Programming Assignment 2 - Computer Integrated Surgery I

Pranhav Sundararajan and Luiza Brunelli

November 2025

1 Overview

This programming assignment extends the first calibration and registration task to include electromagnetic (EM) distortion modeling and its application in a complete navigation workflow. The objective is to fit polynomial functions that characterize EM field distortion, use these functions to "dewarp" the EM tracker space, and then apply the corrected data to repeat the probe pivot calibration, compute registration to the CT coordinate system, and finally report the probe tip positions in CT coordinates for a sequence of navigation frames.

1.1 Context of the Problem

The scenario models a stereotactic navigation system that uses an EM tracking device to measure 3D marker positions relative to a fixed EM base. These measurements are distorted by the EM field. To compensate, an optical tracking system with negligible measurement error provides accurate ground-truth positions for calibration. The calibration object contains both optical and EM markers at known positions in its own coordinate frame, and additional optical markers are placed on the EM base. By observing the calibration object in multiple poses, the relationship between the optical and EM coordinate systems can be established.

The experimental setup (Figure 1) consists of the EM tracker base, the optical tracker on a tripod, the calibration object with LEDs and EM coils, and an EM pointer probe used for pivot calibration and navigation. The optical tracker provides the distortion-free reference required to model the EM distortion field.

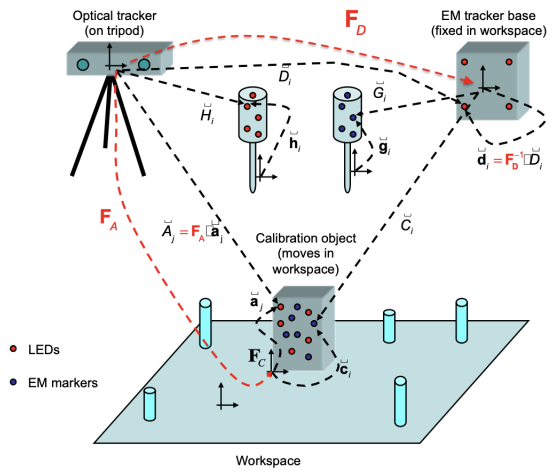

Figure 1: Problem scenario diagram from the assignment.

1.2 Assignment Goals and Methods

1. Process the calibration body data to compute corresponding marker positions in each calibration frame.
2. Fit a polynomial distortion correction function, as described in class, to model the mapping between measured and true EM positions.
3. Apply this distortion correction to the EM probe data and repeat the pivot calibration to obtain an improved probe tip position.
4. Using the corrected probe calibration, compute the locations of fiducial points with respect to the EM tracker base coordinate system.
5. Determine the registration frame F_{reg} that maps EM coordinates to CT coordinates.
6. Apply the distortion correction and registration transform to all subsequent EM readings to compute the probe tip positions in CT coordinates.

The methods used include 3D point-set registration to estimate rigid transformations, polynomial distortion fitting for EM correction, and least-squares pivot calibration for computing the probe tip position. Together, these steps implement the full calibration to navigation pipeline required for accurate stereotactic tracking, where each frame of EM data can be transformed and visualized in the CT image coordinate system.

2 Mathematical Approach

2.1 3D Point-Set to Point-Set Registration

Given two corresponding point sets $\mathbf{P} = \{p_1, p_2, \ldots, p_N\}$ and $\mathbf{Q} = \{q_1, q_2, \ldots, q_N\}$, where each point $p_i, q_i \in \mathbb{R}^3$, we seek a rigid transformation consisting of rotation matrix $\mathbf{R} \in SO(3)$ and translation vector $\mathbf{t} \in \mathbb{R}^3$ that transforms the point set P to the point set Q, while minimizing the least-squares error:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|q_i - (\mathbf{R}p_i + \mathbf{t})\|^2 \tag{1}$$

Step 1: Compute Centroids

The first step is to compute the mean of both point sets, P and Q.

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i, \quad \bar{q} = \frac{1}{N}\sum_{i=1}^{N} q_i \tag{2}$$

Step 2: Center the Point Sets

Next, using the average of each point-set, each individual point must be centered by subtracting the average from it.

$$p_i^c = p_i - \bar{p}, \quad q_i^c = q_i - \bar{q} \tag{3}$$

Now we can convert this problem to simply find the rotation matrix that satisfies the following approximation:

$q_i^c \approx \mathbf{R} p_i^c$

Step 3: Construct Cross-Covariance Matrix

Next, the cross-covariance matrix needs to be calculated which will later be decomposed to calculate the rotation matrix.

$$\mathbf{H} = (\mathbf{P}^c)^T \mathbf{Q}^c = \sum_{i=1}^{N} (p_i^c)(q_i^c)^T \tag{4}$$

where $\mathbf{P}^c$ and $\mathbf{Q}^c$ are $N \times 3$ matrices with rows p_i^c and q_i^c.

Step 4: Singular Value Decomposition

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5}$$

where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{3\times3}$ are orthogonal matrices and $\mathbf{\Sigma}$ is a diagonal matrix.

Step 5: Compute Rotation Matrix

To ensure a proper rotation with $\det(\mathbf{R}) = +1$:

$$\mathbf{S} = \text{diag}(1, 1, \det(\mathbf{V}\mathbf{U}^T)) \tag{6}$$

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T \tag{7}$$

Step 6: Compute Translation

With the best-fit R calculated, we can now solve for the translation vector by assigning it as the difference between the average of the transformed point set, Q, and the rotated average of the point-set P.

$$\mathbf{t} = \bar{q} - \mathbf{R}\bar{p} \tag{8}$$

Corner Case: Degenerate Configurations

When input points are coplanar or collinear, $\mathbf{H}$ becomes rank-deficient ($\sigma_i = 0$ for some singular values), causing $\det(\mathbf{V}\mathbf{U}^T) = 0$. The rotation is then undetermined in the degenerate direction(s).In such cases, the plane normal or line direction can be used to complete the rotation matrix; otherwise, at the very least, singular values should be checked to detect degeneracy and raise a flag when this could occur [1].

2.2 Pivot Calibration

The Pivot calibration technique determines the probe tip position $\mathbf{p}_{\text{tip}}$ in probe coordinates and the dimple position $\mathbf{P}_{\text{dimple}}$ in tracker coordinates. The probe tip remains stationary at a fixed dimple position while the probe is rotated through N_{frames} poses.

At each frame k, the probe and the dimple positions are related by the equation:

$$\mathbf{P}_{\text{dimple}} = \mathbf{R}[k] \cdot \mathbf{p}_{\text{tip}} + \mathbf{p}[k] \tag{9}$$

where $[\mathbf{R}[k], \mathbf{p}[k]]$ is the probe pose obtained via point-set registration between the probe marker positions in probe coordinates $\mathbf{g}$ and measured positions $\mathbf{G}[k]$.

The probe coordinate system is defined using the first frame by centering markers at their centroid:

$$\mathbf{g}_j = G_j[1] - \frac{1}{N_G} \sum_{i=1}^{N_G} G_i[1] \tag{10}$$

Rearranging Equation (9) and stacking all frames yields the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$:

$$\begin{bmatrix} \mathbf{R}[1] & -\mathbf{I} \\ \vdots & \vdots \\ \mathbf{R}[N] & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{\text{tip}} \\ \mathbf{P}_{\text{dimple}} \end{bmatrix} = \begin{bmatrix} -\mathbf{p}[1] \\ \vdots \\ -\mathbf{p}[N] \end{bmatrix} \tag{11}$$

The least-squares solution is $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$, yielding $\mathbf{p}_{\text{tip}} = \mathbf{x}(1:3)$ and $\mathbf{P}_{\text{dimple}} = \mathbf{x}(4:6)$ [1, 3].

2.3 Distortion Correction

The electromagnetic tracker exhibits systematic, repeatable distortion that varies spatially across the workspace. To accurately model the distortions we characterized them as Bernstein polynomials to and corrected them using calibration data.

2.3.1 Measuring Distortion

For each calibration frame k and marker i, we compute the distortion vector:

$$\boldsymbol{\delta}_{k,i} = \mathbf{C}_{\text{measured}}[k, i] - \mathbf{C}_{\text{expected}}[k, i] \tag{12}$$

where $\mathbf{C}_{\text{expected}}$ are the true marker positions (computed via optical tracking and registration) and $\mathbf{C}_{\text{measured}}$ are the raw EM readings.

2.3.2 Polynomial Distortion Model

We model the distortion as a 3D polynomial function using Bernstein basis polynomials of degree 5. Bernstein basis polynomials were used because of their stability and predictability during computation and their smooth interpolation on the normalized domain [0,1]. The polynomials are modeled as follows [2].

First, for position $\mathbf{q} = [q_x, q_y, q_z]^T$, the distortion is:

$$\boldsymbol{\delta}(\mathbf{q}) = [\delta_x(\mathbf{q}), \delta_y(\mathbf{q}), \delta_z(\mathbf{q})]^T \tag{13}$$

Now, each component of the distortion vector is modeled separately using the 3D Bernstein polynomial:

$$\delta_x(u, v, w) = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=0}^{n} c_{x,i,j,k} \cdot B_{i,n}(u) \cdot B_{j,n}(v) \cdot B_{k,n}(w) \tag{14}$$

The Bernstein basis function is:

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{15}$$

and $(u, v, w) \in [0, 1]^3$ are normalized coordinates computed from the workspace bounding box $[\mathbf{q}_{\min}, \mathbf{q}_{\max}]$:

$$u = \frac{q_x - q_{x,\min}}{q_{x,\max} - q_{x,\min}}, \quad v = \frac{q_y - q_{y,\min}}{q_{y,\max} - q_{y,\min}}, \quad w = \frac{q_z - q_{z,\min}}{q_{z,\max} - q_{z,\min}} \tag{16}$$

2.3.3 Fitting the Polynomial

For each dimension (x, y, z), we construct a design matrix $\mathbf{A} \in \mathbb{R}^{M\times(n+1)^3}$ where each row m contains the Bernstein basis functions evaluated at the normalized position (u_m, v_m, w_m):

$$A_{m,\ell} = B_{i,n}(u_m) \cdot B_{j,n}(v_m) \cdot B_{k,n}(w_m) \tag{17}$$

where ℓ is the flattened index corresponding to coefficients (i, j, k).

The coefficients are obtained via the least-squares method using the design matrix $\mathbf{A}$:

$$\mathbf{c}_x = \mathbf{A} \backslash \boldsymbol{\delta}_x, \quad \mathbf{c}_y = \mathbf{A} \backslash \boldsymbol{\delta}_y, \quad \mathbf{c}_z = \mathbf{A} \backslash \boldsymbol{\delta}_z \tag{18}$$

where $\boldsymbol{\delta}_x, \boldsymbol{\delta}_y, \boldsymbol{\delta}_z \in \mathbb{R}^M$ are vectors of the measured distortions in each dimension.

2.3.4 Applying the Correction

Now, we can correct the raw EM readings $\mathbf{q}_{\text{raw}}$ using the following procedure:

1. Normalize to $[0, 1]^3$ using Equation (18)
2. Evaluate the Bernstein polynomial to compute $\boldsymbol{\delta}(\mathbf{q}_{\text{raw}})$ using the fitted coefficients
3. Apply correction: $\mathbf{q}_{\text{corrected}} = \mathbf{q}_{\text{raw}} - \boldsymbol{\delta}(\mathbf{q}_{\text{raw}})$

2.4 Frame Transformations

There are multiple frame transformations required to solve this assignment and attain the probe tip position in CT coordinates for each respective data frame. However, they all use the rotation matrix and translation vector format to represent these transformations [1].

Expected EM markers: The first transformation was to compute $\mathbf{C}_{\text{expected}}$, for which the transformation from the optical tracker to the EM base frame and from the calibration object to the optical tracker have to be calculated first. With these transformations the procedure to compute $\mathbf{C}_{\text{expected}}$ was as follows: For calibration frame k, let transformations from known marker positions to optical observations be: $\mathbf{F}_D = [\mathbf{R}_D, \mathbf{t}_D]$ (EM base) and $\mathbf{F}_A = [\mathbf{R}_A, \mathbf{t}_A]$ (cal object). Expected positions are:

$$\mathbf{C}_{\text{expected}}[k, i] = \mathbf{R}_D^T(\mathbf{R}_A \mathbf{c}_i + \mathbf{t}_A - \mathbf{t}_D) \tag{19}$$

Fiducial positions: To attain the fiducial positions in EM coordinates, the transformation rotation matrix and translation vector from the corrected markers were used as follows: For fiducial j, with probe pose $[\mathbf{R}_G, \mathbf{t}_G]$ from corrected markers,:

$$\mathbf{b}_j = \mathbf{R}_G \mathbf{p}_{\text{tip}} + \mathbf{t}_G \tag{20}$$

Registration: To find $\mathbf{F}_{\text{reg}}$, which is the transformation to map EM to CT coordinates the previously described 3D point-set to 3D point-set transformation algorithm was used:

$$[\mathbf{R}_{\text{reg}}, \mathbf{t}_{\text{reg}}] = \text{find\_transformation}(\mathbf{b}^{\text{EM}}, \mathbf{b}^{\text{CT}}) \tag{21}$$

Navigation: Putting the above transformations together, to calculate the probe tip in CT coordinates for frame j:

$$\mathbf{v}_j = \mathbf{R}_{\text{reg}}(\mathbf{R}_G \mathbf{p}_{\text{tip}} + \mathbf{t}_G) + \mathbf{t}_{\text{reg}} \tag{22}$$

where $[\mathbf{R}_G, \mathbf{t}_G]$ is computed from distortion-corrected markers.

3 Algorithmic Approach

3.1 Programming Language and Libraries

The implementation of this algorithm uses **MATLAB** with only standard MATLAB functions and no external libraries.

3.2 3D Point-Set Registration

Algorithm 1 find_transformation(P, Q)

Input: $P, Q \in \mathbb{R}^{N \times 3}$ (corresponding point sets)
Output: $\mathbf{R} \in \mathbb{R}^{3 \times 3}$, $\mathbf{t} \in \mathbb{R}^{3 \times 1}$
Compute centroids: $\bar{p} = \text{mean}(P)$, $\bar{q} = \text{mean}(Q)$
Center points: $P^c = P - \bar{p}$, $Q^c = Q - \bar{q}$
Compute cross-covariance: $\mathbf{H} = (P^c)^T Q^c$
Perform SVD: $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \text{svd}(\mathbf{H})$
Set $\mathbf{S} = \mathbf{I}_3$, $\mathbf{S}(3,3) = \det(\mathbf{V}\mathbf{U}^T)$
Compute rotation: $\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T$
Compute translation: $\mathbf{t} = \bar{q} - \mathbf{R}\bar{p}$
Return $\mathbf{R}, \mathbf{t}$

3.3 Distortion Correction Fitting

Algorithm 2 Fit Distortion Correction Model

Input: Calibration frames $\{\mathbf{C}_{\text{measured}}[k], \mathbf{C}_{\text{expected}}[k]\}$, degree $n = 5$
Output: Distortion coefficients and bounding box
Initialize empty arrays: `all_positions`,`all_distortions`
for each frame k and marker i **do**
  Append $\mathbf{C}_{\text{measured}}[k, i]$ to `all_positions`
  Compute $\boldsymbol{\delta} = \mathbf{C}_{\text{measured}}[k, i] - \mathbf{C}_{\text{expected}}[k, i]$
  Append $\boldsymbol{\delta}$ to `all_distortions`
end for
Compute bounding box: $\mathbf{q}_{\min} = \min(\texttt{all\_positions})$, $\mathbf{q}_{\max} = \max(\texttt{all\_positions})$
Normalize positions: $\mathbf{u} = (\texttt{all\_positions} - \mathbf{q}_{\min}) \oslash (\mathbf{q}_{\max} - \mathbf{q}_{\min})$
Build Bernstein design matrix: $\mathbf{A} = \text{create\_bernstein\_matrix}(\mathbf{u}, n)$
for dimension $d \in \{x, y, z\}$ **do**
  Solve: $\mathbf{c}_d = \mathbf{A} \backslash \texttt{all\_distortions}(:, d)$
end for
Return $\{\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z, \mathbf{q}_{\min}, \mathbf{q}_{\max}\}$

3.4 Apply Distortion Correction

Algorithm 3 correctDistortion(q_raw, coeffs)

Input: $\mathbf{q}_{\text{raw}} \in \mathbb{R}^{1 \times 3}$, distortion coefficients
Output: $\mathbf{q}_{\text{corrected}} \in \mathbb{R}^{1 \times 3}$
Normalize: $\mathbf{u} = (\mathbf{q}_{\text{raw}} - \mathbf{q}_{\min}) \oslash (\mathbf{q}_{\max} - \mathbf{q}_{\min})$
Evaluate Bernstein: $\mathbf{A}_{\text{row}} = \text{create\_bernstein\_matrix}(\mathbf{u}, n)$
Compute distortion: $\delta_x = \mathbf{A}_{\text{row}} \cdot \mathbf{c}_x$, $\delta_y = \mathbf{A}_{\text{row}} \cdot \mathbf{c}_y$, $\delta_z = \mathbf{A}_{\text{row}} \cdot \mathbf{c}_z$
Correct: $\mathbf{q}_{\text{corrected}} = \mathbf{q}_{\text{raw}} - [\delta_x, \delta_y, \delta_z]$
Return $\mathbf{q}_{\text{corrected}}$

3.5 Pivot Calibration

Algorithm 4 pivot_calibration(G_frames)

Input: Cell array of marker frames $\{\mathbf{G}[k]\}_{k=1}^{N_{\text{frames}}}$
Output: $\mathbf{P}_{\text{dimple}} \in \mathbb{R}^{1\times 3}$
Define probe coordinates: $\mathbf{g} = \mathbf{G}[1] - \text{mean}(\mathbf{G}[1])$
Initialize: $\mathbf{A} = \text{zeros}(3N_{\text{frames}}, 6)$, $\mathbf{b} = \text{zeros}(3N_{\text{frames}}, 1)$
for $k = 1$ to N_{frames} **do**
  Compute transformation: $[\mathbf{R}[k], \mathbf{p}[k]] = \text{find\_transformation}(\mathbf{g}, \mathbf{G}[k])$
  $\text{row\_start} = 3(k-1) + 1$
  $\mathbf{A}(\text{row\_start} : \text{row\_start} + 2, 1:3) = \mathbf{R}[k]$
  $\mathbf{A}(\text{row\_start} : \text{row\_start} + 2, 4:6) = -\mathbf{I}_3$
  $\mathbf{b}(\text{row\_start} : \text{row\_start} + 2) = -\mathbf{p}[k]$
end for
Solve: $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$
Return $\mathbf{P}_{\text{dimple}} = \mathbf{x}(4:6)$

Note: The probe tip $\mathbf{p}_{\text{tip}} = \mathbf{x}(1:3)$ is also computed but can be extracted separately using the same system when needed for fiducial localization.

3.6 The complete PA2 pipeline

Algorithm 5 PA2 Main Processing Script

Step 1: Compute Expected Values and Fit Distortion
Read calibration data: $\{\mathbf{D}[k], \mathbf{A}[k], \mathbf{C}_{\text{measured}}[k]\}$
for each frame k **do**
  $[\mathbf{R}_D, \mathbf{t}_D] = \text{find\_transformation}(\mathbf{d}, \mathbf{D}[k])$
  $[\mathbf{R}_A, \mathbf{t}_A] = \text{find\_transformation}(\mathbf{a}, \mathbf{A}[k])$
  for each marker i **do**
    $\mathbf{C}_{\text{expected}}[k,i] = \mathbf{R}_D^T(\mathbf{R}_A\mathbf{c}_i + \mathbf{t}_A - \mathbf{t}_D)$
  end for
end for
Use distortion model in Algorithm 2 to calculate: `distortion_coeffs` using $\mathbf{C}_{\text{measured}}$ and $\mathbf{C}_{\text{expected}}$.
Step 2: Corrected Pivot Calibration
Read EM pivot data: $\{\mathbf{G}_{\text{pivot}}[k]\}$
for each frame k and marker i **do**
  $\mathbf{G}_{\text{corrected}}[k,i] = \text{correctDistortion}(\mathbf{G}_{\text{pivot}}[k,i], \texttt{dist\_coeffs})$
end for
$\mathbf{P}_{\text{post}} = \text{pivot\_calibration}(\mathbf{G}_{\text{corrected}})$
Compute probe tip: $\mathbf{p}_{\text{tip}} = \text{compute\_probe\_tip}(\mathbf{G}_{\text{corrected}}, \mathbf{g}, \mathbf{P}_{\text{post}})$

Step 3: Compute Fiducial Positions in EM Coordinates
Read fiducial data: $\{\mathbf{G}_{\text{fiducials}}[j]\}$ for $j = 1$ to N_B
for each fiducial j **do**
  for each marker i **do**
    $\mathbf{G}_{\text{corrected}}[j,i] = \text{correctDistortion}(\mathbf{G}_{\text{fiducials}}[j,i], \texttt{dist\_coeffs})$
  end for
  $[\mathbf{R}_G, \mathbf{t}_G] = \text{find\_transformation}(\mathbf{g}, \mathbf{G}_{\text{corrected}}[j])$
  $\mathbf{b}_j^{\text{EM}} = \mathbf{R}_G\mathbf{p}_{\text{tip}} + \mathbf{t}_G$
end for

Step 4: Registration to CT Coordinates
Read CT fiducials: $\{\mathbf{b}_j^{\text{CT}}\}$ for $j = 1$ to N_B
$[\mathbf{R}_{\text{reg}}, \mathbf{t}_{\text{reg}}] = \text{find\_transformation}(\mathbf{b}^{\text{EM}}, \mathbf{b}^{\text{CT}})$

Step 5: Process Navigation Data
Read navigation data: $\{\mathbf{G}_{\text{nav}}[j]\}$ for $j = 1$ to N_{nav}
for each navigation frame j **do**
  for each marker i **do**
    $\mathbf{G}_{\text{corrected}}[j,i] = \text{correctDistortion}(\mathbf{G}_{\text{nav}}[j,i], \texttt{dist\_coeffs})$
  end for
  $[\mathbf{R}_G, \mathbf{t}_G] = \text{find\_transformation}(\mathbf{g}, \mathbf{G}_{\text{corrected}}[j])$
  $\mathbf{v}_j^{\text{EM}} = \mathbf{R}_G\mathbf{p}_{\text{tip}} + \mathbf{t}_G$
  $\mathbf{v}_j^{\text{CT}} = \mathbf{R}_{\text{reg}}\mathbf{v}_j^{\text{EM}} + \mathbf{t}_{\text{reg}}$
end for

Step 6: Write Output
Write $\{\mathbf{v}_j^{\text{CT}}\}$ to output file in requested format.

4 Overview of Program Structure

4.1 Program Organization

The implementation is organized into modular MATLAB functions grouped by functionality. The main processing script (pa2_run_script.m) orchestrates the complete PA2 pipeline by calling specialized functions for data I/O, core

algorithms, and output generation.

4.2 Function Descriptions

4.2.1 Data Reading Functions

Helper Function:

- `M = read_block(fid, N)`: Reads N lines of xyz coordinates from file. Returns $N \times 3$ matrix. Skips blank lines and extracts numeric tokens using regular expressions.

Calibration Data:

- `[d, a, c] = read_calbody(path)`: Reads calibration body description file containing known marker positions. Returns three matrices: $\mathbf{d}$ ($N_D \times 3$, EM base markers), $\mathbf{a}$ ($N_A \times 3$, optical markers on cal body), and $\mathbf{c}$ ($N_C \times 3$, EM markers on cal body), all in their respective local coordinate frames.
- `[Dcells, Acells, Ccells] = read_calreadings(path)`: Reads calibration frame measurements. Returns three cell arrays where each cell k contains observations for frame k: `Dcells{k}` ($N_D \times 3$), `Acells{k}` ($N_A \times 3$), `Ccells{k}` ($N_C \times 3$).

Pivot Calibration Data:

- `Gcells = read_pivot_data(path)`: Reads EM pivot calibration data. Returns cell array where `Gcells{k}` contains $N_G \times 3$ marker positions for frame k.

Fiducial and Navigation Data:

- `Gcells_fiducials = read_emfiducials(path)`: Reads EM fiducial localization data. Returns cell array with N_B cells, where each cell contains $N_G \times 3$ marker positions when probe touches that fiducial.
- `b_CT = read_ct_fiducials(path)`: Reads CT fiducial coordinates from preoperative imaging. Returns $N_B \times 3$ matrix of fiducial positions in CT coordinate system.
- `Gcells_nav = read_emnav(path)`: Reads navigation frame data. Returns cell array where each cell contains $N_G \times 3$ marker positions for one navigation frame.

4.2.2 Core Algorithm Functions

Registration:

- `[R, t] = find_transformation(P, Q)`: Computes rigid transformation mapping $P \to Q$ using SVD-based point-set registration.

Distortion Correction:

- `distortion_coeffs` = Output of Algorithm 2 which fits degree-5 Bernstein polynomial model to measured distortion data (Section 2.3).
- `q_corrected = correctDistortion(q_raw, coeffs)`: Applies distortion correction to a single EM reading.
- `A = create_bernstein_matrix(u, degree)`: Constructs design matrix for Bernstein polynomial evaluation. Each row contains basis functions $B_{i,n}(u) \cdot B_{j,n}(v) \cdot B_{k,n}(w)$ evaluated at normalized coordinates.
- `B = bernstein_basis(i, n, t)`: Evaluates single 1D Bernstein basis function $B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$.

Calibration:

- `P_dimple = pivot_calibration(G_frames)`: Performs pivot calibration to determine post position (Section 2.2).
- `p_tip = compute_probe_tip(G_frames, g_markers, P_post)`: Extracts probe tip position in probe coordinate system from pivot calibration data.

4.2.3 Processing Pipeline Functions

Calibration Processing:

- `C_expected = compute_C_expected(d, a, c, Dcells, Acells)`: Computes expected (true) EM marker positions using optical tracking measurements as ground truth (Section 2.4.1).

Fiducial Processing:

- `b_EM = compute_fiducials_EM(Gcells_fid, g_markers, p_tip, coeffs)`: Computes fiducial landmark positions in EM tracker coordinates (Section 2.4.2).

Navigation Processing:

- `v_CT = compute_navigation_CT(Gcells_nav, g_markers, p_tip, R_reg, t_reg, coeffs)`: Computes probe tip positions in CT coordinate system for all navigation frames (Section 2.4.4).

4.2.4 Output Functions

- `write_output2(filename, v_CT)`: Writes navigation results to formatted text file.
  - *Format:* Line 1: `N_frames, filename`; Lines 2–$(N+1)$: `vx, vy, vz` with 2 decimal places

4.3 Main Processing Flow

The main script (`pa2_run_script.m`) executes the following sequence:

1. **Initialize:** Set input data paths, create `OUTPUT` directory if not exists (two levels above program directory)
2. **Read calibration data:**
   - Load calibration body definition: $[\mathbf{d}, \mathbf{a}, \mathbf{c}] =$ `read_calbody`$(\cdots)$
   - Load calibration measurements: $[$`Dcells`, `Acells`, `Ccells`$] =$ `read_calreadings`$(\cdots)$
3. **Compute expected values:**
   - Call `C_expected` = `compute_C_expected`$(\mathbf{d}, \mathbf{a}, \mathbf{c},$ `Dcells`, `Acells`$)$
   - This provides ground truth for distortion measurement
4. **Fit distortion model:**

- Concatenate all position-distortion pairs from calibration frames
- Calculate `coeffs` from `Algorithm 2` using (`positions`, `distortions`, 5)

5. **Corrected pivot calibration:**
   - Read EM pivot data: `Gcells_raw` = `read_empivot`$(\cdots)$
   - Apply distortion correction to all markers in all frames
   - Define probe coordinate system: $\mathbf{g}$ = `Gcells_corrected`$\{1\}$ − mean(`Gcells_corrected`$\{1\}$)
   - Call $\mathbf{P}_{\text{post}}$ = `pivot_calibration` (`Gcells_corrected`)
   - Call $\mathbf{p}_{\text{tip}}$ = `compute_probe_tip` (`Gcells_corrected`, $\mathbf{g}, \mathbf{P}_{\text{post}}$)
6. **Compute fiducial positions:**
   - Read fiducial data: `Gcells_fid` = `read_emfiducials`$(\cdots)$, $\mathbf{b}^{\text{CT}}$ = `read_ct_fiducials`$(\cdots)$
   - Call $\mathbf{b}^{\text{EM}}$ = `compute_fiducials_EM`(`Gcells_fid`, $\mathbf{g}, \mathbf{p}_{\text{tip}}$, `coeffs`)
7. **Compute registration:**
   - Call $[\mathbf{R}_{\text{reg}}, \mathbf{t}_{\text{reg}}]$ = `find_transformation`$(\mathbf{b}^{\text{EM}}, \mathbf{b}^{\text{CT}})$
8. **Process navigation data:**
   - Read navigation frames: `Gcells_nav` = `read_emnav`$(\cdots)$
   - Call $\mathbf{v}^{\text{CT}}$ = `compute_navigation_CT`(`Gcells_nav`, $\mathbf{g}, \mathbf{p}_{\text{tip}}, \mathbf{R}_{\text{reg}}, \mathbf{t}_{\text{reg}}$, `coeffs`)
9. **Write output:**
   - Construct output filename (e.g., `pa2-debug-a-output2.txt`)
   - Call `write_output2`(`output_path`, $\mathbf{v}^{\text{CT}}$)

4.4 Program Output

The program generates one output file per dataset:

- **Filename:** `NAME-output2.txt` (e.g., `pa2-debug-a-output2.txt`)
- **Format:**
  - Line 1: `N_frames, filename`
  - Lines 2 to $N_{\text{frames}}+1$: `vx, vy, vz` formatted with 2 decimal places (probe tip position in CT coordinates for each navigation frame)
- **Location:** `output/` directory, two levels above the main program directory

5 Validation Approach

To ensure the correctness of our program, we created unit tests with fake data for core helper functions, analyzed the residual error of our distortion correction model on real data, and performed an end-to-end comparison of our final program output against the provided ground-truth files for all debug datasets.

5.1 Unit Test: Point Set Registration (`find_transformation`)

The `find_transformation` function is a critical component, as it is used for determining probe pose and for the final EM to CT registration.

Testing Process: We created a unit test script, `test_find_transformation.m`, which performed two tests on our function using known fake data.

1. **Perfect Data:** A set of 6 known 3D points, P, was defined. A ground-truth rotation matrix R_{true} (45° around Z-axis) and translation vector t_{true} were created. A "perfect" target point set $Q_{perfect}$ was generated by applying this transformation. Our function was then called to find the transformation between P and $Q_{perfect}$.
2. **Noisy Data:** Gaussian noise (std. dev. = 0.1) was added to $Q_{perfect}$ to create Q_{noisy}. Our function was called to find the transformation between P and Q_{noisy}.

Results:

- **Perfect Data:** The error between the calculated and true transformations was very small.
  - Rotation Error (Frobenius Norm): 1.007313e-15 (using `R_error = norm(R_true - R_calc, 'fro')`
  - Translation Error (Euclidean Norm): 3.552714e-15 (using `t_error = norm(t_true - t_calc)`
- **Noisy Data:** We evaluated the Fiducial Registration Error (FRE), which is the RMS distance between the transformed source points and the noisy target points, defined as:

$$\text{FRE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||(R_{calc}P_i + t_{calc}) - Q_{noisy,i}||^2}$$

  Our calculated FRE was **0.129640**. This value is consistent with the 0.1 noise added, confirming the function's accuracy under realistic conditions.

The success of both tests provided high confidence in our registration algorithm.

5.2 Unit Test: Pivot Calibration (`pivot_calibration`)

This function calculates the probe tip's position relative to the tracker.

Testing Process: We created a `test_pivot_calibration.m` script.

1. We defined a local probe geometry g (4 markers) and two "ground-truth" vectors: $p_{tip,probe}$ (the pivot's location in the probe's local frame) and $p_{tip,tracker}$ (the fixed pivot location in the tracker's frame).
2. We simulated 20 frames of pivot data. For each frame k, we generated a random rotation R_k and calculated the corresponding translation $p_k = p_{tip,tracker} - R_k \cdot p_{tip,probe}$. The marker positions for the frame were then calculated: $G_k = (R_k g' + p_k)'$.
3. Our `pivot_calibration` function was called with this cell array of simulated G_k frames. The test was run once with this perfect data and again after adding Gaussian noise (std. dev. = 0.1) to each G_k.

Results: We measured the Euclidean distance between our calculated pivot $P_{dimple,calc}$ and the known $p_{tip,tracker}$.

- **Perfect Data:** The calculated error was very small: 1.819877e-13.
- **Noisy Data:** The calculated error was 0.054712. This small error is attributable to the injected noise and confirms our implementation is correct.

5.3 Analysis: Distortion Correction Model

For distortion correction, we validated our model by calculating its "training error" (the residual error after applying the correction to the same `calreadings` data used to compute the coefficients).

Testing Process: We ran our model on `debug-c`, which contains distortion but no noise. We fit a 5th-degree Bernstein polynomial to the distortion data. We then applied this correction to all the raw calibration points (`all_positions`) and calculated the RMSE between our corrected points and the ground-truth points (`all_expected`).

$$\text{RMSE}_{\text{distortion}} = \sqrt{\frac{1}{M}\sum_{i=1}^{M} ||p_{\text{corrected},i} - p_{\text{expected},i}||^2}$$

(where $M = N_{frames} \times N_{markers}$)

Results (for dataset `debug-c`, degree 5): The training RMSE for our distortion model was **0.028914**. This extremely low error indicates that our 5th-degree polynomial model provides an excellent fit and is highly effective at removing the systematic distortion from the EM tracker.

6 Results

The results section is divided into an analysis of the debug datasets, for which we have ground-truth outputs, and the final reported coordinates for the unknown datasets.

6.1 Debug Dataset Analysis

We ran our full pipeline on all 6 debug datasets. The final end-to-end error was calculated by comparing our program's final output file (`pa2-debug-X-output2.txt`) against the provided ground-truth file. The error metric is the navigation RMSE, defined as:

$$\text{RMSE}_{\text{nav}} = \sqrt{\frac{1}{N_{\text{frames}}}\sum_{i=1}^{N_{\text{frames}}} ||v_{\text{CT, calc},i} - v_{\text{CT, truth},i}||^2}$$

The results are presented in Table 1, along with the known error sources for each dataset.

Table 1: End-to-End RMSE for Debug Datasets vs. Error Source

table-format=1.7

Dataset	Error Sources	Final Nav. RMSE (mm)
`debug-a`	None	0.011180
`debug-b`	EM Noise	0.100000
`debug-c`	EM Distortion	0.017321
`debug-d`	OT Jiggle	0.007071
`debug-e`	All	0.110905
`debug-f`	All	0.159138

Discussion of Results: The results in Table 1 align perfectly with the expected behavior of the system.

- `debug-a` **(Ideal Case):** The baseline error is 0.011, which is small.
- `debug-c` **(Distortion Only):** The error is 0.017, which is almost identical to the ideal case. This shows that our 5th-degree polynomial distortion correction is extremely effective, removing most systematic error.
- `debug-b` **(Noise Only):** The error is 0.100. This isolates the effect of EM noise, indicating it is a significant source of error.
- `debug-d` **(Jiggle Only):** The error is 0.007, which is negligible and on par with the baseline. This suggests that random tracker "jiggle" may be averaged out during the pivot calibration and registration process, having little to no effect on the final result.
- `debug-e` **and** `-f` **(All Errors):** These datasets show the highest errors (0.111 and 0.159). Their error magnitudes are dominated by the EM noise (comparing to `debug-b`). This confirms that in a realistic scenario, once systematic distortion is corrected, random EM noise is the primary remaining source of error.

6.2 Unknown Dataset Results

Our program was run on the 2 unknown datasets. All unknown datasets include distortion, noise, and jiggle. While we cannot calculate a final RMSE without ground truth, we can report our calculated probe tip positions in the CT coordinate system, as shown in Table 2. We expect the true accuracy of these points to be similar to datasets `debug-e` and `debug-f`.

Table 2: Calculated Probe Tip Positions for Unknown Datasets

Dataset	x (mm)	y (mm)	z (mm)
`unknown-g` (Frame 1)	90.3942	82.9857	40.1636
`unknown-g` (Frame 2)	44.9459	74.7948	73.9016
`unknown-g` (Frame 3)	52.7327	117.6388	111.4476
`unknown-g` (Frame 4)	34.5287	65.1011	65.1999
`unknown-h` (Frame 1)	145.1652	39.0829	134.8244
`unknown-h` (Frame 2)	69.9514	53.4844	158.7247
`unknown-h` (Frame 3)	50.8205	146.0109	111.7214
`unknown-h` (Frame 4)	93.9497	41.9895	151.6971

7 Contributions

Both partners worked on the coding portion together. Pranhav focused on the mathematical and algorithmic approach sections of the report, and Luiza focused on the validation and results sections.

References

[1] Taylor, R.H., "Registration – Part 1," Computer-Integrated Surgery I Lecture Slides, Johns Hopkins University, Fall 2025.

[2] Taylor, R.H., "Interpolation and Deformations: A Short Cookbook," Computer-Integrated Surgical Systems and Technology Lecture Notes, Johns Hopkins University, Updated September 17, 2025.

[3] Taylor, R.H., "Linear Least Squares Review," Computer-Integrated Surgery I Lecture Notes, Johns Hopkins University, Fall 2018.