ME 384R ASBR: THA3 - Programming Assignment Report

Daniyal Maroufi and Anas Yousaf

April 15, 2025

1 Introduction

For this programming project, we were tasked with developing MATLAB functions that allowed us to investigate calibration, registration, and tracking for a stereotactic navigation system as well as hand-eye calibration. The stereotactic navigation system allows us to understand the importance of proper calibration methods and how the nature of different equipment affects calibration. The hand-eye calibration algorithms allows us to understand why proper data collection is essential by comparing clean data with noisy data.

2 Stereotactic Navigation System

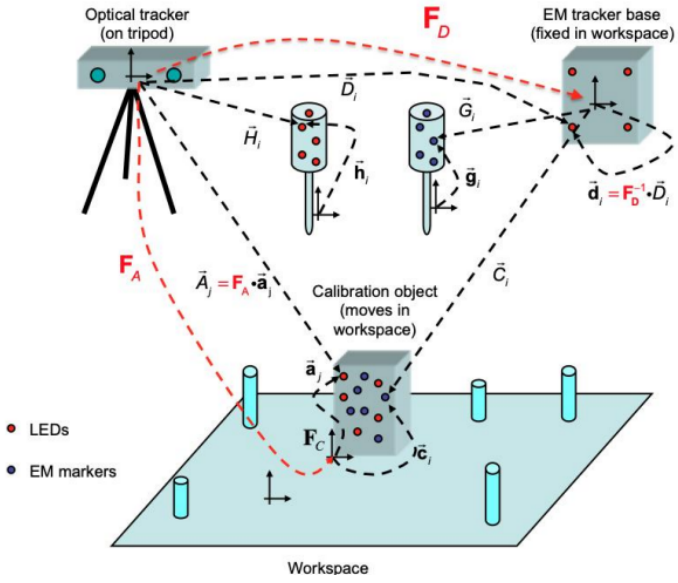

Figure 1: The stereotactic navigation system with an electromagnetic positional tracking device, an optical tracker, and a calibration object.

The stereotactic navigation system used for Problem 1 is seen in Figure 1. This system consists of an electromagnetic (EM) positional tracking device, an optical tracker, and the calibration object. The calibration object has EM markers and optical LED markers at known positions on the object. There are also optical LED markers on the EM tracking system at known locations. The calibration object has been placed at various locations and the EM and optical markers positions have been recorded at these various frames. Additionally, the workspace has dimpled calibration posts that are fixed but at unknown locations. Two pointer probes, one with EM and one with LED markers at unknown but fixed locations, have been used to gather pivot calibration data.

2.1 3D Point Set Registration

2.1.1 Method

To perform an correspondence-based registration using the SVD between two 3D point sets, we first calculate the means of each point set, a and b, and then center each point set by the mean.

$$\overline{a} = \frac{1}{N}\sum_{i=1}^{N} a_i \qquad \overline{b} = \frac{1}{N}\sum_{i=1}^{N} b_i \tag{1}$$

$$\tilde{a}_i = a_i - \overline{a} \qquad \tilde{b}_i = b_i - \overline{b} \tag{2}$$

The centered vectors are then used to construct the H matrix.

$$H = \sum_i \begin{bmatrix} \tilde{a}_{i,x}\tilde{b}_{i,x} & \tilde{a}_{i,x}\tilde{b}_{i,y} & \tilde{a}_{i,x}\tilde{b}_{i,z} \\ \tilde{a}_{i,y}\tilde{b}_{i,x} & \tilde{a}_{i,y}\tilde{b}_{i,y} & \tilde{a}_{i,y}\tilde{b}_{i,z} \\ \tilde{a}_{i,z}\tilde{b}_{i,x} & \tilde{a}_{i,z}\tilde{b}_{i,y} & \tilde{a}_{i,z}\tilde{b}_{i,z} \end{bmatrix} \tag{3}$$

We then take the singular value decomposition of the H matrix to construct the rotation matrix R [1].

$$H = U\Sigma V^T \tag{4}$$

$$R = VU^T \tag{5}$$

After calculating the R matrix, it is important to check that *det(R)* = 1. If the *det(R)* = -1, we check if one of the singular values is equal to zero. When one of the singular values is zero, we can recalculate V by changing the sign of the 3rd column and use this new V' to calculate R. However, if there is no zero singular value, the algorithm will fail.

Finally, we can calculate the translation vector p.

$$p = \tilde{b} - R\tilde{a} \tag{6}$$

This will allow us to create the transformation matrix representing the transformation between the two 3D point sets.

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{7}$$

2.1.2 Code Details

The MATLAB function `correspondence_registration_SVD` implements the above algorithm:

```
transfmat = correspondence_registration_SVD(a,b)
```

The function inputs

- `a`: N x 3 matrix containing 3D point set for vectors in the a coordinates, where N is the number of a vectors
- `b`: M x 3 matrix containing 3D point set for vectors in the b coordinates, where M is the number of b vectors

and outputs

- `transfmat`: 4x4 matrix representation transformation from a coordinates to b coordinates (b = transfmat*a)

2.2 Pivot Calibration

2.2.1 Method

The pivot calibration function is used to determine the position of the tool tip with respect to the base frame, as well as the position of the calibration post where the tool tip is inserted. To calculate these positions, the function separates the input transformation matrices into the corresponding rotation matrices, R_k and translation vectors, p_k. These are used to calculate the positions using the least squares optimization problem shown below.

$$\begin{bmatrix} \vdots & \vdots \\ R_k & -I \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} p_{tip} \\ p_{post} \end{bmatrix} = \begin{bmatrix} \vdots \\ -p_k \\ \vdots \end{bmatrix} \tag{8}$$

2.2.2 Code Details

The MATLAB function `pivot_calibration` implements the above algorithm:

```
[b_tip, b_post] = pivot_calibration(F)
```

The function inputs

- `F`: cell array of 4x4 transformation matrices containing the transformation from the tool to the sensor frame

and outputs

- `b_tip`: 3x1 vector containing the estimated position of the tool tip in the tool frame
- `b_post`: 3x1 vector containing the estimated position of the calibration post in the sensor frame

2.3 Computing Expected C_i

2.3.1 Method

To calculate the expected coordinates of the EM markers on the calibration object, we first compute the transformation, $\mathbf{F}_d$ between the optical tracker and EM tracker coordinates.

$$D_j = \mathbf{F}_d \cdot \mathbf{d}_j \tag{9}$$

We also calculate the transformation, $\mathbf{F}_a$ between the calibration object and optical tracker coordinates.

$$A_j = \mathbf{F}_a \cdot \mathbf{a}_j \tag{10}$$

Finally, using these calculated transformations as well as the input data of the EM markers on the calibration object, we can calculate the expected coordinates of the EM markers on the calibration object for a distorted calibration data set.

$$C_i^{(expected)} = \mathbf{F}_d^{-1} \cdot \mathbf{F}_a \cdot \mathbf{c}_i \tag{11}$$

2.3.2 Code Details

The MATLAB function `compute_expected_Ci` implements the above algorithm:

```
C_expected = compute_expected_Ci(data_base_body, data_base_readings, data_opt_obj_body,
                 data_opt_obj_readings, data_EM_obj_body, N_frames)
```

The function inputs

- `data_base_body`: N_base x 3 matrix of positions of optical markers on EM base based on calibration object

- `data_base_readings`: N_frames number of matrices of size N_base x 3 of positions of optical markers on EM base based on sensor readings
- `data_opt_obj_body`: N_opt_obj x 3 matrix of positions of optical markers on calibration object
- `data_opt_obj_readings`: N_frames number of matrices of size N_opt_obj x 3 of positions of optical markers on calibration object based on sensor readings
- `data_EM_obj_body`: N_EM_obj x 3 matrix of positions of EM markers on calibration object
- `N_frames`: Number of data frames of data from sensor readings

and outputs

- `C_expected`: Cell array with N_frames number of data containing the calculated coordinates of the EM markers on the calibration object.

2.3.3 Results

For the debug data cases provided (*a,b,c,d,e,f,g*), the expected coordinates of the EM markers on the calibration object was calculated and compared to the actual coordinates given to us in the output data files. Specifically, the body calibration and calibration readings data files were read for each debug test case and then used to calculate the expected coordinates. These expected coordinates were then compared against the actual coordinates in the output data file that was read for that specific test case. Finally, the mean error in *(x,y,z)* for each test case was calculated and can be found in Table 1.

Table 1: The mean error calculated for each debug test case in the expected coordinates of the EM markers on the calibration object.

	A	B	C	D	E	F	G
Error in C_i	0.0049	0.4709	0.3841	0.0115	1.5668	1.7237	1.7661

For the data cases with unknown output values (*h,i,j,k*), the expected coordinates of the EM markers on the calibration object were calculated as well. These values were stored in the same format found as the output files for the debug cases in files found in the *"Output"* folder of the code base.

2.4 EM Probe Pivot Calibration

2.4.1 Method

The EM tracking data can be used to perform a pivot calibration for the EM probe and determine the position of the dimple in the calibration post relative to the EM tracker base coordinate system. To perform the pivot calibration calculation, the first frame of the input data is used to define a local probe coordinate system, F_{origin}. This origin frame is then centered by the mean, similar to Section 2.1.1. This allows for the positions of the EM markers to be calculated in the tool frame, g_j. Using these positions, we can iterate through each frame of pivot calibration data and compute the transformation matrices, F_G, using the `correspondence_registration_SVD` function such that:

$$G_j = \mathbf{F}_G[k] \cdot \mathbf{g}_j \tag{12}$$

Finally, these transformations can be used with the `pivot_calibration` function to find the tool tip position and calibration post position.

2.4.2 Code Details

The MATLAB function `pivot_calibration_EM` implements the above algorithm:

```
[EM_post] = pivot_calibration_EM(data_EM_probe, N_frames)
```

The function inputs

- `data_EM_probe`: N_frames number of matrices of size N_EM_probe x 3 of positions of EM markers on probe
- `N_frames`: Number of data frames of data from sensor readings

and outputs

- `EM_post`: 3x1 vector of the position of the EM calibration post in the EM probe coordinate frame

2.4.3 Results

For the debug data cases provided (*a,b,c,d,e,f,g*), the position of the calibration post dimple in the EM probe coordinate frame was calculated and compared to the actual position given in the output data files. Specifically, the EM pivot data files were read to calculate the calibration post position. This result was compared against the actual position found in the provided output data file for each specific test case. Finally, the error in *(x,y,z)* for each test case was calculated and can be found in Table 2.

Table 2: The error calculated for each debug test case in the position of the calibration post dimple in the EM probe coordinate frame.

	A	B	C	D	E	F	G
Error in EM_{post}	0.0024	0.0041	0.0028	0.0027	0.0100	0.0190	0.0122

For the data cases with unknown output values (*h,i,j,k*), the position of the calibration post dimple in the EM probe coordination frame was also calculated. This value can be found in Table 3 and was also stored in the same format found as the output files for the debug cases in files found in the *"Output"* folder of the code base.

Table 3: The position of the calibration post dimple in the EM probe coordinate frame for each unknown test case.

	H	I	J	K
P_{EM_post}	(181.05,182.96,199.87)	(201.39,207.12,196.52)	(207.74,205.29,188.11)	(198.40,201.69,205.08)

2.5 Optical Probe Pivot Calibration

2.5.1 Method

The optical probe tracking data can be used to perform a pivot calibration for the optical probe and determine the position of the dimple in the calibration post relative to the optical tracker base coordinate system. The process to calculate the position of the calibration post is similar to Section 2.4, however, there is one key change. First, the transformation between the optical frame and EM frame is calculated using the EM tracker optical markers with the `correspondence_registration_SVD` function. This transformation is then applied to all of the tool markers so that the correspondence registration and pivot calibration results in positions with respect to the EM frame instead of the optical frame.

2.5.2 Code Details

The MATLAB function `pivot_calibration_opt` implements the above algorithm:

```
[opt_post] = pivot_calibration_optical(data_opt_base, N_opt_probe, data_opt_probe, N_frames,
                                       data_body_base)
```

The function inputs

- `data_opt_base`: N_frames number of matrices of size N_base x 3 of positions of optical markers on EM base

- `N_opt_probe`: number of optical markers on probe
- `data_opt_probe`: N_frames number of matrices of size N_opt_obj x 3 of positions of optical markers on probe
- `N_frames`: Number of data frames of data from sensor readings
- `data_body_base`: N_base x 3 matrix of positions of optical markers on EM base

and outputs

- `opt_post`: 3x1 vector of the position of the optical calibration post in the EM probe coordinate frame

2.5.3 Results

For the debug data cases provided (a,b,c,d,e,f,g), the position of the optical calibration post dimple in the EM probe coordinate frame was calculated and compared to the actual position given in the output data files. Specifically, the optical pivot data files were read to calculate the calibration post position. This result was compared against the actual position found in the provided output data file for each specific test case. Finally, the error in *(x,y,z)* for each test case was calculated and can be found in Table 4.

Table 4: The error calculated for each debug test case in the position of the optical calibration post dimple in the EM probe coordinate frame.

	A	B	C	D	E	F	G
Error in opt_{post}	0.0070	0.0060	0.0013	0.0055	0.0041	0.0010	0.0055

For the data cases with unknown output values (h,i,j,k), the position of the optical calibration post dimple in the EM probe coordination frame was also calculated. This value can be found in Table 5 and was also stored in the same format found as the output files for the debug cases in files found in the *"Output"* folder of the code base.

Table 5: The position of the optical calibration post dimple in the EM probe coordinate frame for each unknown test case.

	H	I	J	K
P_{opt_post}	(393.53,400.35,194.06)	(405.90,407.60,209.66)	(396.35,402.93,190.79)	(391.34,397.05,196.42)

3 Hand-eye Calibration

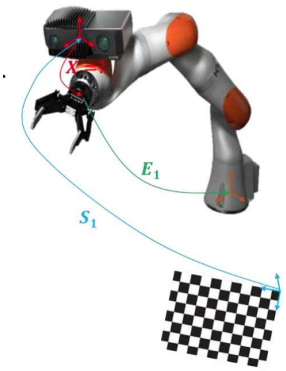

Figure 2: The transformation from robots end-effector to its base frame and transformation of calibration board to camera's reference frame.

3.1 Method

The hand-eye calibration problem as shown in Figure 2 can be formulated as finding the transformation matrix X that satisfies the following equation for a set of N measurements

$$AX = XB$$

where:

- A is the transformation matrix representing relative motion of robot's end-effector.
- B is the transformation matrix representing relative motion of camera.
- X is the unknown transformation matrix to be computed.

The transformation matrix X is composed of a rotation matrix R_x and a translation vector T_x:

$$X = \begin{bmatrix} R_x & T_x \\ 0 & 1 \end{bmatrix}.$$

The rotation component R_x is computed using quaternion algebra [2]. For each measurement, the quaternion representation of the robot's end-effector orientation (q_A) and the camera's orientation (q_B) are used to construct a matrix M:

$$M = \begin{bmatrix} s_A - s_B & -(v_A - v_B)^T \\ v_A - v_B & (s_A - s_B)I + \text{skew}(v_A - v_B) \end{bmatrix},$$

where:

- s_A and s_B are the scalar parts of the quaternions q_A and q_B, respectively.
- v_A and v_B are the vector parts of the quaternions q_A and q_B, respectively.
- `skew` is a function that converts a vector into a skew-symmetric matrix.

The M matrices for all measurements are stacked into a tall matrix, and Singular Value Decomposition (SVD) is performed [3]:

$$M_{\text{stacked}} = U\Sigma V^T.$$

The last column of V corresponds to the quaternion q_x representing the rotation. This quaternion is then converted to a rotation matrix R_x.

The translation component T_x is computed by solving a least-squares problem. For each measurement, the following equation is constructed:

$$(R_A - I)T_x = R_x p_B - p_A,$$

where:

- R_a is the rotation matrix derived from the robot's quaternion q_A.
- p_A and p_B are the translation vectors of the robot's end-effector and the camera, respectively.
- I is the identity matrix.

The matrices $R_A - I$ and $R_x p_B - p_A$ are stacked into tall matrices:

$$R_{\text{stacked}} = \begin{bmatrix} R_A^{(1)} - I \\ R_A^{(2)} - I \\ \vdots \\ R_A^{(N)} - I \end{bmatrix}, \quad T_{\text{stacked}} = \begin{bmatrix} R_x p_B^{(1)} - p_A^{(1)} \\ R_x p_B^{(2)} - p_A^{(2)} \\ \vdots \\ R_x p_B^{(N)} - p_A^{(N)} \end{bmatrix}.$$

The least-squares solution is computed as [1]:

$$T_x = \text{lsqr}(R_{\text{stacked}}, T_{\text{stacked}}).$$

The final transformation matrix X is constructed by combining R_x and T_x:

$$X = \begin{bmatrix} R_x & T_x \\ 0 & 1 \end{bmatrix}.$$

3.2 Code Details

The MATLAB function `handeye_calibration` implements the above algorithm. Below are the details of the function:

3.2.1 Function Signature

```
X = handeye_calibration(q_robot_config, q_camera_config, t_robot_config, t_camera_config)
```

3.2.2 Inputs

- `q_robot_config`: $N \times 4$ matrix of quaternions representing the robot's end-effector orientations (each row: [scalar, vector]).
- `q_camera_config`: $N \times 4$ matrix of quaternions representing the camera's orientations (each row: [scalar, vector]).
- `t_robot_config`: $N \times 3$ matrix of translations representing the robot's end-effector positions (each row: [x, y, z]).
- `t_camera_config`: $N \times 3$ matrix of translations representing the camera's positions (each row: [x, y, z]).

3.2.3 Outputs

- `X`: 4×4 transformation matrix that includes the rotation (R_x) and translation (T_x) components, representing the hand-eye calibration.

3.3 Results

The results of the hand-eye calibration function are presented below. The function was tested using both noise-free and noisy data, as well as subsets of the data. The transformation matrix X, which relates the camera's frame to the robot's end-effector frame, was computed in each case. The results are as follows:

3.3.1 Noise-Free Data

Using the complete noise-free dataset, the computed transformation matrix X is:

$$X = \begin{bmatrix} -0.2136 & 0.9769 & 0.0000 & 0.0760 \\ -0.9769 & -0.2136 & 0.0000 & -0.0482 \\ 0.0000 & -0.0000 & 1.0000 & 0.0085 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

The quaternion representation of the rotation part of the transformation is:

$$R_q = \begin{bmatrix} 0 \\ 0 \\ -0.778967 \\ 0.6270649 \end{bmatrix}$$

3.3.2 Noisy Data

Using the complete noisy dataset, the computed transformation matrix X is:

$$X_{\text{noisy}} = \begin{bmatrix} -0.2137 & 0.9769 & -0.0000 & 0.0758 \\ -0.9769 & -0.2137 & -0.0000 & -0.0484 \\ -0.0000 & 0.0000 & 1.0000 & 0.0083 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

$$R_q{}^{\text{noisy}} = \begin{bmatrix} 0 \\ 0 \\ -0.779006 \\ 0.6270165 \end{bmatrix}$$

The Euclidean distance of the translation part of the calculated transformation using the noisy data from the translation part of the calculated transformation using the noise-free data is calculated as the error introduced by the noise.

$$\text{Translation Error}_{\text{noisy}} \approx 0.000346$$

The error for the rotation part of the transformation is calculated by norm of the difference of two quaternions as:

$$\text{Rotation Error} \approx 0.0000621$$

3.3.3 Noise-Free Data (First Half)

Using the first half (5 configurations) of the noise-free dataset, the computed transformation matrix X is:

$$X_{\text{half}} = \begin{bmatrix} -0.1312 & 0.9695 & -0.2072 & 0.4941 \\ -0.3872 & -0.2425 & -0.8896 & 0.4483 \\ -0.9126 & -0.0365 & 0.4072 & 0.2884 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

3.3.4 Noise-Free Data (Second Half)

Using the second half (5 configurations) of the noise-free dataset, the computed transformation matrix X is:

$$X_{\text{half2}} = \begin{bmatrix} -0.8864 & 0.4494 & 0.1115 & -0.1039 \\ -0.3167 & -0.7641 & 0.5620 & -0.1104 \\ 0.3377 & 0.4629 & 0.8196 & -0.1265 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

3.3.5 Noisy Data (First Half)

Using the first half (5 configurations) of the noisy dataset, the computed transformation matrix X is:

$$X_{\text{half\_noisy}} = \begin{bmatrix} -0.1312 & 0.9695 & -0.2071 & 0.4938 \\ -0.3872 & -0.2424 & -0.8896 & 0.4478 \\ -0.9126 & -0.0365 & 0.4072 & 0.2880 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

3.3.6 Noisy Data (Second Half)

Using the second half (5 configurations) of the noisy dataset, the computed transformation matrix X is:

$$X_{\text{half2\_noisy}} = \begin{bmatrix} -0.8864 & 0.4494 & 0.1115 & -0.1041 \\ -0.3167 & -0.7641 & 0.5620 & -0.1105 \\ 0.3377 & 0.4629 & 0.8196 & -0.1266 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

4 Discussion

The results from the stereotactic navigation system analysis show the importance of proper calibration methods and how calibration methods need to be adjusted for different types of equipment. Overall, we see that our methods produced very minimal error between the calculated values and the debug cases provided. Particularly for the pivot calibration calculations with the EM probe and optical probe, we see that the error between the calculated and given values are very small. While the error for the expected coordinates in the EM markers on the calibration object are somewhat higher than the pivot calibration errors, they are still small. This provides confidence in our methods viability and the values produced for the unknown test cases. Furthermore, we see the practicality of modular functions as the correspondence registration and pivot calibration functions were repeatedly used in our calculations. Finally, the importance of properly reading the data was seen. By ensuring that we created read functions for all of the input data that organized the data effectively and properly, it allowed for easy implementation when testing with our calculation functions.

The results from the hand-eye calibration demonstrated the differences between using noise-free and noisy data. The transformation matrices computed using the complete noise-free and noisy datasets are very similar, with only minor differences in the translation components. This indicates that the algorithm is robust to small amounts of noise in the input data. When using only half of the data (either the first or second half), the computed transformation matrices differ significantly from those obtained using the full dataset. This is particularly evident in the rotation components, which show larger variations. This suggests that using fewer data points reduces the accuracy and reliability of the calibration. The transformation matrices computed using noisy subsets are very similar to those computed using noise-free subsets. However, the presence of noise introduces small deviations in the translation components. This further highlights the robustness of the algorithm to noise, even when using limited data. The results indicate that the algorithm performs best when using the complete dataset, as it provides more accurate and consistent results. Using subsets of the data, especially in the presence of noise, leads to less reliable calibration. Therefore, it is recommended to use as much data as possible for hand-eye calibration to ensure accuracy.

5 Conclusion

The stereotactic navigation system analysis allowed us to implement modular functions that conduct a correspondence-based registration method with the SVD algorithm and a pivot calibration algorithm. This allows us to calculate the expected coordinates of the EM markers on the calibration object, the position of the calibration post dimple relative to the EM tracker coordinate system, and the position of the optical tracker calibration post dimple relative to the EM tracker coordinate system. The significantly low errors calculated with the debug data provide confidence for the values calculated with the unknown data.

The hand-eye calibration function successfully computes the transformation matrix X, which relates the camera's frame to the robot's end-effector frame. The results demonstrate that the algorithm is robust to noise, as the transformation matrices computed using noisy data are very similar to those obtained with noise-free data. However, the accuracy and reliability of the calibration decrease when using subsets of the data, particularly in the rotation components. This highlights the importance of using the complete dataset for optimal performance.

6 Helper Functions

Several helper functions were created to make the implementations of the core functions easier. The new helper functions for THA3 include:

- `[N_base, data_base, N_opt_obj, data_opt_obj, N_EM_obj, data_EM_obj] = read_calbody(filename)`
- `[N_base, data_base, N_opt_obj, data_opt_obj, N_EM_obj, data_EM_obj, N_frames] = read_calreadings(filename)`
- `[N_EM_probe, data_EM_probe, N_frames] = read_empivot(filename)`
- `[N_base, data_base, N_opt_probe, data_opt_probe, N_frames] = read_optpivot(filename)`
- `[N_EM_obj, data_EM_obj, N_frames, P_EM_probe, P_opt_probe] = read_output(filename)`
- `write_output(filename, N_EM_obj, data_EM_obj, N_frames, P_EM_probe, P_opt_probe)`

7 Contributions

Daniyal Maroufi worked on PA1 and PA2.

Anas Yousaf worked on PA1.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, Sep. 1987, ISSN: 1939-3539. DOI: 10.1109/TPAMI.1987.4767965.

[2] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, vol. 4, no. 4, pp. 629–642, Apr. 1987, ISSN: 1520-8532. DOI: 10.1364/JOSAA.4.000629.

[3] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.