all the cameras using the built in commands and is displaying the message that the system can now be used for capturing images and video. Anne clicks on the 'Capture motion' command and proceeds into the dome wearing a suit she designed for her character. She dances a small routine and also records motions like walking, jumping, crawling and somersaults.

Meanwhile, Anne's partner Sheila reaches the lab to work on their project. Seeing Anne in the dome, Sheila turns on the 'Real-time viewer' to see Anne's antics live from all the angles. The live feed is jerky initially, but once Sheila double-clicks on one of the images, it maximizes into a full-screen window which displays the motion live from one camera. Once Anne finishes, Sheila clicks on ‘Stop capture’ and waits for the system to load the captured images. After all the images are in place, a pop-up notification appears stating that a 3D model of the images can be built. Anne clicks on ‘Build 3D model’ and a 3D reconstruction of the images shows-up on the screen.

Very happy with the results, Anne and Sheila load the 3D data and images on their hard disk. All they would have to do now is tweak the images into the character Anne had created and their project would be ready! They shut down the system and leave the lab talking about dinner plans...after all, a task that would usually have taken them up to a week of running around and asking favors was done within a couple of days and they can spare the time off!

4. SYSTEM LEVEL REQUIREMENTS

LEGEND	
Requirement:	Description of the project requirement
Priority:	High / Low
Type:	Mandatory/Desirable
Functionality:	Functional/Non-functional
Deadline:	Fall 2013 / Spring 2014
TPM:	Technical Performance Measure

USER INTERFACE

#1.1 – The UI shall be intuitive and user friendly	
Requirement:	The UI should be intuitive so that a person without any technical knowledge can operate it
Priority:	High
Type:	Mandatory
Functionality:	Non-functional
Deadline:	Spring 2014
TPM:	To have a non-technical persons without any background of camera networks and computer vision use the camera network via the UI – (testing based on survey – have at least 80% of the volunteers judge it to be intuitive)
#1.2 – The commands from the user shall be processed appropriately	
Requirement:	To ensure that the UI reads in the command given by the user and processes it appropriately by communicating it to the correct sub-system
Priority:	High
Type:	Mandatory
Functionality:	Functional
Deadline:	Fall 2013
TPM:	To ensure all commands to all sub-systems run correctly
#1.3 – The final output shall be displayed on the screen	
Requirement:	To have the UI display the final output as desired by the user
Priority:	High
Type:	Mandatory
Functionality:	Functional
Deadline:	Fall 2013
TPM:	To use the UI with test data and ensure the output displayed is as expected

CALIBRATION INTERFACE

#2.1 – The 480 VGA cameras shall be calibrated within time-constraint	
Requirement:	To calibrate the entire camera network of 480 VGA cameras faster than the time currently taken (10 hours)
Priority:	High
Type:	Mandatory
Functionality:	Non-functional
Deadline:	Fall 2013
TPM:	To calibrate camera network in significantly lesser time as compared to ten hours (at least half)
#2.2 – The calibration shall be accurate	
Requirement:	To develop an algorithm to calibrate the camera network accurately
Priority:	High
Type:	Mandatory
Functionality:	Functional
Deadline:	Fall 2013
TPM:	To achieve a re-projection error lesser than one pixel
#2.3 – Manual control of the quadrotor shall be achieved after attaching the tether	
Requirement:	To achieve manual control of the quadrotor after tethering it to the floor of the dome
Priority:	High
Type:	Mandatory
Functionality:	Functional
Deadline:	Fall 2013
TPM:	To be able to manually control the quadrotor to fly in an approximately helical pattern
#2.4 – The quadrotor shall be automated to fly along a pre-programmed path with the tether attached	
Requirement:	To automate the quadrotor to fly along a pre-programmed path after tethering it to the floor of the dome
Priority:	High
Type:	Mandatory
Functionality:	Functional
Deadline:	Spring 2014
TPM:	To achieve autonomous flight of quadrotor (after tethering) along the pre-programmed path while maintaining stability

REAL TIME VIEWER

#3.1 – The required network interfacing between the cameras and RTV shall be set-up	
Requirement:	The network interfacing between all the cameras of the dome and the real-time viewer screen must be set up
Priority:	High
Type:	Mandatory
Functionality:	Non-functional
Deadline:	Fall 2013
TPM:	Ensure outputs from all 480 cameras are visible on the screen
#3.2 – The data from the image capture shall be displayed live for each camera	
Requirement:	The captured images/video must be displayed real-time on the screen of the real-time viewer
Priority:	High

Type:	Mandatory
Functionality:	Non-functional
Deadline:	Spring 2014
TPM:	Ensure the view of the subject/object being captured is displayed live without delay, at the rate of 25 frames per second (speed of capture)

DYNAMIC MOTION CAPTURE & ANALYSIS

#4.1 – The large amounts of data being generated shall be effectively managed	
Requirement:	The data being generated during the motion capture (~10GB per minute) must be processed and stored appropriately without any loss of data for ease of access for the user in the future
Priority:	High
Type:	Mandatory
Functionality:	Non-functional
Deadline:	Spring 2014
TPM:	Ensure that data for a test routine is stored without any loss and can be accessed by simple commands via the UI
#4.2 – The data generated shall be analyzed for generating a 3D replay of the motion capture	
Requirement:	The data generated must be analysed for generating a 3D replay of the motion capture
Priority:	Low
Type:	Desirable
Functionality:	Functional
Deadline:	Spring 2014
TPM:	Compare the 3D model to the actual test routine of movements to check if data was analysed appropriately

3D RECONSTRUCTION

#5.1 – The large amounts of data being generated shall be effectively managed	
Requirement:	The data being generated during the image capture (~10GB per minute) must be processed and stored appropriately without any loss of data for ease of access for the user in the future
Priority:	High
Type:	Mandatory
Functionality:	Non-functional
Deadline:	Fall 2013
TPM:	Ensure that data for a test routine is stored without any loss and can be accessed by simple commands via the UI
#5.2 – The visual hull algorithm shall be used for 3D reconstruction	
Requirement:	Using the Visual Hull algorithm in place of the existing algorithms for 3D reconstruction
Priority:	Low
Type:	Mandatory
Functionality:	Functional
Deadline:	Spring 2014
TPM:	Check if image reconstructed agrees with the test subject captured by comparing dimensions and approximate positioning of interest points

5. FUNCTIONAL ARCHITECTURE

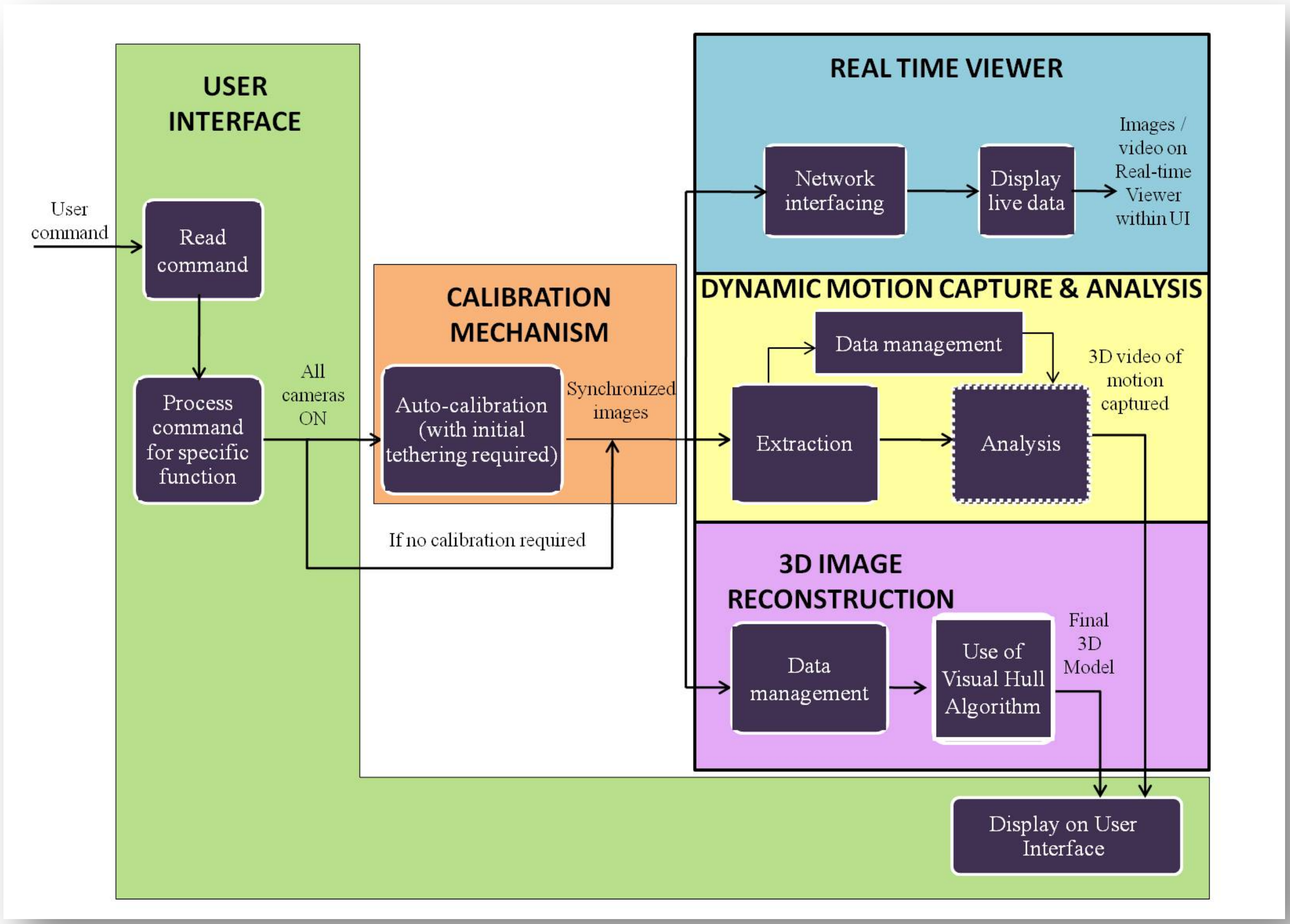

USER INTERFACE
User command
Read command
Process command for specific function
All cameras ON
CALIBRATION MECHANISM
Auto-calibration (with initial tethering required)
Synchronized images
If no calibration required
REAL TIME VIEWER
Network interfacing
Display live data
Images / video on Real-time Viewer within UI
DYNAMIC MOTION CAPTURE & ANALYSIS
Data management
Extraction
Analysis
3D video of motion captured
3D IMAGE RECONSTRUCTION
Data management
Use of Visual Hull Algorithm
Final 3D Model
Display on User Interface

Figure 2: Functional architecture (Dotted line – desirable functions)

This project involves five major subsystems namely - the user interface, calibration mechanism, real-time viewer, dynamic motion capture and analysis, and 3D image reconstruction (Figure 2).

5.1. USER INTERFACE

The user interface is an intuitive and efficient medium of communication between the user and the system. The UI reads in a user command and processes it to interface it with the appropriate hardware demanded by the user. Once the process specified is completed, the UI is also responsible for displaying the final output. The UI for the initial automation sends a command to the quadrotor to execute its pre-programmed flight path. The UI for the calibration and 3D reconstruction call the functions for the required image data set.

5.2. CALIBRATION MECHANISM

The calibration sub-system (Figure 3) is one of the most critical sub-systems, as without appropriate initial calibration, the dynamic motion capture and analysis and the 3D reconstruction sub-systems cannot carry out their functions correctly and the real-time viewer will not receive and display correct data.

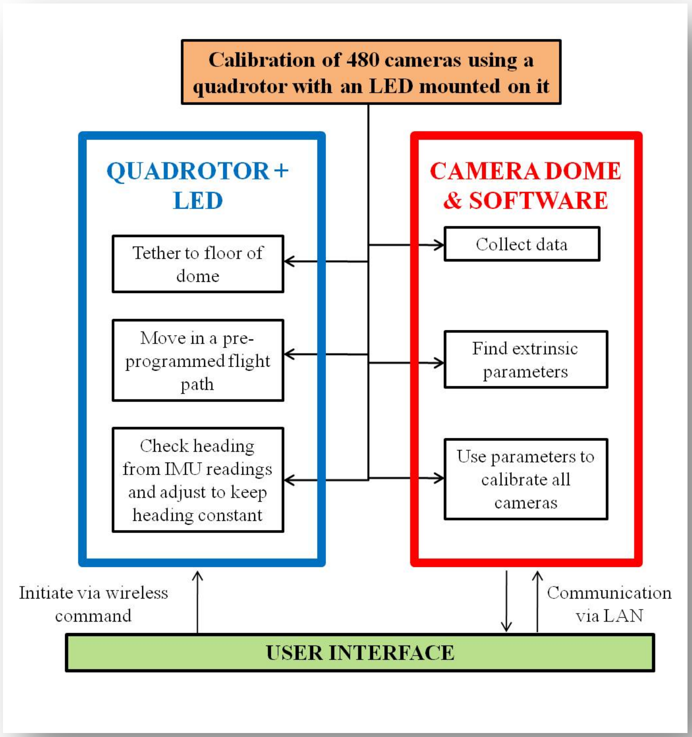

Calibration of 480 cameras using a quadrotor with an LED mounted on it
QUADROTOR + LED
Tether to floor of dome
Move in a pre-programmed flight path
Check heading from IMU readings and adjust to keep heading constant
CAMERA DOME & SOFTWARE
Collect data
Find extrinsic parameters
Use parameters to calibrate all cameras
Initiate via wireless command
Communication via LAN
USER INTERFACE

Figure 3: Break-up of functional structure of calibration mechanism

There are two major parts in this sub-system viz. the quad-rotor and the camera-dome.

- The Quadrotor

  The quadrotor with an LED mounted on it is the 'virtual calibration object' tracked by the cameras in the dome. It moves autonomously around the dome in a pre-programmed flight path, which is optimized for the best calibration results. During this, using the internal IMU (inertial measurement unit) of the quadrotor, the heading (yaw) of the quadrotor is monitored constantly and maintained within a range of 20 degrees. The quadrotor is tethered to the base of the dome in order to protect the sensitive equipment of the dome in case of any malfunction of the quadrotor.

- The Camera Dome

  The cameras in the dome track the progress of the LED and send the image data via optical fibers to the main processing CPU. This data is then used to find the intrinsic and extrinsic parameters of the cameras and estimate the pose of the cameras accordingly.

5.3. REAL TIME VIEWER

The major function of this sub-system is to give the user access to the live data being received from the cameras. It requires interfacing of the camera network with the screen along with communication between the screen and the UI in order to display the data being provided by all the cameras in one frame.

5.4. DYNAMIC MOTION CAPTURE & ANALYSIS

This sub-system involves the generation of the motion capture data, its storage, and its analysis (desirable). As huge amounts of data are to be processed, data management is an important function. The user can use this data for dynamic motion capture and have the final output displayed on the user interface.

5.5. 3D IMAGE RECONSTRUCTION

Using the data received from the cameras in the dome through the UI, the images can be processed to form a 3D reconstruction using the visual hull algorithm. The 3D reconstruction is also a measure of the accuracy of the calibration process. For this, as in 1.4, we require good data management for processing the large amounts of data being received.

6. TRADE STUDIES

6.1. QUADROTOR

We did a trade study for the best quadrotor platform that we could use for the purpose of our project which is summarized below.

Table1. Comparison of Different Quadrotors

Criteria	Weight	DJI Phantom Aerial (Point)	UDI RC U816A (Point)	AR Parrot Drone 1.0 (Point)	AR Parrot Drone 2.0 (Point)
Cost	10%	599.00$ (0)	45.00$ (10)	279.99$ (6)	329.99$ (4)
Flying time	20%	10-20 min (10)	8-10 minutes (2)	12mins (5)	12-18minutes (10)
Carry load	15%	Around 400g (10)	<=50g (2)	>=200g (5)	>=200g (5)
Weight	5%	6 pounds (0)	1.4 pounds (6)	1.5 pounds	4 pounds (10)
Indoor Frame for Safety	20%	No (7)	Yes (10)	Yes (9)	Yes (10)
Product Dimensions	10%	17 x 17 x 8 inches (10)	38.1 x 25.4 x 9.4 cm (5)	28 x 28 x 5.5 inches (with hull) (7)	23 x 0.5 x 23 inches(Without hull) (8)
Ability to Automate Flying	20%	No (2)	No (2)	No (3)	Yes (10)
Final Score	100%	66%	57%	59%	89%

Blue stands for 0-3 points, Yellow stands for 4-7 points, Green stands for 8-10 points

The features that we require for our system include the following: quadrotor flying for over 10 minutes supporting certain payload, equipped with an indoor frame for safety and hold the ability of automate flying. According to the comparison above, the AR Parrot Drone 2.0 gets the highest points and therefore become our best choice.

6.2. GUI PROGRAMMING LANGUAGE

Table 2. Comparison of Different Programming Language

Criterion	**C#**	**C++**	**Python**	**Matlab**	**Java**
Library	QT/MFC/Other (7)	QT/MFC/Other (8)	Tkinter/ Other (6)	Visual Basic/Other (5)	Swing/Other (10)
Tool kit	Built-in Toolbox (10)	Built-in Toolbox (10)	Need to Code (2)	Toolbox Modules present(5)	Built-in Toolbox (10)
Speed	Fast (10)	Fast (10)	Slow (3)	Medium (6)	Fast (10)
Proficiency of Team Members	Experienced(9)	Good (5)	Good (7)	Average (4)	Experienced (10)
Final Score	36	33	18	20	40

Blue stands for 0-3 points, Yellow stands for 4-7 points, Green stands for 8-10 points

We choose Java as the programming language for our GUI as our team members who are in charge of developing this are proficient in these languages. It also provided the required API for the automation of the AR Drone unlike many of the other languages.

6.3. CALIBRATION ALGORITHM

Table 3. Comparison between different algorithm

Criterion	**Single point calibration object**	**Single global coordinate system**	**Silhouettes of moving objects**
Description	Using a bright LED on top of a moving object to get the extrinsic parameters.	Tracking co-ordinate or a point in the world space (without an LED/specific point).	Used silhouettes of these moving objects visible in a pair to compute the epipolar geometry of that camera pair
Requirements	Dark space/LED	Repeat multiple times	Normal Light for silhouette
Efficiency	LED is easy to track and therefore results in high efficiency.	Needs to be repeated multiple times, and hence is time-consuming	The process is complex, making this algorithm less efficient

Blue stands for 0-3 points, Yellow stands for 4-7 points, Green stands for 8-10 points

From Table 3, after the comparison of all aspects, especially efficiency, we chose the first algorithm as it is the most efficient. We chose a quadrotor as the object for calibration of the cameras because a flying moving object can reach the heights at which cameras exist. Also it can be taken into the camera-dome and out of it very easily, unlike any other complicated mechanical structure. Also if we use a quadrotor it is possible that the calibration algorithm used can be generalized and used for other camera networks (indoor and outdoor) as well.

7. PHYSICAL ARCHITECTURE

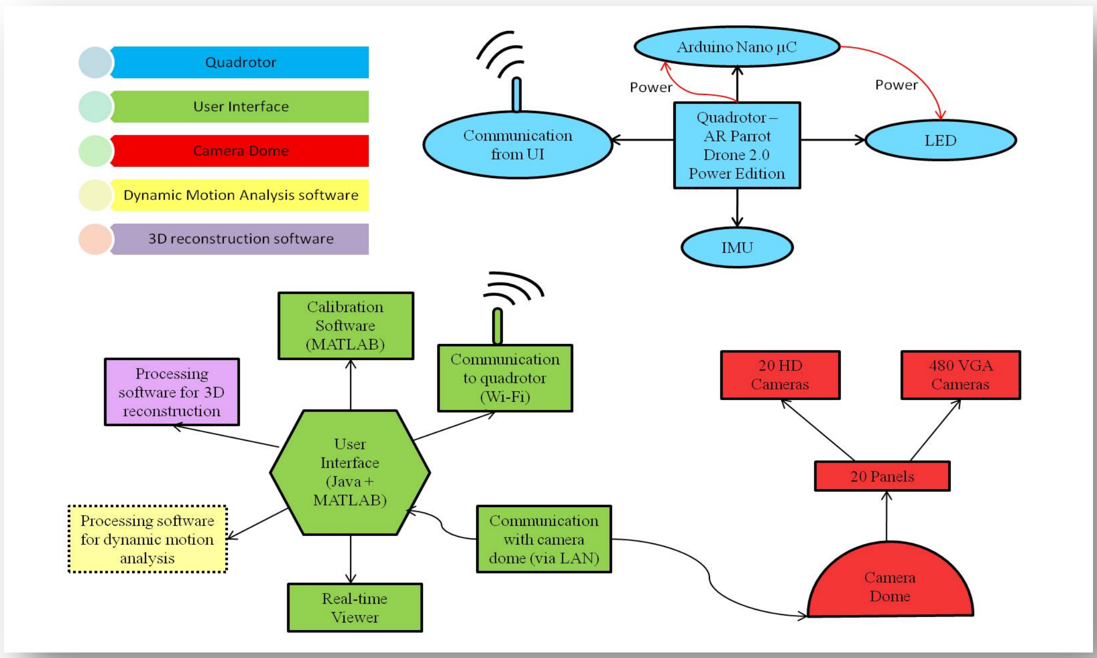

Quadrotor
User Interface
Camera Dome
Dynamic Motion Analysis software
3D reconstruction software
Arduino Nano µC
Power
Power
Communication from UI
Quadrotor – AR Parrot Drone 2.0 Power Edition
LED
IMU
Calibration Software (MATLAB)
Processing software for 3D reconstruction
Communication to quadrotor (Wi-Fi)
20 HD Cameras
480 VGA Cameras
User Interface (Java + MATLAB)
20 Panels
Processing software for dynamic motion analysis
Communication with camera dome (via LAN)
Camera Dome
Real-time Viewer

Figure 4: Physical Architecture

The physical architecture (Figure 4) of our project via which we expect to realize the functional architecture is shown in the figure above. We have three major subsystems, which are divided into various physical elements as follows.

7.1. QUADROTOR

An AR-Parrot Drone 2.0 Power Edition has been chosen to act as the moving object in the calibration mechanism. It includes:

- A mounted LED powered by the power distribution board
- An inertial measurement unit (IMU) present in the AR Drone's internal architecture
- An Arduino Nano microcontroller board mounted on the frame
- Wi-Fi communication capability to and from any computer to the quadrotor

7.2. USER INTERFACE

The user interface is divided into various modules as follows:

- An interface designed in Java that displays the roll and pitch, yaw and battery status of the quadrotor in flight

- A guided set-up procedure tool designed in Java including pictures, text and videos to enable the user to set up the quadrotor
- A comprehensive capture system built upon the existing Qt framework to make it easier for the user to capture images in the dome and view them.
- An interface for the user to calibrate the system using the images extracted from the image capture in the dome designed in MATLAB
- This MATLAB interface also includes an interface for the user to form a 3D reconstruction of a particular frame from the images captures

7.3. CALIBRATION SOFTWARE

The calibration software has been developed in MATLAB over an existing framework called the 'Multi-Camera Self-Calibration' toolbox by Tomas Svoboda. The flowchart of the algorithm has been described in detail in the next section.

7.4. PROCESSING SOFTWARE FOR 3D RECONSTRUCTION

The 3D reconstruction has been carried out using the 'Visual Hull' algorithm which takes in several images of an object and outputs its 3D reconstruction using the silhouettes seen in the images. This has been described in detail in the next section

7.5. CAMERA DOME

This is the camera-network present in the 'Virtualization Studio' present in B510 of Newell Simon Hall, Carnegie Mellon University. It is on this network that the various capabilities of the system will be tested and used. It has twenty panels containing 24 VGA cameras and 1 HD camera each leading to 480 VGA cameras and 20 HD cameras in total.

8. SYSTEM DESCRIPTION AND EVALUATION

Our system (Figure 5) consists of three major subsystems further divided into various components. These are described in detail below.

8.1. CALIBRATION MECHANISM

This sub-system includes the quadrotor, the user interface for its set-up, the power distribution board, the UI for the calibration algorithm and the calibration algorithm itself. The power distribution board is mounted on the quadrotor and connected to its Li-Po battery using a USB cord to extract power. We also have the LED, which can be switched on or off using the switch present on the power distribution board (Figure 6).

The quadrotor is connected to user interface on the system through Wi-Fi. A GUI (Figure 7) explains in detail to the users the steps they need to perform to setup the quadrotor system before launching it. The user uses the UI to launch the calibration system by initiating the flight of the quadrotor along with the capture simultaneously. The quadrotor has been automated to follow a pre-programmed path inside the dome such that the LED point mounted on it is visible to all the cameras. The automation was done using a Java API for the AR Parrot Drone by modifying their functions[9][10][11].

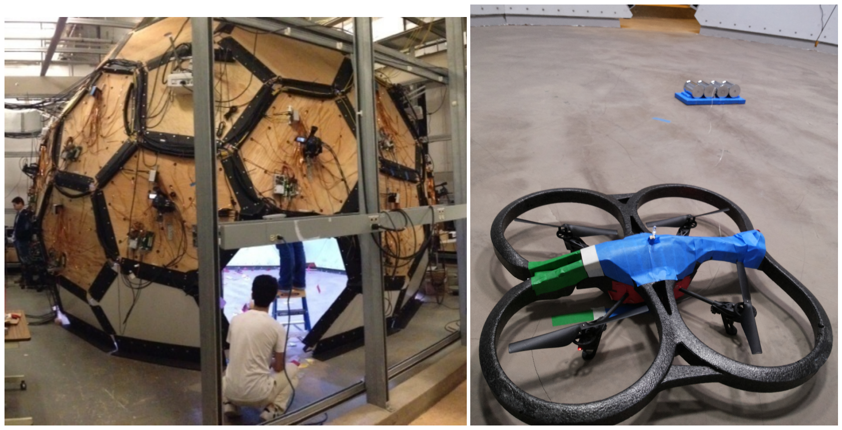


Figure 5: Full System Depiction

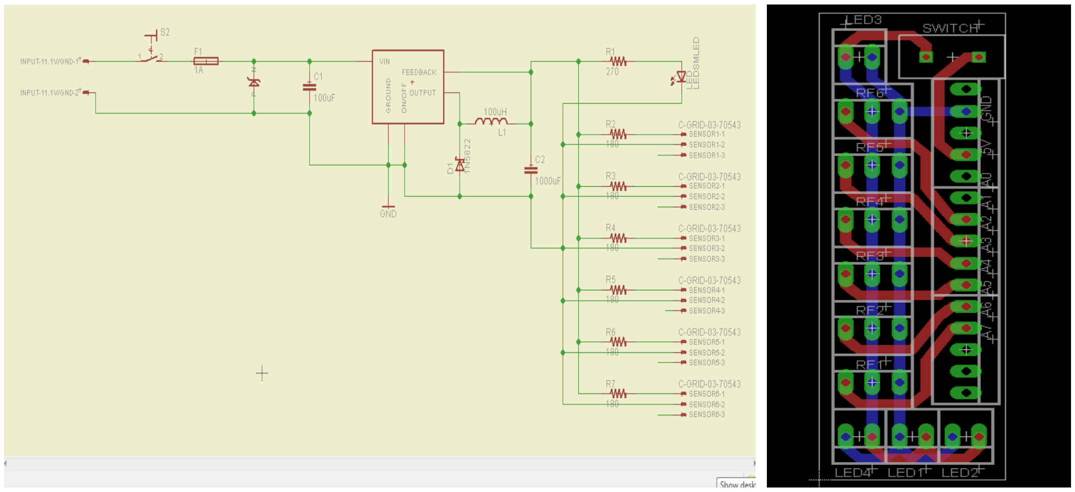
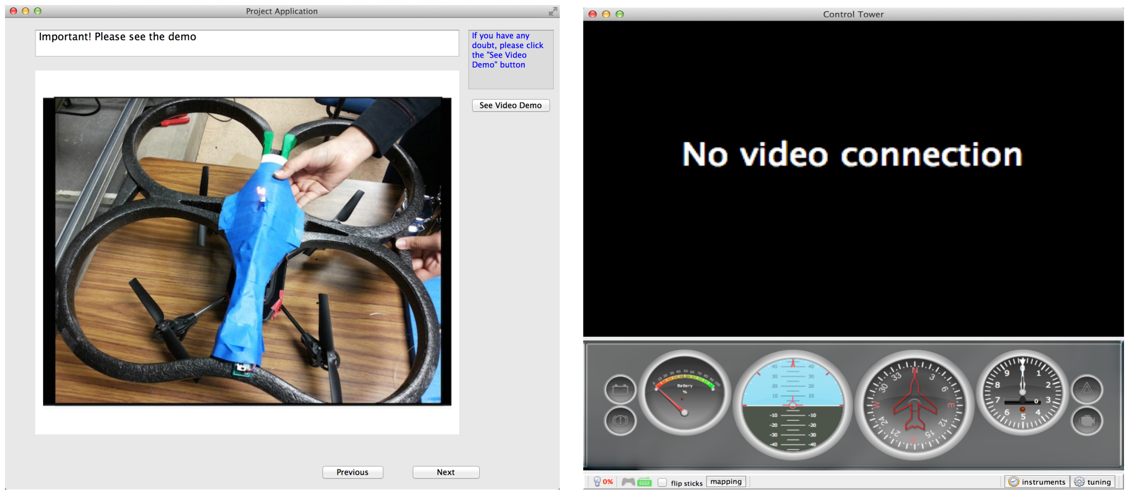


Figure 6: Power distribution board schematic (left) and layout (right)


Important! Please see the demo

If you have any doubt, please click the "See Video Demo" button

No video connection

Figure 7: Graphical User Interface

The capture then results in images of the LED mounted on the quadrotor from all the 480 cameras in the dome. The UI of the calibration algorithm then walks the user through the extraction process by providing the destination path for the images extracted and also providing the folder name for it. Once the extraction is completed the user can enter the required the details in the UI and start the calibration algorithm at the end of which we obtain the intrinsic and extrinsic parameters of all the cameras along with their reprojection error. The flowchart of the calibration algorithm is shown in Figure 8.

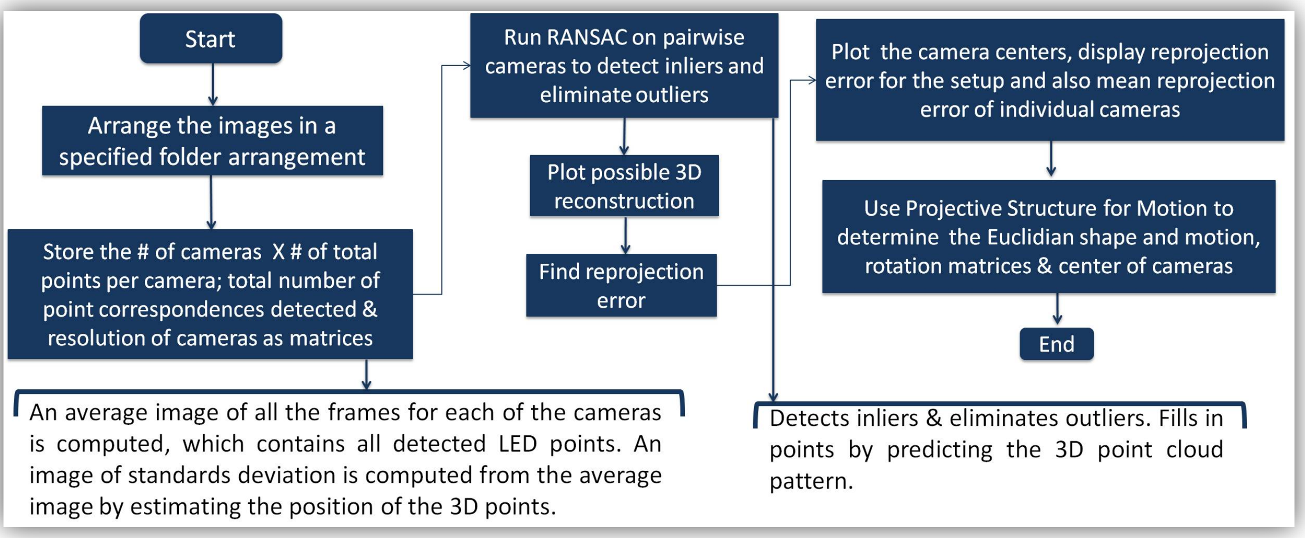
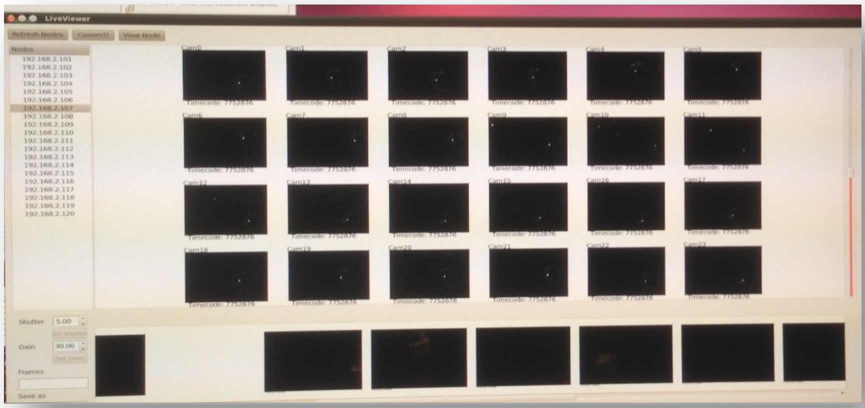

Start
Arrange the images in a specified folder arrangement
Store the # of cameras X # of total points per camera; total number of point correspondences detected & resolution of cameras as matrices
An average image of all the frames for each of the cameras is computed, which contains all detected LED points. An image of standards deviation is computed from the average image by estimating the position of the 3D points.
Run RANSAC on pairwise cameras to detect inliers and eliminate outliers
Plot possible 3D reconstruction
Find reprojection error
Detects inliers & eliminates outliers. Fills in points by predicting the 3D point cloud pattern.
Plot the camera centers, display reprojection error for the setup and also mean reprojection error of individual cameras
Use Projective Structure for Motion to determine the Euclidian shape and motion, rotation matrices & center of cameras
End

Figure 8: Flowchart of the Calibration Algorithm

8.2. USER INTERFACE

Our user interface is divided into three modules. The first module helps the user set-up the quadrotor required for the calibration mechanism and the second module guides them through the calibration process and 3D reconstruction process as described in the previous section. The third module is the existing GUI in the 'Virtualization Studio' that is used for image captures from all the 480 VGA cameras. This GUI was initially very non-intuitive and did not display all the information needed by the user. We developed our GUI[1] on the basic framework of this existing GUI such that we could integrate all the required features for us to launch calibration, view the real-time images and take an image capture. This sub-system is formed by the input, output and display systems. We modified the existing code and added several functions according to the user's feedback after using the system, such that the GUI would be more intuitive for the future users.

We implemented parallel viewing of all the cameras from one panel in a matrix form and made it easy for the user to choose a panel using the scroll bar on the bottom of the screen. We also enabled the user to see the full-screen output of any camera they wished by double-clicking on the image (Figure 9).



Nodes
192.168.2.101
192.168.2.102
192.168.2.103
192.168.2.104
192.168.2.105
192.168.2.106
192.168.2.107
192.168.2.108
192.168.2.109
192.168.2.110
192.168.2.111
192.168.2.112
192.168.2.113
192.168.2.114
192.168.2.115
192.168.2.116
192.168.2.117
192.168.2.118
192.168.2.119
192.168.2.120

Shutter 5.00
Gain 30.00
Frames
Save as

Figure 9: Graphical User Interface

8.3. REAL-TIME VIEWER

For this sub-system, a network interface between all the cameras from one panel was set-up. This interface was embedded into the GUI such that it could receive the data and commands from the GUI and display the live feed from all the cameras at the rate of 25 frames per second.

8.4. 3D RECONSTRUCTION

We implemented 3D reconstruction[4][5] of an object in the dome using the images on one frame from the 480 cameras in the dome. This was implemented using the 'Visual Hull' algorithm[6]. Figure 10 shows some sample images from the cameras of the subject who is a football player in a red jersey.

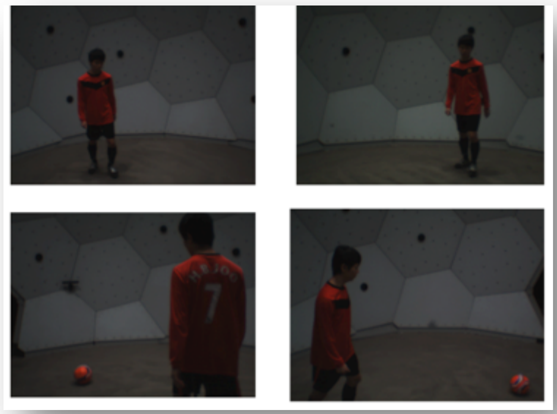


Figure 10: Test images for 3D reconstruction

We reconstruct the 3D visual hull by using the reprojection of each voxel in space and use the ground truth to validate these voxels[7][8]. This scheme is shown in Figure 11.

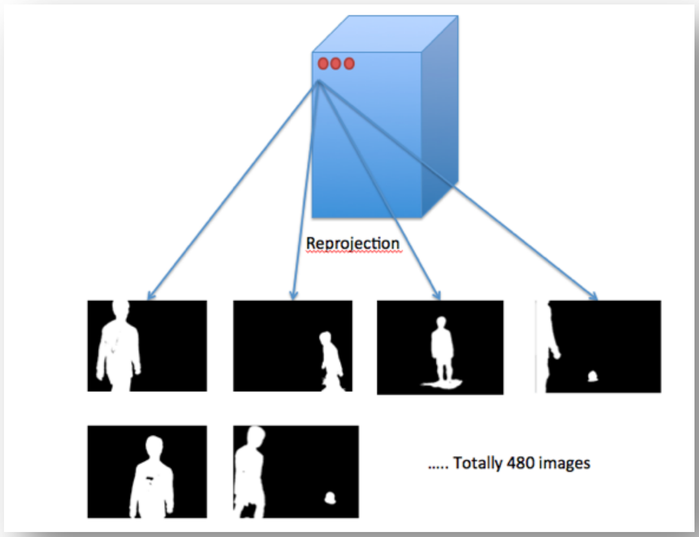


Figure 11: Principle for reprojection in visual hull reconstruction

With the results from the calibration subsystem, we can generate a set of parameters with reprojection error of less than one pixel using which we can reconstruct a 3D visual hull with the same reprojection error. Figure 12 shows the flowchart for the algorithm used for constructing the visual hull 3D image.

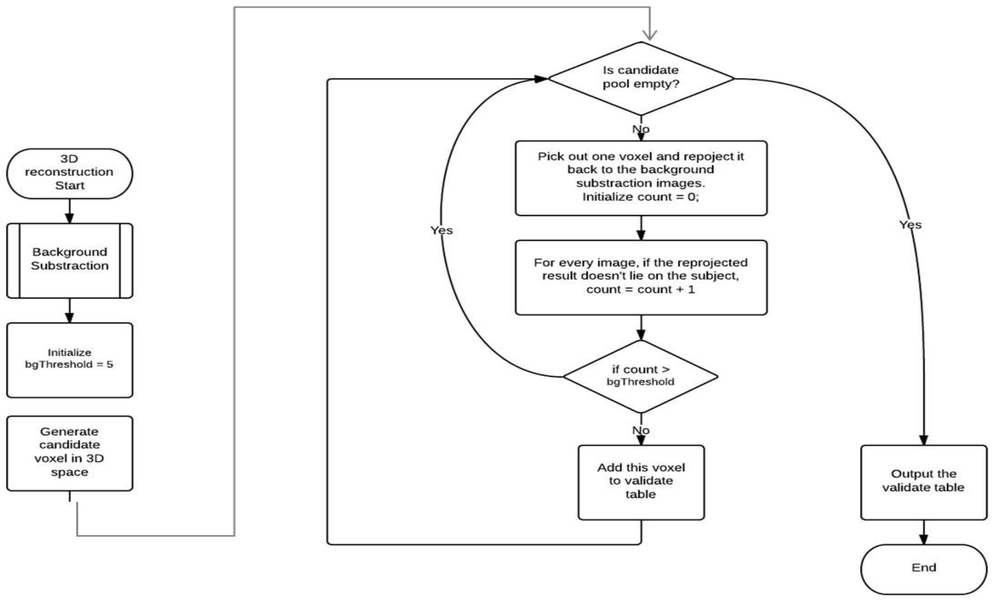

3D reconstruction Start
Background Substraction
Initialize bgThreshold = 5
Generate candidate voxel in 3D space
Is candidate pool empty?
No
Pick out one voxel and repoject it back to the background substraction images. Initialize count = 0;
For every image, if the reprojected result doesn't lie on the subject, count = count + 1
if count > bgThreshold
Yes
No
Add this voxel to validate table
Yes
Output the validate table
End

Figure 12: Flowchart for 3D reconstruction

Figure 13 shows the results of the 3D reconstruction using this algorithm. By parallelizing this with 10 cores of a CPU and a high-performance GPU, we can generate the visual hull in 60 seconds and the RGB model in 134 seconds.

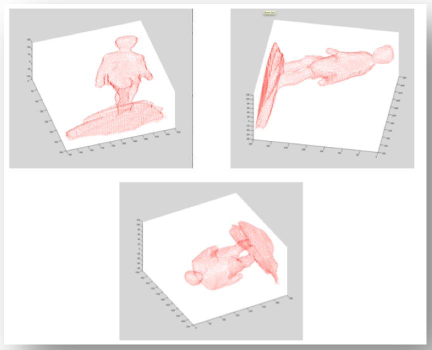
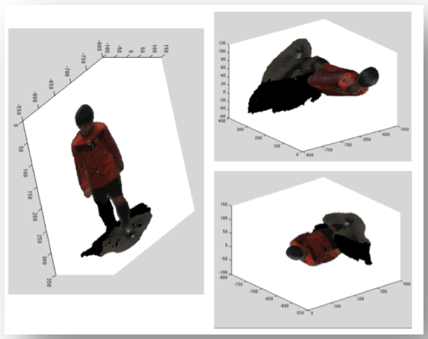



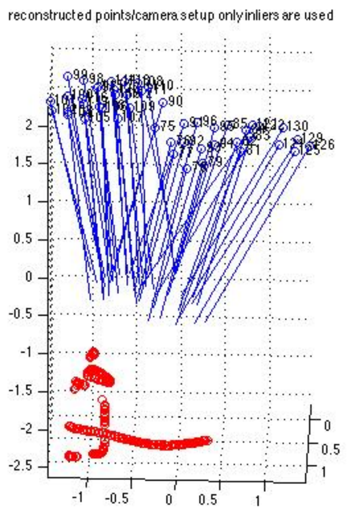
Figure 13: 3D reconstruction results (left) with RGB data (right)

8.5. MODELLING, ANALYSIS & TESTING

8.5.1. CALIBRATION ALGORITHM

We have implemented the toolbox called the 'Multi-camera Self-Calibration' Toolbox by Tomas Svoboda and others for the calibration of the multi-camera system of 480 cameras. This toolbox is designed to detect point correspondences of a bright spot in an image. We have tweaked some configuration parameters and updated the code scripts such as to adapt to the system of cameras for our experiment. Last semester we worked with the algorithm with manual flight of the quadrotor however this semester we automated the quadrotor motion. We experimented with the path of the quadrotor in terms the accuracy of the calibration and settled on a path that provided us with maximum image points of the LED in the camera images.

The algorithm requires at least some minimum number of points seen by all the cameras in the setup for it to successfully calibrate the cameras. In the case of our setup, this was not possible as the cameras are in different planes. Thus we refined the configuration parameters to run the calibration algorithm multiple times in batched of 70-90 cameras per batch to calibrate the whole setup. Running each batch required about 2-2.5 hours and the thus the calibration of the whole system took approximately 8 hours in total. On successful calibration of the setup, we get plots showing the position and orientation of the cameras and graphs showing the mean reprojection error of every camera as shown below in Figure 14. For our experiment, we achieved a mean reprojection of less than 0.5 pixels for almost all cameras.



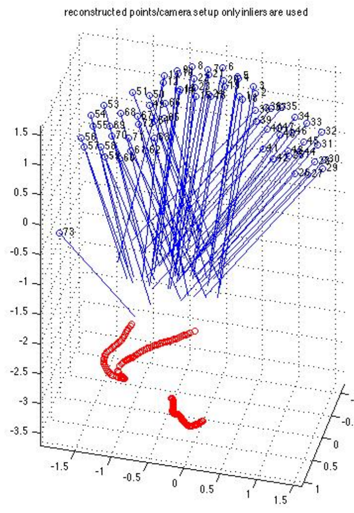


Figure 14: Calibration result for two batches of 70-90 cameras each showing the positions and orientations of the cameras on the panels of the dome

8.5.2. QUADROTOR

In order to incorporate some safety measures, we decided to attach a tether to the quadrotor so that even if the quadrotor lost stability inside the dome, it would not damage the cameras and the other expensive equipment. We initially designed a base structure (Figure 15) for the quadrotor in order to attach the tether securely and ensure that it never got entangled with the propellers but after testing the quadrotor several times with and without the base we found that the performance of the quadrotor was optimal without the base, and the tether did not create any problems.

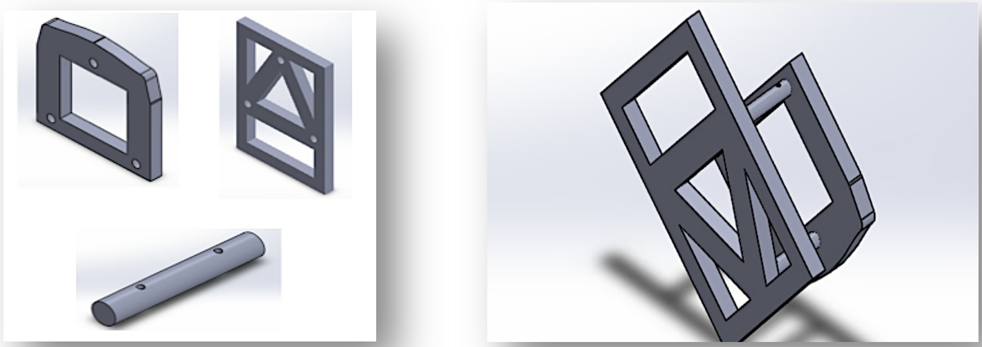


Figure 15: Parts and assembly of the quadrotor base

While automating the quadrotor, we also had to model several paths in order to adjust the quadrotor's dynamics to handle the effects of air-thrust inside the dome. We initially tried a helical path for the quadrotor in an open area which worked well but this was not possible inside the confined area of the dome. Hence we used the values from the internal IMU of the quadrotor to get the heading (yaw) as the quadrotor executed the pre-programmed path in discrete steps (Figure 16) while maintaining the heading direction, forming a closed control loop. This ensured that the quadrotor was more stable inside the dome.

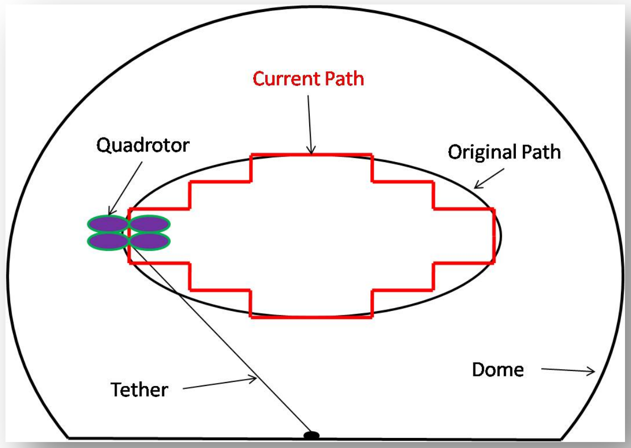


Figure 16: Path programmed to be followed by the quadrotor inside the dome

Although the values from the IMU helped in maintaining the stability of the quadrotor, there was still a significant amount of drift from the programmed path. In order to remedy this, we tried to use distance sensors to get the position of the quadrotor but this approach failed as the quadrotor could not carry the payload consisting of the Arduino Uno and XBee shield required to transmit the values from the sensors to the computer. Hence, we decided to estimate the original position and use this data as the drift in path did not affect the results of the calibration in any way.

8.5.3. USER INTERAFCE

The main purpose of this test was to prove that the changes implemented by us and the various new features added would work satisfactorily in all the GUI modules, with the users actually finding it intuitive and easy to use. Figure 17 shows the practical test for the image capture GUI. The changes made in the GUI made it possible for users to view 24 cameras from one panel at the same time and improve the user experience by providing high-resolution color-windows for all images. Users could switch between the high-resolution mode and all-window mode by double clicking on the window. Also, all the images from each camera could be displayed in our live-viewer in real-time and the key parameters such as time stamp and camera information could also be displayed at the same time.

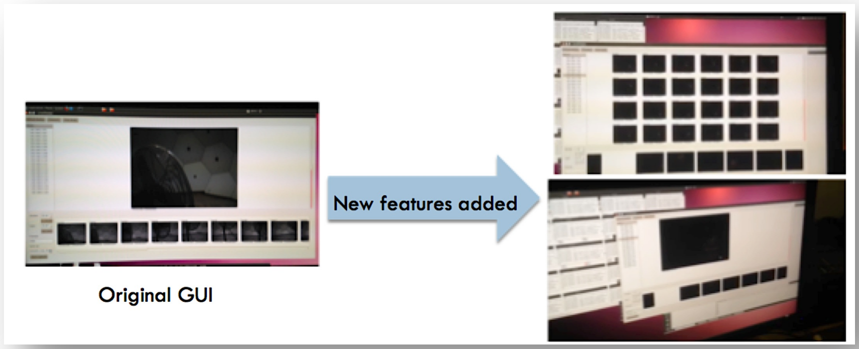


Figure 17: Modified Original GUI with new features added

For the quadrotor set-up GUI module, we conducted a user study where users from various backgrounds followed the instructions on the GUI to set up the quadrotor prior to the calibration. We iteratively improved the GUI according to the feedback from the users. Once a good model was built, we included users currently working in the 'Virtualization Studio' in the user study. Our user study indicated that on an average, users could finish setting-up the quadrotor in the dome in approximately 8-10 minutes and, from their feedback, the GUI was well designed and helped them understand the step-by-step instructions. Further, the videos accompanying the instructions also helped a lot when they got confused about any of the procedures. We also had the users evaluate the design of the calibration and 3D reconstruction GUI and they confirmed that it was intuitive and easy to follow. The final results of the user study are attached in the appendix.

8.5.4. REAL TIME VIEWER

In this test, we asked one person to stand in the dome and play ball, using the camera to record images. The results proved that all cameras showed live images (Figure 18).

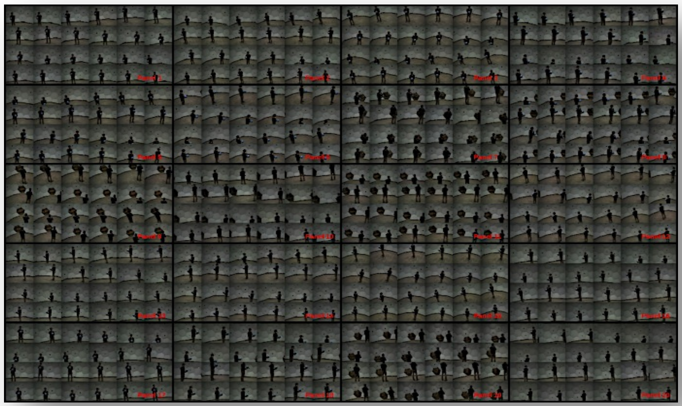
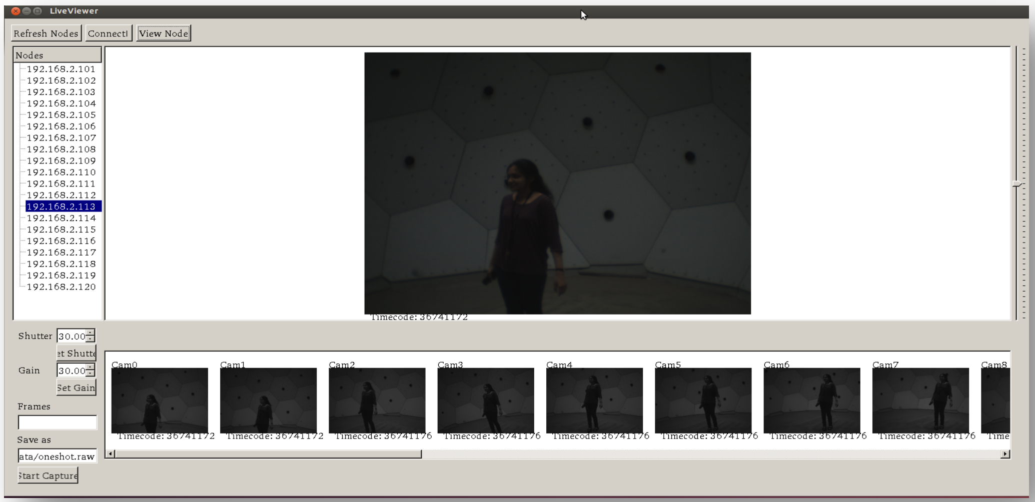


Figure 18: All cameras showed live images and displayed on GUI

Also, all the images were automatically saved on the disk, easy to extract and displayed on computer (Figure 19).



Shutter 30.00
Gain 30.00
Save as ata/oneshot.raw

Figure 19: Saved images displayed on the computer

8.5.5. 3D RECONSTRUCTION

Several tests were done with subjects wearing different colored clothes, carrying out different activities and with multiple subjects in the dome. We found that the algorithm worked very robustly whenever the contrast between the background of the dome and the subject was high but the algorithm could not estimate the actual position of the subject if there was low contrast between the subject and the background. The shortcoming of the current system is that the algorithm is vulnerable to light noise. If the light inside the dome changes a little bit, our background subtraction causes an error, which makes the final output unstable, but this should not affect the algorithm much as the dome is a controlled environment where the light will not change drastically.

8.6. PERFORMANCE VALUATION AGAINST FINAL PROJECT GOALS (SVE)

The test plan in Table 1 shows the final requirements and tests that we would conduct for our spring validation experiment (Appendix B).

Table 1: High Level Test plan

Test No	Purpose of the test	Test performed	Subsystem being tested	Relevant Requirement
1	Speed of calibration	Run algorithm for 480 cameras in <= 5 hours	Calibration software	2.1
2	Test the real-time viewer	Check if data is displayed live at required rate (25fps)	Real-time Viewer	3.1 & 3.2

3	Performance of GUI, simplicity and easy to understand	Let non-tech users operate on GUI, getting their feedback in a survey format	User Interface	1.1/1.2
4	Capability of Wi-Fi communication between the computer, quad-rotor and the microcontroller on-board	Check if commands sent from computer are received appropriately by quad-rotor and data from sensors is visible on the GUI	Quadrotor/ calibration software	2.1
5	Capability of the quad-rotor to follow a programmed path autonomously	Program a predetermined path and check the accuracy of the quad-rotor following it	Quadrotor/ calibration software	2.4
6	Accuracy of 3D reconstruction (Desirable)	Check the accuracy of the 3D reconstruction by visual comparison	3D reconstruction	5.2

The following table shows our performance against the test plan we had set-up.

Table 2: Spring Validation targeted requirements and performance evaluation against it

Subsystem	Requirements targeted to accomplish	What we had said we would do?	What did we actually do?	Result
User Interface	**1.3** Display final output	The GUI would display the images and videos captured by the cameras in the dome setup	GUI displays the output from all the panels and also from all the 480 cameras individually	Success
	1.1 & **1.2** Make UI intuitive & process commands appropriately	Let non-tech users operate on GUI, getting their feedback in a survey format	Iterated on design based on feedback from sponsor + current users of the hardware setup	Success
Calibration System	**2.4** Automated flight of quadrotor after attaching the tether	Tether the quadrotor and have it follow a pre-programmed path successfully	Used the final set-up to tether the quadrotor and automated the flight path successfully	Success
	2.3 Manual control of quad-rotor after attaching tether	Tether the quad-rotor to the ground and manually fly it around the dome in helical pattern	Tethered the quad-rotor using a temporary setup and manually flew the quad-rotor in 3 steps to cover a helical pattern around the dome	Success
	2.2 Develop algorithm for accurate calibration	480 cameras would be calibrated to a reprojection error of less than 1 pixel	All cameras were calibrated within a reprojection error of 1 pixel with almost all lower than 0.5 pixels	Success
	2.1 Calibrate 480 cameras within time-constraint	The calibration of 480 cameras would be completed within 5 hours	480 cameras are calibrated within 8.6 hours	Partial Success
Real Time	**3.2** Use the input	The real time viewer	The real time viewer	Success

viewer	from the GUI to operate the system	would display a live feed from the cameras at 25 frames/sec	displayed the live feed from all the cameras	🟩
3D Reconstruction	**5.2** To use the visual hull algorithm for 3D reconstruction	Reconstruct an object using the Visual Hull algorithm that is visually similar to the actual object	Reconstructed the object using the required algorithm that was visually similar and within a reprojection error of one pixel	Success
Note: Requirements 3.1 and 5.1 were done by the lab's staff and students for the purposes of their experiments and hence we did not carry out any further work for them				

8.7. CONCLUSIONS AFTER THE SPRING VALIDATION EXPERIMENT

The strong points of our system:

- The users found it easy to set-up the quadrotor with the instructions on the GUI and several of them mentioned that the videos were helpful.
- The quadrotor worked well with the tether attached inside the area of the dome.
- The GUI was effectively integrated with the current system for real-time viewing and image captures.
- The calibration algorithm gave accurate results.
- The set-up time required for calibration was significantly reduced. Earlier the set-up time for the calibration was over 2 hours whereas the set-up time using our algorithm is 15 minutes or lesser.

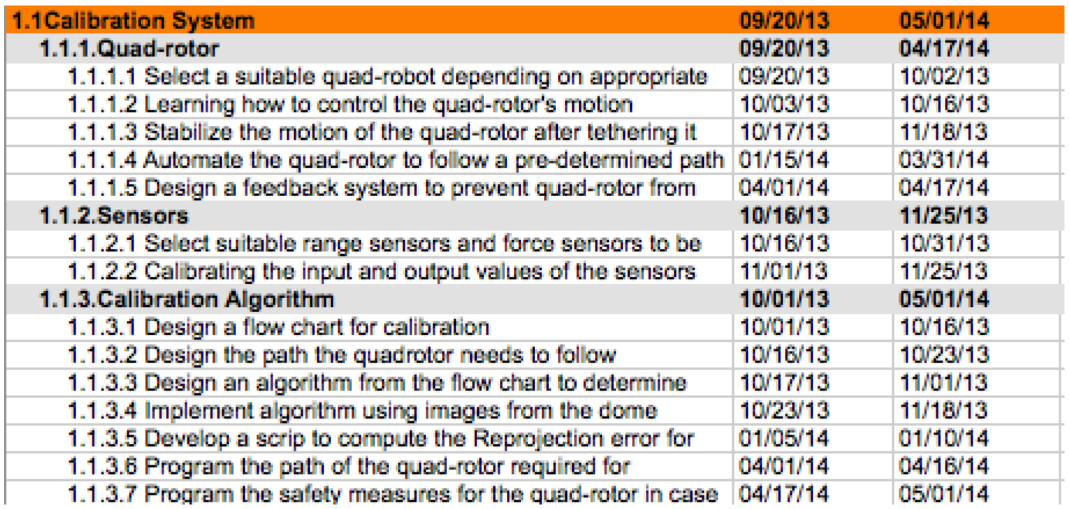
The weak points of our system where refinement is needed:

- The quadrotor has a drift from its pre-programmed path position due to the effects of the air-thrust which could not be controlled fully due to the inability to put more sensors on the quadrotor. However, this does not affect the results of the calibration algorithm, and hence does not make any significant difference to the entire system as a whole.
- The calibration algorithm is accurate but it takes too much time to run. This is a major issue caused due to the large amount of data being processed. However, although we could not meet the time requirement for the running of the calibration algorithm, we did significantly reduce the set-up time ensuring that there is minimal amount of human-intervention required in the entire process.

9. PROJECT MANAGEMENT

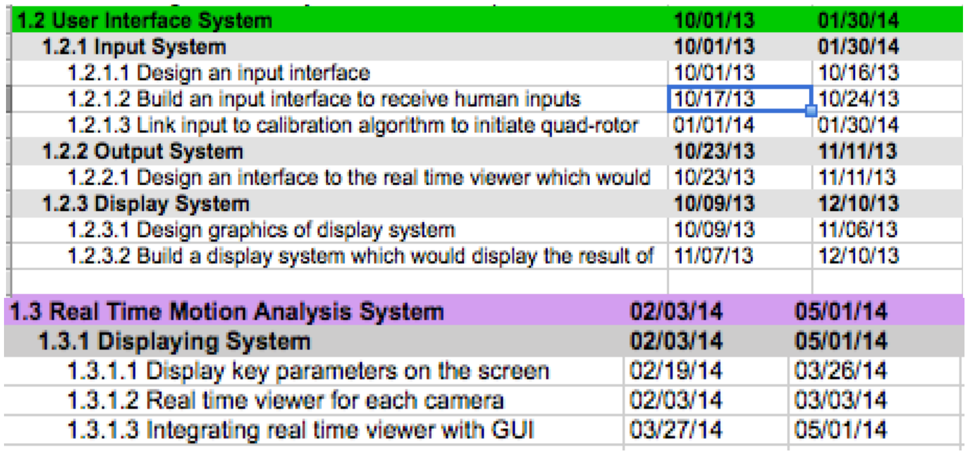
9.1. SCHEDULE

The schedule we had designed for our project is shown below (Figure 20&21). We designed this in the last semester and overall we have been successful in following this. One of the major time-consuming tasks that we did not estimate earlier was the time required for the extraction of the images after an image capture. This, added to the time it took to run the calibration algorithm made it very difficult to debug errors in a time-efficient manner. We also spent a lot of time on sensor calibration and forming the control loop using XBee shields and sensors but in the end we realized that the quadrotor could not carry the required payload. This ended up wasting a large portion of our time relative to the progress towards the completion of the project.



1.1Calibration System	**09/20/13**	**05/01/14**
1.1.1.Quad-rotor	**09/20/13**	**04/17/14**
1.1.1.1 Select a suitable quad-robot depending on appropriate	09/20/13	10/02/13
1.1.1.2 Learning how to control the quad-rotor's motion	10/03/13	10/16/13
1.1.1.3 Stabilize the motion of the quad-rotor after tethering it	10/17/13	11/18/13
1.1.1.4 Automate the quad-rotor to follow a pre-determined path	01/15/14	03/31/14
1.1.1.5 Design a feedback system to prevent quad-rotor from	04/01/14	04/17/14
1.1.2.Sensors	**10/16/13**	**11/25/13**
1.1.2.1 Select suitable range sensors and force sensors to be	10/16/13	10/31/13
1.1.2.2 Calibrating the input and output values of the sensors	11/01/13	11/25/13
1.1.3.Calibration Algorithm	**10/01/13**	**05/01/14**
1.1.3.1 Design a flow chart for calibration	10/01/13	10/16/13
1.1.3.2 Design the path the quadrotor needs to follow	10/16/13	10/23/13
1.1.3.3 Design an algorithm from the flow chart to determine	10/17/13	11/01/13
1.1.3.4 Implement algorithm using images from the dome	10/23/13	11/18/13
1.1.3.5 Develop a scrip to compute the Reprojection error for	01/05/14	01/10/14
1.1.3.6 Program the path of the quad-rotor required for	04/01/14	04/16/14
1.1.3.7 Program the safety measures for the quad-rotor in case	04/17/14	05/01/14

Figure 20: Original Schedule

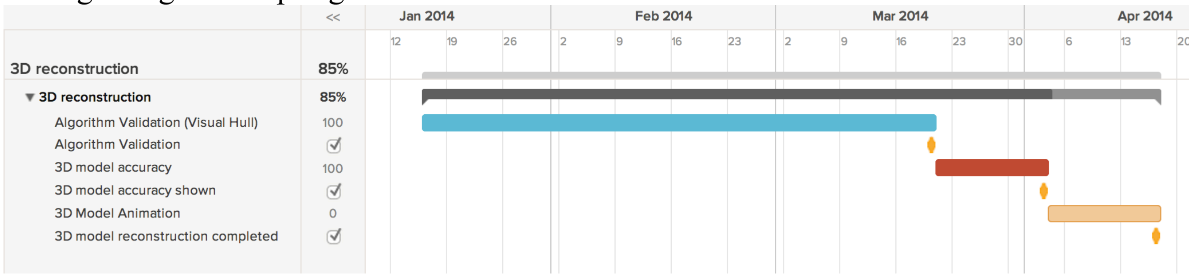


1.2 User Interface System	**10/01/13**	**01/30/14**
1.2.1 Input System	**10/01/13**	**01/30/14**
1.2.1.1 Design an input interface	10/01/13	10/16/13
1.2.1.2 Build an input interface to receive human inputs	10/17/13	10/24/13
1.2.1.3 Link input to calibration algorithm to initiate quad-rotor	01/01/14	01/30/14
1.2.2 Output System	**10/23/13**	**11/11/13**
1.2.2.1 Design an interface to the real time viewer which would	10/23/13	11/11/13
1.2.3 Display System	**10/09/13**	**12/10/13**
1.2.3.1 Design graphics of display system	10/09/13	11/06/13
1.2.3.2 Build a display system which would display the result of	11/07/13	12/10/13

1.3 Real Time Motion Analysis System	**02/03/14**	**05/01/14**
1.3.1 Displaying System	**02/03/14**	**05/01/14**
1.3.1.1 Display key parameters on the screen	02/19/14	03/26/14
1.3.1.2 Real time viewer for each camera	02/03/14	03/03/14
1.3.1.3 Integrating real time viewer with GUI	03/27/14	05/01/14

Figure 21: Original Schedule

We increased the scope of our project in spring to include 3D reconstruction as a mandatory requirement. Shown below is the schedule we designed for this (Figure 22) at the beginning of the spring semester.



3D reconstruction	85%
▼ 3D reconstruction	85%
Algorithm Validation (Visual Hull)	100
Algorithm Validation	✓
3D model accuracy	100
3D model accuracy shown	✓
3D Model Animation	0
3D model reconstruction completed	✓

Figure 22: Schedule designed for 3D reconstruction

9.2. BUDGET

Our budget estimation and management was definitely a lot easier than several other teams. This was mainly because we had the dome set-up ready for most of our work. The only major costs were that of the quadrotor and some electronic modules like the XBee. Thus, our budget was well managed and under the limit by a very high margin. The total amount we spent in nine months is $1,157.31 which is much lower than the $4000 budget given to us.

Table 3: Budget spent in fall semester

Brand	Description	Model	Estimated Cost	Actual Number	Actual Cost
Parrot	AR drone	B00D8UP6I0	$ 369.99	1	$369.99
Arduino	Arduino Nano	ARD-NANO30	$ 14.00	1	$ 7.94
Sparkfun	Super Bright LED	COM-00531	$ 8.6	20	$19
Berkley	Fishing Wire	BTBGMFCS	$ 4.99	215 meters	$8.11
Sharp	Sharp Sensor	GP2Y0A21	$ 13.5	6	$83.70
Sugatsune	Latch & Hook for tether	EN-K84	-	2	$36.82
McMASTER-CARR	Steel Eyebolt	3014T471	$3.81	1	$3.81
Arduino Nano	XBee	XBEE-4NANO	$24.95	1	$24.95
DealExtreme	6DOF IMU Sensor Module	GY-85	$11.81	1	$11.81
XBee	XBee to USB adapter	Not provided	$19.99	1	$19.99
XBee	XBee 802.15.4 (Series 1) 1mW Point-to-Multipoint RF Module with Chip Antenna	XB24-ACI-001	$24.00	1	$24.00
XBee	XBee Module - Series 1 - 1mW with Wire Antenna	1 - 1mW	$22.95	2	$68.85
Gino	Pin Headers	Not provided	-	1	$3.60

AMW	2Kg Calibration Weight	Not provided	19.95	4	$79.80

9.3. EVALUATION OF RISK MANAGEMENT

During our project, we came up with various problems and risks. In order to handle them well, we needed to analyze the problems that could occur and the chance that would happen[3]. Tables 5 and 6 show the risk estimates and their likelihood as we identified them to be for our project
.

Table 5: Risk of projects and their likelihood

Likelihood

Risk of damage to the quadrotor		The risk of the quadrotor being unstable in flight
Risk of non-efficiency of 3D modelling	Risk of UI being complex	The risk of the calibration algorithm not performing

Consequence

Table 6: Risks and their likelihood

Risk No.	Risk Title	Description	Risk Type
1	Risk of damage to the quadrotor	The AR Drone 2.0 is expensive. When we try to launch the tests or try to implement automate features on our AR drone, we might face the risk of causing damage to it and need to buy a new one. This might be a big load to our whole budget.	Cost
2	The risk of the calibration algorithm not performing	We may face the problem of a trade-off between memory space and time. There is a great chance that our method might be fast enough but consumes a lot space on memory, which is also another issue that might be considered.	Technical
3	Risk of non-efficiency of 3D modelling	The efficiency of 3D modeling and motion capture requires not only good strategies and a well-designed pattern but also needs high performance of our hardware and system. The higher efficiency is the requirement for our system and equipment.	Technical
4	Risk of UI being complex	The risk for UI design lies in the definition of “Easy” and “Friendly”. Thus, there might be the possibility that we think our ways of developing is considerate enough to help users guide through all the operations, but as a matter of fact, users might still find it hard to understand all those buttons, textboxes and grids.	Technical

5	The risk of the quadrotor being unstable in flight	The control of AR Drone to fly in a stable manner may not be perfect when automated. This needs to be considered in terms of the extra time that might be required to make it stable.	Schedule

Overall we managed to identify almost all the risks that we faced. One of the major risks that we did not consider initially was that of the schedule not being followed due to the excess time taken for data management, i.e., extraction of images etc. Towards the end of the project there were several times when we could not get newer data due to time restrictions. Had we considered this risk earlier, we could have planned our schedule around that and started the testing of the data right from the beginning of the semester instead of waiting for the data from the final, automated run of the quadrotor. We also faced issues sometimes when the lab was occupied by the research scholars who work there for the purpose of their experiments. Had we considered this risk earlier, we could have set-up a lab schedule such that we would always know when another student would be using it.

A risk that we did consider was that of the UI being too complex, and we conducted a thorough user study in order to ensure that it was intuitive. We also considered the risk of damage to the quadrotor and took precautionary measures like adding tape in weak areas from the beginning to reinforce those areas. We also had a back-up drone from the inventory in case any damage occurred to our drone in the last minute before a demonstration.

10. CONCLUSION

This section describes the lessons learnt through the course of the project and what we would be different if we started once again. It also puts forth the future work possible.

10.1. LESSONS LEARNT

Elucidating lessons learnt give us a chance to reflect on events and activities during the project and helps bring closure to the project by understanding the various issues and nuances faced in its designing, building and testing. Some of the major lessons that we learnt in this semester are listed below.

- Clearly define goals for everyone in the team
- Conduct meetings regularly to keep everyone in the team updated about progress on different modules and also to brainstorm about problems faced
- Maintain proper written communication with sponsor to make sure the expectations match the actions
- Documentation must be developed as a habit with respect to design iterations, videos of test runs etc.
- Dive deep into a topic before starting the design process for anything, be it the system architecture, software algorithm or mechanical parts.
- Order parts only from well-known websites to avoid mishaps such as parts arriving late which cause an unnecessary delay in the project
- Simplifying the design of the PCB by using existing micro-controller boards wherever possible.

- Develop the software in robust, modular and efficient way to avoid any kind of re-writing.
- Completion of testing needs to be done well in advance before a demo to avoid battery or part failure during the demo
- Testing the most uncertain part of any subsystem before building the other parts. We built the entire control loop using sensor feedback before we mounted it onto the quadrotor and realized that the quadrotor could not carry the required payload which ended up wasting a lot of our time.

10.2. CHANGES WE WOULD MAKE IF WE COULD RESTART THE PROJECT

The main aim of our project was to calibrate the multi-camera setup of 480 cameras and display one potential application of such a setup. There are two ways in which a multi-camera system can be calibrated, either using a checkerboard or by asking a volunteer to stand in the center of setup and wave a bright object in a pattern. The first method uses the features present in a checkerboard image to calibrate the setup whereas the second method calibrates on the basis of visible points of the bright object waved and tracking the pattern of these points by using the famous 'PNP' approach followed by bundle adjustment. We implemented the second method by using a quadrotor with a LED to imitate the pattern of the bright object as waved by the volunteer. However, due to the dome like arrangement of the multi-camera setup using the quadrotor was not a good idea. The air-flow within the dome is not uniform because of the construction of the dome.

Thus, given a chance to restart the work on the project we would prefer using a ground vehicle with an extending telescopic arm on it. The height of the telescopic arm would be continuously varied, as the ground vehicle would move within the dome and this would replicate having a series of the LED points move in a pattern in the air whose images would be captured by the camera and the algorithm would continue approximating the intrinsic and extrinsic parameters. The calibration algorithm and the application designed that is 3D reconstruction in our case would continue to be the same. In this manner we would also not require any setup time from the user before starting the calibration as the vehicle can be made autonomous and its path and navigation into the dome can be preprogrammed.

10.3. FUTURE WORK

If we could continue working on our project further, we would have liked to use the dome to build an application using dynamic motion capture. Within these 9 months, although we worked on the set-up in order to make it easier to use, we did not get a chance to build any applications except for 3D reconstruction. Considering the vast capabilities of the set-up, we would definitely have explored some real-world application possible using the framework they have built, like real-time 3D transmission of live events. We would also have liked to integrate all the GUI's and develop the entire code in C++ for the calibration algorithm.

We would also have liked to use fewer cameras for some applications to compare the difference in results in order to validate the usage of the high number of cameras in the dome.

REFERENCES

1. Blanchette, Jasmin; Summerfield, Mark (June 2006). "A Brief History of Qt"; C++ GUI Programming with Qt 4 (1st edition) Prentice-Hall. pp. xv–xvii. Retrieved 5 August 2013
2. Tomas Svoboda, Daniel Martinec, Tomas Pajdla, Jean-Yves Bouguet, Tomas Werner, Ondrej Chum, ‘Multi-Camera Self-Calibration Toolbox’, May 2005
3. Risk Management Process for DoD Acquisition,Sixth Edition, August 2006 (through the Systems Engineering class by Dimitrios Apostolopoulos)
4. B. Baumgart. Geometric Modeling for Computer Vision. PhD thesis, Stanford University, 1974
5. J. S. D. Bonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. InProc. of ICCV’99, Sept. 1999.
6. S. Lazebnik, E. Boyer, and J. Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In Proc. of CVPR’01, Dec. 2001.
7. G. K. M. Cheung. Visual hull construction, alignment and refinement across time. Technical Report CMU-RI-TR-02-05, Carnegie Mellon University, January 2002.
8. C. Bregler and J. Malik. Video motion capture. Technical Report CSD-97-973, UCB, 1997
9. https://code.google.com/p/javadrone/
10. https://code.google.com/p/javadrone/wiki/Beginner_Dev_Guide
11. https://code.google.com/p/java-simple-serial-connector/

APPENDIX A

User #	Name	Time taken (min)	Errors made	Feedback
1	Konduri Vamsi	6.40	• Tether not attached correctly (over the frame instead of under it)	• Tether attachment not clear; • Text not always visible clearly
2	Naina Thangaraj	15.30	• Frame not attached correctly • Tether not attached correctly	• Confusion about how to attach the frame – mention about flat alignment • What is the base? Show a picture again during the attachment of tether • USB cable attachment is tricky – needs more instructions • Tether is a problem – mention adding a simple double knot • GUI still non-intuitive – users cannot be sure they're right • Alignment of base required?
3	Karthik C.L.	12.19	• None – although frame attachment not done initially	• What is the frame? • How to tether? Is it correct? • Make the demo button bigger • Alignment of base required?
4	Akshay Phatak	9.02	• None – although does not like the fact that the tether has to be knotted	• What does 'tether is tight' in the last instruction mean? • Overall fairly intuitive
5	Spencer Krause	7.02	• None – though weights placed in an incorrect configuration at first	• Make the 'see video demo' link more visible • Resistance to go in the dome and come back out before seeing the next instruction
6	Prathamesh Kini	6.53	• None	• Good interface – will be good if videos have subtitles
7	Lei Tan	8.14	• Frame attached incorrectly before seeing the video 0 corrected after seeing the video	• Resistance to go in the dome and come out again – tried to view all instructions before doing it
8	Zijun Wei	10.28	• None	• Was confused before watching videos
9	Minh Vo	7.45	• None	• Wanted final image before starting

General Feedback:

- GUI is intuitive and set-up can be understood but most users tend to skip over the videos if they see the pictures thinking they can do it themselves – made text specifically say 'watch the video demo'

- Ensuring that the users go into the dome and come back out before going to the next instruction is difficult – add a time lag and dialog box to ensure they do it
- People tend not to read text – add colours where it is particularly crucial
- Only pictures are not enough – added video demos to make sure that all the instructions are clear
- GUI for calibration and 3D reconstruction is very obvious and easy to use – general marking on clarity above 9 on 10

APPENDIX B

1. TEST PROCESS

- Test 1: Speed of calibration
  - Step 1: Tether the quad-rotor according to instruction on GUI
  - Step 2: Move the quad-rotor in helical pattern and start capture from all cameras simultaneously
  - Step 3: Manually land quad-rotor
  - Step 4: Start timer and Run calibration algorithm
  - Step 5: Stop timer when calibration is completed
  - Step 6: Test succeeds time < = 5hours else fails

- Test 2: Test the real-time viewer
  - Step 1: Click the input button on the GUI for motion capture using VGA cameras.
  - Step 2: Check if real-time viewer shows live data.
  - Step 3: Check number of frames per second in the video stored from the real-time viewer.
  - Step 4: Test succeeds if data is shown live

- Test 3: Performance of GUI, simplicity and easy to understand
  - Step 1: Switch on system
  - Step 2: Let volunteer (non-tech) use the system via the GUI to calibrate, do an image capture, load previous data etc. and record their responses in the survey format given.
  - Step 3: Test succeeds if 80% users find the GUI simple and easy to understand.

- Test 4: Capability of Wi-Fi communication of computer with the quad-rotor and the microcontroller on-board
  - Step 1: Tether the quad-rotor according to instruction on GUI
  - Step 2: Send command for quad-rotor flight using GUI
  - Step 3: Check if data returned by the microcontroller is shown on the screen.
  - Step 4: Send command for the quad-rotor to land using GUI.
  - Step 5: Test succeeds if the quad-rotor takes-off and lands according to the commands sent and if the data from sensors is visible on the GUI screen.

- Test 5: Capability of the quad-rotor to follow a programmed path autonomously
  - Step 1: Tether the quad-rotor according to instructions on GUI
  - Step 2: Send command for quad-rotor flight using GUI
  - Step 3: Test succeeds if it follows an approx. path programmed and lands without getting damaged

- Test 6: 3D reconstruction (Desirable option)
    - Step 1: Get image data from the image-capture done in the dome
    - Step 2: Build the 3D model by using our visual hull algorithm
    - Step 3: Test succeeds if the model represent real object and is visually recognizable