Group 5 Project Report
Prototype of a Micro-Surgical Tool Tracker

Students: Sue Kulason, Yejin Kim
Mentors: Marcin Balicki, Balazs Vagvolgyi, Russell Taylor

May 13, 2013

Abstract

Ophthalmic surgery consists of a number of delicate procedures that require a level of accuracy and precision that is difficult to achieve with current methods. Although robot-assistance can alleviate hand tremor, sucess of these procedures still suffers from a lack of quantitative information and poor decision making. By providing the position of a tool with respect to the eye, hospitals can monitor surgical protocol, assess surgical skill, and provide information for better decision making. An optical tracking system that provides positional feedback was implemented and tested. Results show that the detection algorithm must be improved to eliminate outliers, but when the target is correctly detected, the tracking system is accurate within 2 mm in optimal lighting conditions.

1 Introduction

According to a study conducted in 2009, ophthalmic surgery has the highest number of incorrect procedures in the operating room [1]. These procedures suffer from inaccuracy due to hand tremor, and a lack of quantitative information about the tool in relation to the eye [3]. Specifically, surgeons currently use a 2-channel microscope to amplify their view of the patients' eye, which provides a narrow field of view [3]. The narrow field of view and lack of quantitative information leads to poor decisions that can result in unwanted collisions. New robot-assisted surgical methods are being developed to cope with issues in accuracy due to hand tremor [3]. However, both traditional ophthalmic surgery and robot-assisted ophthalmic surgery would benefit from positional feedback of the tool in relation to the eye.

Several methods have been proposed to provide positional feedback. One such system utilizes electromagnetic trackers placed around the operating room to track tool position.

Electromagnetic trackers have continuous visibility of markers, do not interfere with the surgeon workspace, and have markers with negligible size and weight [4]. However, there are several key disadvantages to electromagnetic trackers. First, these trackers are sensitive to metal, which is found in nearly all tools and equipment. Furthermore, it has been shown that electromagnetic trackers are not as accurate as optical trackers [4]. Although optical trackers provide high accuracy and are not sensitive to metals, this system also has some drawbacks. Optical trackers rely on continuous visibility of markers, has potential to interfere with surgeon workspace, and the tool tracking markers tend to add weight to the tools [4].

The goal of this project was to prove that an optical tracking system can calculate accurate tool position in relation to the eye. The drawbacks of a typical optical tracking system were eliminated by implementing a system of four cameras to account for occlusion, and painted tools of negligible additional weight. The aims were to develop a device to rigidly house four cameras in place without interfering with surgeon workspace, implement a tracking algorithm to provide accurate positional feedback in real-time, and an evaluation of the tracking accuracy of this system.

Tool position can be utilized in a variety of ways to improve ophthalmic surgery. It can monitor surgical protocol to ensure tasks such as lubricating the eye are done periodically. It can also assess surgical skill, which has been shown to correlate with the number and type of strokes taken during a procedure [5]. With quantitative information on tool position with respect to the eye, hazard warnings to improve surgical safety can also be implemented. Optical tracking is compatible with both standard practices and robot-assisted surgery, and can also be adapted to other types of microsurgeries. For example, optical tracking could also prove useful for the insertion of cochlear implants.

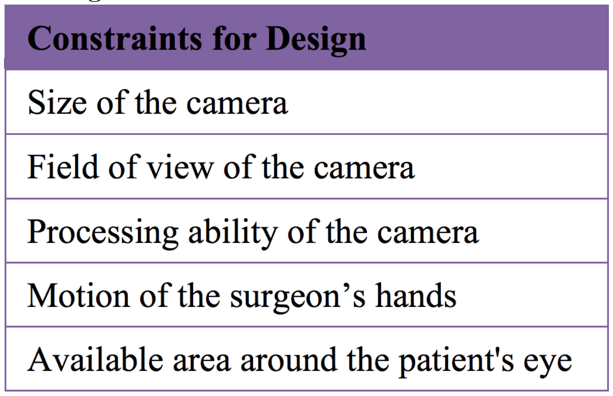
2 Technical Approach

2.1 Mechanical Device

The first phase of device design was to determine ideal optical sensor specifications. The primary specifications taken into account are shown below in Figure 1. Other considerations were type of camera, cost, resolution, and software compatibility. Although IR cameras would have been ideal for the dark surgical environment, it would require an IR light source and modification of the tools that went beyond the scope of our proof-of-concept. Therefore, an RGB camera was chosen.

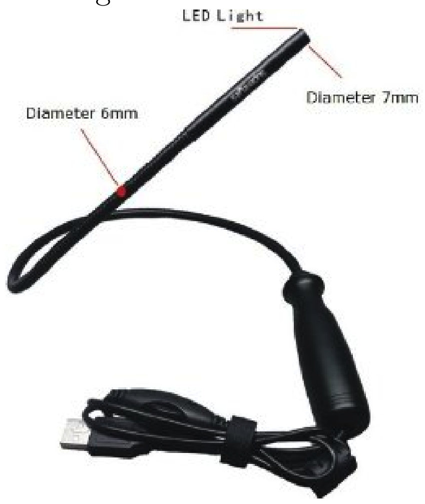
The camera that was selected was an endoscope camera for $76.98 as shown in Figure 2. The camera had a small, 7 mm camera tip to minimize interference with the surgeon's work space, a USB interface that made it easy to connect to tracking software, and an undetermined field of view as this data was not provided. However, a test of the camera showed that it had a narrower field of view than expected. At a 7 mm distance, the camera could see 20 mm by 30 mm. As the tool tip was estimated to be 30mm, it became necessary to scale the prototype by 1.5 times larger than life. Camera resolution was 640 x 480 pixels, which meant at least 16 pixels could be seen per mm at the desired distance. This was determined to be reasonable for the purposes of a proof-of-concept.

Figure 1: List of Camera Constraints



Constraints for Design
Size of the camera
Field of view of the camera
Processing ability of the camera
Motion of the surgeon’s hands
Available area around the patient's eye

Figure 2: Chosen Camera



The next step was to develop stands on which the cameras could be rigidly attached. It was key for the cameras to stay still once calibrated in order to ensure optimal accuracy of tracking. To this end, stands with 7 mm diameter clips were printed and tested. These clips were also utilized in an initial calibration test and placed as shown in Figure 3. However, unexpected issues with cameras rotating within its case was discovered and may contribute to the tracking errors discussed later.

Figure 3: Initial Prototype

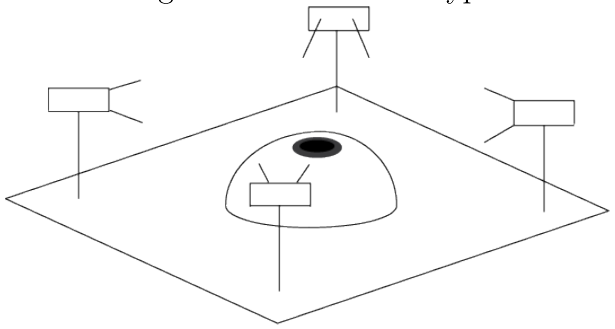
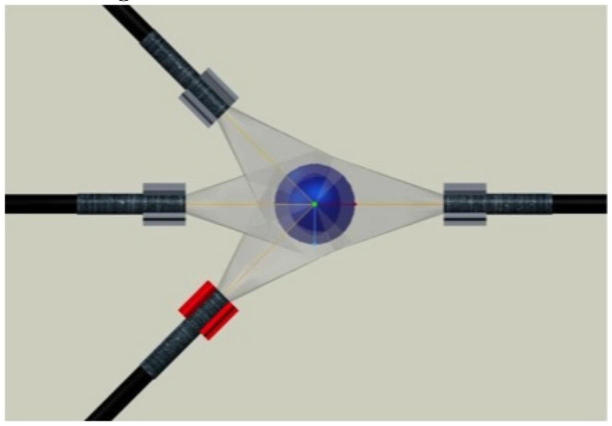
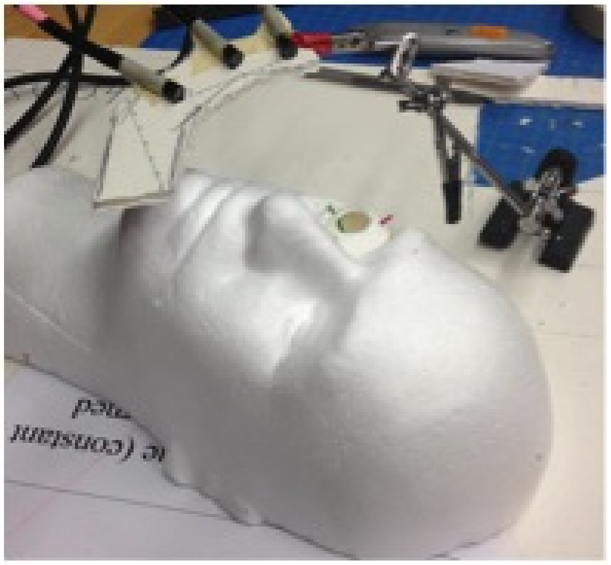


Figure 4: CAD of Camera Views



The third phase of device design was to determine optimum placement of the four cameras. In order to provide positional feedback of the tool and trocars, it was necessary to place cameras in a way that each had a complete view of the eye environment. Constraints for camera placement include the camera’s field of view, surgeon’s hand motion during operations, and the shape of the face. Below in Figure 4 is the finalized design of camera placement.

Figure 5: Mockup of Device



In order to verify camera placement, a styrofoam face was purchased and modified to house a silicon eye provided by Marcin Balicki. Yellow, green, and red wires were used to represent trocars and the tool was painted blue. A mock-up of the final device was designed as shown in Figure 5. The orientations between cameras were changed to account for the profile of face and view of the eye environment. More detailed documentation of the device's dimensions can be found in Apendix A.

Once the mock-up was verified, the next step was to CAD the final device design, as shown in Figure 6. After some final adjustments, the final device was built as shown in Figure 7.

Figure 6: CAD of Device

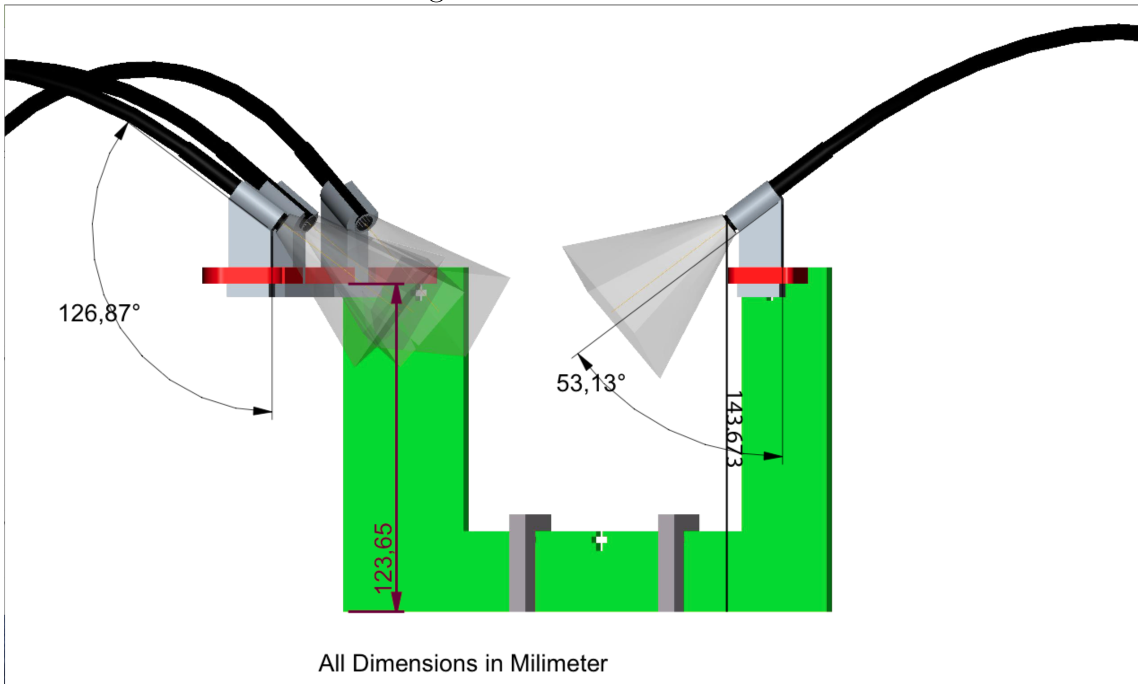
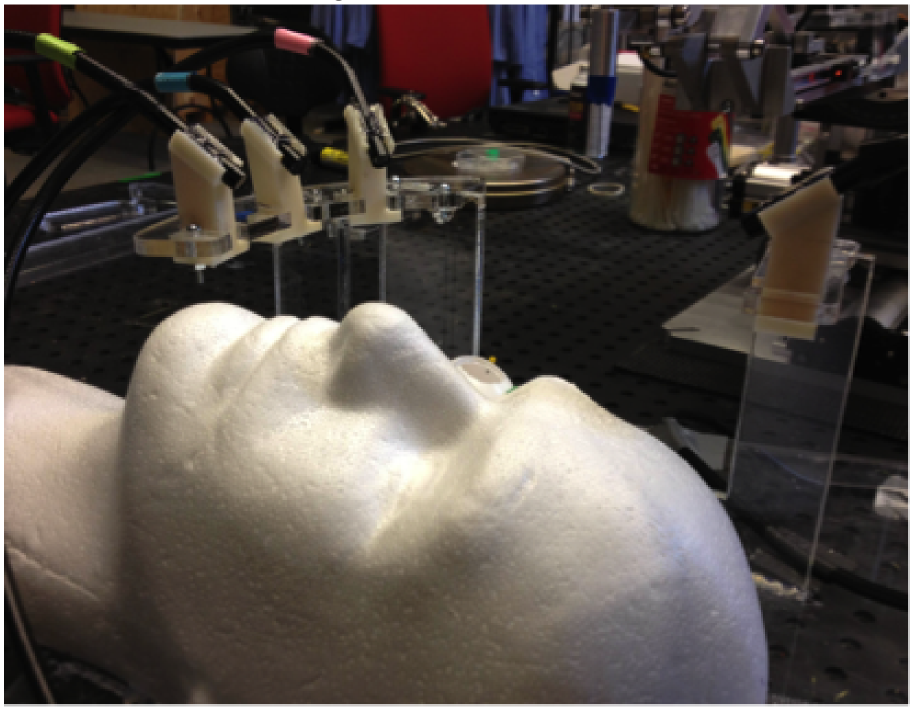
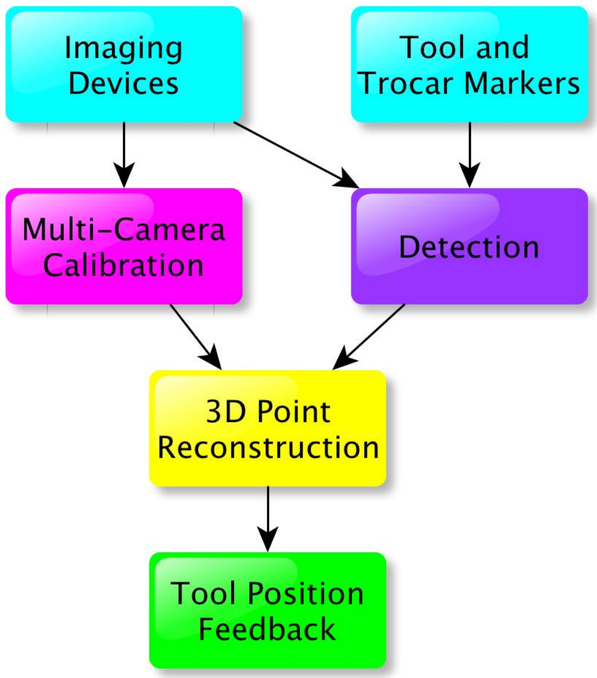


Figure 7: Final Device



2.2 Tracking Software

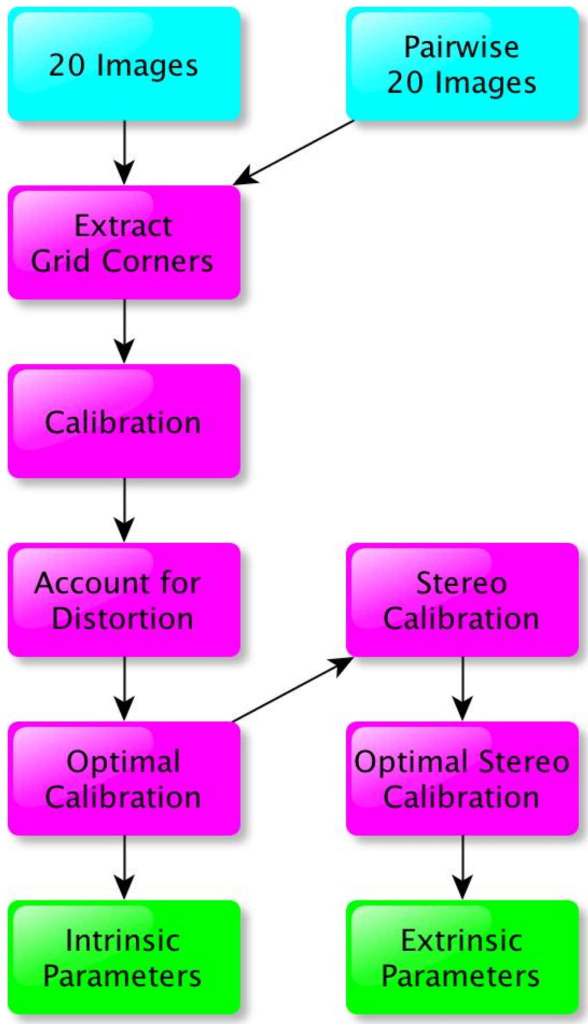
Figure 8: Tracking System Flow Chart



The conceptual approach of optical tracking has three major steps: multi-camera calibration, detection, and 3D point reconstruction. Figure 8 shows the overall tracking scheme. First, images are taken from all four cameras to perform multi-camera calibration. This is performed offline. Next, new images are taken with the tool and three trocars marked visibly in different colors. This is run through the fast-blob color segmentation algorithm that detects pixel locations of each color of interest for each camera view. Finally, the output from calibration and detection is combined to perform 3D point reconstruction. The final tool and trocar positions are printed and drawn in a 3D graphic display.

Figure 9 is a more detailed view of calibration. Since single camera calibration is generally more accurate than pair-wise camera calibration, the results from single camera calibration were used to determine intrinsic parameters. For each camera, there is a set of 20 images that is run through a pipeline to extract grid corners, calibrate, account for distortion, and optimally calibrate in order to determine the intrinsic parameters. This is a pipeline provided by Matlab Calibration Toolbox under Example 1.

Figure 9: Calibration Flow Chart



For extrinsic parameters, it is necessary to perform pair-wise camera calibration. In this case, the accuracy is limited by the shared field of view where the planar checkerboard can be placed. Similar to single camera calibration, each set of 20 images must be run through a pipeline to extract grid corners, calibrate, account for distortion, and optimize calibration. These results were then taken to determine stereo camera calibration, and then the optimal stereo camera calibration, which determined intrinsic and extrinsic parameters for each camera. Only the extrinsic parameters were used in the tracking algorithm. The pipeline for stereo camera calibration is provided in detail by Matlab Calibration Toolbox under Example 5.

As described before, the next step in tracking is detecting the four color markers. Figure 10 describes the general process of detection. First, a frame taken from a camera is converted from the RGB color space to YUV color space. This is because the YUV color space is more robust to changes in illumination. Next, the image was thresholded for each color of interest. The ranges of each color were determined by guess and check for multiple images with varying illumination and camera views. This noisy image is improved with a series of morphological operators such as dilate, erode, open, and close. Then, using OpenCV, the connected components were determined from the binary image. The largest one is assumed to be the marker and the center was calculated using moments. From test results, it is clear that assuming the largest connected component to be the marker led to issues when the marker was occluded. Currently the algorithm does not account for instances when a marker is not present but the color is. The final output from this portion of the algorithm is the pixel location of the center of each marker.

The detection algorithm was tested by overlaying original RGB images with the color thresholds. Specifically, detection was tested for varying camera views and illumination settings. Appendix A has information on where these results are stored.

Figure 10: Detection Flow Chart

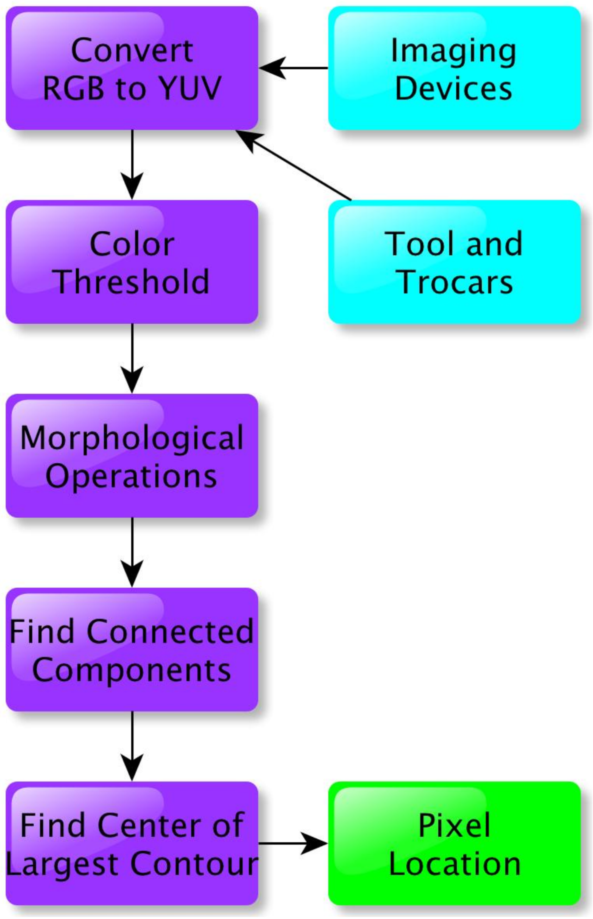
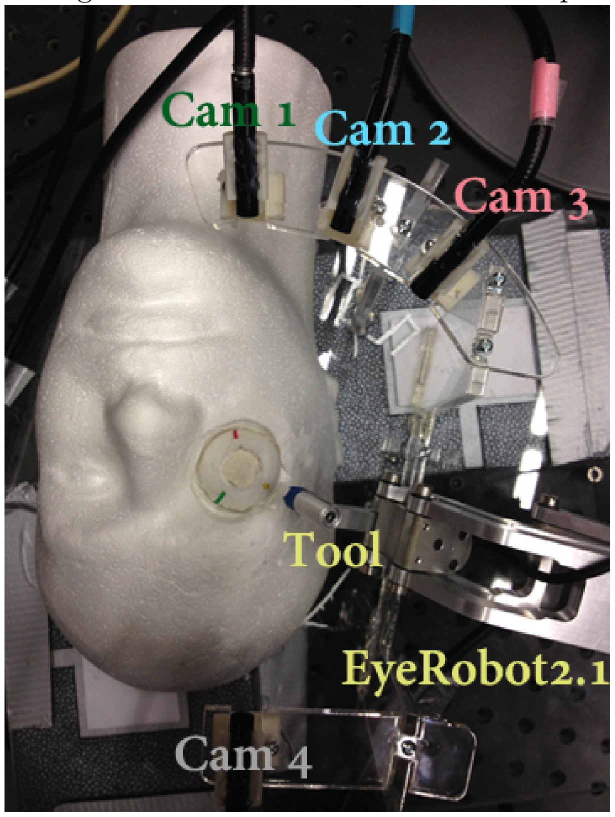


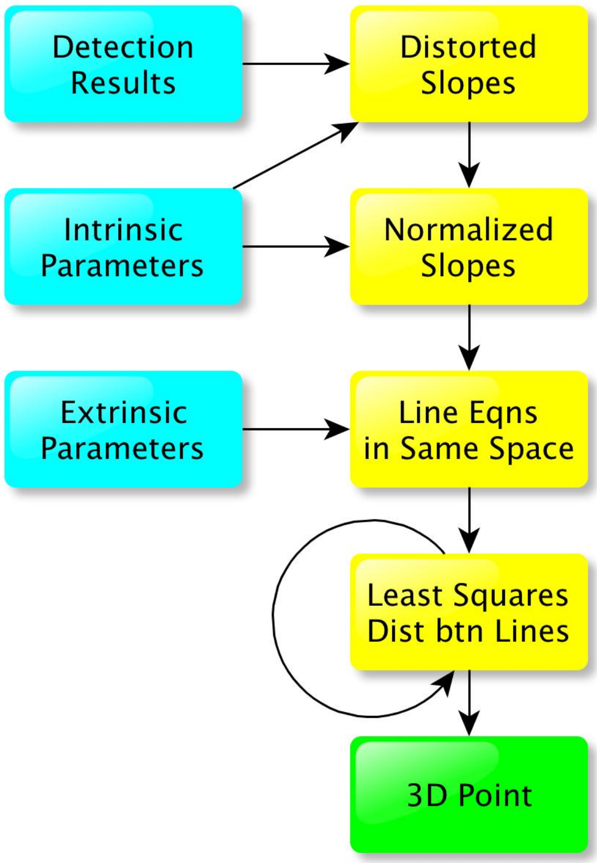
Figure 11: Color Coded Camera Setup



The final step in the tracking algorithm was the 3D point reconstruction for each marker, as shown in Figure 12. Each of the 4 pixel location outputs for each marker from the detection step was transformed to find the distorted pinhole image projection utilizing the camera matrix calculated during calibration. This point must then be normalized with the distortion parameters, which were also calculated during calibration. Specifically, there is no algebraic solution to undistort a point, so two corresponding mesh grids of pixels must be used to estimate the normalized point. These normalized values represent the slope of the line from the camera origin to a detected marker center. In order to compare between cameras, these line equations must be transformed to the same camera space using the extrinsic properties calculated from calibration. We chose to transform everything to the space of Camera 1, the green camera shown in Figure 11. Next, the goal was to find the point of intersection between all the lines. However, due to noise there may not be a point of intersection. Therefore, we chose to look through each pair of lines and find the pair of points with the shortest distance between the lines through a least squares approach:

$$Slope_0 = X_1 - X_0 \tag{1}$$
$$Slope_1 = Y_1 - Y_0 \tag{2}$$
$$F = Slope_0 \times Slope_1 / \sqrt{(}Slope_0 \cdot Slope_1) \tag{3}$$
$$G = F * (Slope_0 - Slope_1) \tag{4}$$
$$A = [Slope_0 | Slope_1] \tag{5}$$
$$b = -G * F - X_0 + Y_0 \tag{6}$$
$$[U, S, V^t] = svd(A) \tag{7}$$
$$x = U^t \cdot 1/S^t \cdot V \cdot b \tag{8}$$
$$Point_0 = Slope_0 * x[0] + X_0 \tag{9}$$
$$Point_1 = Slope_1 * x[1] + X_1 \tag{10}$$

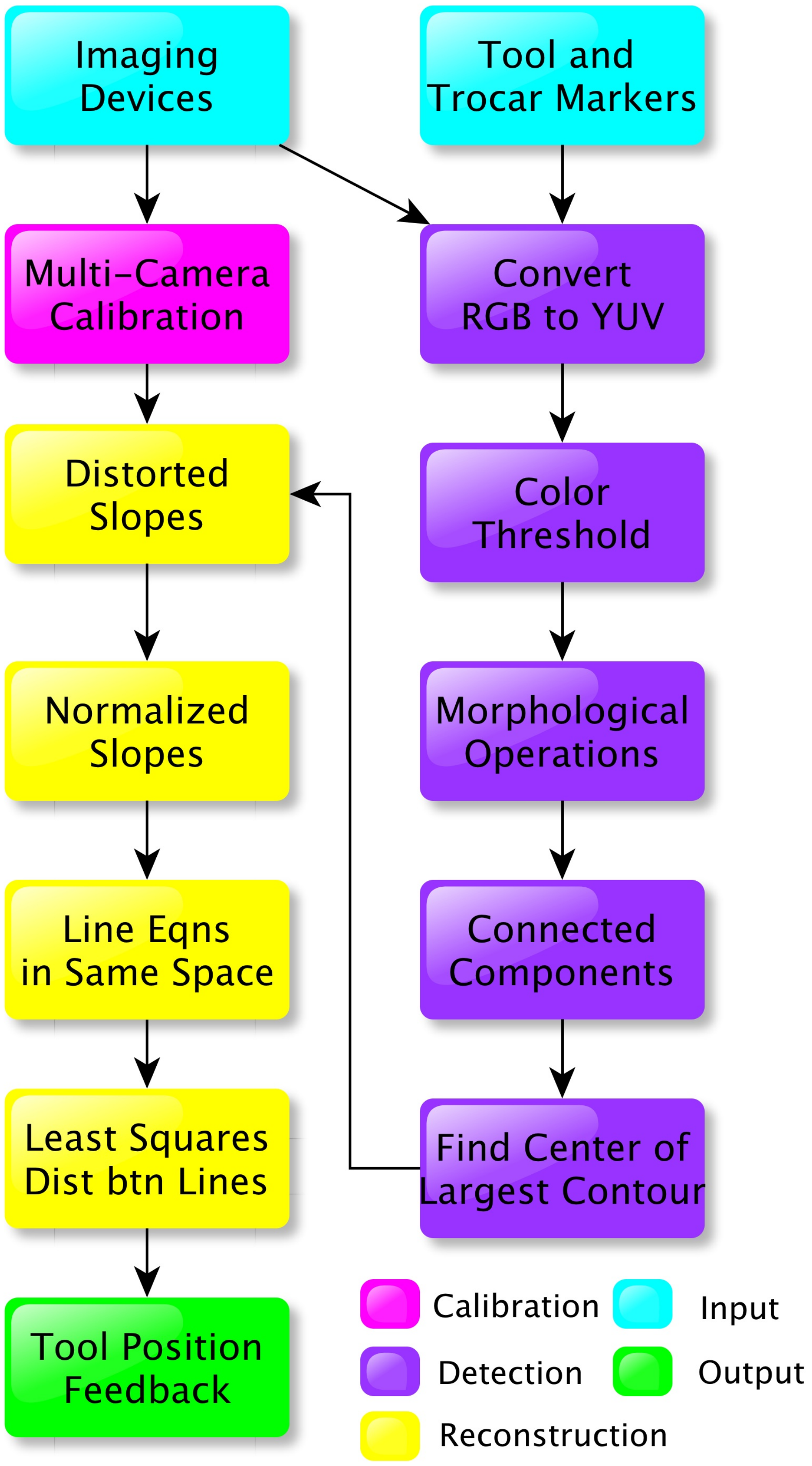
Figure 12: 3D Reconstruction Flow Chart



Once the two corresponding points are found, the average was taken to be the true location of the marker. This was repeated for each pair of lines and an average of the averages was used to define the final 3D point location. These four resulting points were put in a graphic display to provide positional feedback.

An unexpected problem arose that led to modifications in the technical approach. The cameras were not compatible with image grabbing software available through Python, OpenCV, or video streaming software such as DivX. Therefore, it became unrealistic to achieve real-time tracking. Instead, we chose to opt for offline tracking of images and videos. However, image and video acquisition through 4 synchronized cameras also proved challenging. The CISST script for stereo camera viewing was modified by Balazs Vagvolgyi to stream two unsynchronized sets of two cameras. Compressed videos depended on DivX, so it was necessary to modify the script to acquire uncompressed videos and use CISSTs video converter for post-processing. The difference between frame times is not accounted for in our algorithm, although this issue should be minimal for the slow movements characteristic of ophthalmic surgery.

Figure 13: Final Tracking Flow Chart

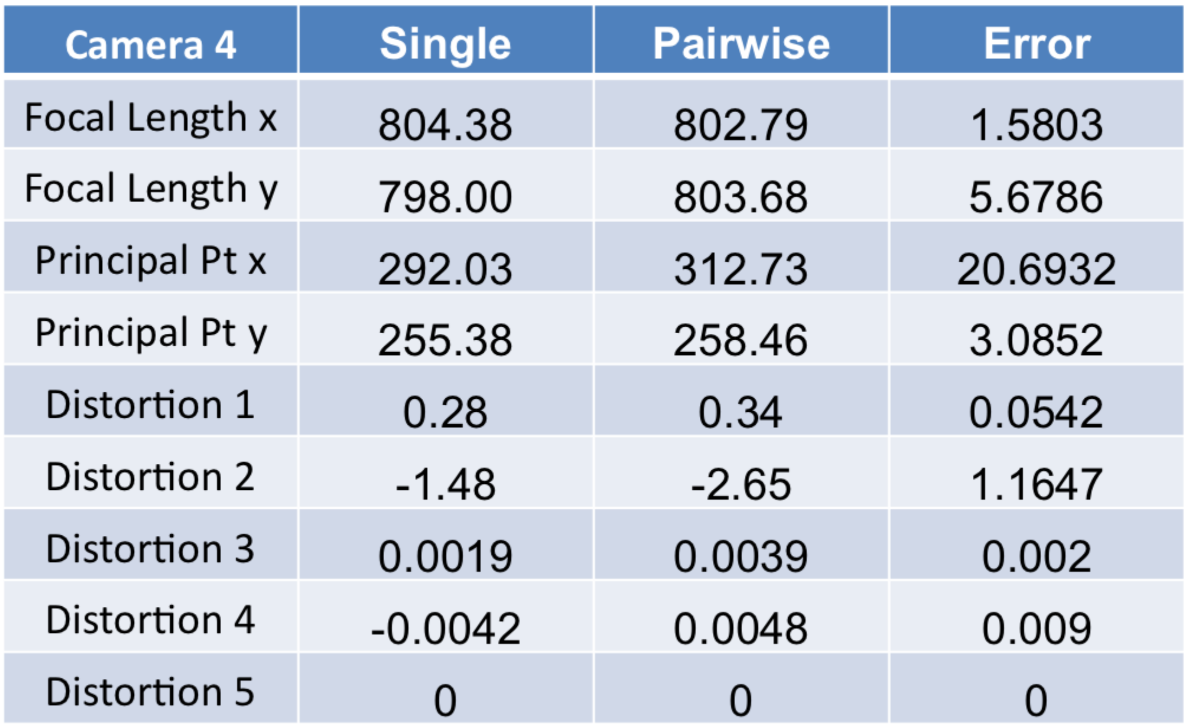


3 Results

3.1 Calibration Test

The purpose of the calibration test was to determine the accuracy of pair-wise calibration utilized in our tracking algorithm. For each camera, the intrinsic values were compared between single camera calibration and pair-wise camera calibration. This is because the limited field of overlapping view is a major factor in calibration error. Accuracy of extrinsic properties was also examined by comparing differences in designed, measured-by-hand, and measured-by-calibration extrinsic parameters. Figure 14 shows the results for Camera 4, whose pair-wise calibration had the smallest field of view and was therefore prone to the most error. Results for the other three cameras revealed similar results and can be found in Appendix A. The error in determining the principal point in the x direction was 20.69 pixels which was larger than ideal. Overall the errors were small, suggesting that pair-wise calibration is a reasonable approach for our tracking project. In fact, many of the errors found for intrinsic properties of Camera 4 were smaller than other cameras. The biggest source of error seems to come from Camera 2, whose resolution was visibly worse than the other three cameras. Other sources of possible error for both types of calibration include subpixel corner detection error and warping of the planar checkerboard.

Figure 14: Camera 4 Intrinsic Test



Camera 4	Single	Pairwise	Error
Focal Length x	804.38	802.79	1.5803
Focal Length y	798.00	803.68	5.6786
Principal Pt x	292.03	312.73	20.6932
Principal Pt y	255.38	258.46	3.0852
Distortion 1	0.28	0.34	0.0542
Distortion 2	-1.48	-2.65	1.1647
Distortion 3	0.0019	0.0039	0.002
Distortion 4	-0.0042	0.0048	0.009
Distortion 5	0	0	0

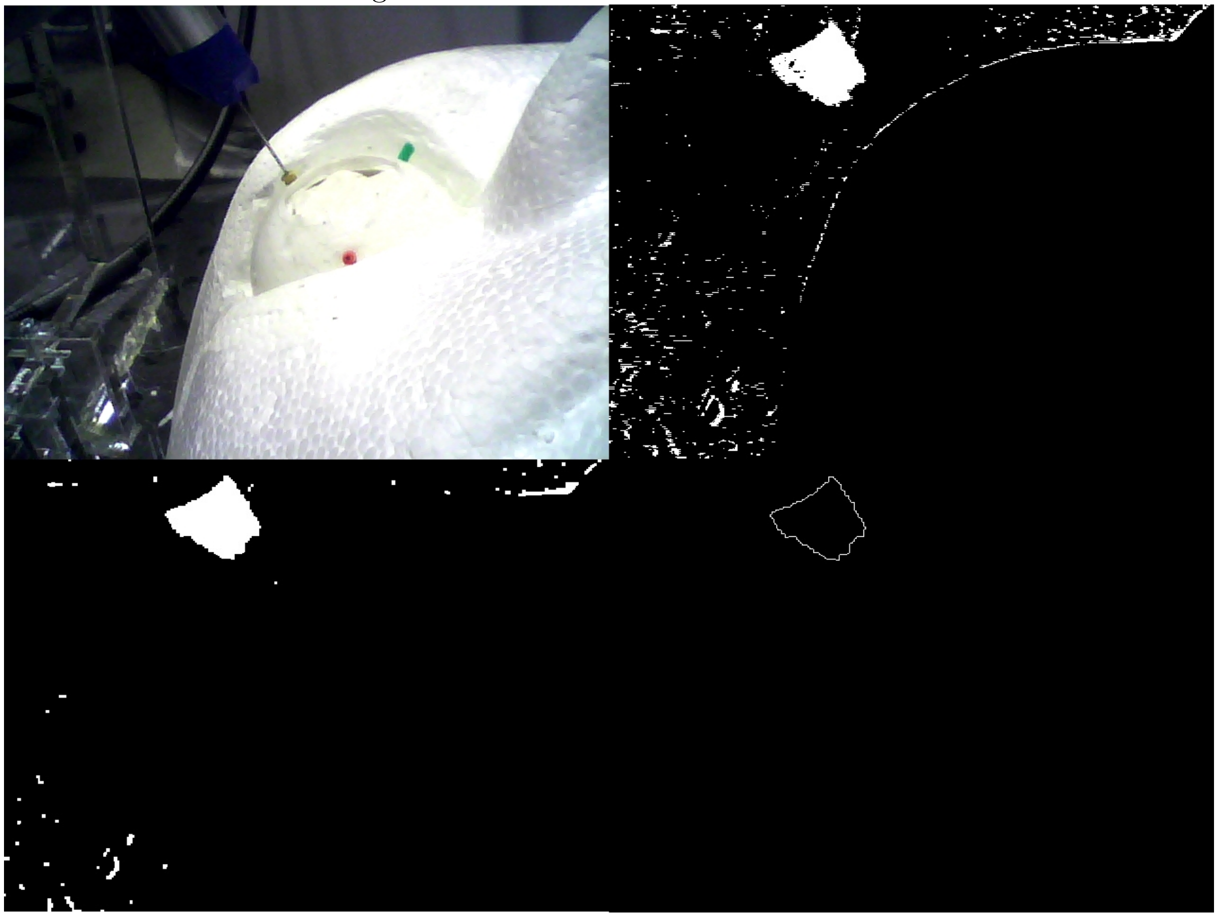
Other than the intrinsic results, the extrinsic results of pair-wise calibration were compared to designed measurements between cameras. There is a possibility of error from device design to creation that cannot be accounted for. Nonetheless, the extrinsic results give an idea of the milimeter accuracy of this calibration technique. As shown in Figure 15, the largest error was observed between Camera 2 and Camera 3, which had a large overlapping field of view but a problem with Camera 2 resolution. The goal of less than 1mm error was not achieved. This was taken into account when analyzing the results of the tracking system.

Figure 15: Extrinsic Test

Camera	1 to 2	2 to 3	3 to 4
Distance Designed (mm)	31.21	31.21	149.56
Distance from Calibration (mm)	31.94	33.64	147.82
Error (mm)	0.72	2.42	1.74

3.2 Detection Test

Figure 16: Blue Marker Detection Test



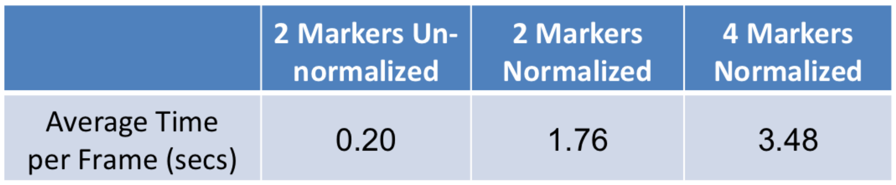
The purpose of the detection test was to determine whether colors can be accurately detected under different illuminations and camera views. Specifically, three sets of four images under varying illuminations and camera views were examined. The original image was compared to the initial threshold, image after morphological operations, the contour, and the location of the detected center. Figure 16 shows an example of a blue marker detection test. Although these tests were successful, the detection tests did not account for cases when the markers were occluded. Tracking accuracy evaluation results and a look at the thresholds show that when a color marker was not present, an incorrect center was still calculated. This was taken into account when analyzing tracking evaluation results.

3.3 3D Reconstruction Test

The original purpose of the 3D reconstruction test was to determine whether the tracking algorithm was working properly and efficiently. However, due to several delays in device building and sharing of Eye Robot 2.1, it was decided to remove most of the 3D reconstruction tests and perform these tests as part of the tracking accuracy evaluation. The revised 3D reconstruction test was to test the time it took to reconstruct markers from 4 corresponding frames, and to perform a test of the graphic display.

As expected, calling a Matlab function from Python was slow. Specifically, the tracking algorithm utilized the Matlab Calibration Toolbox normalize() function to estimate the normalized image projection from distorted values. Figure 17 shows that detection and 3D reconstruction of 2 un-normalized markers took an average of 0.2 seconds per frame. 2 normalized markers took 1.76 seconds, and 4 normalized markers took 3.48 seconds. This is much slower than the .04 seconds per frame collected from the videos. Therefore, the algorithm currently does not have the capability to perform real-time tracking.

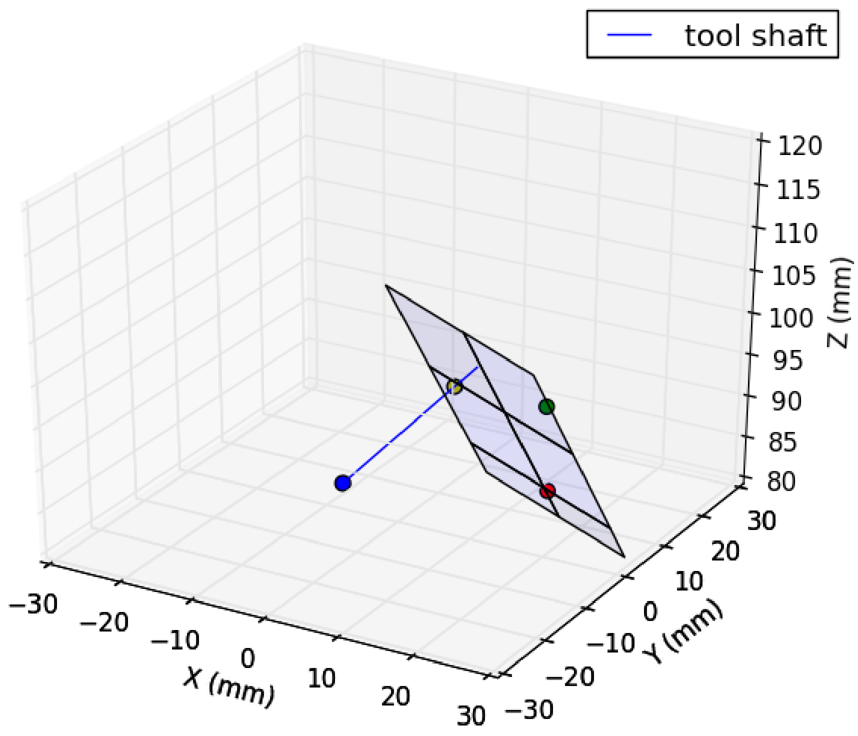
Figure 17: Reconstruction Time Test



	2 Markers Un-normalized	2 Markers Normalized	4 Markers Normalized
Average Time per Frame (secs)	0.20	1.76	3.48

The last piece of this project was to display the positional feedback in a useful way. For four markers, the display shows the center point of each marker in its corresponding color. It also draws a 20 mm tool from the tool through the correct, pre-defined trocar. A plane is also shown to convey the orientation of the eye, assuming that the three trocars define the plane of the eye. Below is a frame representing a test of the graphic output.

Figure 18: Graphic Example



3.4 Tracking Accuracy Evaluation

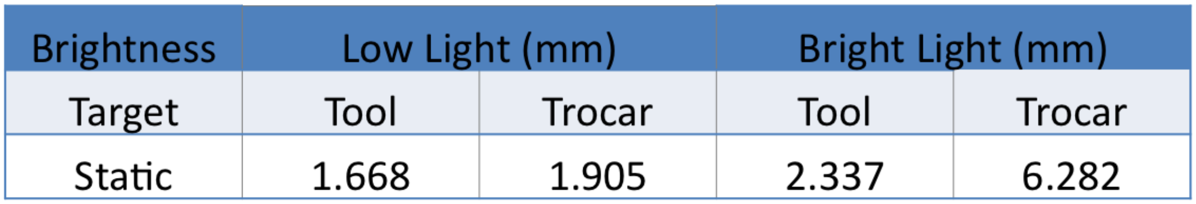
The goal of this project was to track tool tip within 1 mm of accuracy. However, due to the 3 mm difference in the designed and measured position of Camera 2 with respect to Camera 3, it became clear that this goal would be unattainable due to calibration errors. Eye Robot 2.1 was utilized to control translational movement and measure actual displacement. It is important to note that there may be small errors in tracking accuracy of the Eye Robot. However, these errors are not on the same order of magnitude as the calibration errors, and therefore, were excluded from consideration.

In order to evaluate tracking accuracy, the original plan was to track the tool tip for static and dynamic cases under varying circumstances of illumination and occlusion. Furthermore, the plan was to look at both translational and rotational motions of the tool. However, it became clear that it would be difficult to instruct the Eye Robot to rotate about a point, and due to the lack of information about the transformation from the Eye Robot to Camera 1 coordinate system, rotational testing was excluded. Several delays in the schedule also interrupted plans to test for occlusion.

Static testing was conducted by inserting the tool with a blue marker into a yellow trocar, and moving this tool in known displacements for five trials under two light settings. The displacements were calculated using the equation below. Average error in mm is shown in Figure 19, with the most noticeable error of 1.9 mm for low light and 6.3 mm for bright light. This suggests that the yellow trocar was not being detected accurately in bright light conditions. The rest of the errors can be attributed to errors from calibration.

$$d = \sqrt((X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z1 - Z2)^2) \tag{11}$$

Figure 19: Average Static Error Results



Brightness	Low Light (mm)		Bright Light (mm)	
Target	Tool	Trocar	Tool	Trocar
Static	1.668	1.905	2.337	6.282

Next, dynamic tests of tool tracking were performed. Since the transformation from the Eye Robot to Camera 1 is not known, it was not possible to compare the slope between frames to the slope programmed in the Eye Robot. Instead, the average and standard deviation of the slope from frame to frame was examined for each of the five trials under two light conditions.

Figure 20: Average Dynamic Error Results

Currently this table does not exist because it has not been made.

It is clear that the current detection method needs to be improved to minimize maximum tracking errors. Specifically, the color threshold must be modified to identify more specific ranges of colors. Furthermore, the algorithm should incorporate bounding errors on the size of the trocar and marker in order to exclude detection of other objects of the same color. However, when the correct marker is detected, the tracking algorithm proved to be accurate to about 2 mm. Since much of this error can be attributed to calibration, it may be worthwhile to implement a more accurate calibration scheme as well. One other possible source of error is the inevitable shift in camera orientation due to the quality of the camera construction. As described earlier, there was a noticeable shift in camera view during testing. This was minimized by allowing the system to settle for half an hour before calibration before the final tracking evaluation. Tracking tests show that there is potential for an accurate optical tracking system that can be implemented for ophthalmic surgery.

4 Management Summary

4.1 Credits

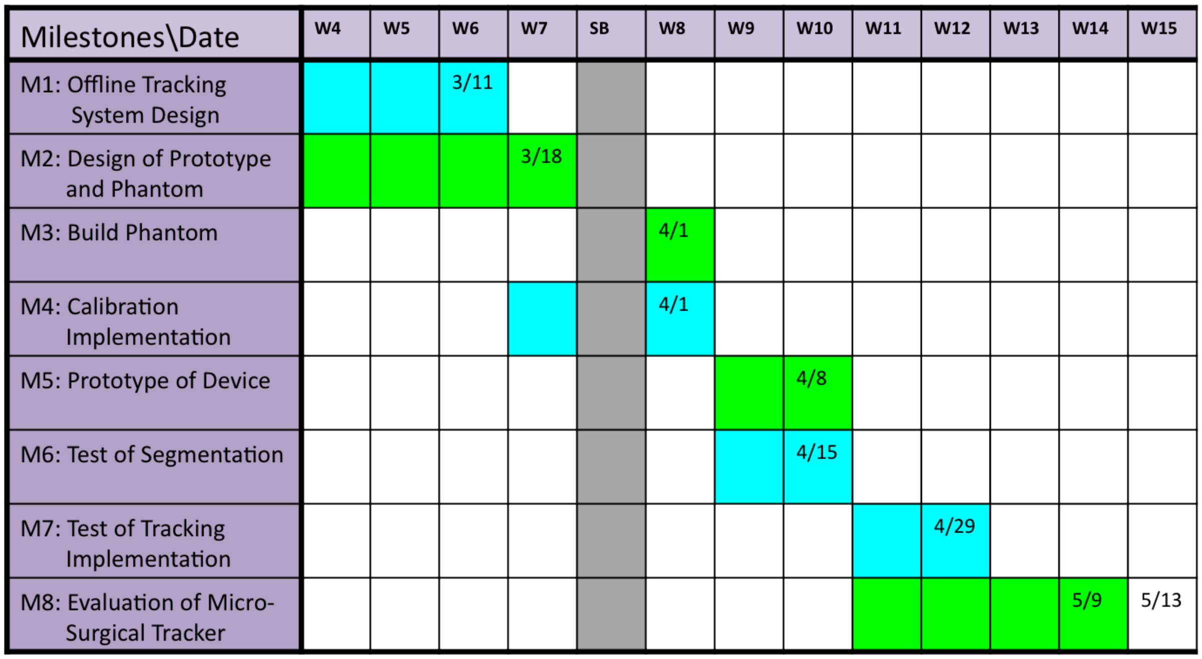
Yejin	Sue	Both
CAD of camera placement	Survey of tracking papers	Research of camera options
Sketch of device	Calibration, detection, 3D point reconstruction design	Obtaining clinical input
Mock-up of device	Implementation of detection	Obtaining/making trocars and tool
CAD of device	Implementation of 3D point reconstruction	Survey of camera specification papers
3D print of stands	Calibration evaluation	Test of camera field of view
Laser cutting of device	Detection evaluation	Tracking data acquisition
Tracking evaluation	Reconstruction evaluation	Proposal report
Purchase of equipment	Wiki-page updates	Proposal presentation
Purchase of phantom	Installation of software dependencies	Checkpoint presentations
Seminar presentation	Implementation of multi-camera video capture	Final report
Seminar report	Seminar presentation	Final poster
	Seminar report	

4.2 Deliverables

Minimum (3/18)	Expected (4/26)	Maximum (5/13)
CAD design of prototype ✔	A scaled prototype ✔	Life-size prototype ✖
Design of phantom ✔	A scaled phantom ✖	Life-size phantom ✔
Specifications of equipment ✔	Offline multi-camera calibration ✔	Evaluation of tracking accuracy ✔
Calibration scheme ✔	Offline segmentation/ tracking algorithms ✔	Real-time tracking ✖
Segmentation/tracking scheme ✔		

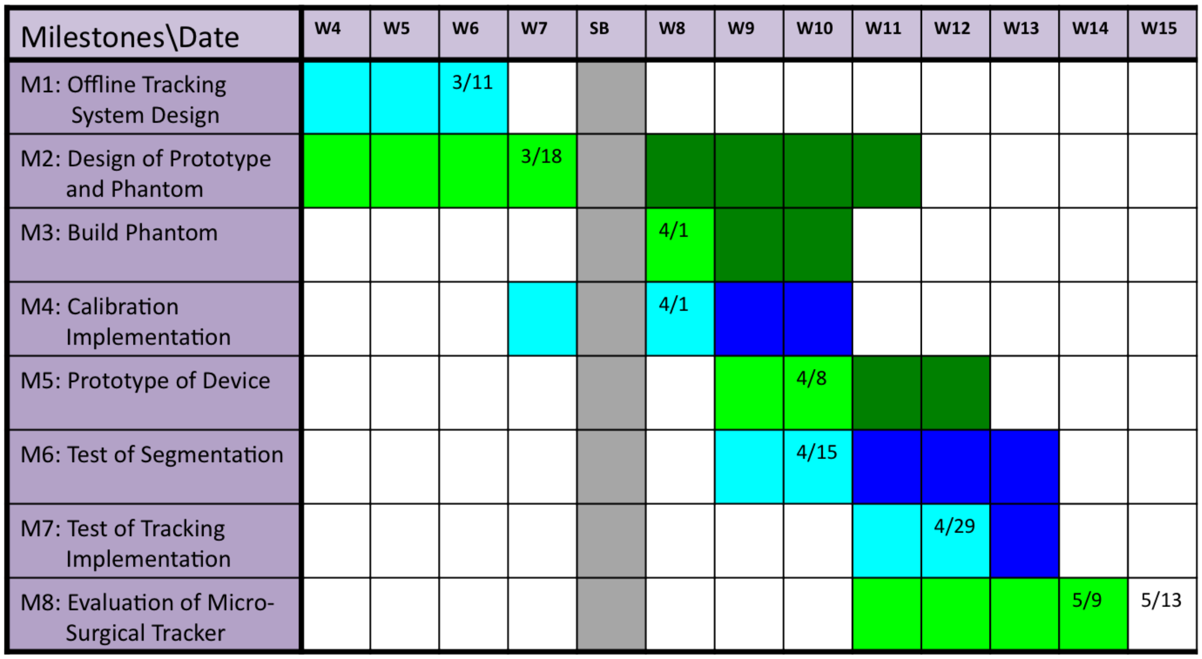
4.3 Timeline

Figure 21: Original Timeline



Milestones\Date	W4	W5	W6	W7	SB	W8	W9	W10	W11	W12	W13	W14	W15
M1: Offline Tracking System Design	🟦	🟦	🟦 3/11		▨								
M2: Design of Prototype and Phantom	🟩	🟩	🟩	🟩 3/18	▨								
M3: Build Phantom					▨	🟩 4/1							
M4: Calibration Implementation				🟦	▨	🟦 4/1							
M5: Prototype of Device					▨		🟩	🟩 4/8					
M6: Test of Segmentation					▨		🟦	🟦 4/15					
M7: Test of Tracking Implementation					▨				🟦	🟦 4/29			
M8: Evaluation of Micro-Surgical Tracker					▨				🟩	🟩	🟩	🟩 5/9	5/13

Figure 22: Final Timeline



Milestones\Date	W4	W5	W6	W7	SB	W8	W9	W10	W11	W12	W13	W14	W15
M1: Offline Tracking System Design	🟦	🟦	🟦 3/11		▨								
M2: Design of Prototype and Phantom	🟩	🟩	🟩	🟩 3/18	▨	■	■	■	■				
M3: Build Phantom					▨	🟩 4/1	■	■					
M4: Calibration Implementation				🟦	▨	🟦 4/1	🔷	🔷					
M5: Prototype of Device					▨		🟩	🟩 4/8	■	■			
M6: Test of Segmentation					▨		🟦	🟦 4/15	🔷	🔷	🔷		
M7: Test of Tracking Implementation					▨				🟦	🟦 4/29	🔷		
M8: Evaluation of Micro-Surgical Tracker					▨				🟩	🟩	🟩	🟩 5/9	5/13

4.4 Future Directions

The nature of our project was to show proof that a multi-camera optical tracking system could provide useful positional feedback to surgeons performing ophthalmic surgery. As such, there are many future directions this project could be taken in.

1) Purchase or build better cameras
There were four major issues with the 7mm boroscope cameras that were purchased: narrow field of view, relatively poor resolution (480x640), durability, and compatibility with software for real-time tracking. Improvement in tracking accuracy and moving to real-time tracking is highly dependent on cameras that fit these requirements.

2) Implement 4-camera calibration
Currently, our device utilizes three pairs of pair-wise camera calibration through Matlab‘s camera calibration toolbox. However, it would be less time consuming and possibly more accurate to implement a multi-camera calibration that uses a calibration object visible to all cameras to calculate homography, and subsequently the appropriate rotation and translation matrices.

3) Calibration update while tracking
One problem that occurred during our project was the movement of cameras during tracking experiments. Although part of the problem would be resolved with cameras that are more durable and a device that can handle more weight, it is reasonable to assume a camera may be moved accidentally during surgery. Therefore, it is necessary to implement a calibration method such that calibration parameters can be updated during tracking procedures.

4) Synchronize cameras
Since the current implementation runs two streams that each grabs images from two cameras, all four cameras are not synchronized. Under the assumption of slow movement, which is reasonable for ophthalmic surgery, this is not an issue. However, it would be better to have all four cameras synched.

5) Implement real-time tracking
The 7mm boroscope cameras were not compatible with python‘s opencv or ffmpeg modules. As such, it was not possible to implement real-time tracking. One simple fix would be to purchase cameras that are known to be compatible with one of these modules. Another more time consuming approach would be to implement the tracking algorithm in C and write functions to grab images.

Time tests also showed that the current implementation is too slow for real-time tracking. One fix to dramatically increase efficiency is to implement the normalize function in Python. The next fix would be to move the whole system to C++, which is much faster than Python.

Another way to improve efficiency of the tracking algorithm is to implement a bounding box for detection. Specifically, information on maximum velocity and acceleration coupled with current position would be necessary to define a bounding box on where the markers could be next. This way, only a portion of the next frame needs to be searched to detect the markers.

6) Design more realistic device to fit the face
The final device design was rigidly attached to a platform with the phantom in order to minimize errors due to movement. However, a real device would need to be attached to the patient's face. Further steps could be taken to attach the device to a phantom rather than to a platform.

4.5 Lessons Learned

One of the biggest obstacles we encountered was the lack of compatibility between the cameras and software. Cameras could not be viewed at the same time unless they are run on different buses. Furthermore, more problems exist with video capture due to the limitations of DivX to two cameras at a time. The cameras were also not recognized by camera capture modules in Python. These were all problems that were not accounted for in our time management plans, so no one had been assigned responsibility for resolving these issues. Time management became stressful when accounting for these unforseen problems. In the future, we should plan for obstacles when dealing with new hardware or software.

Also, several issues arose from using group-owned equipment. For example, there were several instances when UPrint was not available due to use for other projects, the laser cutter was broken, the eye robot was unavailable, and the network was down on the desktop. Although all these issues could not have been avoided, it would have helped to schedule with other groups and communicate needs.

Finally, communication of expectations and task management is key. Although we made a detailed management plan and meeting schedule, the expectations for how rigorously to follow this plan differed. We would have also benefited from a more in depth search of dependencies, as several unexpected dependencies emerged.

References

[1] Neily, Mills, et al. “Incorrect Surgical Procedures Within and Outside of the Operating Room.” Archives of Surgery 16 Nov. 2009: Vol. 144, No.11:1028-1034. Web. 12 Feb. 2013

[2] J. D. Pitcher, J. T. Wilson, S. D. Schwartz, and J. Hubschman, “Robotic Eye Surgery: Past, Present, and Future,” J Comput Sci Syst Biol, pp. 14, 2012.

[3] J.-P. Hubschman, J. Son, B. S. D. Schwartz, and J.-L. Bourges, “Evaluation of the motion of surgical instruments during intraocular surgery,” Eye (London, England), vol. 25, no. 7, pp. 94753, Jul. 2011.

[4] M. Nasseri, E. Dean, S. Nair, and M. Eder, “Clinical Motion Tracking and Motion Analysis during Ophthalmic Surgery using Electromagnetic Tracking System,” in 5th International Conference on BioMedical Engineering and Informatics (BMEI 2012). 2012.

[5] G. M. Saleh, G. Voyatzis, Y. Voyazis, J. Hance, J. Ratnasothy, and A. Darzi, “Evaluating surgical dexterity during corneal suturing,” Archives of ophthalmology, vol. 124, no. 9, pp. 12636, Sep. 2006.

[6] K. Guerin, G. Vagvolgyi, A. Deguet, C.C.G. Chen, D. Yuh, and R. Kumar, “ReachIN: A Modular Vision Based Interface for Teleoperation,” in the MIDAS Journal - Computer Assisted Intervention, Aug. 2010.

[7] J. Y. Bouguet. Camera Calibration Toolbox for Matlab. 2008.

[8] Tomas Svoboda. “A Software for Complete Calibration of MultiCamera Systems.” Talk given at MIT CSAIL. Jan 25, 2005.

[9] K. Zimmermann, J. Matas, and T. Svoboda. “Tracking by an Optimal Sequence of Linear Predictors.” IEEE Transactions on Pattern Analysis and Machine Intelligence. 31(4), 2009

[10] A. Borkar, M. Hayes, and M. T. Smith, “A Non Overlapping Camera Network: Calibration and Application Towards Lane Departure Warning” IPCV 2011: Proceedings of the 15th International Conference on Image Processing, Computer Vision, and Pattern Recognition. 2011.

[11] D.L. Pham, C. Zu, and J.L. Prince. “Current Methods in Medical Image Segmentation.” Annual Review of Biomedical Engineering Vol. 2 pp 315-337. August 2000.

[12] Y. Deng. Color Image Segmentation. Computer Vision and Pattern Recognition, 1999 IEEE Computer Society Conference.

[13] J. Bruce, T. Balch, and M. Veloso, “Fast and inexpensive color image segmentation for interactive robots,” in Proc. IEEE Intl. Conf. Intell. Robot. Syst., 2000, pp. 20612066.

[14] M. K. Hu, “Visual pattern recognition by moment invariants,” Information Theory, IRE Transactions on, vol. 8, no. 2, pp. 179187, 1962.

[15] K. Kim, L. S. Davis. “Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering.” Computer Science Volume 2953, pp 98-109. 2006.

[16] A. Yilmaz, O. Javed, M. Shah. “Object tracking: A survey.” ACM Computing Surveys Volume 38 Issue 4, Article No. 13. 2006.

5 Appendix A

5.1 Device Design

Figure 23: CAD of Device View 1

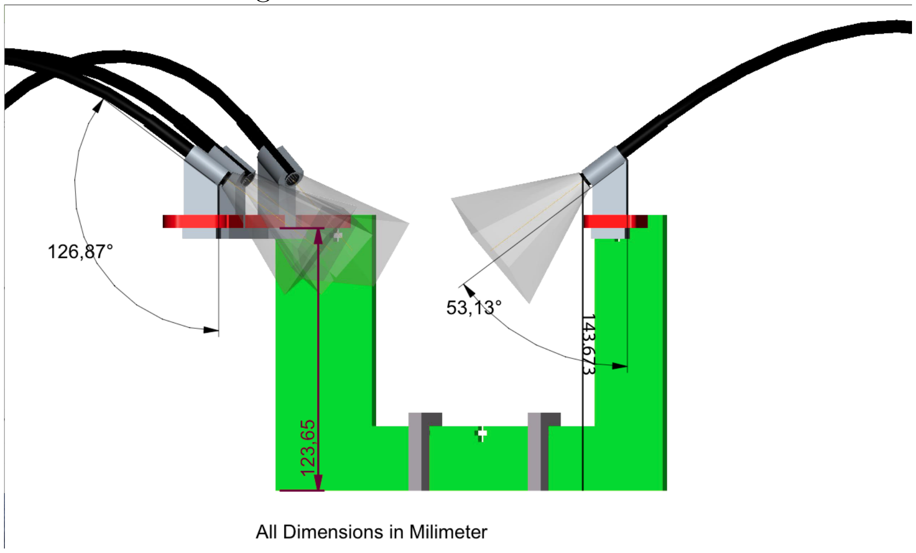
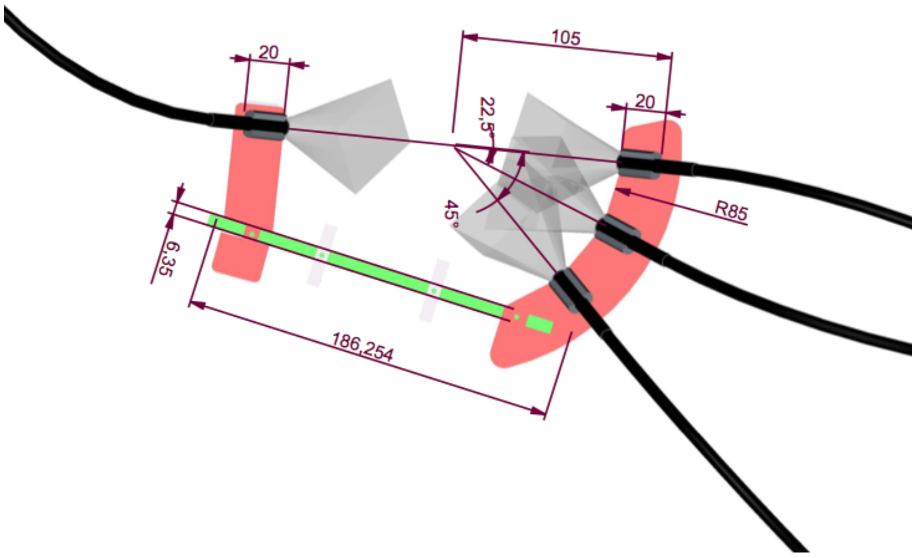
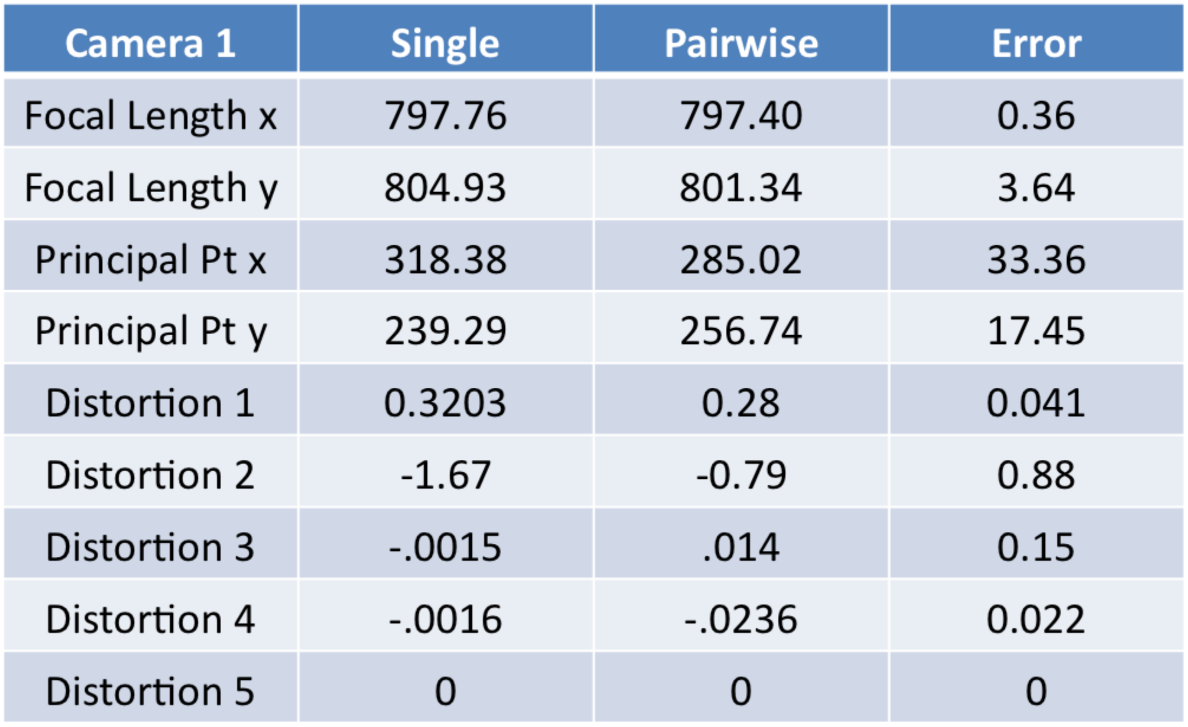


Figure 24: CAD of Device View 2



5.2 Calibration Test Results

Figure 25: Camera 1 Intrinsic Test



Camera 1	Single	Pairwise	Error
Focal Length x	797.76	797.40	0.36
Focal Length y	804.93	801.34	3.64
Principal Pt x	318.38	285.02	33.36
Principal Pt y	239.29	256.74	17.45
Distortion 1	0.3203	0.28	0.041
Distortion 2	-1.67	-0.79	0.88
Distortion 3	-.0015	.014	0.15
Distortion 4	-.0016	-.0236	0.022
Distortion 5	0	0	0

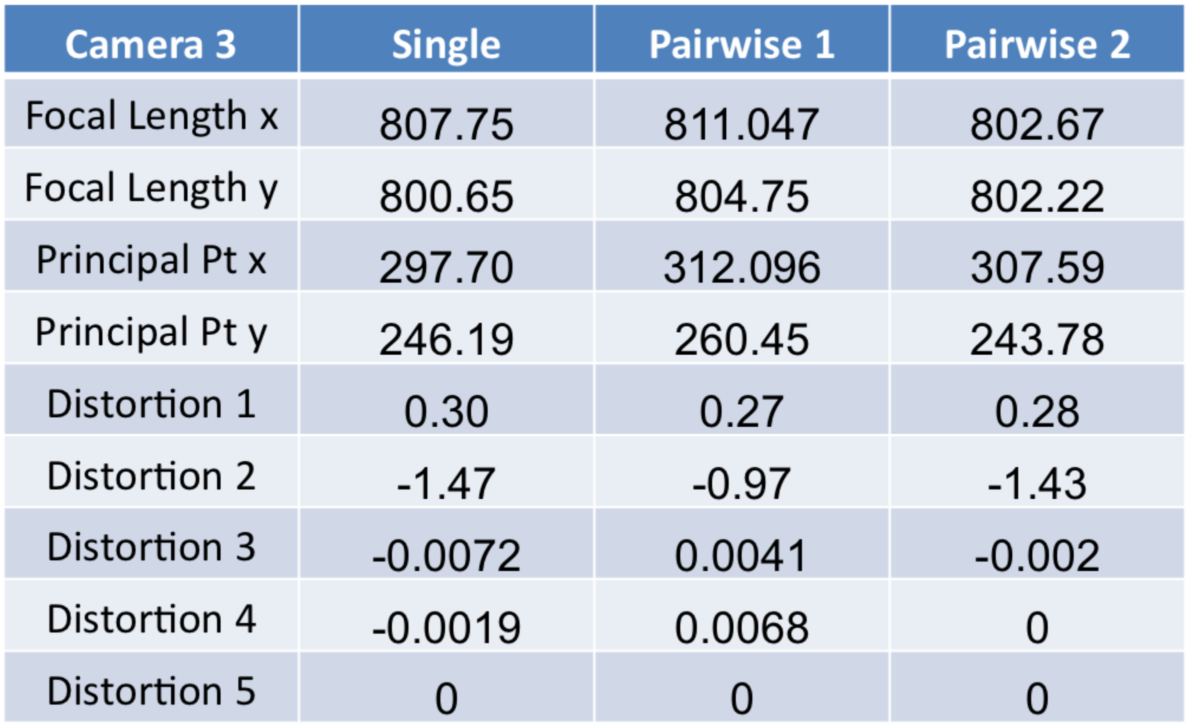
Figure 26: Camera 2 Intrinsic Test

Camera 2	Single	Pairwise 1	Pairwise 2
Focal Length x	803.89	808.14	810.97
Focal Length y	810.45	808.71	812.41
Principal Pt x	305.89	261.84	263.38
Principal Pt y	233.42	218.52	204.82
Distortion 1	0.35	0.25	0.21
Distortion 2	-2.06	-0.47	-0.36
Distortion 3	-0.0023	-0.016	-0.024
Distortion 4	0.0051	-0.028	-0.028
Distortion 5	0	0	0

Figure 27: Camera 2 Error

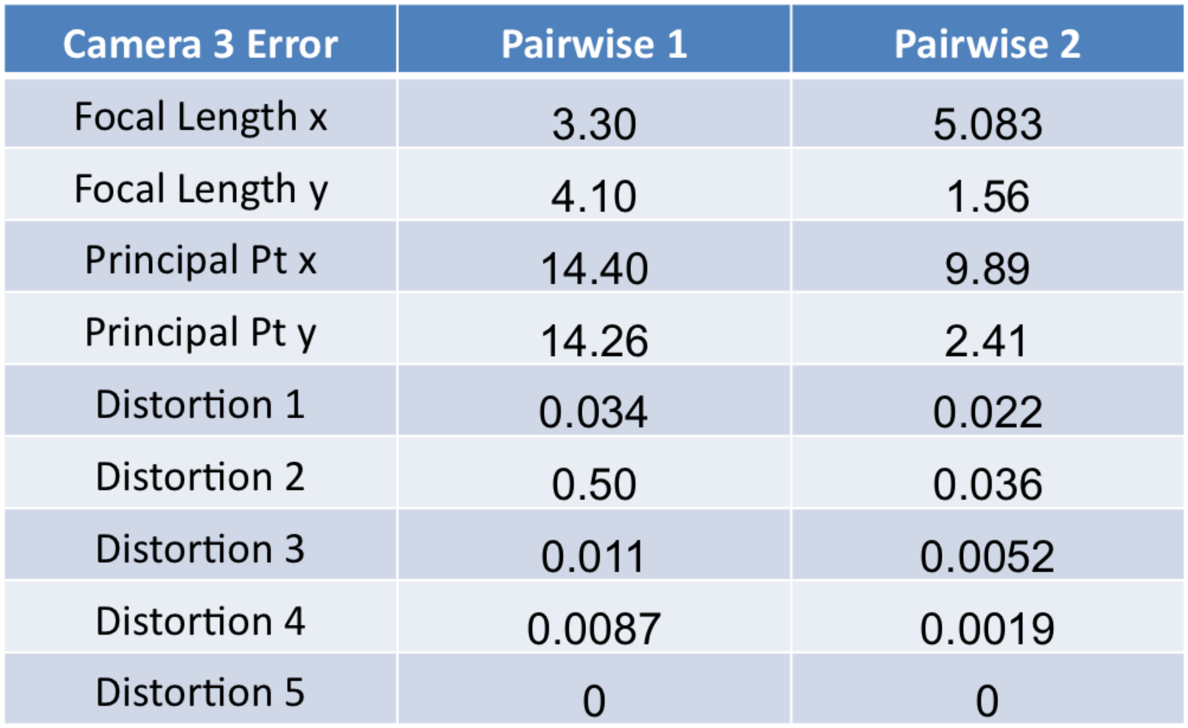
Camera 2 Error	Pairwise 1	Pairwise 2
Focal Length x	4.25	7.08
Focal Length y	1.73	1.97
Principal Pt x	44.05	42.5
Principal Pt y	14.9	28.6
Distortion 1	0.098	0.14
Distortion 2	1.59	1.71
Distortion 3	0.013	0.21
Distortion 4	0.034	0.033
Distortion 5	0	0

Figure 28: Camera 3 Intrinsic Test



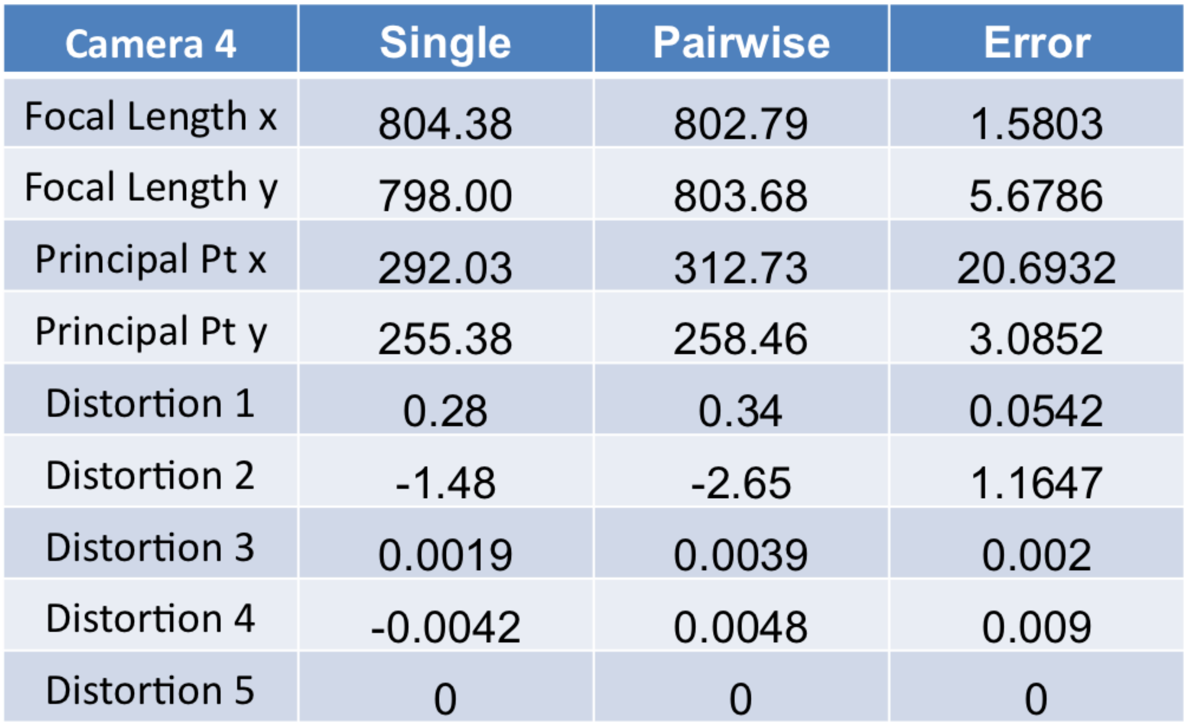
Camera 3	Single	Pairwise 1	Pairwise 2
Focal Length x	807.75	811.047	802.67
Focal Length y	800.65	804.75	802.22
Principal Pt x	297.70	312.096	307.59
Principal Pt y	246.19	260.45	243.78
Distortion 1	0.30	0.27	0.28
Distortion 2	-1.47	-0.97	-1.43
Distortion 3	-0.0072	0.0041	-0.002
Distortion 4	-0.0019	0.0068	0
Distortion 5	0	0	0

Figure 29: Camera 3 Error



Camera 3 Error	Pairwise 1	Pairwise 2
Focal Length x	3.30	5.083
Focal Length y	4.10	1.56
Principal Pt x	14.40	9.89
Principal Pt y	14.26	2.41
Distortion 1	0.034	0.022
Distortion 2	0.50	0.036
Distortion 3	0.011	0.0052
Distortion 4	0.0087	0.0019
Distortion 5	0	0

Figure 30: Camera 4 Intrinsic Test



Camera 4	Single	Pairwise	Error
Focal Length x	804.38	802.79	1.5803
Focal Length y	798.00	803.68	5.6786
Principal Pt x	292.03	312.73	20.6932
Principal Pt y	255.38	258.46	3.0852
Distortion 1	0.28	0.34	0.0542
Distortion 2	-1.48	-2.65	1.1647
Distortion 3	0.0019	0.0039	0.002
Distortion 4	-0.0042	0.0048	0.009
Distortion 5	0	0	0

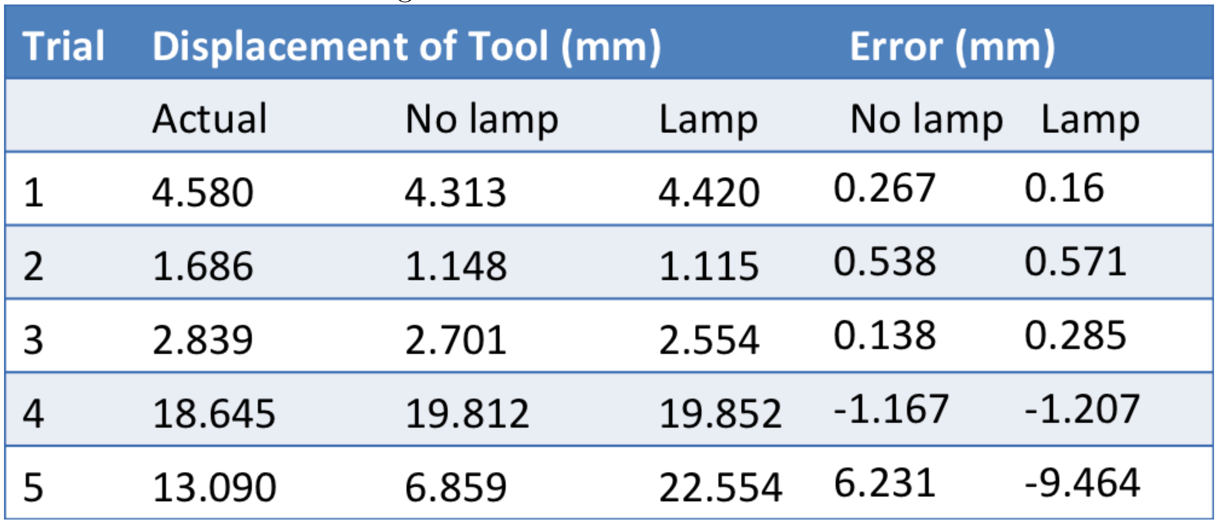
Look in the calibration.xlsx file in the Calibration folder for more information.

5.3 Detection Test Results

Look in the Detection folder for a complete list of all images and output used in the detection test.

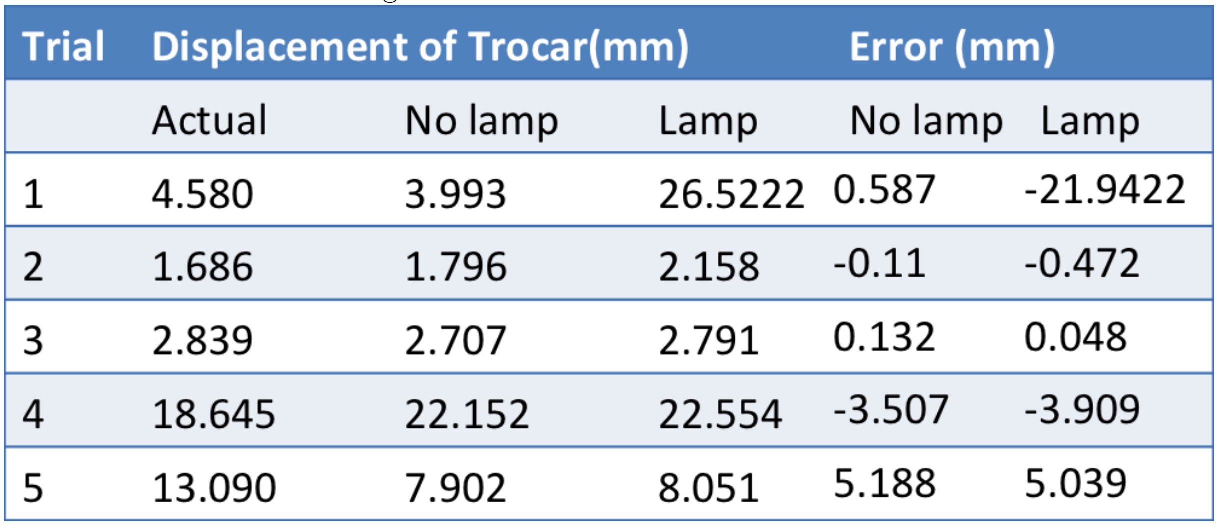
5.4 Tracking Accuracy Test Results

Figure 31: Tool Static Test Results



Trial	Displacement of Tool (mm)			Error (mm)	
	Actual	No lamp	Lamp	No lamp	Lamp
1	4.580	4.313	4.420	0.267	0.16
2	1.686	1.148	1.115	0.538	0.571
3	2.839	2.701	2.554	0.138	0.285
4	18.645	19.812	19.852	-1.167	-1.207
5	13.090	6.859	22.554	6.231	-9.464

Figure 32: Trocar Static Test Results



Trial	Displacement of Trocar(mm)			Error (mm)	
	Actual	No lamp	Lamp	No lamp	Lamp
1	4.580	3.993	26.5222	0.587	-21.9422
2	1.686	1.796	2.158	-0.11	-0.472
3	2.839	2.707	2.791	0.132	0.048
4	18.645	22.152	22.554	-3.507	-3.909
5	13.090	7.902	8.051	5.188	5.039

Figure 33: Dynamic Test Results

xx.png Not currently included.

6 Appendix B

6.1 Calibration

Single Camera Calibration
Step 1: take 20 pictures of the checkerboard from each camera
Step 2: open calib_gui and load images
Step 3: manual corner detection
Step 4: calibration
Step 5: analyze error for outliers
Step 6: re-define corners for outliers. if still problematic, exclude images.
Step 7: final optimized calibration

Pair-wise Camera Calibration
Step 1: take 20 pictures of the checkerboard from camera pairs (1,2), (2,3), and (3,4)
Step 2: perform single camera calibration for each set
Step 3: open stereo_gui and load left and right calibrations appropriately
Step 4: calibrate
Step 5: optimize calibration

6.2 Data acquisition

Step 1: start up in Windows on desktop with modified CISST code
Step 2: attach four cameras, each to a different bus
Step 3: run SVLxStereoCameraViewer.exe (found in Users/dev/cisst/build/cisst/bin/Release/)
Step 4: answer dialog with proper device indices, video names (.cvi), and frame rates (25 fps)
Step 5: take pictures by pressing space
Step 6: record videos by pressing 'r' to start, 'p' to pause
Step 7: rename files to reflect correct camera number as shown in Figure 11

6.3 Preprocessing videos

Step 1: run SVLxVideoConverter.exe (found in Users/dev/cisst/build/cisst/bin/Release/)
Step 2: open appropriate .cvi video
Step 3: insert converted name .avi video
Step 4: save number of frames for later use

6.4 Tracking Algorithm Use

Segmentation and Tracking of Image:
Step 1: modify path to image and calibration results in main_image.py

Step 2: run main_image.py
Note: for two markers instead of four, modify and run main_image2.py
More instructions can be found in README files

Segmentation and Tracking of Video: Step 1: modify path to video, number of frames, and calibration results in main_video.py
Step 2: run main_video.py
Note: for two markers instead of four, modify and run main_video2.py
More instructions can be found in README files