Programming Assignment 5 – 600.445/645 Fall 2016

Score Sheet (hand in with report) Also, PLEASE INDICATE WHETHER YOU ARE IN 600.445 or 600.645

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 _____ _____

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	

NOTE: This is an optional assignment.

If you hand it in, I will use the grade to replace the lowest other programming assignment or written homework assignment with one exception:

You may not drop **both** HW#3 and HW#4. If these are your two lowest grades, then I will drop the lower of those two under the drop 1 homework scenario and replace the next lowest grade (other than the other of HW#3-4) with this score

CIS PA 5

MENGZE XU, LIUJIANG YAN

Summary
This report is for programming assignment 5 deformable registration implementation beyond rigid iterative closest points algorithm, containing following parts,

* Mathematical Approach and Algorithm
* Programming Structure
* Validation and Results
* Discussion
* Summary of Unknown Data

The folder contains following subfolders and Matlab files,

* DeformableRegistration
  - deform_triangle_set.m - update mesh
  - solve_F_Lambda.m - solve linear system for F_{reg} and λ
* IterativeClosestPoint - Functions for finding closest points in mesh
  - find_closest_point_in_triangle.m
  - linear_search_brute_force.m - find the closest point in mesh by brute force
  - registration.m: 3D point set to 3D point set registration
  - registrationBySVD.m: 3D point set to 3D point set registration by SVD method
  - BoundingSphere&Box
    * radius_center_of_sphere.m
    * linear_search_bounding_spheres.m
    * bound_of_box.m
    * linear_search_bounding_boxes.m
  - Tree
    * OcTree
    * KdTree
    * TreeSearch
* Parse - Matlab functions for parsing data files
* PA5Driver.m - driver script for PA5
* Solver
  - PA5Solver_1.m - Solver script via updating separately
  - PA5Solver_2_Linear.m - Solver script via updating simultaneously by linear search
  - PA5Solver_2_Tree.m - Solver script via updating simultaneously by tree search
* PA234-StudentData - Data for PA5
* PA5Validation.m - Matlab script for validation and error analysis
* PA5OutputData - Output data for different data set
* PA5OutputFig - Output figures for different search methods

Mathematical Approach and Algorithm

This programming assignment contains two parts: iterative closest point algorithm and deformable registration. In short, all the two methods are finding ways to update rigid transformation matrix and update lambda for deformable registration. We have gone through the ICP algorithm for rigid transformation in PA3 and PA4. As a result, here we first introduce how we update lambda for deformable registration, followed by combining two methods.

(a) iterative closest points algorithm

The iterative closest points is based on PA3's one time closest point algorithm, by introducing registration updating process. Also another key point is to choose a proper stopping condition.

The stopping condition are combined by: accuracy, maximum iterations and convergence. The accuracy are measured by the distance between rigid transformed local sample points and the corresponding closest points. The maximum iterations is set default as 100 times. The convergence is measured by the delta term of the update process, if the delta term is almost identity matrix then we consider it convergence.

Algorithm 1 iterative closest point

Freg $\leftarrow I_{4\times4}$ (for initial guess)
while error > 0.1 and norm(ΔFreg - $I_{4\times4}$)>0.01 or iteration<max iteration **do**
  $s \leftarrow$ rigid transformation(d, Freg)
  $c \leftarrow$ find closest points in mesh(s)
  error$\leftarrow |s - c|$
  ΔFreg$\leftarrow$registration(s,c)
  Freg$\leftarrow$ ΔFreg$\cdot$Freg
end while

(b) deformable registration

The process to update lambda for deformable registration contains following steps:

(1) deform the mesh by current deformable registration coefficients
(2) From ICP find the sample points' local coordinates and corresponding closest points
(3) Solve linear system for coefficients of triangle vertexes for each closest point
(4) Solve linear system for updating lambda

First of all, we should deform the triangle set by current deformable registration coefficients λ.

$$m_{=}m_{0,t} + \sum_{i=1}^{N} \lambda_i m_i$$

Then, we look at the process for coefficients of triangle vertexes for each closest point. The ICP algorithm guarantees that the closest point should lies in some triangle so that we can always represent the closest point by some linear combination of the vertexes of the corresponding triangle.

$$c_i = \alpha_i m_t + \beta_i m_u + \gamma_i m_v$$

and,

$$\alpha_i + \beta_i + \gamma_i = 1$$

where m_t, m_u and m_v are the corresponding vertexes. Then we could construct linear system as,

$$\begin{bmatrix} m_{t,x} & m_{u,x} & m_{v,x} \\ m_{t,y} & m_{u,y} & m_{v,y} \\ m_{t,z} & m_{u,z} & m_{v,z} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} c_x \\ c_y \\ c_z \\ 1 \end{bmatrix}$$

Finally we could solve for updating *lambda* by following equations. From the expression of c_i above, we could represent c_i by some combinations of mesh vertexes as,

$$c_i = \alpha_i m_t + \beta_i m_u + \gamma_i m_v$$

then,

$$c_i = q_{0,i} + \sum_{m=1}^{N} q_{m,i}$$

where,

$$q_{m,i} = \alpha_i m_{m,t} + \beta_i m_{m,u} + \gamma_i m_{m,v}$$

Given assumption that $s_i = c_i$, the linear system is like,

$$s_i - q_{0,i} = \sum_{m=1}^{N} \lambda_i q_{m,i}$$

Then we could perform least square method solving for λ.
Combine the two steps listed above, here we present the algorithms for deformable registration.

Algorithm 2 deformable registration

$\lambda \leftarrow \vec{0}$ (as initial guess)
Freg $\leftarrow I_{4\times4}$ (as initial guess)
while norm($\Delta\lambda$)>0.01 and norm(ΔFreg - $I_{4\times4}$)>0.01 or iteration<max iteration **do**
    Update mesh by current λ
    $s \leftarrow$ rigid transformation(d, Freg)
    $c \leftarrow$ find closest points in mesh(s)
    error$\leftarrow |s - c|$
    ΔFreg$\leftarrow$registration(s,c)
    Freg$\leftarrow$ ΔFreg$\cdot$Freg
    Solve linear system for α, β, γ of each closest point
    Solve linear system for updating λ
end while

(c) update rigid and deformable registration coefficients at the same time.
The method list above update rigid registration and deformable registration one by one. We could also update them at the same time by solving a linear system which contains delta term for each part.
Recalling the expression for sample points

$$F_{reg}d = q_0 + \sum_{m=1}^{N} \lambda_m q_m$$

Introducing ΔF_{reg} and $\Delta\lambda$, we have,

$$\Delta F_{reg} F_{reg} d = q_0 + \sum_{m=1}^{N} \lambda_m q_m + \sum_{m=1}^{N} \Delta\lambda_m q_m$$

Given the expression for c as

$$c = q_0 + \sum_{m=1}^{N} \lambda_m q_m$$

we have,

$$\Delta F_{reg} s = c + \sum_{m=1}^{N} \Delta\lambda_m q_m$$

and $\Delta F_{reg} = [R(\alpha), \varepsilon]$, we could give out the final equation as,

$$s \times \alpha - \varepsilon + \sum_{m=1}^{N} \Delta\lambda_m q_m = s - c$$

Then we could present our unknowns as a vector, and the linear system is given as,

$$\begin{bmatrix} \cdots & & & & & & & & \\ 0 & -s_{i,z} & s_{i,y} & -1 & 0 & 0 & q_{x,1} & ... & q_{x,N} \\ s_{i,z} & 0 & -s_{i,x} & 0 & -1 & 0 & q_{y,1} & ... & q_{y,N} \\ -s_{i,y} & s_{i,x} & 0 & 0 & 0 & -1 & q_{z,1} & ... & q_{z,N} \\ \cdots & & & & & & & & \end{bmatrix} \begin{bmatrix} \alpha \\ \varepsilon \\ \lambda \end{bmatrix} = \begin{bmatrix} \cdots \\ s_{i,x} - c_{i,x} \\ s_{i,y} - c_{i,y} \\ s_{i,z} - c_{i,z} \\ \cdots \end{bmatrix}$$

$$A_{3n\times 12} t_{12\times 1} = b_{3n\times 1}$$

From *alpha* to ΔR we should use the proper rotation. Here we present the exponential way to compute the rotation.
Given any α, we have $w = \alpha/||\alpha||$ and $\theta = ||\alpha||$

$$R = e^{\hat{w}\theta} = I + \hat{w} sin\theta + \hat{w}^2(1 - cos\theta)$$

Algorithm 3 deformable registration: another approach

1: $\lambda \leftarrow \vec{0}$ (as initial guess)
2: Freg $\leftarrow I_{4\times4}$ (as initial guess)
3: **while** norm($\Delta\lambda$)>0.01 and norm(ΔFreg - $I_{4\times4}$)>0.01 or iteration<max iteration **do**
4: Update mesh by current λ
5: $s \leftarrow$ rigid transformation(d, Freg)
6: $c \leftarrow$ find closest points in mesh(s)
7: Solve linear system for α, β, γ of each closest point
8: Solve linear system for updating both ΔF_{reg} and $\Delta\lambda$
9: $F_{reg} \leftarrow \Delta F_{reg} F_{reg}$ and $\lambda \leftarrow \Delta\lambda + \lambda$
10: **end while**

Programming Structure
(a) Phase Functions
We implements parseBody.m for reading markers' information, parseMesh.m for mesh information and parseSample.m for markers' local coordinates.

(b) Programming Structure
We implemented both methods separately. And here we briefly sum up the process we performed to implement the method in our driver script.

Algorithm 4 programming assignment 5 driver by both methods

initializing the workspace
add some file path
read markers and mesh information
build the bounding sphere, bounding box, octree or kdtree
for each data set **do**
    read local markers coordinates, registration, compute sample points d
    while deformable registration process **do**
        perform deformable registration process to update F_{reg} and λ
    end while
end for

Besides, we also implemented a real time plot for error term and the changes of λ while executing the PA5Driver script. The plot will clearly show the iterative process of deformable registration and how the process meets its stopping condition, like below. Also we display some essential computational information during the execution of driver script, including elapsed time, registration residual error, iteration experienced, error term, etc.

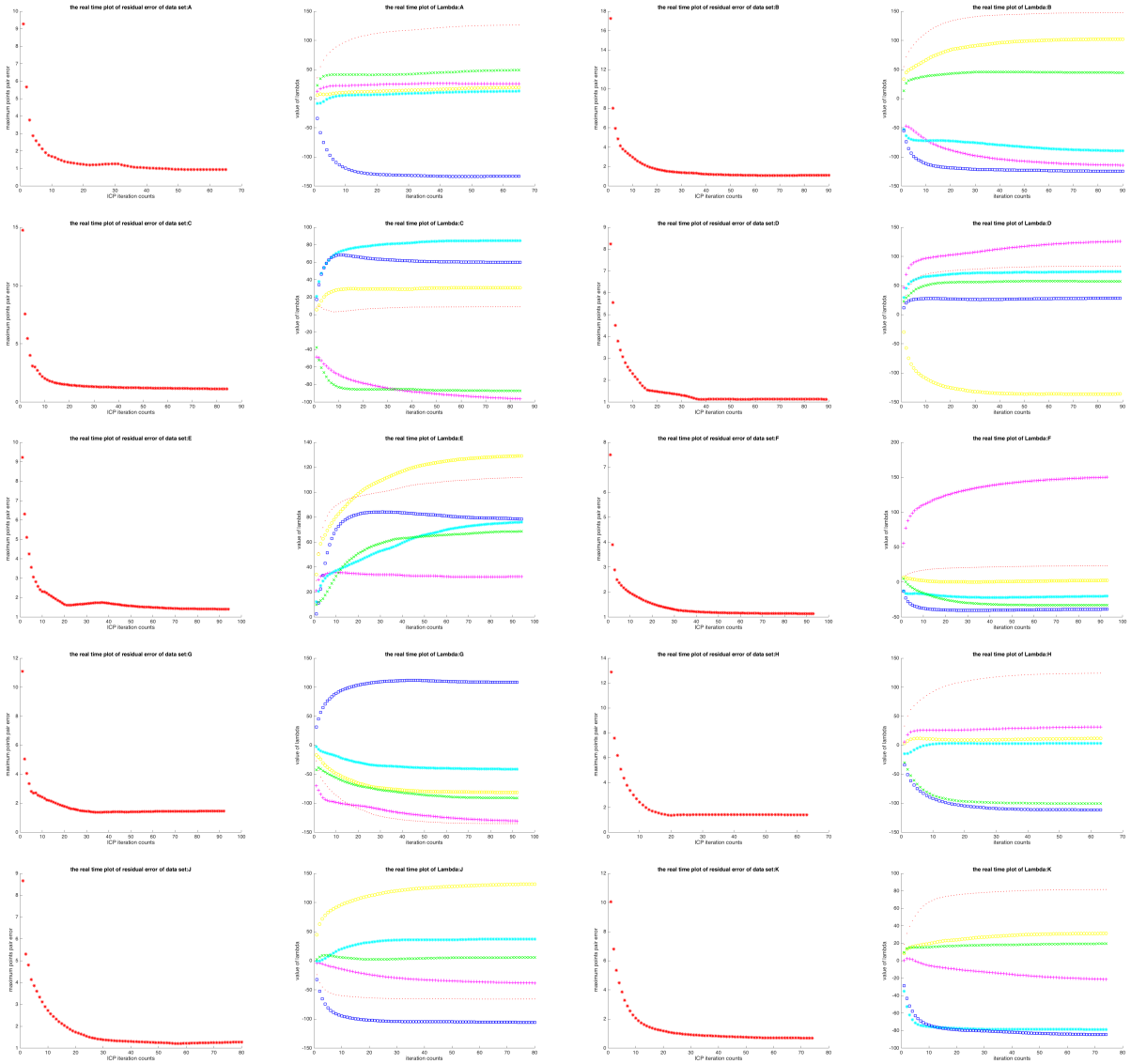

FIGURE 1. Method 1: iterative process for each data set

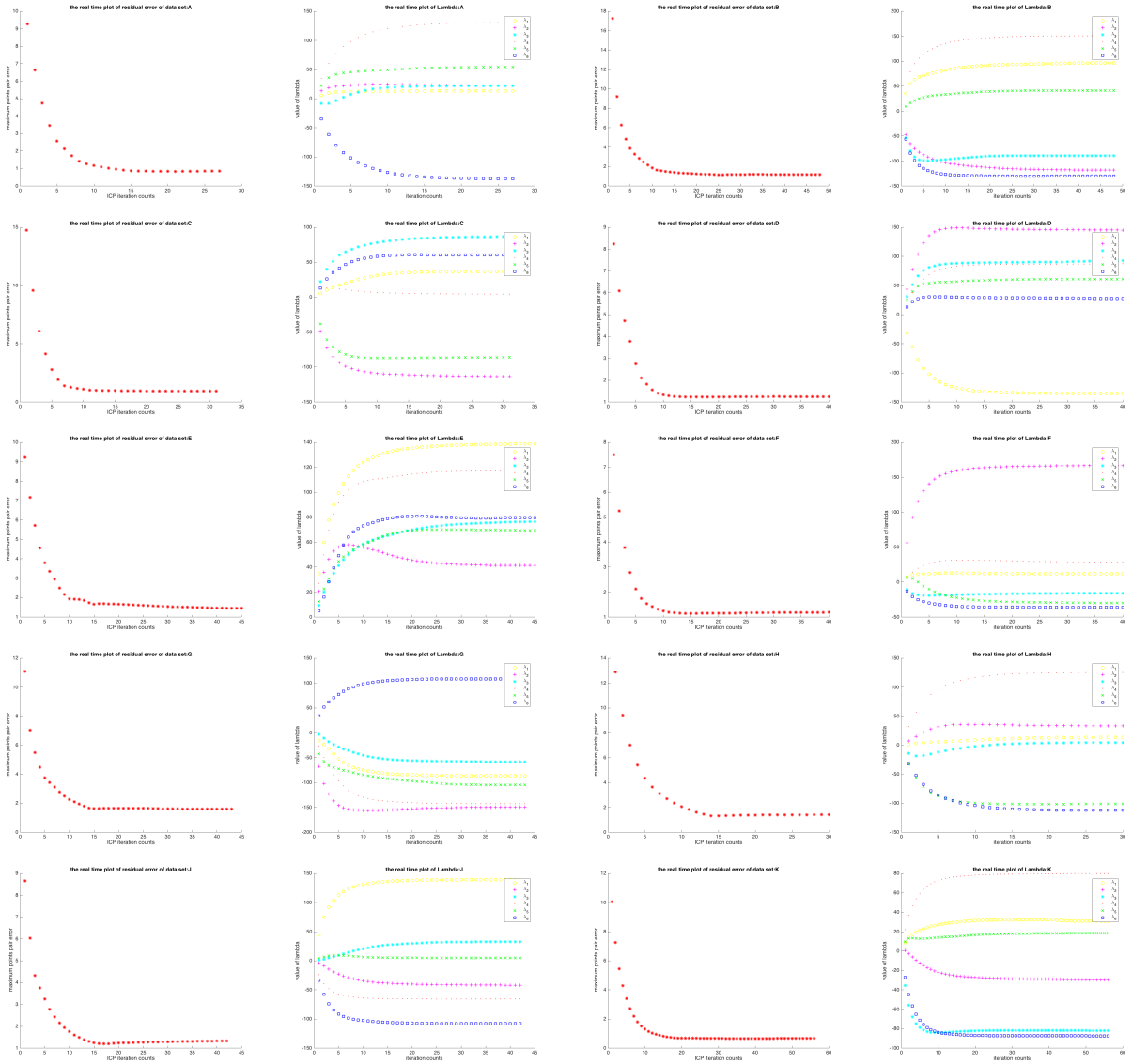

FIGURE 2. Method 2: iterative process for each data set

Validation and Results

We previously validated our ICP methods in PA3 and PA4. In this part, we intend to validate the whole deformable registration process, by comparing our computed results of the sample points' local coordinates, closest points and the distance between them, to the debug result files. More specifically, we compute the norm of each points pair in the same data set.

Here we list how the validation script execute,

LISTING 1. PA5Validation.m

```
%% initialization
clear; clc;
format compact;

%% add the path
addpath('PA234 - Student Data/');
addpath('PA5OutputData/');
addpath('Parse/');

%% validate the algorithm
for index = 1:2
    lambda_diff_norm = zeros(6,1);
    s_diff_max = zeros(6,1);
    c_diff_max = zeros(6,1);
    distance_diff_max = zeros(6,1);
    for char = 'A':'F'
        % read the given debug result file
        validation_set_path = strcat('PA234 - Student Data/PA5-',...
            char, '-Debug-Output.txt');
        [lambda, s_set, c_set, dist_set] = ...
            parseOutput(validation_set_path);

        % read the computed resul file
        computed_set_path = strcat('PA5OutputData/Method-',num2str(index),...
            '/PA5-',char,'-Output-',num2str(index),'.txt');
        [computed_lambda, computed_s_set, computed_c_set, computed_dist_set] = ...
            parseComputedOutput(computed_set_path);

        % get the difference of three terms
        lambda_diff_norm(abs(char)-64) = norm(lambda - computed_lambda);

        s_diff = sum((s_set - computed_s_set).^2, 2).^1/2;
        s_diff_max(abs(char)-64) = max(s_diff);

        c_diff = sum((c_set - computed_c_set).^2, 2).^1/2;
        c_diff_max(abs(char)-64) = max(c_diff);

        distance_diff = dist_set - computed_dist_set;
        distance_diff_max(abs(char)-64) = max(distance_diff);
```

```
        % display the results
        disp(strcat('data set:', char));
        disp(strcat('the difference of Lambda:', ...
            num2str(lambda_diff_norm(abs(char)-64))));
        disp(strcat('the difference of sample points coordinates:', ...
            num2str(s_diff_max(abs(char)-64))));
        disp(strcat('the difference of closest points coordinates:', ...
            num2str(c_diff_max(abs(char)-64))));
        disp(strcat('the difference of closest distance:', ...
            num2str(distance_diff_max(abs(char)-64))));
        disp('------------------------------------------------------------------------');
    end
end
```

The difference of each term are showed as bar plot below,

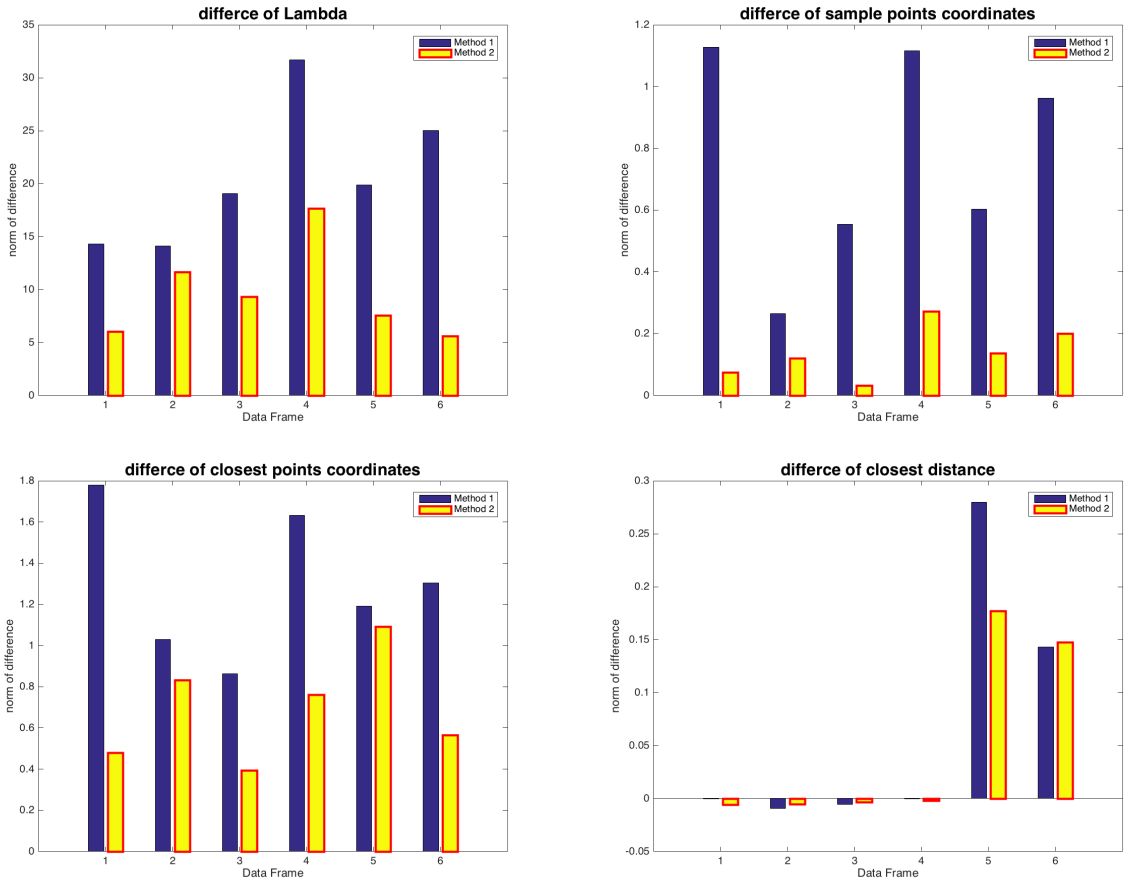

FIGURE 3. norm of difference for each data set

From the plot we can see, the second method (solve ΔF_{reg} and $\Delta\lambda$ at the same time) has more similar result to the given output data file than the first method. However, all the two methods give the acceptable result with difference around 1 mm. Then we believe that the accuracy of those two methods are guaranteed.

Discussion

Besides accuracy, we would like to discuss about the efficiency of those two methods, by comparing both time consuming and iterations taking to converge. Noting that we use the same stopping condition, the same search method for ICP, so that the comparison is effective.

Here we recorded and compared those two methods for deformable registration' time consuming and iterative taking for every data set.

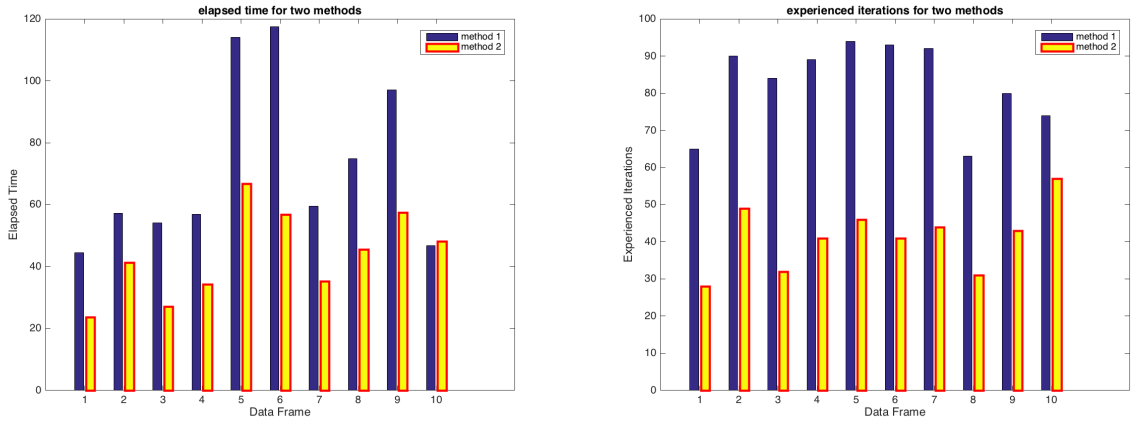

FIGURE 4. Time Consuming and Iteration Taking for Each Method

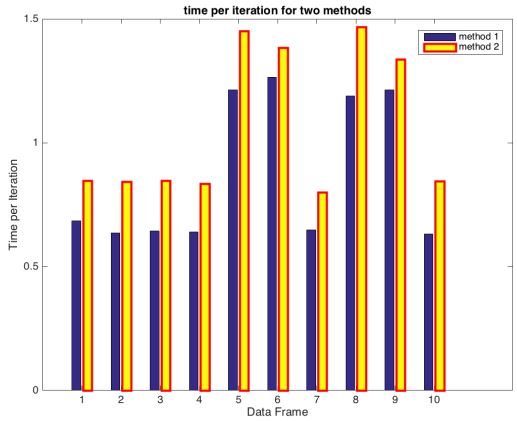

FIGURE 5. Time Consuming and Iteration Taking for Each Method

We could see that, the second method, solving linear system for both part at the same time, strictly dominates the first method at time consuming and iteration taking, though takes more time per iteration than the first method.

Furthermore, we would like to talk about the efficiency of different search methods in deformable registration. Above we use the linear search method by bounding sphere. Though we know that tree search is more efficient than linear search, the deforming of mesh each loop requires a re-computation for the boundary of each node of the tree. As for linear search

by bounding sphere, the only effort is to re-compute the radius and center for each triangle, which is a lot less.
Here we recorded and compared those two search methods' time consuming for every data set.

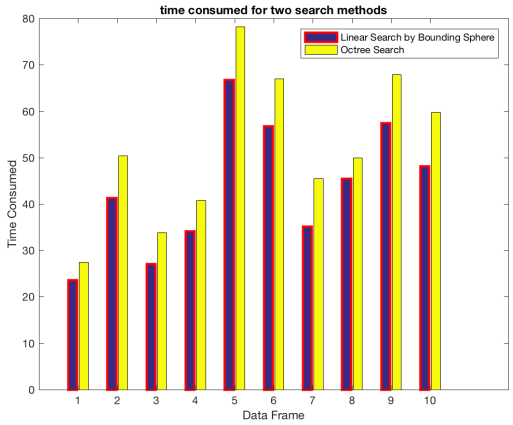

FIGURE 6. Time Consuming for Each Search Method

We could see that, though according to our PA4 result, depth first search of hierarchy data structure strictly dominates linear search methods in rigid transformation, the construction of octree each iteration consumes significant time and performs worse than linear search method by bounding sphere. Still, introducing parallel programming speeds up twice (for your information, my cpu is quad-core).

Summary of Unknown Data

(1) All the result files are in path **\PA5OutputData** with file name as **\PA5-*index*-Output-Method**.
(2) Inside the data file, the first line is the deformable registration coefficients λ. The following result are listed line by line as, x y z coordinates of d_k, x y z coordinates of c_k, and magnitude of difference.
(3) The iterative process for different search methods are in path **\PA5LogFile** with file name as. The figures for the iterative process of different search methods are in **\PA5OutputFig**.