**EN.530.666**
**Group 1**
**Magnetically Actuated and MRI Compatible Robots**
**Lab 9: MRI Guided Needle Insertion**

## Errors identified from the live demonstration and their respective solutions that were implemented:

### 1) X and Y were flipped

After the demonstration, we went through our code line by line and found that while the coordinates were being converted from the 3D slicer to the real-world, the X was displayed as Y and vice versa. Now, that issue has been resolved by correctly resolving X and Y in the real-world coordinates. This error also affected our value of theta which was mirrored about the horizontal which has also been resolved now.

### 2) Conversion factor was off by 0.1 but caused major offsets

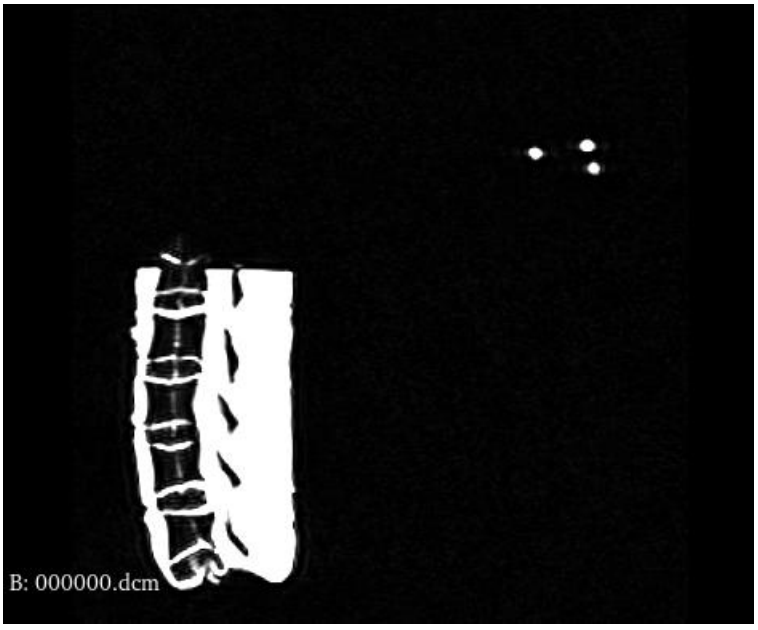
We obtained the conversion factor from 3D slicer software (as directed in the lab 9 document) but due to some difference in fiducial marker positions between our robot and the ones in the scan, there was an offset in the conversion value of 0.16 (It was given as 1.367 by 3D slicer but the measured value is 1.20225). Though it may not seem like a high number, it gave a difference in x-value of about 15 mm and y-value of about 7 mm for the 1$^{st}$ slot (more for the slots further away from home) which is pretty significant and matches up with the errors occurring during our testing and in the live demo (We had accounted for it with an offset but this is scalable so a fixed value would not fix the issue).

### 3) Horizontal line identification for theta

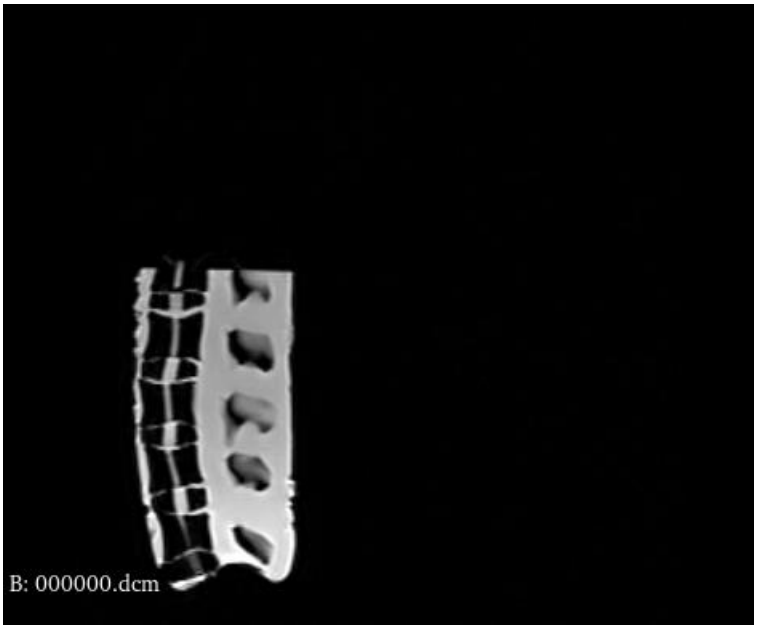
This was a comparatively minor error compared to the other two. We had initially calculated the horizontal and vertical reference lines of the robot in the slides using MATLAB to identify the fiducial markers. However, the small difference in pixel values had a significant effect on theta which in turn had an even greater effect on the final position and insertion angle of the needle tip. This error was considerably reduced when we just assumed that the reference horizontal line is one with slope equal to zero which gave us a theta value of around 10 degrees from the horizontal compared to the 20 degrees we were getting before. This significantly reduced our offset values and made the system much more accurate.

**Screenshots from 3D Slicer and rationale behind choosing your slices. (10 points)**

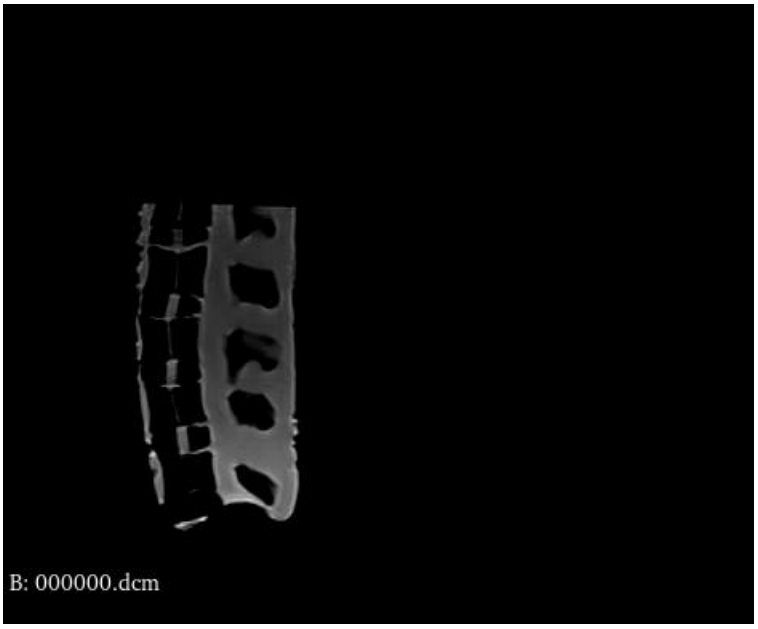
Firstly, slices were chosen such that one shows all the 3 robot fiducials clearly, and the other shows all the vertebrae contours clearly. This was done so that we could get the best possible and most accurate locations and representation of the fiducials and vertebrae respectively for image processing – required for fiducial locating and the vertebrae needle path planning. Right after selecting the slices in 3D slicer, their image window-levels were adjusted so that we could get the best possible representation for image processing. Then the image was exported from Slicer. For the image with the fiducials, we window-leveled it so that the fiducials were as bright as possible, thus getting a very high contrast compared to its immediate surrounding background which allowed me to determine the location of the fiducial centers and radii easily. Image saved from the yellow slice in 3D slicer as shown:




For the image showing the spine vertebrae, the image was window-leveled to reduce the amount of gray noise in the vertebrae, to make the image processing more robust. Images were still saved from the yellow slices. Before and after windowing vertebrae images as shown:
Before window-leveling:




After window-leveling:



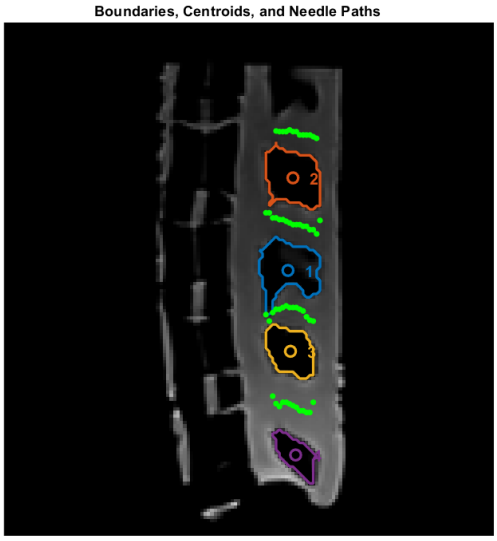

**Screenshots, observations, and rationale for code used to detect needle insertion path. (10 points)**

Note that the images exported from 3D Slicer had to be resized into 256 by 256 image when read into our Matlab code, as the conversion factor obtained from Slicer was using a 256 by 256 image. The general process to determine optimal needle insertion path for each of the four slots is as follows:

Perform morphological closing (Matlab imclose function) on the image and then find the boundaries (bwboundaries function in Matlab) to identify each vertebrae. Compute properties of

the vertebraes (Matlab regionprops) such as centroids, area, perimeter. For the space of interest, compute a path between the two specified vertebrae. For the top vertebrae, get the centroid. Take all boundary values from that vertebrae that fall under the y centroid value (aka greater than the y values, in Matlab image coordinates since y increases downwards). That is the set of upper boundary points. For the lower vertebrae, get the centroid. Take all boundary values from that vertebrae that are above the y centroid value (aka less than the y values, in Matlab image coordinates since y decrease upwards). That is the set of lower boundary points.
To find the ideal path between these, maximizing the distance from upper and lower vertebrae contours. This can be done by averaging the y values between the upper and lower boundary points. For each point in the upper and lower boundary that share the same x value, average the y values to find the point in the middle of those upper and lower boundary points. Do this for all the points in the upper and lower boundary and connect the points to get an ideal path. The ideal paths are all shown below:

All ideal paths computed from the code are plotted and shown in green on the image below:

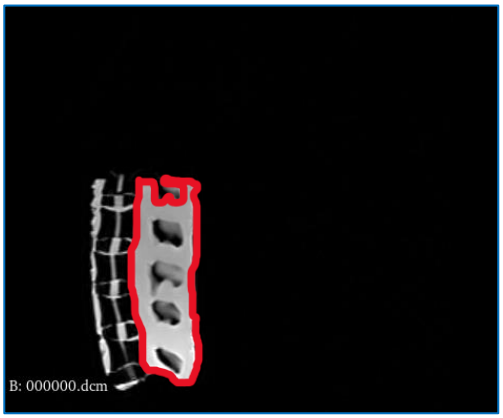



Since the needle is a straight line, and some of these points are likely to hit bone, the set of points was also manually inspected and an optimal straight path, consisting of two points, was determined:

| **Path** | **Matlab Coordinate Points (x,y)** |
|---|---|
| L1/L2 | (93,105)<br>(81, 102.833) |
| L2/L3 | (92, 132)<br>(80, 128.5) |
| L3/L4 | (90, 157.75)<br>(79, 159) |
| L4/L5 | (92, 183)<br>(80, 181.167) |

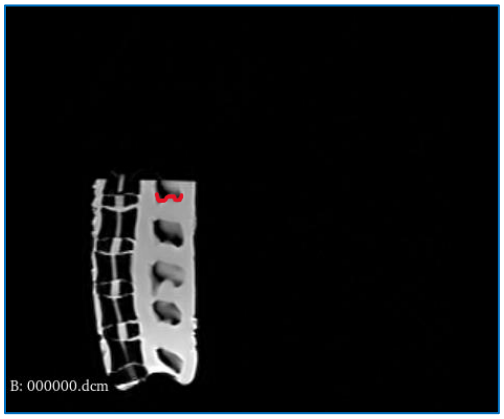
**Note:** We refer to vertebrae from top to bottom, L1 through L5 as shown oriented in the images above (NOT flipped). We chose not to flip the images for our image processing, since the ideal path itself should remain the same regardless. Only the coordinate values in the Matlab image will change, but the flipped orientation was accounted for in our conversion from Matlab coordinates to the robot commands, rather than in our image processing scripts itself.

**Note:** For the upper topmost vertebrae shown in the above slicer images (the one where its contour is not completely closed), its boundary detection was done a little differently. For this specific vertebrae, only the values in its lower half were detected and used. This was done in a separate custom written script, L1_vertebrae_code.m. This script finds the contour of the right-hand-side portion of the spine phantom (aka all the white/light gray area:




Traced image for demonstration purposes only

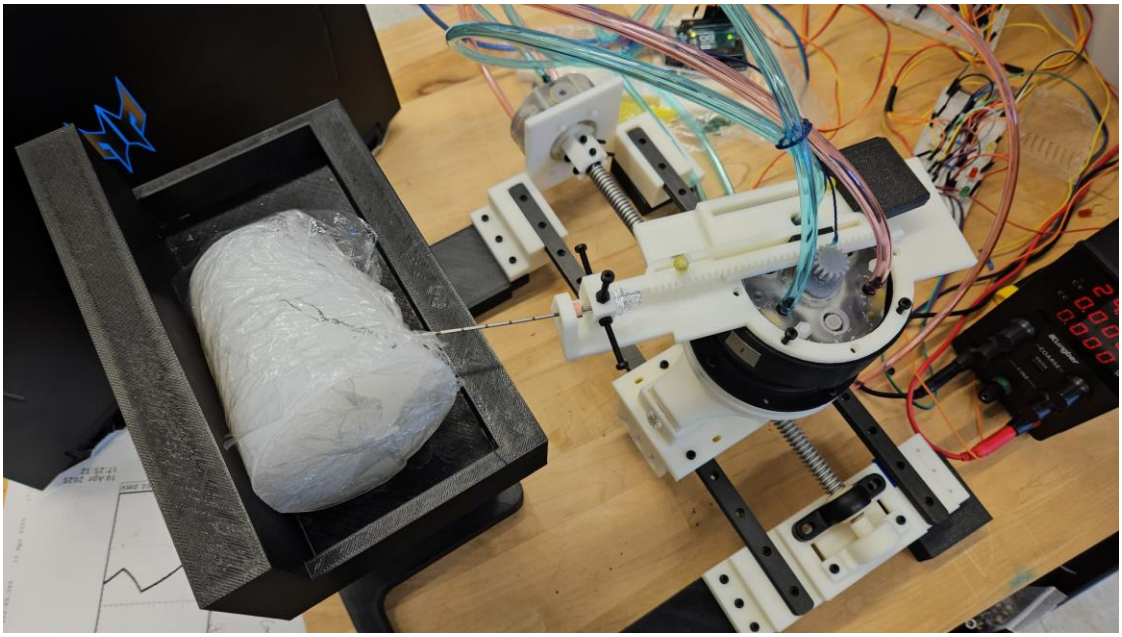
Then the top open position of the contour was manually inspected, and the values on that boundary in that region were extracted to form that vertebrae's lower boundary:




Traced image for demonstration purposes only

The rest of the ideal needle path computations were computed as described previously, just with this boundary forming the upper bound in the path involving the top vertebrae.

**Screenshots and observation for the resistant force tests on silicone block. (10 points)**



We observed that there was a difference of less than 2 mm between the desired depth and the depth the needle reaches. This was observed after slowing down the speed of stage 3 to about half of the speed of other stages. Another observation was that the needle was more easily inserted when it was at an angle rather than completely perpendicular to the block. The system was run at 25 psi and the needle depth is computed in the code itself using trigonometry based on the X, Y and theta values provided to run the robot. The offsets measured during these tests were averaged and provided as insertion_offset in our code.
(Note: We initially did not take videos when we ran this experiment using our system. So, after the live demonstration, we ran the test again using team 5's robot with their permission as they still had some part of the needle fixed in their robot)

**Screenshots and observations for the Targeting test on the 3D printed mimic of the Spine model. (10 points)**

This section has been divided into two parts, before the demo and after the demo to provide more details.

Before the demo:

We had not realized that our X and Y values were flipped so we had given considerably large offsets for each of the 4 slots. We had tested out the code for the 1st to 3rd slot but had not tested out the performance for the 4th slot and it turned out that the offset provided was not sufficient. The offsets included an offset in needle insertion depth, X direction offset, and Y direction offset.

After the demo:

Once we figured out the issue with the code i.e., the flipped X and Y and scaling factor, the offset was very small (within a few millimeters) and desirable performance of the robot was observed. Minor offsets were present to account for some errors in the assembly of the robot, but no major offsets were required in this case.

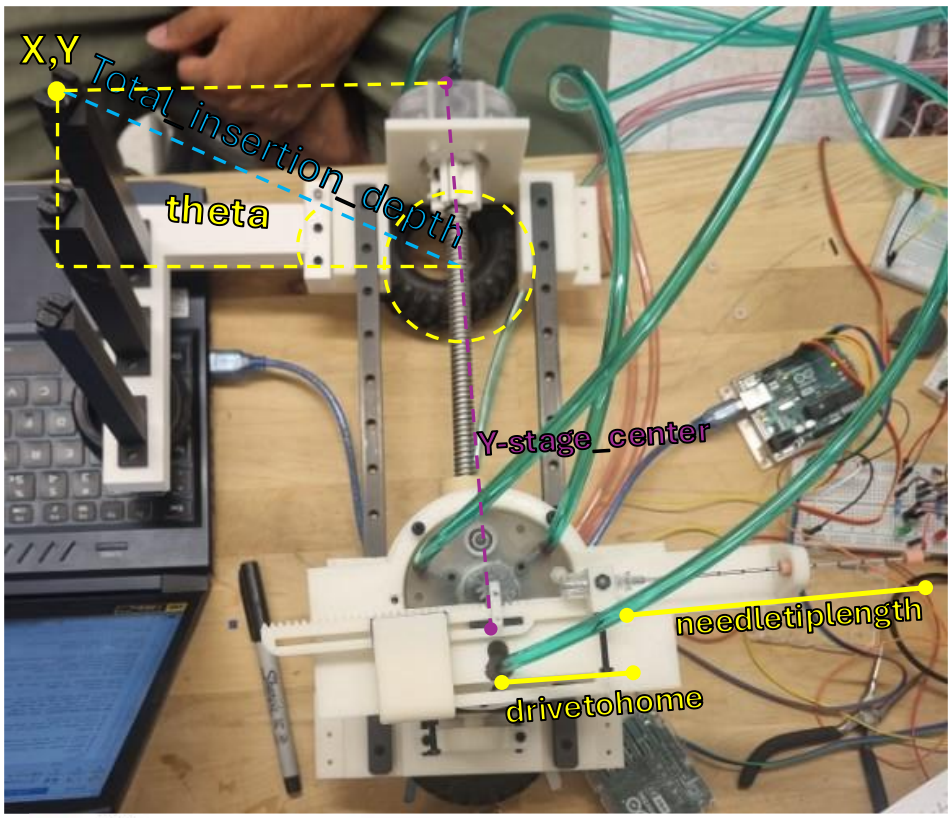
**Observations for your final demonstration and any improvements to your robot or targeting protocols. (10 points)**

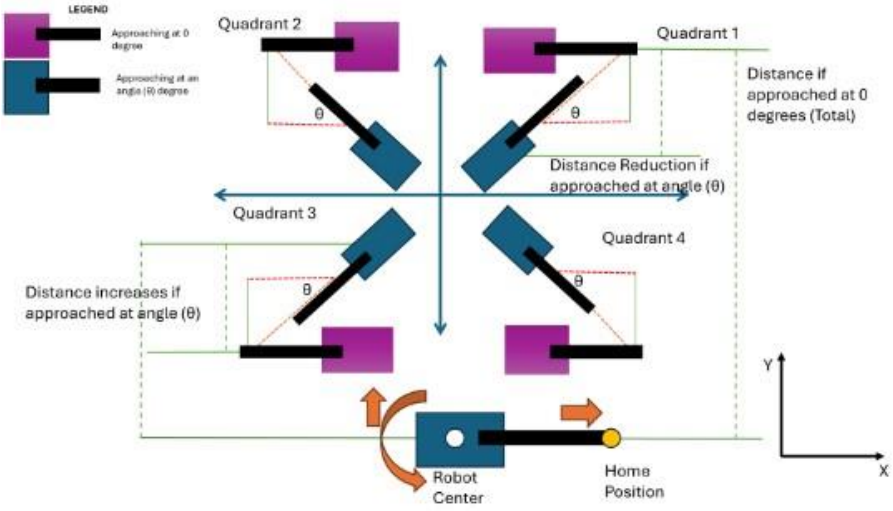
As presented on the first page of this report, our final demonstration had several issues. The changes that have been made in the code after the live demonstration have made the robot much more accurate. The improvements would be:

- Better tolerancing on parts as we had several parts that would wobble and affect the accuracy of the system. It also made it difficult to distinguish errors in our code from errors in our system.
- Reliability of the motor was a factor that affected our testing as we would often see that the motor would skip steps or not move in some cases if one screw was a little tight.
- For path planning, we believe we had one of the better algorithms in place. However, the usage of the path identified by this algorithm has been improved by taking reference as the horizontal line instead of a line identified by fiducial markers.
- There needs to be a more accurate way of calculating the conversion factor as even small changes in its value can have a large impact on the working of the system.

**Inverse Kinematics Explanation:**

In this section, we will explain the inverse kinematics equations that have been used. The inverse kinematics in this case are for a 2D problem as the height cannot be varied. Stage 1 can only affect one direction which we have chosen to be Y while stage 2 and stage 3 affect both X and Y. Variables used: stage_center is the distance from edge of stage 1 platform to its center which is 35 mm (Thickness of stage 1 platform is 70 mm); needletiplength and drivetohome are marked in the figure below. Using trigonometry with these variables, we can calculate the required distance to be travelled for stage 1 and stage 3 and the angle for stage 2 after which we apply the step conversions.




**Two videos showing needle insertion into the Silicone block and the mimic-model. (10 points)**

The videos are present in the zip file.