dVRK Stereo Camera Hand-Eye Calibration and Model Registration

Group 12: Mengze Xu & Peter Ahn

Mentors: Preetham Chalasani and Anton Deguet

Abstract

da Vinci Surgical System is a widely used surgery robot system, and da Vinci Research Kit (dVRK) provides open- source control tools to it. Though da Vinci System provides visual feedback to surgeons, the vision system is not registered to its robot arm and the visual information is only used by surgeons instead of the whole system. In this project, we equipped da Vinci system with registered stereo camera to enable the robot arm to see the patient. Two cameras are set up as the stereo camera and marker tags are used to implement hand-eye calibration between cameras and robot arm. A feature detection algorithm is developed to automatically register phantom to robot system. The error of the whole system is around 5 mm. These are our minimum and expected deliverable and we failed to complete maximum deliverable.

1 Introduction

1.1 Background

The da Vinci Surgical System is a widely used surgery robot that helps surgeons to perform delicate and complex operations through a few small incisions[1]. da Vinci Research Kit (dVRK) is an "open-source mechatronics" system, consisting of electronics, firmware, and software that is being used to control research systems based on the first-generation da Vinci system[2]. The current optical system of da Vinci provides stereo scope view with minimum depth information, which is not enough for robot to sense the world and make decision. Therefore, the registered stereo camera will be very helpful, which enables the robot to see the patient and to have more autonomy.

Many works have been done to increase the autonomy of da Vinci system. Currently we have a system which can scan the phantom of organ surface by touching and provides the robot with the surface model. There are two problems of this system: first it needs an initial estimation of the phantom position to start touching, and an estimation of how large the phantom is to avoid moving out.

Second is that it's not always feasible to touch the surface in real case, especially during the surgery. We tried to solve these two problems using registered stereo camera system.

Finally we set up two Intel RealSensor R200 camera by 90 degree as a stereo camera. A marker tag is attached to robot arm (da Vinci Patient Side Manipulators, PSM). The marker's pose relative to camera can be detected, and the robot pose is reported by dVRK. From these two poses, the transformation between cameras and robot base is obtained through hand-eye calibration. Then a phantom is set up, and the four corners of the phantom will be automatically detected in the images of two camera. The 3D position can be calculated to register the phantom to the robot.

1.2 Problem Description

Current surface detection software lacks accurate information of phantom position. And in some cases, it's not feasible to touch the surface and the software can not work.

1.3 Project Goal

Our goal is to provide accurate information of phantom position (within 2 mm) to the robot. Then we want to develop a system to scan the surface without touching.

2 Hardware

2.1 da Vinci PSM

da Vinci PSM is a 6 degree of freedom robot arm. PSM has two modes. Figure 1 shows an example of PSM and tool tips. In our project, the PSM is manually moved. The pose of the robot is reported by dVRK.
The error of the pose is reported around 1mm by other users. We tested it by moving around the PSM while the tool tip remains the same position. The result is in table 1 and is within the reported error. We suppose the PSM works normally.

Direction	x	y	z	norm
Std [mm]	1.3	0.6	1.0	1.7

Table 1: PSM Tool Tip Position

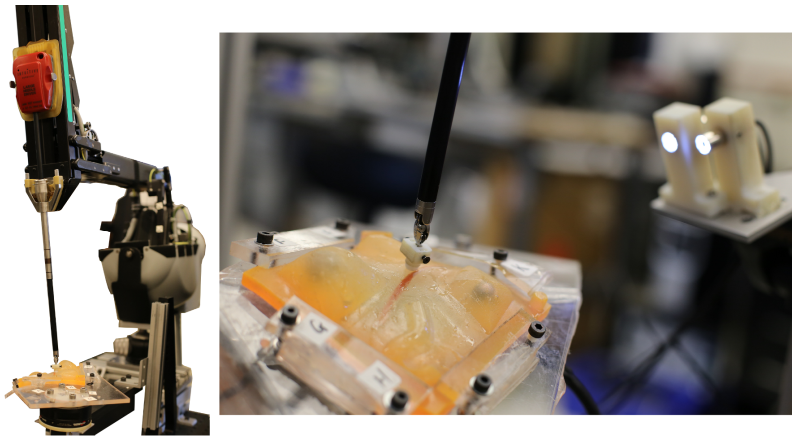

Figure 1: da Vinci PSM [3]

2.2 RealSense Camera (R200)

RealSense Camera (R200) is a stereo camera developed by Intel. It uses structure infrared light to produce depth information and provides easy-to-use Robot Operation System (ROS) interface[5]. In our project, since the error of depth is much larger than our goal, we abandon the it and only use R200 as RGB camera. The resolution of RGB camera is 640×480 with fixed focal length.

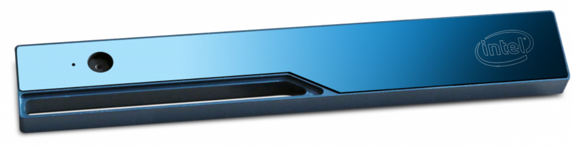

Figure 2: RealSense Camera (R200) [4]

We calibrated the two cameras using 9×7 3mm checkerboard and Matlab Camera Calibration Tool. The results are below with the parameters reported by Intel. Lens distortion is modified using ROS package image_proc[6].

	Parameters	fx [mm]	fy [mm]	cx [pixel]	cy [pixel]
Camera1	Calibration	619.8	622.6	318.6	249.7
	Reported	623.0	629.1	321.3	250.9
Camera2	Calibration	626.7	628.9	340.0	233.3
	Reported	629.6	635.8	341.4	228.9

Table 2: PSM Tool Tip Position

2.3 Marker and Adapter

The adapter is designed to fix the marker onto the PSM tool.

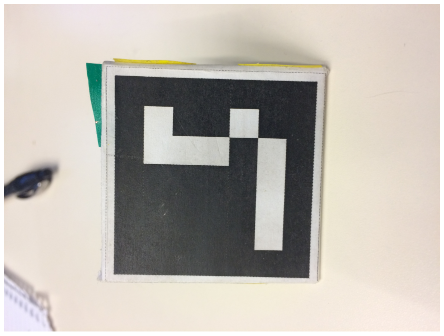

Figure 3: Marker with Adapter

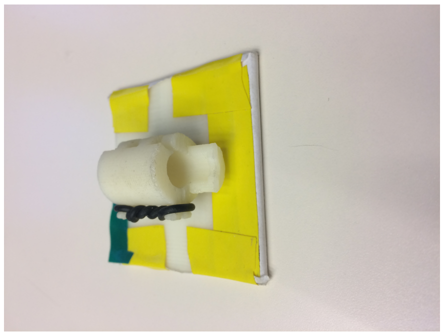

Figure 4: Marker and Adaptor

3 Method

3.1 Marker Detection

We used Aruco_ros package [7] which is able to detect the marker pose relative to camera. The marker is a black square (Fig 3) with white background similar with checkerboard. Therefore the edge of markers are very clear in images. As Fig. 5 shows, Aruco package works in this way [8]:

1. finds edge points from (a) using Canny edge detection algorithm to get (b)
2. extracts contours from (b) edge points to get (c)
3. keeps outside rectangular contours to get (d)

4. calculates marker pose for both (d) and pre-defined marker size (length of black square).

5. uses inside pattern to recognize ID (not used in this project).

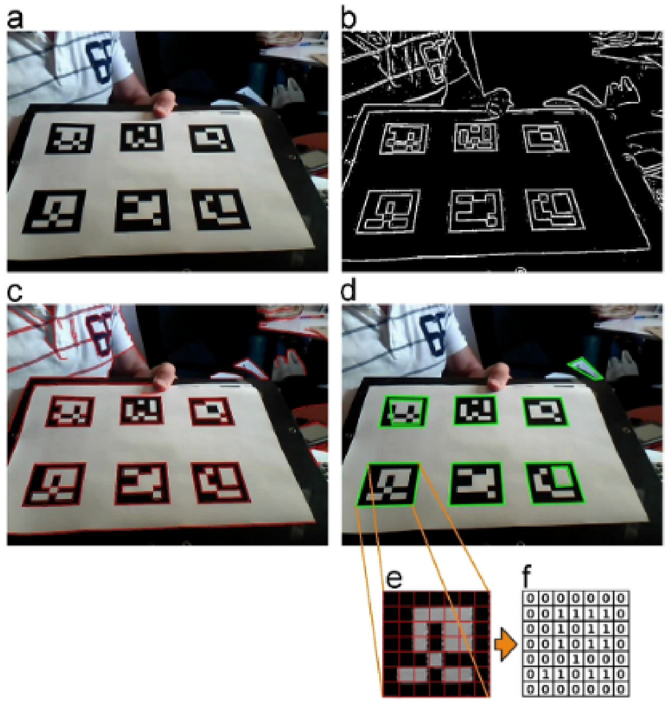

Figure 5: How Aruco works

3.2 Hand-eye Calibration

Hand-eye calibration registers sensors (in our case, cameras) to motion system (robot arm). As Fig. 6 shows, the transformation X from robot arm end-effector to marker and the transformation Z from robot base to camera are constant but unknown. What we know it the transformation A from marker to camera and the transformation B from end-effector to robot base. We have

$$AX = ZB$$

X and Z can be solved from this equation [9]. To do this, first we need to move the robot to different poses to get as leat 3 sets of A and B. The more pose, the less the error is. In practice, we used 10 different poses. It should be noticed that these poses can not be too close or similar.

To calculate X and Z, we first calculate the rotation part R_X and R_Z. We have $R_A R_X = R_Z R_B$.We define Kronecker product of two matrix A($m \times n$ and B $p \times q$ as a $mp \times nq$ matrix:

$$A \bigotimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{bmatrix}$$

In our case, A and B are both 3×3 matrix. If we moved the marker to N different poses, we can define K as a 9×9 matrix

$$K = \sum_{i=1}^{N} R_{B_j} \bigotimes R_{A_j}$$

Then we did Singular Value Decompostion (SVD) to K as

$$K = U \Sigma V$$

It can be proved [9] that K must have singular value N, which is the number of poses, and the left singular vector u_N and right singular vector v_N (9×1) can be reshaped to rotation matrix R_X and R_Z. A scaling is needed here to make sure $|R| = 1$.

Once rotation part is known, translation is easy to calculate. First we rewrite the equation $AX = ZB$ as

$$\begin{pmatrix} I_{3\times3} & -R_{A_i} \end{pmatrix} \begin{pmatrix} t_Z \\ t_X \end{pmatrix} = t_{A_i} - R_Z t_{B_i}$$

This is simple linear system and can be solved by least square.

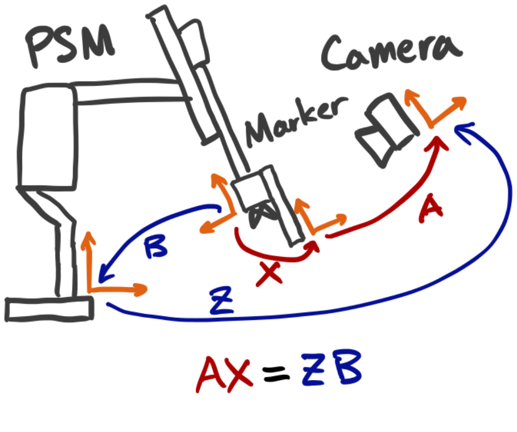

Figure 6: Hand-eye Calibration

3.3 Stereo Camera Setup

As Fig. 7 shows, we set up two cameras by 90 degree. This setup had long baseline and was not sensitive. The cost was the small field of view. In our case, this was not a serious problem

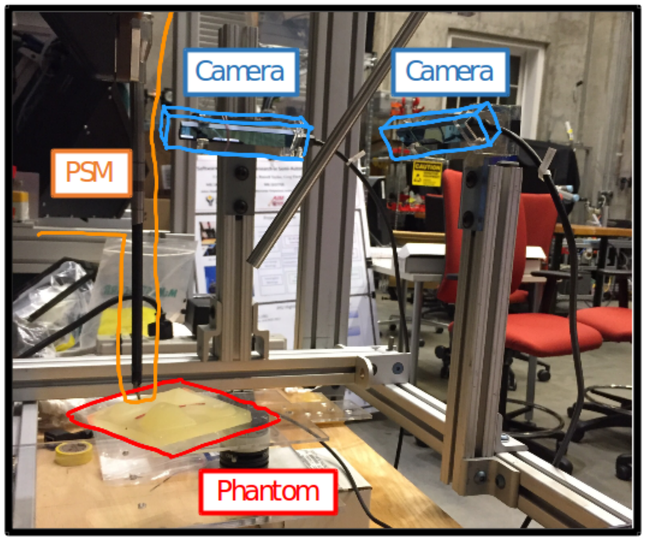

Figure 7: Stereo Camera Setup

Camera parameters of two camera, $P^{(1)}$ and $P^{(2)}$, were known from camera calibration. The pose of two cameras, $Y^{(1)}$ and $Y^{(2)}$, were known from hand-eye calibration, and the position of the corner in two images, (u_1, v_1) and (u_2, v_2). Then we would use these data to calculate the corner position (x, y, z) in robot frame.

Using pin hole model and homogeneous coordinate, we have

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \sim P^{(i)} * (Y^{(i)})^{-1} * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

We set $T = P * Y^{-1}$ and rewrite it in linear form as

$$\begin{pmatrix} T_{11}^{(1)} - u_1 T_{31}^{(1)} & T_{12}^{(1)} - u_1 T_{32}^{(1)} & T_{13}^{(1)} - u_1 T_{33}^{(1)} \\ T_{21}^{(1)} - v_1 T_{31}^{(1)} & T_{22}^{(1)} - v_1 T_{32}^{(1)} & T_{23}^{(1)} - v_1 T_{33}^{(1)} \\ T_{11}^{(2)} - u_2 T_{31}^{(2)} & T_{12}^{(2)} - u_2 T_{32}^{(2)} & T_{13}^{(2)} - u_2 T_{33}^{(1)} \\ T_{21}^{(2)} - v_2 T_{31}^{(2)} & T_{22}^{(2)} - v_2 T_{32}^{(2)} & T_{23}^{(2)} - v_2 T_{33}^{(2)} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u_1 * T_{34}^{(1)} - T_{14}^{(1)} \\ v_1 * T_{34}^{(1)} - T_{24}^{(1)} \\ u_2 * T_{34}^{(2)} - T_{14}^{(2)} \\ v_2 * T_{34}^{(2)} - T_{24}^{(2)} \end{pmatrix}$$

This is a linear system and can be solved by least square.

3.4 Corner Extraction

To improve the accuracy of corner extraction, we added additional black ground to the phantom. As Fig. 8 shows, first, we isolate the phantom surface from the background by color threshold. Then we extracted the edges points.

To find 4 corners, we used the centre of the these edge points as new origin and used polar coordinate (ρ, θ). As Fig 9 shows, we plotted the relationship between ρ and ϕ of edge points. The peaks constitute to the 4 corners of the phantom.

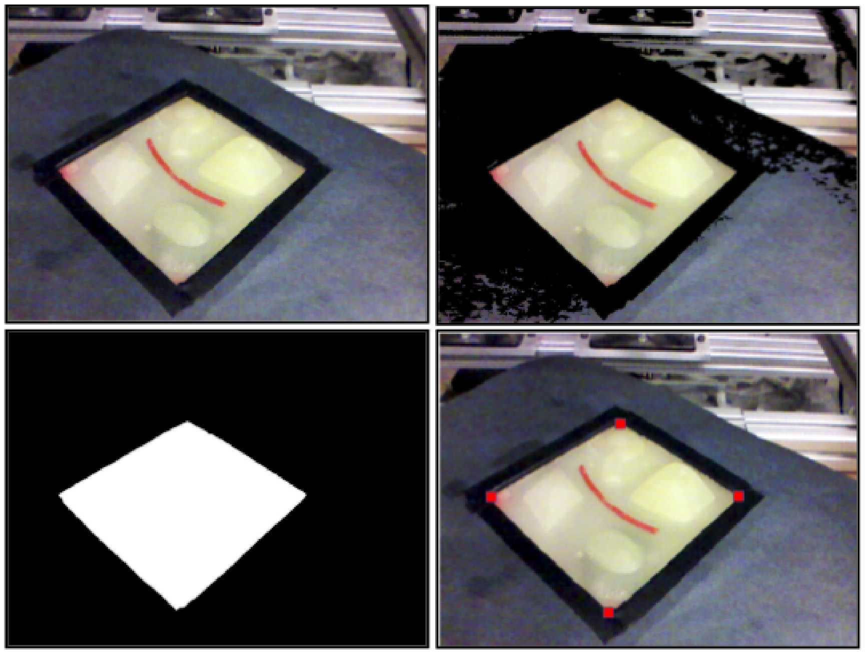

Figure 8: Phantom Corner Detection Process (Top Left: input image; top right: set surrounding background to zero; bottom left: isolate the surface boundaries; bottom right: derived four corners)

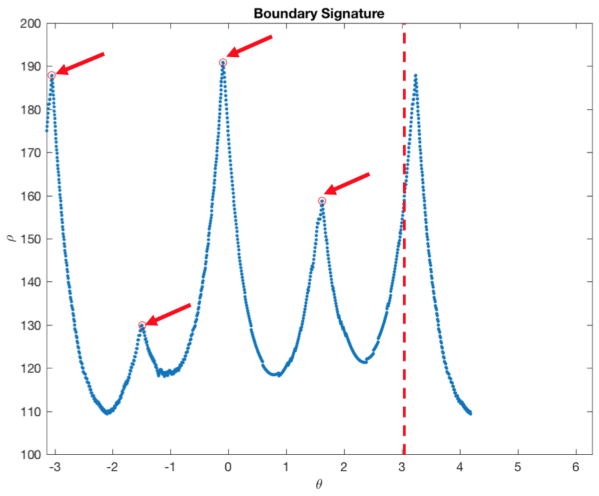

Figure 9: Boundary Signature (red arrows: corner points in polar coordinates)

4 Result

In experiment, we first did hand-eye calibration to get camera pose Y_1 and Y_2, then we set up the phantom and got RGB images. Corners are extracted from two images and the 3D position are calculated in robot frame. Then we move PSM to touch each corners and used the tool tip position as ground truth.

4.1 Whole System Error

10 times of experiment were taken. For each experiment, we reset the adapter and the phantom position. The error vector was in random direction and the average norm is 5.1 mm.

Another 10 times of experiment were taken. Instead of using the result of two camera pose, we only did hand-eye calibration for one camera. Then we did stereo calibration using marker tag to get relative pose between two cameras and calculated the pose of other camera. Other setup were the same. In this case, the error vector was random in XY plane and was similar in Z-axis (vertical). The detect corners were lower than ground truth by average 5.0 mm and the mean norm of error in XY plane was 2.2 mm.

During the first 10 experiment, the error of one experiment was significantly smaller than other groups (mean norm 2.5 mm). Since the camera and robot base were fixed, we used the camera pose from this experiment to do another 5 times of experiment. Without hand-eye calibration, we only reset the phantom and did corner detection. The mean error of these 5 experiments were only 2.3 mm. The result of this experiment illustrated that the error of hand-eye calibration played a more significant role in the whole system error.

4.2 Hand-eye Calibration Error

We saw the fact that the error of hand-eye calibration played an important role. In each experiment, we moved PSM to let the marker be seen by two cameras at the same time and did hand-eye calibration simultaneously for the two cameras. We moved PSM to 10 different poses. Since there was no ground truth for hand-eye calibration to compare, we looked at the consistency of the results. The error were in random direction and the norm were in table 3.

	Rotation [rad]	Translation [mm]
Camera1	0.025	2.7
Camera2	0.023	3.1

Table 3: Hand-eye Calibration Error

For the later 10 experiments in which we did one hand-eye calibration and on stereo calibration, the results were in table 4.

	Rotation [rad]	Translation [mm]
Camera1	0.027	2.9
Stereo	0.008	2.5

Table 4: Hand-eye and Stereo Calibration Error

4.3 Corner Extraction

The results of corner extraction were compared with results of manually selection. The mean deviation was 3 pixel. In our case, the distance between phantom and camera was around 240 mm, and each pixel was equal to 0.3 mm. The error of corner extraction would finally result an error around 1 mm in final results.

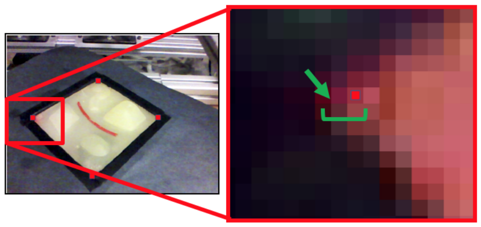

Figure 10: Example error in Corner Detection

4.4 Failure Setup

At the beginning, we try to use the depth image of R200 camera directly. The depth camera works at range from 50 cm to 1.5 m. At the distance of 50 cm, the phantom was only 40 pixel by 40 pixel. The mean error of phantom position was around 3 cm. Also the depth camera just imaged the phantom surface into a plane.

When we set up two cameras at the first time, we used a short baseline design showed in Fig. The mean error of phantom position is 15 mm. It turned out that the system was super sensitive to errors in the direction along baseline.

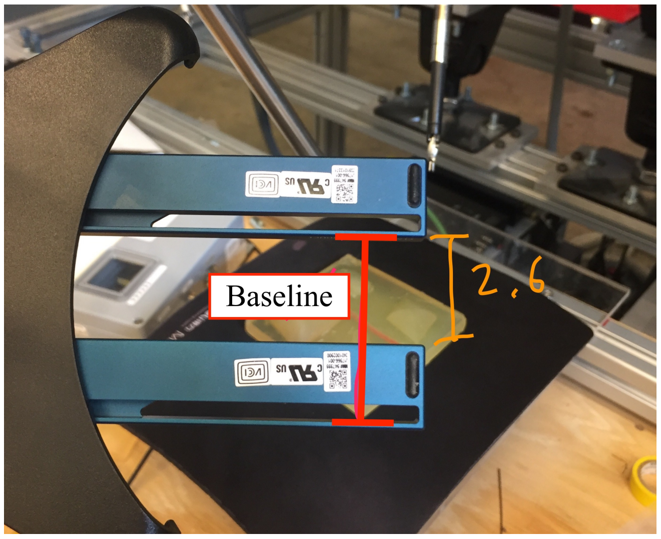

Figure 11: Failure Setup of Two Cameras

5 Discussion and Future Work

The result of the whole system is not bad, but still needs improvement. Considering that the error of pose from dVRK is around 1 mm, the ideal error of the system should be around 2 mm. From the results above, error of hand-eye calibration contributes a large part of the total error. Several points will be discussed and suggestions on future work will be given.

5.1 Marker Detection Error

AruCo is a widely used package to detect marker pose in field of augmented reality. It's convenient to use, but there is not error analysis available. During the experiment, we did very simple test to make sure the error was not too large.

The translation error was less than 3 mm. The marker detection error might contribute a lot to hand-eye calibration error. For future work, a calibration object can be used (Optical Marker or EM marker) to control the error of this part.

5.2 Better Camera

At beginning, we want to directly use depth image and chose R200. Later we set up our own stereo camera system and only used RGB camera of R200. However, the resolution of R200 (640*480) is quite poor. For future work, R200 can be substitue with 4K better cameras. This can reduce the error of corner reduction by 2-3 times and can also reduce the total error if the camera has less lens distortion.

5.3 Surface Detection

This is our maximum deliverable. Once the stereo camera system gets more accurate, we can find each points from two images, pair them and get the point cloud of the surface. The largest challenge is to pair the points from two images. There are few obvious features on phantom and every point looks similar. The poor resolution of the camera makes it worse. This will very help if implemented.

5.4 Code Frame

To fast prototype, we used a mixture of C++ (ROS) and Matlab with help of Matlab ROS API. For future work, the codes should be integrated into ROS packages in C++.

6 Lessons Learned

6.1 Think Patiently Before Prototype

We should think through the performance of each hardware and software package before integrating them, especially their error and sensitivity. However, we hurried to prototype and wanted to get a result. We wasted a lot of time on meaningless attempt.

6.2 Do not Trust Documentation

We learned to test anything before using it. At beginning, we just trusted the documentation. Our mentors taught us not to trust robot and test everything, even the dVRK.

6.3 How to Analyze Coupled System

We learned a lot from mentors on how to analyze the system. We learned on how to design experiments to quantitatively test each unit. We also learned to avoid designing coupled component which can't be tested separately.

7 Conclusion

In this project, we established a prototype of vision system for dVRK, which was able to detect single point within 5 mm. Although the accuracy failed to reach within 2 mm, it provided useful information on the performance of each unit and several software packages to be used in future development.

Management Summary

Deliverable

Our minimum deliverable is to complete hand-eye calibration between PSM and camera. It was completed in March 24, a week later than proposal.

Our expected deliverable is to complete phantom detection. It was completed in April 28, a week later than proposal.

Our maximum deliverable is to improve the detection error from 5 mm to 2mm, and then to detect the surface of the phantom. We spent last two weeks on it but failed to work it out.

Credit

Mengze Xu is responsible of all software except corner detection and error analysis. Peter Ahn is responsible of all hardware and codes for corner detection. Experiments are done by both.

Acknowledgement

Anton and Preetham gave us a lot guidance on how to design and analyze the vision system. They were always responsible and kind. We had a good time to work with them and thank them very much.

References

[1] https://www.intuitivesurgical.com/products/davinci_surgical_system/

[2] https://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki

[3] https://s3.amazonaws.com/vu-my/wp-content/uploads/sites/2254/2017/01/23115309/fig1_PSM_robot_setup.png

[4] https://software.intel.com/en-us/realsense/r200camera

[5] http://wiki.ros.org/RealSense

[6] http://wiki.ros.org/image_proc

[7] http://wiki.ros.org/aruco_ros

[8] Garrido-Jurado, Sergio, et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion." Pattern Recognition 47.6 (2014): 2280-2292.

[9] Shah, Mili. "Solving the robot-world/hand-eye calibration problem using the Kronecker product." Journal of Mechanisms and Robotics 5.3 (2013): 031007.