PA2-REPORT

Score Sheet

Name 1	
Email	
Other contact information (optional)	
Name 2	
Email	
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment Jiaming Zhang [illegible]

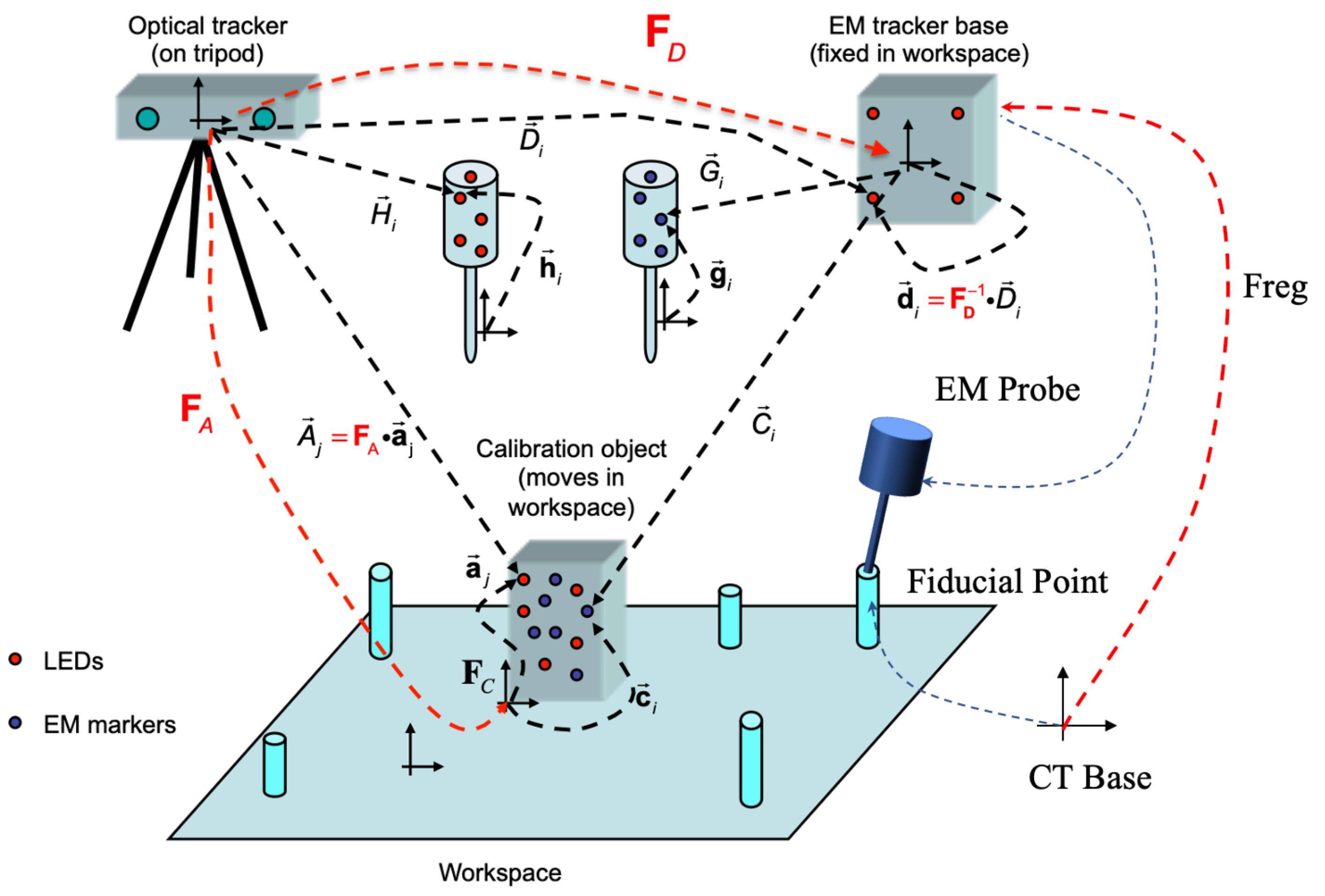
Grade Factor		
Program (40)		
Design and overall program structure	20	
Reusability and modularity	10	
Clarity of documentation and programming	10	
Results (20)		
Correctness and completeness	20	
Report (40)		
Description of formulation and algorithmic approach	15	
Overview of program	10	
Discussion of validation approach	5	
Discussion of results	10	
TOTAL	100	

I. Mathematics & Algorithms Implementation

This section introduces the mathematical principles and implemented algorithm for the 6 problems in Programming Assignment 2.

0. Scenario

The goal is to correct the distortion of the electromagnetic tracker and apply the corrected data to locate the fiducial markers in the CT coordinate system.



1. Registration and Calibration Recap

In Programming Assignment 1, we developped a math package to describe frame transformation in Cartesian Space. We also implemented point cloud to point cloud registration and pivot calibration methods. The mathematical methods and code implementation are explained and described in Ref 1. Here we'll go through a quick recap for what we included in PA1.

1) Cartesian Math Package

Rigid body transformations can be represented by matrix multiplcations. Suppose there are two Cartesian coordinate frames, A and B. Position vectors in frame A and frame B is denoted as $\vec{p}_a$ and $\vec{p}_b$, respectively. $F_{AB} = \{R_{AB}, t_{AB}\}$ represents the rigid body transformation from frame B to frame A. The relationship between $\vec{p}_a$ and $\vec{p}_b$ is $\vec{p}_b = R_{AB}\vec{p}_b + t_{AB}$.

To do this in a more compact way, we can expand the transformation and position vector to their homogeneous form, i.e.

$$F_{AB} = \begin{bmatrix} R_{AB} & t_{AB} \\ \vec{0} & 1 \end{bmatrix}, \quad p = \begin{bmatrix} \vec{p} \\ 1 \end{bmatrix} \tag{2}$$

Hence, the rigid body can be simply represented as $F_1 \cdot F_2$ and $F \cdot p$. A function that computes the homogeneous form of the given transformation frame of position vector is provided in the **"../cispa/CarteFrame.py"** .

2) Registration

Our implementation uses a non-iterative least-squares approach to match two sets of 3D points. Suppose we have two point sets and denote them as $P = \{p_i, i \in 0, 1, 2, \ldots, N\}$ and $P' = \{p'_i, i \in 0, 1, 2, \ldots, N\}$, and they are representing the same rigid body described at different poses. Hence, they should follows: $p'_i \approx Rp_i + t$, in which the error is stated as: $ei = p'_i - Rp_i - t$. Arun's method applied Singular Value Decomposition to find the R and t that minimize the following cost function:

$$\Sigma^2 = \sum_{i=1}^{N} ||p'_i - Rp_i - t||^2 \tag{3}$$

And to solve the problem, we took a two step method, i.e. Solve R first and find the corresponding t. To do this, we have to centralize the point sets to annihilate the coupling between rotation and translation as $p_{ic} = p_i - \bar{p}$, $p'_{ic} = p'_i - \bar{p}'$. Therefore, the problem reduces to:

$$\hat{R} = \underset{R}{\text{argmax}} \sum ||p'_{ic} - Rp_{ic}||^2 \tag{4}$$

And t can be found simply by:

$$\hat{t} = \bar{p}'_i - \hat{R}\bar{p}_i \tag{5}$$

The code is implemented in the **"../cispa/Registration.py"** and the function is defined as **regist_matched_points(X,Y)**. Here X and Y represent two point sets.

3) Pivot Calibration

The model of pivot calibration is described as: $\vec{p}_{pivot} = F_i\vec{p}_t = R_i\vec{p}_t + \vec{p}_i$
It holds for all point pairs on the tool. F_i is the rigid transformation from tool coordinate frame to the EM tracker coordinate frame. The calculation of pivot vector can be described as a least square problem like Eq (5).

$$\begin{bmatrix} \cdots & \cdots \\ R_i & -I \\ \cdots & \cdots \end{bmatrix} \begin{bmatrix} p_t \\ p_{pivot} \end{bmatrix} = \begin{bmatrix} \cdots \\ -p_i \\ \cdots \end{bmatrix} \tag{6}$$

By solving this least square problem, we can get the pivot vector p_{pivot} . The code is implemented in the **"../cispa/PivotCalibration.py"**. The function is defined as **calib_pivot_points(F)**. Here F is a list of homogeneous transformations.

@misc{benjamindkilleen2022Sep,
author = {Killeen, Benjamin D.},
title = {{cispa: Template for CIS I programming assignments at Johns Hopkins}},
journal = {GitHub},
year = {2022},
month = {Sep},
url = {https://github.com/benjamindkilleen/cispa}
}