Programming Assignments 1 & 2 (circle one)
600.445/645 Fall 2016

Score Sheet (hand in with report)

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	

CIS PA 2

MENGZE XU, LIUJIANG YAN

Summary
This report is for programming assignment 2 distortion calibration and application, 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,

* PA2 - Essential Matlab functions
  - scale_to_box.m: scale the experimental distorted data and return its maximum and minimum value
  - BernsteinMat.m: return the Bernstein matrix of scaled points set
  - distortion_matrix_compute.m: compute the distortion matrix
  - distortion_correct.m: correct the measured distorted data by distortion matrix
  - correct_em_pivot_calibration.m: corrected version of EM pivot calibration
  - fiducials_in_EM.m: compute the locations of the fiducial points relative to the tracker base frame.
* PA12-StudentData - Data for PA2
* parse - Matlab functions for parsing data files
* pa2output - Output data derived by PA2
* pa2driver.m - driver script for PA2 and leads to pa2output files
* pa2validation.m - Matlab script for validation and error analysis

Mathematical Approach and Algorithm

(a)Compute expected value $C_i^{expected}$ for C_i

Just as what we have done in PA1, to compute the expected value $C_i^{expected}$ for C_i we need to perform composition rule as below,

$$C_i^{expected} = F_{AD}^{-1} F_{AC} c_i$$

In order to solve the expression above, we need to perform registration for transformation from optical tracker to EM tracker and from optical tracker to calibration object, where the transformation stands for,

$$F_{AC} a_i = A_i$$
$$F_{AD} d_i = D_i$$

The Matlab function to compute $C_i^{expected}$ for C_i is shown here,

```
function c_expected = c_expected_value(D, d, A, a, c)
    % perform 3D point set to 3D point set registration
    F_AC = registration(A, a);
    F_AD = registration(D, d);
    % perform composition rule
    F_DC = F_AD\F_AC;
    c(:,4) = 1;
    % write c in homogeneous representation and compute C expected
    c_expected = F_DC*c';
    c_expected = c_expected(1:3,:)';
end
```

(b) Distortion Correction Function

In order to compute the distortion correction function for 3D tracker, let Q stand for experimental distorted data, and P stand for experimental expected data. So the relation between Q and P is,

$$F(Q)C = P$$

where C is the distortion correction function. Here we construct the interpolation matrix by fifth degree Bernstein polynomials, where

$$F_{ijk}(u_x, u_y, u_z) = B_{5,i}(u_x) B_{5,j}(u_y) B_{5,k}(u_z)$$

The function for compute F are shown below,

```
function F = BernsteinMat(q)
    % get the number of experiental distorted data points
    [height, ~] = size(q);
    % predefine the size of Bernstein Matrix
    F = zeros(height, 216);
    % row by row
    for n=1:height
        for i=0:5
            for j=0:5
                for k=0:5
                    h = n;
                    w = i*6^2 + j*6 + k + 1;
```

```
                    % compute Bx By Bz respectively
                    Bx = nchoosek(5,i)*((1-q(n,1))^(5-i))*q(n,1)^i;
                    By = nchoosek(5,j)*((1-q(n,2))^(5-j))*q(n,2)^j;
                    Bz = nchoosek(5,k)*((1-q(n,3))^(5-k))*q(n,3)^k;

                    % assign the product
                    F(h,w) = Bx*By*Bz;
                end
            end
        end
    end
end
```

Bernstein works well when u lies in range [0,1]. So before we compute F and perform least square for C, we need to scale Q in box [0,1]. We record the maximum value as $qmax$ and $qmin$, so that

$$u_{ix} = \frac{Q_{ix} - q_{min,x}}{q_{max,x} - q_{min,x}}$$
$$u_{iy} = \frac{Q_{iy} - q_{min,y}}{q_{max,y} - q_{min,y}}$$
$$u_{iz} = \frac{Q_{iz} - q_{min,z}}{q_{max,z} - q_{min,z}}$$

Following are the scale to box function,

```
function [u, qmax, qmin] = scale_to_box(q)
    % get the maximum and minimum value of each column (x, y, z)
    qmax = max(q);
    qmin = min(q);

    % scale each row into [0,1]
    u = zeros(size(q));
    for i=1:length(q)
        u(i,:) = (q(i,:) - qmin)./(qmax - qmin);
    end
end
```

With given functions above, we are able to construct the linear system for solving distortion matrix,

$$F(u)C = P$$

Here presents the function for distortion matrix computation, with experimental distorted and expected data as input. For a quick note, we write the maximum and minimum of experimental distorted data as one of the output, for future use to correct the distorted data.

```
function [C, qmax, qmin] = distortion_matrix_compute(C_distorted, C_expected)
    % scale the distorted data into [0,1]
    % record the maximum, minimum value for future distortion correction
    [u, qmax, qmin] = scale_to_box(C_distorted);
```

```
    % compute the Berstein Matrix of scaled distorted data
    F = BernsteinMat(u);
    % solve the least sqaure problem and get the distortion matrix
    C = linsolve(F,C_expected);
end
```

(c) Repeat pivot calibration for EM probe with distortion correction.
In order to perform the EM probe pivot calibration, we need to correct the measured distorted data $G^{distorted}$ via distortion matrix derived above.
The expression to correct measured distorted data is,

$$F(u^{distorted})C = G^{expected}$$

where u are the scaled measured data with maximum and minimum value from experimental distorted data.

$$u^{distorted} = \frac{G^{distorted} - q_{min}}{q_{max} - q_{min}}$$

Apply Berstein Matrix computation function as above to get F(u) and then multiply distortion matrix to get the expected data.
The Matlab function to perform distortion correction are shown below,

```
function G_expected = distortion_correct(G, C, qmax, qmin)
    % scale the measured data to [0,1]
    % with experimental distorted data's maximum and minimum value
    G_norm = zeros(size(G));
    for i=1:length(G)
        G_norm(i,:) = (G(i,:)-qmin)./(qmax-qmin);
    end

    % get the Bernstein matrix of normalized distorted data
    F = BernsteinMat(G_norm);
    % apply the distortion correction matrix to get the expected value
    G_expected = F*C;
end
```

Once we get the expected value $G^{expected}$ we could perform the pivot calibration as PA1 did.

```
function [pivot, g_1st_frame] = corrected_em_pivot_calibration...
                    (Ng, Nframes, G, C, qmax, qmin)
    % correct distortion
    G_expected = distortion_correct(G, C, qmax, qmin);
    % pivot calibration
    [pivot, g_1st_frame] = em_pivot_calibration(Ng, Nframes, G_expected);
end
```

(d) Compute the fiducial points' position with respect to the EM tracker base
From the data file we have the position of EM markers in probe with respect to EM tracker base, recorded as G. As above, we need to correct G in the first step.
The position of fiducial points' position with respect to the EM tracker is the same as the pivot value of probe.

From the EM tracker pivot calibration, we record the first data frame which has the EM trackers' position relative to the local coordinate, marked as g. Also, the first term of pivot value is relative to the local coordinate, marked as g_{pivot}.
In order to get the pivot value relative to EM tracker base, we need to perform registration for g and G in order to get,

$$F_k g_i = G_i$$

So that,

$$F_k g_{pivot} = G_{pivot}$$

which is the desired result we are looking for.
The Matlab function to compute pivot value relative to EM tracker base is shown below,

```
function B = fiducials_in_EM...
                    (NG, NB, G, C, pivot, g_1st_frame, qmax, qmin)
    % record the position of fiducial points with respect to EM tracker
    B = zeros(NB,3);

    % correct the distorted point
    G_expected = distortion_correct(G, C, qmax, qmin);

    % get the homogeneous representation of pivot value
    g_pivot = pivot(1:3);
    g_pivot(4) = 1;

    % loop each data frame
    for n=1:NB
        % get the EM markers' position to EM tracker and local frame
        startpoint = (n-1)*NG + 1;
        endpoint = n*NG;
        G = G_expected(startpoint:endpoint,:);
        % compute the transformation
        FG = registration(G, g_1st_frame);

        % get the fiducial point with respect to EM tracker
        B_pivot = FG*g_pivot;
        B(n,:) = B_pivot(1:3)';
    end
end
```

(e) Compute F_{reg}
The registration transformation is to compute the transformation from the position B relative to EM tracker base to the position of fiducials points b relative to CT frame.

$$F_{reg} B = b$$

Here we use the registration function derived in PA1 to perform the registration.

```
function Fform = registration(P, p)
    % 3D point set to 3D point set registration
    % Fa = A -> solve linear system At=b
```

```
    A = [];
    b = [];
    for i=1:length(p)
        subA = [p(i,1) p(i,2) p(i,3) 1 0 0 0 0 0 0 0 0;
                0 0 0 0 p(i,1) p(i,2) p(i,3) 1 0 0 0 0;
                0 0 0 0 0 0 0 0 p(i,1) p(i,2) p(i,3) 1];
        subb = [P(i,1); P(i,2); P(i,3)];
        A = [A; subA];
        b = [b; subb];
    end
    % solve the least square
    h = linsolve(A,b);
    Ffrom = reshape(h,4,3);
    Fform = [Ffrom'; 0 0 0 1];
end
```

(f) Compute the pointer tip with respect to EM tracker base, apply F_{reg} to compute the tip with respect to CT frame.
First of all, we correct the distorted position of points relative to EM tracker frame by distortion matrix derived above.

$$F(u^{distorted})C = G^{expected}$$

where,

$$u^{distorted} = \frac{G^{distorted} - q_{min}}{q_{max} - q_{min}}$$

Just as the method stated to compute the fiducial points' position relative to EM tracker frame, we performed the same method the compute the position of test points relative to EM tracker.

$$F_k g_i = G_i$$

So that the test points' postion can be derived by,

$$F_k g_{pivot} = G_{pivot}$$

Then we apply the transformation matrix F_{reg}, transforming the position relative to EM tracker to CT frame.

$$F_{reg} G_{pivot} = v$$

The Matlab command flow to compute the positions of probe tip in CT frame is shown below,

```
[Ng, Nf, M] = parseEMNAV...
    (strcat('pa2-',type,'-',char,'-EM-nav.txt'));
% get the corrected test point pivot value with respect to EM tracker
V = fiducials_in_EM...
    (Ng, Nf, M, distortion_matrix, pivot, g_1st_frame, qmax, qmin);
V(:,4) = 1;
% transform to CT frame
v = Freg*V';
v = v(1:3,:)';
```

Programming Structure

(a) Phase Functions

Here we show how we phase the data file and get what we want.

Take calbody for example, we return the number of the markers or data frame, as well as the raw data matrix following.

Other functions to parse other data files are almost in the same structure with difference in the return.

```
function [Nd, Na, Nc, M] = parseCALBODY(filepath)
    FID = fopen(filepath);
    file = fgetl(FID);
    % get the first line
    header = textscan(file,'%f%f%f%s','delimiter',',');

    % extract the numerical data
    Nd = header{1,1};
    Na = header{1,2};
    Nc = header{1,3};
    % get the raw data
    M = csvread(filepath, 1, 0);
end
```

(b) Programming Structure

The main script for programming assignment 2 is **pa2driver.m**.

The first part of it is to add some essential paths and clear the workspace.

```
% programming assignment 2 driver
clc
clear
format compact

addpath('PA1');
addpath('PA2');
addpath('PA12-StudentData');
addpath('parse');
```

Then, we loop each debug or unknown data files. We specify the type of files we are reading.

```
for char = 'a':'j'
    % specify the file trype
    if ismember(char, 'a':'f')
        type = 'debug';
    elseif ismember(char, 'g':'j')
        type = 'unknown';
    end

    % the main functions are running inside here
end
```

We read the position of optical markers in EM tracker base relative to EM tracker base stored as d, the position of EM markers in calibration object relative to calibration object base as c and the position of optical markers in calibration object relative to calibration object base as a.

```
% read positions of EM tracker and Optical trakcer
% that describes the calibration object
[Nd, Na, Nc, M] = parseCALBODY...
    (strcat('pa2-',type,'-',char,'-calbody.txt'));
d = M(1:Nd,:);
a = M(Nd+1:Nd+Na,:);
c = M(Nd+Na+1:end,:);
```

Then we read the position of optical markers in EM tracker base relative to optical tracker base as D, the position of optical markers in calibration object relative to optical tracker base as A and the position of EM trackers in calibration object relative to EM tracker base as C.
With given data above, we perform the function shown above to compute the expected value $C_i^{expected}$ for C_i for each data frame.
At last we combine all the data frame's experimental distorted data as C_d and experimental expected data as C_e.

```
% compute the expected value C_expected for C
[ND, NA, NC, NFrames, M] = parseCALREADINGS...
    (strcat('pa2-',type,'-',char,'-calreadings.txt'));
C_e = [];
for i=1:NFrames
    start = (i-1)*(ND+NA+NC)+1;
    D = M(start:start+ND-1,:);
    A = M(start+ND:start+ND+NA-1,:);
    C = M(start+ND+NA:start+ND+NA+NC-1,:);
    C_expected = c_expected_value(D, d, A, a, c);
    C_e = [C_e; C_expected];
end
```

With the derived experimental distorted data and expected data, we could compute the distortion matrix by calling the function introduced above. Along with the distortion matrix, we also store the maximum and minimum value of experimental distorted data for following distortion correction use.

```
% compute the distortion correction matrix: F(q)C=p
% the maximum/minimum value for experimental distorted data
[distortion_matrix, qmax, qmin] = distortion_matrix_compute(C_d, C_e);
```

Then we repeat the pivot calibration process for EM tracker, with distortion correction matrix derived above. Also here we record the first data frame with respect to the local coordinate for following fiducial points' location computation.

```
% get the measured distorted data
[Ng, Nframes, M] = parseEMPIVOT...
    (strcat('pa2-',type,'-',char,'-empivot.txt'));
% apply distortion correction and pivot calibration for pivot value
% the local coordinates of first data frame for following use
[pivot, g_1st_frame] = corrected_em_pivot_calibration...
    (Ng, Nframes, M, distortion_matrix, qmax, qmin);
```

To compute the locations of fiducial points with respect to EM tracker base, first of all, we correct the position of EM markers via distortion correction matrix, which also needs maximum and minimum value of experimental distorted data. Following we perform the points' location computation method derived above.

```
% get the measurement of markers in probe with respect to EM tracker
[NG, NB, G_EM] = parseEMFIDUCIALS...
    (strcat('pa2-',type,'-',char,'-em-fiducialss.txt'));
% compute the fiducials' position with respect to EM tracker
B = fiducials_in_EM...
    (NG, NB, G_EM, distortion_matrix, pivot, g_1st_frame, qmax, qmin);
```

Derive the transformation from EM tracker base to CT image frame, here we perform the registration process by taking the fiduical points' location relative to EM tracker base and relative to CT image frame.

```
% get the measurement of the fiducials' position with respcet to CT
[Nb, b] = parseCTFIDUCIALS...
    (strcat('pa2-',type,'-',char,'-ct-fiducials.txt'));
% get the transformation matrix
Freg = registration(b, B);
```

To compute the locations of test points with respect to EM tracker base, first of all, we correct the position of EM markers. Apply the same method as fiducial points' location computation. Once get the test points' location with respect to EM tracker base, we could apply the transformation matrix F_{reg} derived above to transform the test points' location from EM tracker base to CT image frame.

```
[Ng, Nf, M] = parseEMNAV...
    (strcat('pa2-',type,'-',char,'-EM-nav.txt'));
% get the corrected test point pivot value with respect to EM tracker
V = fiducials_in_EM...
    (Ng, Nf, M, distortion_matrix, pivot, g_1st_frame, qmax, qmin);
V(:,4) = 1;
% transform to CT frame
v = Freg*V';
v = v(1:3,:)';
```

The last part is to output the results in certain form.

```
% output C expected to file
```

```
output1_filename = strcat('pa2output/pa2-',type,'-',char,'-output1.txt');
output1 = zeros(length(C_e)+3,3);
output1(1,1) = NC;
output1(1,2) = NFrames;
output1(4:end,:) = C_e;
% output EM tracker pivot value to file
output1(2,:) = pivot(4:6)';
csvwrite(output1_filename, output1);
% output v to file
output2_filename = strcat('pa2output/pa2-',type,'-',char,'-output2.txt');
output2 = zeros(length(v)+1,3);
output2(1,1) = Nf;
output2(2:end,:) = v;
csvwrite(output2_filename, output2);
```

Validation and Results

With the output result data and given debug output data, we are able to perform some error analysis.

Here, first of all, we present the structure of our validation Matlab script.

```
clc;
clear;
format compact;

addpath('PA1');
addpath('PA2');
addpath('PA12-StudentData');
addpath('parse');

% record the error between computed and output data
testpoint_error = [];
EM_pivot_error = [];
C_expected_error = [];

count = 0;
for char = 'a':'f'
    count = count + 1;

    original_1_filename = strcat('pa2-',type,'-',char,'-output1.txt');
    computed_1_filename = strcat('pa2output/pa2-',type,'-',char,'-output1.txt');

    % get the data
    m_original = csvread(original_1_filename, 1, 0);
    m_computed = csvread(computed_1_filename, 1, 0);

    % store the error
    EM_pivot_error(count,:) = m_original(1,:)-m_computed(1,:);
    C_expected_error(:,:,count) = m_original(3:end,:) - m_computed(3:end,:);

    original_2_filename = strcat('pa2-',type,'-',char,'-output2.txt');
```

```
    computed_2_filename = strcat('pa2output/pa2-',type,'-',char,'-output2.txt');

    % get the data
    pivot_original = csvread(original_2_filename, 1, 0);
    pivot_computed = csvread(computed_2_filename, 1, 0);

    % store the error
    testpoint_error(:,:,count) = pivot_original - pivot_computed;
end
% compute the norm of the error term
% use max() or mean() command to get the statistical data
EM_pivot_error_norm = sum(abs(EM_pivot_error).^2,2).^(1/2);
testpoint_error_norm = sum(abs(testpoint_error).^2,2).^(1/2);
C_expected_error_norm = sum(abs(C_expected_error).^2,2).^(1/2);
```

For the expected value $C_i^{expected}$ for C_i part, we present the mean and maximum error term here.

mean	x	y	z	$\|\|\Delta r\|\|$
a	0	0	0	0
b	0	0	0	0.48
c	0.02	0	0.02	0.42
d	0	0	0	0.02
e	0.16	0.08	0.26	1.78
f	0.16	0.11	0.09	1.65

max	x	y	z	$\|\|\Delta r\|\|$
a	0.01	0.01	0.01	0.01
b	0.51	0.51	0.50	0.84
c	1.48	1.41	1.16	1.75
d	0.03	0.04	0.04	0.05
e	4.04	5.03	3.93	5.89
f	5.17	4.10	6.32	6.65

For EM tracker pivot calibration part, we present the error term here.

EM	x	y	z	$\|\|\Delta r\|\|$
a	0	0	0.01	0.01
b	-0.01	-0.19	0.09	0.2
c	0.04	-0.08	-0.06	0.1
d	-0.01	0	0	0.01
e	-0.37	0.41	0.08	0.55
f	0.30	0.08	-0.22	0.38

For the positions of probe tip in CT image frame, the error between the computed results and the given output data are displayed below.

mean	x	y	z	$\|\|\Delta r\|\|$
a	0	0	0	0
b	0.18	0.02	0.14	0.44
c	0	0	0.04	0.05
d	0	0	0.01	0.01
e	0.55	0.28	-0.17	0.71
f	0.05	0.16	0.08	0.4

max	x	y	z	$\|\|\Delta r\|\|$
a	0	0.01	0.01	0.01
b	0.49	0.29	0.49	0.6
c	0.05	0.02	0.05	0.07
d	0	0	0.01	0.01
e	0.81	0.43	0.54	0.90
f	0.41	0.53	0.28	0.55

Discussion

(a) registration As what we performed in the programming assignment 1, we use linsolve() function in Matlab solving for the transformation matrix, the accuracy of which is dependent of the residual. Also, if we have more data points we will have a more accurate transformation.

(b) the expected value $C_i^{expected}$ for C_i

As we derived in PA1, the optical tracker jiggle term has no influence in the process for expected value computed.

$$\begin{aligned}(C_i^{expected})^* &= R_{AD}^T(R_{AC}c_i + p_{AC} + \Delta p_{AC} - p_{AD} - \Delta p_{AD}) \\ &= C_i^{expected} + R_{AD}^T(\Delta p_{AC} - \Delta p_{AD}) \\ &= C_i^{expected}\end{aligned}$$

And according to the error term between what we have computed and the output file data, the EM tracker's distortion and noise have affect on the accuracy.

(c) pivot calibration with distortion correction matrix

Since we apply the distortion correction matrix to correct the measured value, the error term of pivot calibration and probe tips' location between our computed results and output data is below 1mm, which means that our mathematical approach as well as implementation eliminate the EM tracker's distortion term up to several millimeter but there is still noise term about less than 1 mm deviation.

Summary of Unknown Data

EM Pivot Calibration with distortion correction matrix

EM	x	y	z
g	195.45	204.47	198.67
h	195.42	197.53	190.03
i	192.11	205.39	190.16
j	193.26	198.36	198.5

Tip location relative to CT image frame

g	x	y	z
	114.87	29.688	55.717
	93.058	172.64	53.388
	85.718	126.49	89.492
	78.875	85.515	113.27

h	x	y	z
	156.33	128.7	73.442
	127.06	112.82	60.887
	141.17	161.89	82.938
	162.2	162.75	171.57

i	x	y	z
	79.947	30.482	74.119
	82.04	32.699	30.564
	106.18	118.11	29.749
	133.61	170.55	168.24

j	x	y	z
	39.001	71.772	94.687
	26.545	118.88	155.39
	101.07	148.25	28.316
	168.64	43.239	142

Expected value for C_i

Due to the huge amount of data, please refer to the result output files for unknown data.