CMPE 330 Assignment 3

Genevieve Hayes
November 23rd, 2020

The script **main_A3.m** will call the testing and display functions for Assignment 3. All functions and tests are located in the same folder as main_A3.m. To run each question's tests independently within main_A2.m, use the "Run Section" option while the cursor is placed in the desired section.

Table of Contents

Part 1: Tool Tip Calibration *1*
Method 1
Part 2: Tool Axis Calibration *1*
Method 2
Part 3: Tool Tip Calibration Testing *2*
Method 3
Part 4: Tool Axis Calibration Testing *3*
Method 3
Part 5: Surgical Navigation *4*
Method 4
Bonus Question #1: Tool Tip Calibration using Transformation Offset *9*
Method 10
Testing 10
Bonus Question #2: Tool Tip Calibration using Pose Plane Estimation *10*
Method 10
Testing 11
Appendix: Additional Functions *12*
Fit Plane to Vectors and Find Normal 12
Check Triangles Are Congruent 12
Reconstruct Sphere From Point 13
Generate Orthonormal Frame 13
Frame Transformation to Home 14
Rotation About Frame Axis 15
Number of Intersections between Sphere and Cylinder 16
References 18

Part 1: Tool Tip Calibration

```
function [Tip_tool] = toolTipCalibration(At_track,Bt_track,Ct_track)
```

The tool tip calibration determines the location of the tool tip with respect to the tool marker frame based on arrays of marker A, B and C positions. At_track, Bt_track, Ct_track are matrices with 3 columns and at least 4 rows; the first column with X values, 2nd column with Y values and 3rd columns with Z values of spherical pivot poses determined by rotating the surgical drill with the tool tip at a fixed location.

Method

- First fit a sphere to the arrays of each marker locations using the function reconstructSphereFromPoints(pointsMatrix). This reconstructs three spheres corresponding to the motion of marker A, marker B and marker C respectively. A detailed description of reconstructSphereFromPoints(pointsMatrix) can found in the Appendix. The center of the spheres corresponds to the tip of the tool in tracker frame.
- Take the average of the center locations of the reconstructed spheres to get $C_{trackAVG}$.
- Let N be the number of poses that the markers were captured in. For each pose (N times), generate an orthonormal tool frame from the A, B and C locations corresponding to that pose using generatedFrameTransformationToHome(A1,B1,C1) described in the Appendix.
- Using the orthonormal tool frames, generate a frame transformation matrix that goes from each tool pose to the tracker frame, $T_{track2tool}$ (N times) using generateFrameTrandformationToHome(O,e1,e2,e3) described in the Appendix. There will be N $T_{track2tool}$ matrices, one for each pose.
- Invert $T_{track2tool}$ to generate the transformation matrices that go from the tracker frame to the tool frame for each pose.
- Transform $C_{trackAVG}$ to the tool frame using $T_{track2tool}$ which will generate a C_{tool} for each pose. Note that $C_{trackAVG}$ needs to be padded with a 1 before applying the 4x4 transformation matrix.

$$C_{tool_{padded}} = T_{track2tool} * C_{trackAVG_{padded}}$$

- Take the average of all C_{tool} to determine the location of the tool tip in tool frame, tip_{tool}. tip_{tool} is rounded to 5 decimal places of precision.

Part 2: Tool Axis Calibration

```
function [vax_tool] = toolAxisCalibration(At_track,Bt_track,Ct_track)
```

The tool axis calibration determines the Y axis of the tool with respect to the tool marker frame based on arrays of marker A, B and C positions. At_track, Bt_track, Ct_track are again matrices with 3 columns and at least 4 rows; the first column with X values, 2nd column with Y values and 3rd columns with Z values of poses determined by rotating the surgical drill only about the y-axis with the tool tip at a fixed location.

Method

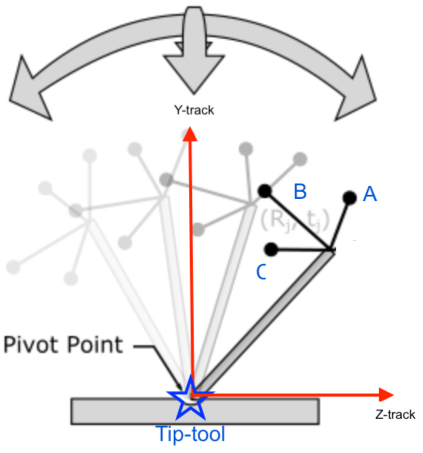
- First fit a plane to the arrays of each marker locations and calculate each surface's normal using the function fitNormal(pointsMatrix). This reconstructs three planes corresponding to the motion of marker A, marker B and marker C respectively. A detailed description of fitNormal(pointsMatrix) can found in the Appendix.
- Take the average of the normals of the reconstructed planes to get $n_{trackAVG}$.
- Let N be the number of poses that the markers were captured in. For each pose (N times), generate an orthonormal tool frame from the A, B and C locations corresponding to that pose using generatedFrameTransformationToHome(A1,B1,C1) described in the Appendix.
- Using the orthonormal tool frames, generate a frame transformation matrix that goes from each tool pose to the tracker frame, $T_{track2tool}$ (N times) using generateFrameTrandformationToHome(O,e1,e2,e3) described in the Appendix. There will be N $T_{track2tool}$ matrices, one for each pose.
- Invert $T_{track2tool}$ to generate the transformation matrices that go from the tracker frame to the tool frame for each pose.
- Transform $n_{trackAVG}$ to the tool frame using $T_{track2tool}$ which will generate a C_{tool} for each pose. Note that $n_{trackAVG}$ needs to be padded with a 1 before applying the 4x4 transformation matrix. Normalize n_{tool} to make it a unit vector.

$$n_{tool_{padded}} = T_{track2tool} * n_{trackAVG_{padded}}$$
$$n_{tool} = \frac{n_{tool}}{\|n_{tool}\|}$$

- Take the average of all n_{tool} to determine the tool Y axis vector in tool frame, tip_{axis}. Tip_{axis} is rounded to 5 decimal places of precision.

Part 3: Tool Tip Calibration Testing

The tool tip calibration is achieved by rotating the markers to different positions with the tool tip fixed at one pivot point as shown in the figure below.



Method

- Generate N random angles between 0 and 30 degrees about the Z axis. Note that since the rotation of the markers is in a cone motion, the pivot about the X and Z axis is proportional, just with a phase shift.
- Generate N random angles between 0 and 360 degrees about the Y axis.
- Make a ground truth model of the tracked tool and markers where Tip_{tool} = [0,-20,0], $At_{trackGT}$ = -Tip_{tool} + [-2,-2,0], $Bt_{trackGT}$ = -Tip_{tool} + [4,-2,0], $Ct_{trackGT}$ = -Tip_{tool} + [-2,4,0].
- Generate N 4x4 rotation matrices for the Z rotation and generate another N 4x4 rotation matrices for the Y rotation using rotationMatrixAboutFrameAxis(axis,angle) described in the Appendix.
- Rotate $At_{trackGT,}$ $Bt_{trackGT}$ and $Ct_{trackGT}$ N times using the N rotation matrices. Note that the ground truth marker locations need to be padded with a 1 before applying the 4x4 rotation matrices.

$$A_{track_{padded_i}} = R_{Z_i} * R_{Y_i} * A_{trackGT_{padded}}$$

- The result is arrays of marker locations at different spherical pivot poses in 3D space for markers A, B and C.

Poses

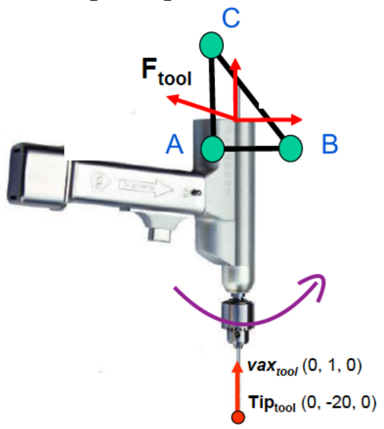
- N = 100 poses were recreated using the 0- to 30-degree rotations about in z axis in combination with the 0- to 360-degree rotations about the y-axis.

Check

- Feeding At_track, Bt_track and Ct_track into toolTipCalibration(A,B,C), the output is [0,-20,0] as expected.

Part 4: Tool Axis Calibration Testing

The tool axis calibration is achieved by rotating the markers only about the y-axis to different positions with the tool tip fixed at one pivot point as shown in the figure below.



Method

- Generate N random angles between 0 and 360 degrees about the Y axis.

- Make a ground truth model of the tracked tool and markers where Tip_{tool} = [0,-20,0], $At_{trackGT}$ = $-Tip_{tool}$ + [-2,-2,0], $Bt_{trackGT}$ = $-Tip_{tool}$ + [4,-2,0], $Ct_{trackGT}$ = $-Tip_{tool}$ + [-2,4,0].
- Generate N 4x4 rotation matrices for the Y rotation using rotationMatrixAboutFrameAxis(axis,angle) described in the Appendix.
- Rotate $At_{trackGT}$, $Bt_{trackGT}$ and $Ct_{trackGT}$ N times using the N rotation matrices. Note that the ground truth marker locations need to be padded with a 1 before applying the 4x4 rotation matrix.

$$A_{track_{padded_i}} = R_{Y_i} * A_{trackGT_{padded}}$$

- The result is arrays of marker locations at different circular pivot poses about the Y axis in 3D space for markers A, B and C.

Poses

- N = 100 poses were recreated using the 0- and 360-degree rotations about the y-axis.

Check

- Feeding At_track, Bt_track and Ct_track into toolAxisCalibration(A,B,C), the output is [0,1,0] as expected.

Part 5: Surgical Navigation

```
function [] = surgicalNavigationSimulation(Tip_tool, vax_tool, ApatCT,
BpatCT, CpatCT, ApatTrack, BpatTrack, CpatTrack, AtoolTrack, BtoolTrack,
CtoolTrack, TumCtrCT, TumRadCT, WinCtrCT, WinRadCT, generatePlot)
```

The surgical navigation simulation computes the tumour and window locations in tracker frame, the tool trajectory in tracker frame, the number of intersections of the tool trajectory with the window and tumour and the depth that the drill needs to continue to reach the depth of the center of the tumour. A navigation scene is also plotted showing the markers, tool tip, tool axis vector, tool trajectory, tumour and window.

Method

- First check that the drill markers and patient markers are congruent with those used for the calibration using checkTrianglesCongruent(A0, B0, C0, A, B, C) described in the Appendix.
- Generate an orthonormal frame from the patient markers in CT frame using generateOrthonormalFrame(ApatCT,BpatCT,CpatCT) described in the Appendix.
- Generate an orthonormal frame from the patient markers in tracker frame using generateOrthonormalFrame(ApatTrack,BpatTrack,CpatTrack).
- Generate a frame transform from patient frame to tracker frame ($T_{pat2track}$) using generateFrameTransformationToHome(Opat-Opathome,e1pat,e2pat,e3pat) described in the Appendix.

- Transform the window and tumour from CT to tracker frame. Note that coordinates are padded with a 1 before applying the 4x4 transformation matrix.

$$TumCTR_{track_{padded}} = T_{pat2track} * TumCTR_{CT_{padded}}$$
$$WinCTR_{track_{padded}} = T_{pat2track} * WinCTR_{CT_{padded}}$$

- Generate an orthonormal frame from the tool markers in tracker frame using generateOrthonormalFrame(AptoolTrack,BtoolTrack,CtoolTrack) described in the Appendix.
- Transform the tool tip from tool frame to tracker frame by offsetting the tip by the origin of the markers in tracker frame.

$$Tip_{track} = \ Omarkers_{track} + Tip_{tool}$$

- Transform the tool axis from the tool frame to tracker frame by offsetting the tip by the y-component of the markers' orthonormal basis in tracker frame. This will ensure the tool is still oriented with the tip below the markers.

$$vax_{track} = \ e2markers_{track} + vax_{tool}$$

- The drill trajectory is taken as the opposite direction of the tool axis.

$$drill_{trajectory} = -vax_{track}$$

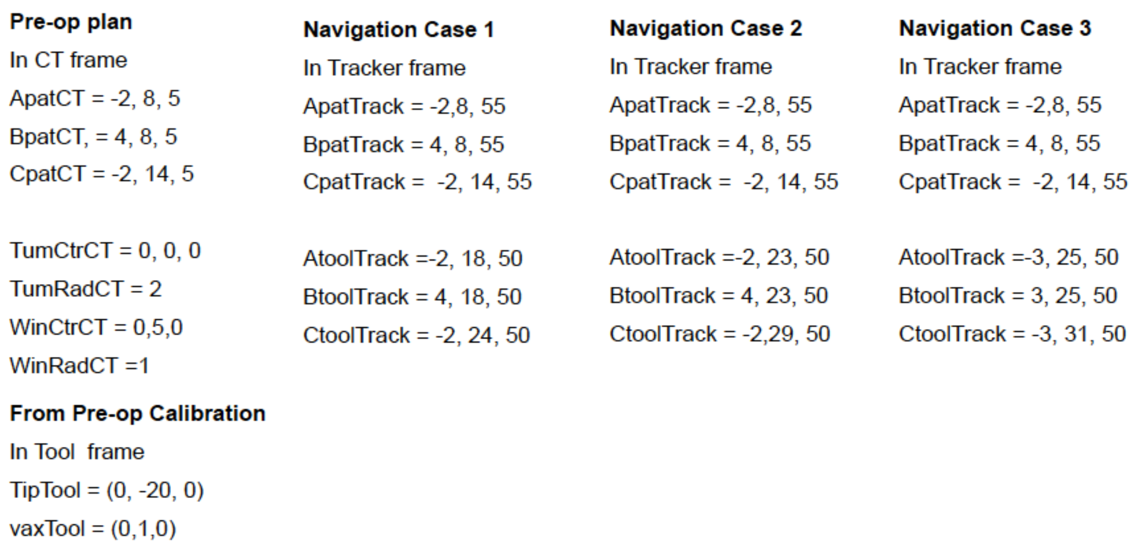
- The drill head is estimated to be a thin cylinder with radius of 0.01 length units, $drill_{rad}$.
- The number of intersections between the drill trajectory and the tumour (with user defined radius) is calculated using the function numIntersectionsOfSphereAndCylinder(TumCtr_track,TumRadCT,drill_Rad,Otip_track, drill_trajectory) described in the Appendix. Intersections outputs are 0, 1 (tangential touch) or 2 (passes in and out of tumour).
- The number of intersections between the drill trajectory and the window (with user defined radius) is calculated using the function numIntersectionsOfSphereAndCylinder(WinCtr_track,WinRadCT,drill_Rad,Otip_track,drill_trajectory). Intersections outputs are 0, 1 (tangential touch) or 2 (passes in and out of window).
- If the drill is found to intersect with the tumour, the depth that the drill needs to go along its trajectory to reach the closest location of the tumour is calculated by taking the difference between the tumour center in tracker frame and the tool tip in tracker frame and normalizing the result.

$$Depth_{tip2tum} = \frac{TumCTR_{track} - Tip_{track}}{\|TumCTR_{track} - Tip_{track}\|}$$

- All the calculated parameters are printed in the command window and an x-y plot of the surgical navigation scene is generated.

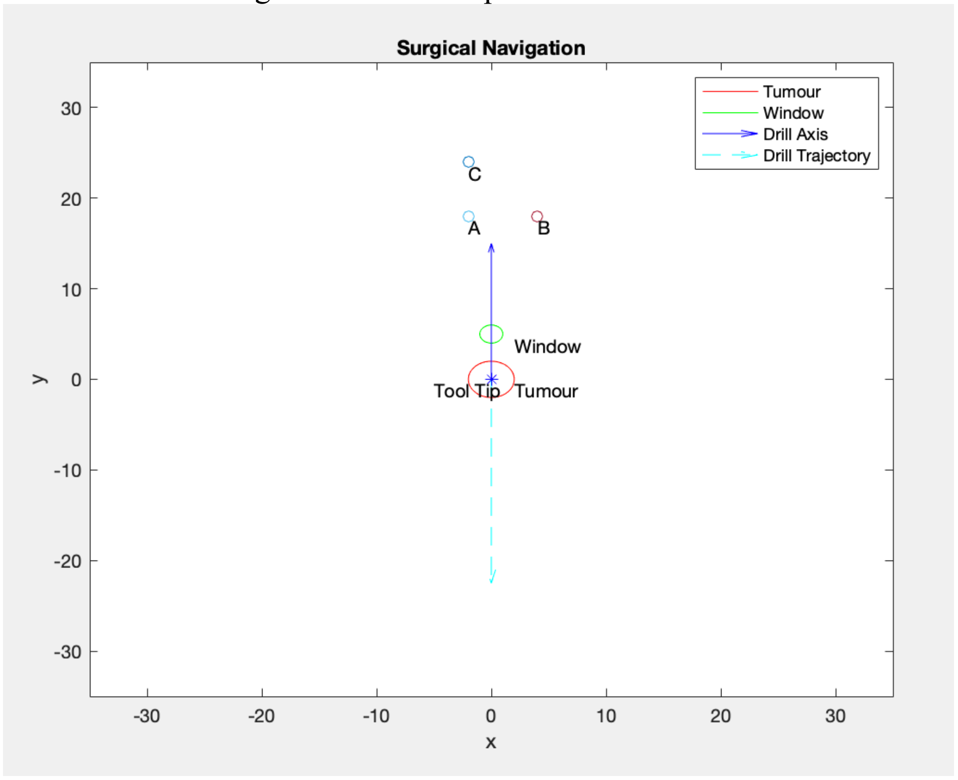
Test Cases

The inputs for the 3 test cases used to evaluate the surgical navigation simulation are presented below.



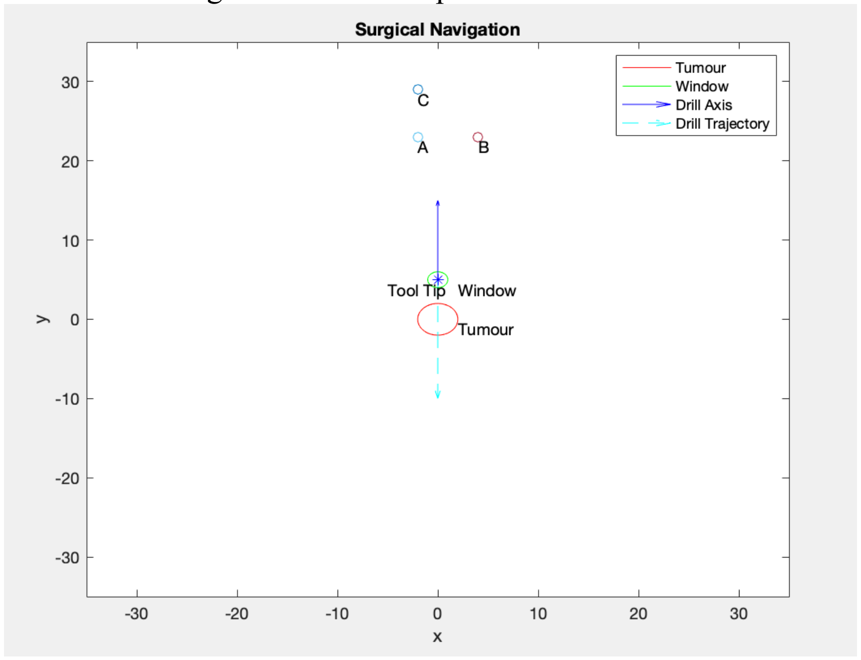
Pre-op plan	**Navigation Case 1**	**Navigation Case 2**	**Navigation Case 3**
In CT frame	In Tracker frame	In Tracker frame	In Tracker frame
ApatCT = -2, 8, 5	ApatTrack = -2,8, 55	ApatTrack = -2,8, 55	ApatTrack = -2,8, 55
BpatCT, = 4, 8, 5	BpatTrack = 4, 8, 55	BpatTrack = 4, 8, 55	BpatTrack = 4, 8, 55
CpatCT = -2, 14, 5	CpatTrack = -2, 14, 55	CpatTrack = -2, 14, 55	CpatTrack = -2, 14, 55
TumCtrCT = 0, 0, 0	AtoolTrack =-2, 18, 50	AtoolTrack =-2, 23, 50	AtoolTrack =-3, 25, 50
TumRadCT = 2	BtoolTrack = 4, 18, 50	BtoolTrack = 4, 23, 50	BtoolTrack = 3, 25, 50
WinCtrCT = 0,5,0	CtoolTrack = -2, 24, 50	CtoolTrack = -2,29, 50	CtoolTrack = -3, 31, 50
WinRadCT =1			
From Pre-op Calibration			
In Tool frame			
TipTool = (0, -20, 0)			
vaxTool = (0,1,0)			

- The results for the Navigation Case 1 are presented below.



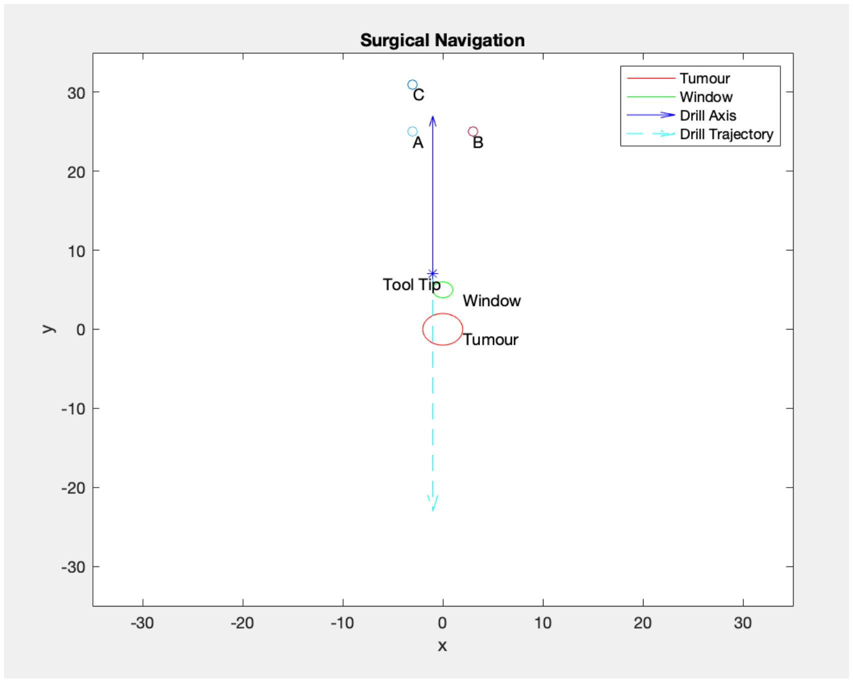
```
Case 1:
Surgical Navigation:
Check that the drill markers are congruent to the calibration drill markers: [ 1 ]
Check that the patient markers are congruent to the calibration patient markers: [ 1 ]
The tumour center in tracker frame is: [ 0  0  50 ]
The window center in tracker frame is: [ 0  5  50 ]
The location of the tool tip in tracker frame is: [ 0  0  50 ]
The direction of the tool axis in tracker frame is: [ 0  1  0 ]
The trajectory of the tool tip in tracker frame is: [ -0  -1  -0 ]
The number of intersections between the drill trajectory and the tumour is: [ 2 ]
The number of intersections between the drill trajectory and the window is: [ 2 ]
The drill tip needs to drill [ 0 ] further along its trajectory to reach the tumour.
```

- The results for the Navigation Case 2 are presented below.



```
Case 2:
Surgical Navigation:
Check that the drill markers are congruent to the calibration drill markers: [ 1 ]
Check that the patient markers are congruent to the calibration patient markers: [ 1 ]
The tumour center in tracker frame is: [ 0  0  50 ]
The window center in tracker frame is: [ 0  5  50 ]
The location of the tool tip in tracker frame is: [ 0  5  50 ]
The direction of the tool axis in tracker frame is: [ 0  1  0 ]
The trajectory of the tool tip in tracker frame is: [ -0  -1  -0 ]
The number of intersections between the drill trajectory and the tumour is: [ 2 ]
The number of intersections between the drill trajectory and the window is: [ 2 ]
The drill tip needs to drill [ 5 ] further along its trajectory to reach the tumour.
```

- The results for the Navigation Case 3 are presented below.



```
Case 3:
Surgical Navigation:
Check that the drill markers are congruent to the calibration drill markers: [ 1 ]
Check that the patient markers are congruent to the calibration patient markers: [ 1 ]
The tumour center in tracker frame is: [ 0  0  50 ]
The window center in tracker frame is: [ 0  5  50 ]
The location of the tool tip in tracker frame is: [ -1  7  50 ]
The direction of the tool axis in tracker frame is: [ 0  1  0 ]
The trajectory of the tool tip in tracker frame is: [ -0  -1  -0 ]
The number of intersections between the drill trajectory and the tumour is: [ 2 ]
The number of intersections between the drill trajectory and the window is: [ 0 ]
The drill tip needs to drill [ NaN ] further along its trajectory to reach the tumour.
```

Bonus Question #1: Tool Tip Calibration using Transformation Offset

```
function [Tip_tool] = toolTipCalibrationTransformOffset
(At_track,Bt_track,Ct_track)
```

The tool tip calibration determines the location of the tool tip with respect to the tool marker frame based on arrays of marker A, B and C positions using the transformation matrix offset. At_track, Bt_track, Ct_track are matrices with 3 columns and at least 4 rows; the first column with X values, 2nd column with Y values and 3rd columns with Z values of spherical pivot poses determined by rotating the surgical drill with the tool tip at a fixed location.

Method

- Let N be the number of poses that the markers were captured in. For each pose (N times), generate an orthonormal tool frame from the A, B and C locations corresponding to that pose using generatedFrameTransformationToHome(A1,B1,C1) described in the Appendix.
- Using the orthonormal tool frames, generate a frame transformation matrix that goes from each tool pose to the tracker frame, $T_{track2tool}$ (N times) using generateFrameTrandformationToHome(O,e1,e2,e3) described in the Appendix. There will be N $T_{track2tool}$ matrices, one for each pose.
- Invert $T_{track2tool}$ to generate the transformation matrices that go from the tracker frame to the tool frame for each pose.
- Take the last column of each $T_{track2tool}$ which corresponds to the offset in tool frame between the center of the tool frame and the center of the tracker frame, $offset_{tool\ i}$.
- Take the average of all N $offset_{tool\ i}$ to determine the location of the tool tip in tool frame, tip_{tool}. tip_{tool} is rounded to 5 decimal places of precision.

Testing

- At_track, Bt_track and Ct_track was generated in the same manner as in the first tool tip calibration method.
- Using these as input to the function, this method generated the expected output of [0.00000, -20.0000, 0.00000] when N =100. This is within 5 decimal places of the ground truth tool tip.

Bonus Question #2: Tool Tip Calibration using Pose Plane Estimation

```
function [Tip_tool] = toolTipCalibrationPosePlane
(At_track,Bt_track,Ct_track)
```

The tool tip calibration determines the location of the tool tip with respect to the tool marker frame based on arrays of marker A, B and C positions by calculating the intersection of normal vectors to the planes formed by each pose's 3 markers. At_track, Bt_track, Ct_track are matrices with 3 columns and at least 4 rows; the first column with X values, 2nd column with Y values and 3rd columns with Z values of spherical pivot poses determined by rotating the surgical drill with the tool tip at a fixed location.

Method

- Let N be the number of poses that the markers were captured in. For each pose (N times), generate an orthonormal tool frame from the A, B and C locations corresponding to that pose using generatedFrameTransformationToHome(A1,B1,C1) described in the Appendix.
- Calculate N-1 mid-points between the origins of the tool frame, call this (N-1)x3 matrix A.

$$A_i = (C_i - C_j)/2$$

- Calculate N-1 distances between the origins of the tool frame, call this (N-1) dimensional array n.

$$n_i = \left\|C_i - C_j\right\|$$

- The tool tip in tracker frame can be found by scaling each Ai by ni then multiplying it by the pseudo inverse of the n array divided by N.

$$Tip_{track} = (n)^{-1}_{Nx1} * \frac{An_{3xN}}{N}$$

- Using the orthonormal tool frames, generate a frame transformation matrix that goes from each tool pose to the tracker frame, $T_{track2tool}$ (N times) using generateFrameTrandformationToHome(O,e1,e2,e3) described in the Appendix. There will be N $T_{track2tool}$ matrices, one for each pose.
- Invert $T_{track2tool}$ to generate the transformation matrices that go from the tracker frame to the tool frame for each pose.
- Transform Tip_{track} to the tool frame using $T_{track2tool}$ which will generate a C_{tool} for each pose. Note that Tip_{track} needs to be padded with a 1 before applying the 4x4 transformation matrix.

$$C_{tool_{padded}} = T_{track2tool} * Tip_{track_{padded}}$$

- Take the average of all N-1 C_{tool} to determine the location of the tool tip in tool frame, Tip_{tool}. Tip_{tool} is rounded to 5 decimal places of precision.

Testing

- At_track, Bt_track and Ct_track was generated in the same manner as in the first tool tip calibration method.
- Using these as input to the function, this planes method was not as accurate as the previous 2 methods, resulting in an output of [0.01000, -19.79000, 0.01000] when N =100. Within 0 decimal places, this is equal to the ground truth tool tip.

Appendix: Additional Functions

Fit Plane to Vectors and Find Normal

```
function n = fitNormal(data)
```

Inspired by Dan Couture's fitNormal(data) function at MathWorks [1].

Method

- Check that the data can be fit with a plane by taking the determinant of the matrix formed from multiply each column by its inverse. The determinant should not be equal to 0.

$$X_{mx} = inv(data_x) * data_x$$
$$\det(X_m) \neq 0$$

- Fit the points with a flat surface by minimizing the sum of the residuals of the 1st order fit.
- Construct and normalize the normal vector to the plane as follows:

$$coeff_x = X_{mx}^{-1} * inv(data_x) * data_x$$
$$norm_x = \frac{-coeff_x}{\|coeff_x\|}$$

Testing

- fitNormal(data) is tested by generating many points on a plane with a known normal vector (the ground truth), feeding them to the function and comparing the output with the ground truth.

Check Triangles Are Congruent

```
function [checkCongruent] = checkTrianglesCongruent(A0,B0,C0,A,B,C)
```

This function checks if the triangle formed by A, B, C is congruent to the A0, B0, C0.

Method

- Calculate the length of sides of the original triangle and of the new triangle. For example, for the length from corner A to B, the side length is $AB = \|A - B\|$.
- If each side of the new triangle side lengths is within 0.0001 of the reference triangles, the triangles are considered congruent and a 1 is returned.

Testing

- Test 1: Case where triangles are congruent. Input of A0 = [-2,-2,0], B0 = [4,-2,0], C0 = [-2,4,0] and A, B, C offset from A0, B0, C0 by -2 in the y directions. The output is 1 as expected.
- Test 2: Case where triangles are not congruent. Input of A0 = [-2,-2,0], B0 = [4,-2,0], C0 = [-2,4,0] and A, B, C offset from A0, B0, C0 by -2 in opposite directions. The output is 0 as expected.

Reconstruct Sphere From Points (Assignment 1, Part 5)

```
function [C,R] = reconstructSphereFromPoints(pointsMatrix)
```

Method

- This function uses a simplified form of least-squares fitting to reconstruct a sphere with defined center and radius best fit to the points. It is simplified because it does not minimize the residuals. It requires at least 4 or more 3-dimensional points as input.
- The sphere equation can be written as $(x-a)^2+(y-b)^2+(z-c)^2=r^2$ where a, b, c and r are the unknowns and x, y and z are the data points.
- For each data point, the above equation can be rewritten as a linear equation containing with a, b, c and r as variable:

$$2\mathrm{ax} - 2\mathrm{by} - 2\mathrm{cz} + (\mathrm{a}^2+\mathrm{b}^2+\mathrm{c}^2-\mathrm{r}^2) = -(\mathrm{x}^2+\mathrm{y}^2+\mathrm{z}^2)$$

Let $\mathrm{D} = (\mathrm{a}^2+\mathrm{b}^2+\mathrm{c}^2-\mathrm{r}^2)$, then the equation become:

$$2\mathrm{ax} - 2\mathrm{by} - 2\mathrm{cz} + \mathrm{D} = -(\mathrm{x}^2+\mathrm{y}^2+\mathrm{z}^2)$$

This can be simplified further using matrices:

$$A * Y = B$$

$$Y = B * A^{-1}$$

Where the first 3 values of the column vector Y are the center of the sphere, C. The last value of Y is equal to D, therefore the radius, R can be found using:

$$R = \sqrt{\mathrm{C}_\mathrm{x}^2 + C_y^2 + \mathrm{C}_\mathrm{z}^2 - \mathrm{Y}_4}$$

The output center values and radius were rounded to 5 decimal places.

Testing

The MATLAB function [X, Y, Z] = sphere was used to generate 20x20 patch test spheres.

- Test 1: Case of unit sphere (R=1) centered at the origin (C = [0,0,0]).
- Test 2: Case of unit sphere (R=1) centered at C = [1,1,0].
- Test 3: Case of sphere with radius R=5 centered at C = [3,3,3].

Generate Orthonormal Frame (Assignment 1, Part 7)

```
function [Oe,e1,e2,e3] = generateOrthonormalFrame(A,B,C)
```

Method

- Use cross products to create 3 pairwise orthogonal vectors:

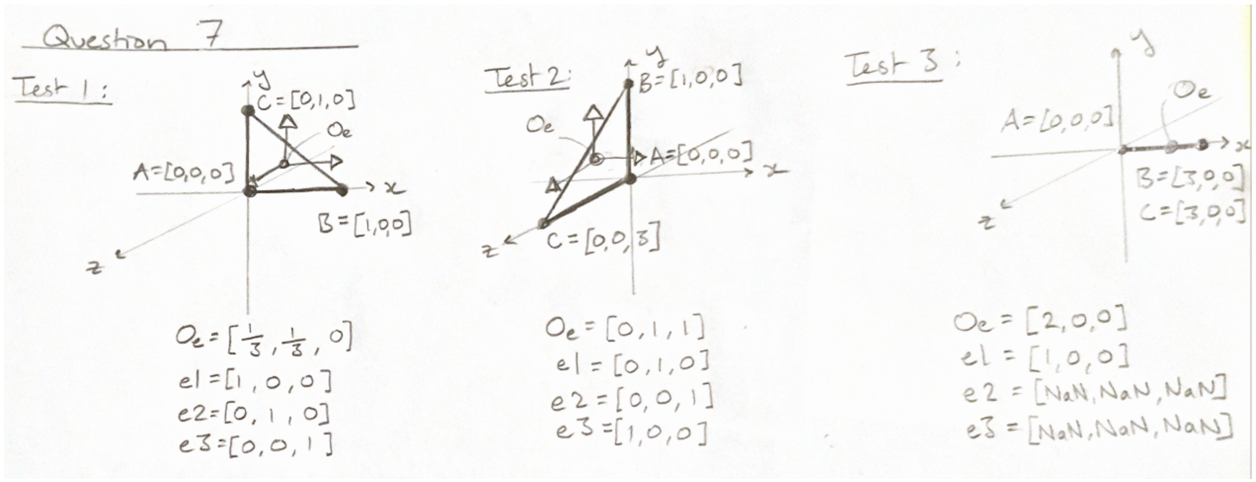
$$i = B - A$$
$$k = i \times (C - A)$$
$$j = k \times i$$

- Normalize each orthogonal vector to create an orthonormal frame (i, k, j).
- Take the Center of Gravity to be the center of the base frame:

$$O_e = \frac{A + B + C}{3}$$

Testing

- Test 1: Case of the origin and x, y unit vectors creating the triangle with corners at A = [0,0,0], B = [1,0,0], C = [0,1,0].
- Test 2: Case of an equilateral triangle centered about the point [0,1,1] with corners at A = [0,0,0], B = [0,3,0], C = [0,0,3].
- Test 3: Case of colinear input points A = [0,0,0], B = [3,0,0], C = [3,0,0]. This is a degenerate case that returns [NaN,NaN,NaN] for at 2 of the base vectors.



Frame Transformation to Home (Assignment 1, Part 9)

```
function [T_e2h] = generateFrameTransformationToHome(Oe,e1,e2,e3)
```

Method

- The translation matrix is given by:

$$Trans_{e\ to\ h} = \begin{pmatrix} 1 & 0 & 0 & O_{ex} \\ 0 & 1 & 0 & O_{ey} \\ 0 & 0 & 1 & O_{ez} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The rotation matrix is given by:

$$Rot_{e\ to\ h} = \begin{pmatrix} e1_x & e2_x & e3_x & 0 \\ e1_y & e2_y & e3_y & 0 \\ e1_z & e2_z & e3_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- These transformations can be used to transform a point in the orthonormal e-basis to the orthonormal h-basis (home frame):

$$P_h = Trans_{e\ to\ h} * Rot_{e\ to\ h} * P_e$$

- This can be written in the form of a single transformation matrix operation:

$$P_h = T_{e\ to\ h} * P_e$$

Where

$$T_{e\ to\ h} =* Trans_{e\ to\ h} * Rot_{e\ to\ h}$$

Testing

In all test cases, Oh = [0,0,0], h1 = [1,0,0], h2 = [0,1,0] and h3 = [0,0,1].

- Test 1 (Pure Translation): Offset the origin of the e basis in the x-axis such that Oe = [1,0,0]. e1=h1, e2=h2 and e3=h3. For point Pe = [0,0,0] the expected output is Ph = [1,0,0].
- Test 2 (Pure Translation): Offset the origin of the e basis in x, y and z such that Oe = [1, -1,1]. e1=h1, e2=h2 and e3=h3. For point Pe = [0,0,1] the expected output is Ph = [1, -1,2].
- Test 3 (Pure Rotation): Make the basis vectors negative such that e1 = [-1,0,0], e2 = [0,-1,0] and e3 = [0,0,-1] and let Oe = Oh. For point Pe = [1,1,1] the expected output is Ph = [-1,-1,-1].
- Test 4 (Pure Rotation): Rotate e1 and e2 by 60 degrees counterclockwise (when looking in the negative z direction) about the z-axis. In this case e1 = [0.5,0.8660,0], e2 = [-0.866,0.5,0] and e3 = [0,0,1]. For point Pe = [√3,1,1] the expected output is Ph = [0,2,1].
- Test 5 (Translation and Rotation): This test case is a combination of Test 1 and Test 3. The origin of the e basis is offset in the x-axis such that Oe = [0,1,0]. The basis vectors are then made negative such that e1 = [-1,0,0], e2 = [0,-1,0] and e3 = [0,0,-1]. For point Pe = [0,-1,-0.5] the expected output is Ph = [0,0,0.5].
- Test 6 (Translation and Rotation): This test case is the same as Test 4 but with an additional offset in the x and y planes such that Oe = [0,1,1]. e1 and e2 are again rotated by 60 degrees counterclockwise (when looking in the negative z direction) about the e3-

axis. For the point Pe = [√3,1,1] the expected output is Ph = [0,3,2] again since Pe "undoes" the basis translation before the rotation-to-home is applied.

Rotation About Frame Axis (Assignment 1, Part 8)

```
function [R3by3,R4by4] = rotationMatrixAboutFrameAxis(axis,angle)
```

Method

- Given θ, the 3x3 rotation matrices are found using:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

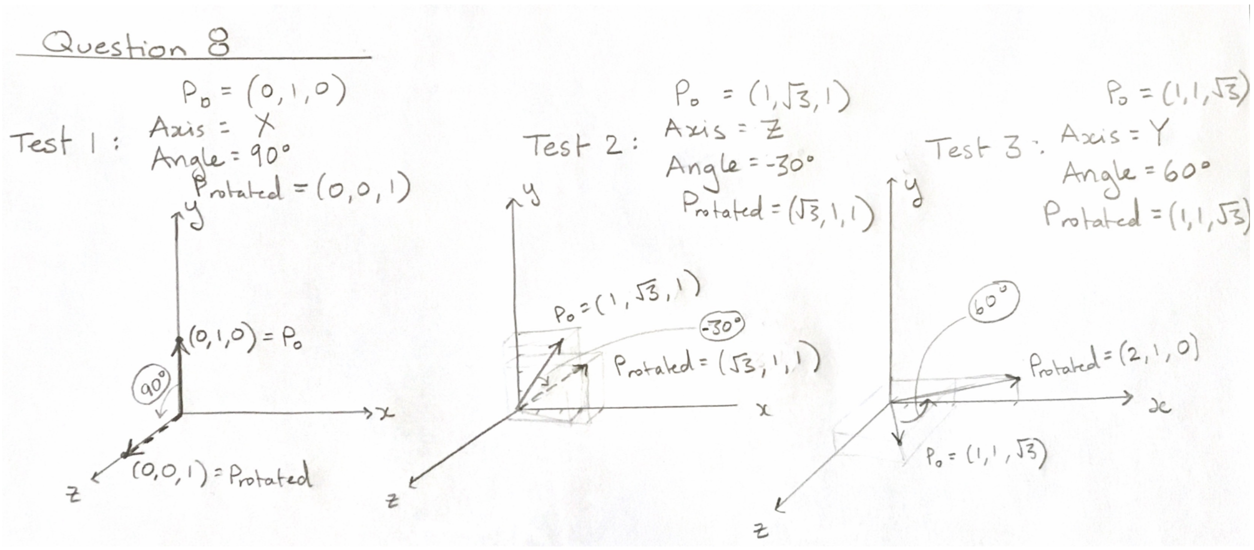
$$R_y = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- These can be made into the 4x4 homogeneous rotation matrices by padding the 4th column with [0,0,0,1] and the 4th row will be [0,0,0,1] as well.

Testing

- Test 1: 90-degree rotation about X-axis of a Y-direction unit vector.
- Test 2: 30-degree rotation about the Z-axis of vector [1,√3,1] (this is a 30-60-90-degree triangle). The expected output is [√3,1,1].
- Test 3: 60-degree rotation about the Y-axis of vector [1,1,√3] (this is a 30-60-90-degree triangle). The expected output is [2,1,0].



Number of Intersections between Sphere and Cylinder (Assignment 1, Part 4)

```
function numInt = numIntersectionsOfSphereAndCylinder(C,R,r,P,v)
```

Method

- Create a unit vector perpendicular to v, the direction vector of the cylinder by solving $v_x \cdot 1 + v_y \cdot 1 + v_z \cdot v_{2z} = 0$. Then $v_2 = \frac{[1,1,v_{2z}]}{norm(v_2)}$
  - Create v_3, V and P such that:

$$v_3 = v_2 \times v_1$$
$$V = \begin{bmatrix} -v_1 \\ v_2 \\ v_3 \end{bmatrix}$$
$$P = \begin{bmatrix} P_{1x} - P_{2x} \\ P_{1y} - P_{2y} \\ P_{1z} - P_{2z} \end{bmatrix}$$

Using the formula of a line $L = P + tV$, determine t of closest intersection by

$$t_{int} = V^{-1} \times P$$

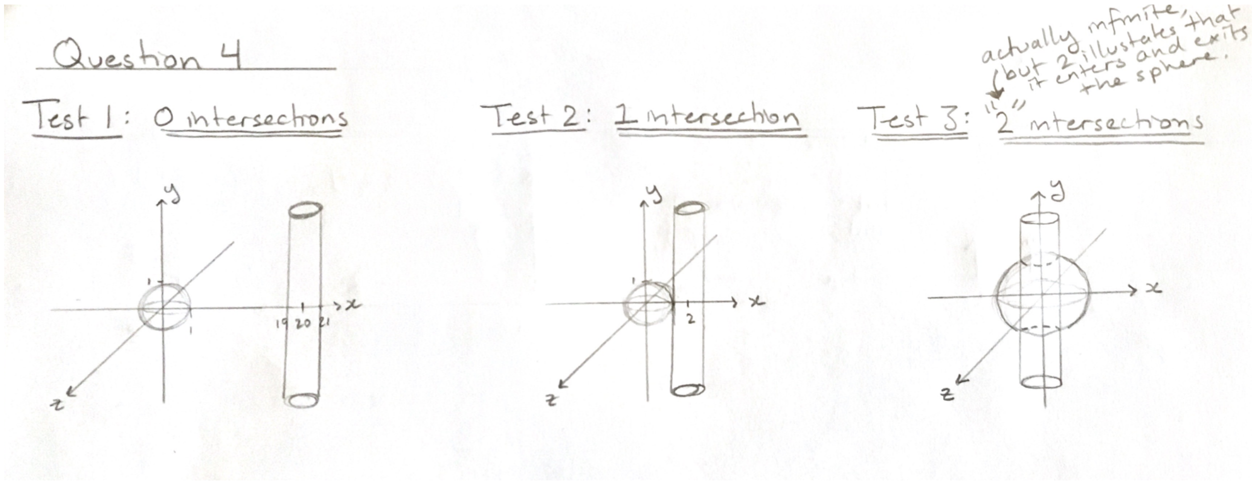
Then compute the location on the line of closest intersection of the line with the sphere center, $L_1 = P_1 + t_{int}V_1$.

Compute the distance $d = abs(L_1 - C)$.

Compute the value $val = \|d\| - R - r$. If val < 0 then there are "2" intersections between the cylinder and the sphere, i.e. the cylinder enters and exits the sphere at different locations. If val = 0 then there is exactly 1 intersection between the cylinder and the sphere. If val > 0, then there are no intersections between the cylinder and the sphere.

Testing

- Test 1: Case of zero intersections between sphere and cylinder. The function return numInt = 0.
- Test 2: Case of single point intersection between sphere and cylinder. Sphere is the unit vector centered at the origin, the cylinder has radius 1 centered at [2,0,0] and extends in the y-direction. The function return numInt = 1.
- Test 3: Case where cylinder enters and exits the sphere at different locations. In this case the function return numInt = 2.


Question 4
Test 1: 0 intersections
Test 2: 1 intersection
Test 3: "2" intersections
actually infinite, but 2 illustrates that it enters and exits the sphere.
y
x
z
1
19 20 21
2

References

[1] Dan Couture (2020). Plane Fitting and Normal Calculation (https://www.mathworks.com/matlabcentral/fileexchange/37775-plane-fitting-and-normal-calculation), MATLAB Central File Exchange. Retrieved November 19, 2020.