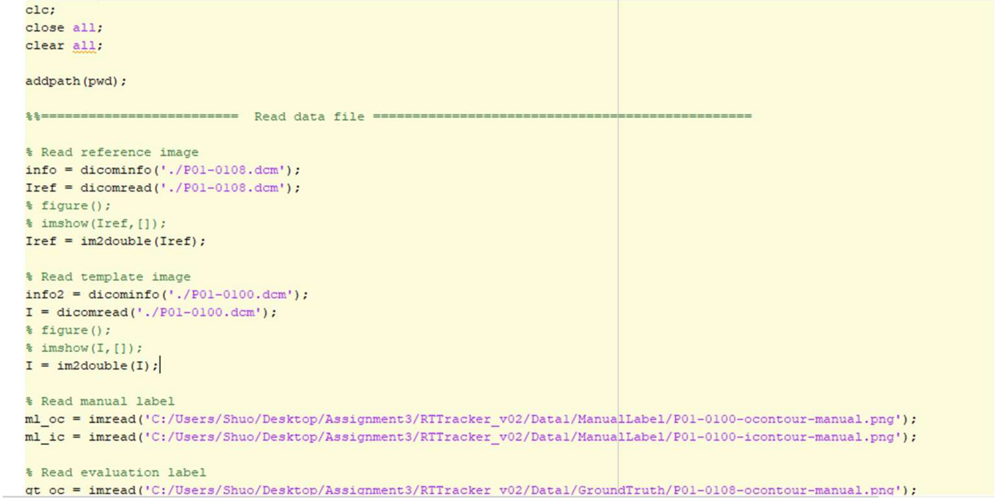
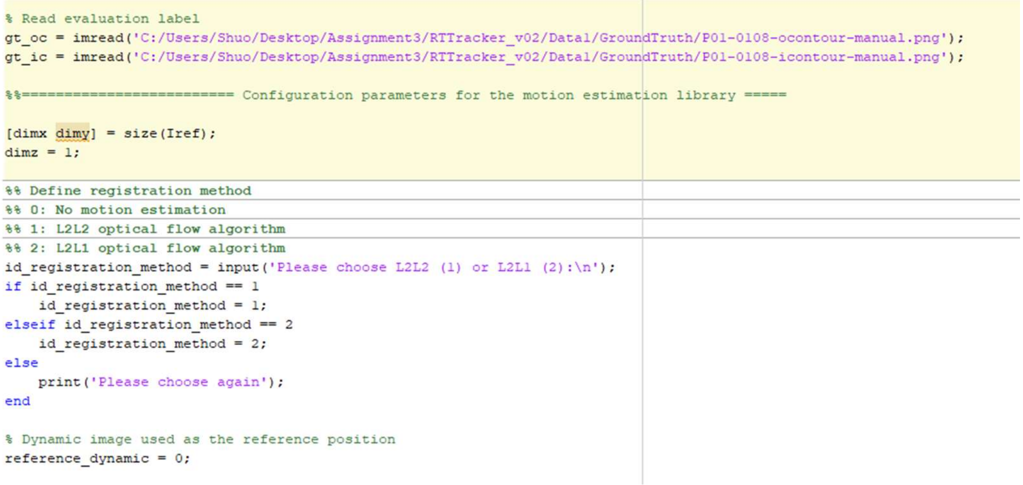
Assignment 3

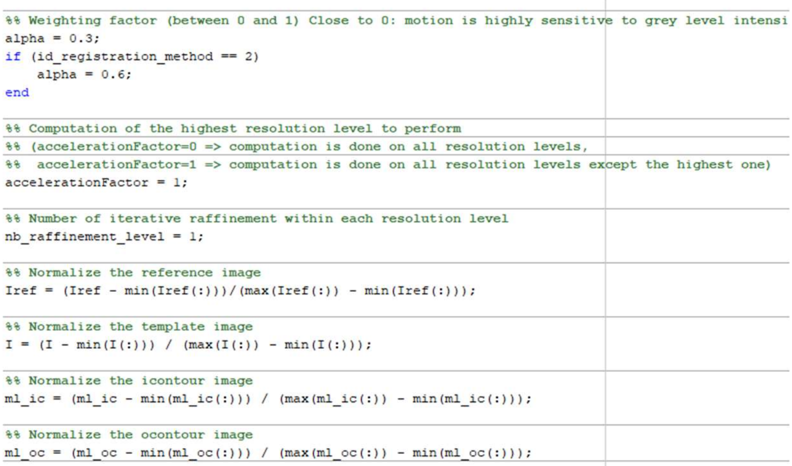
Reference image and template image



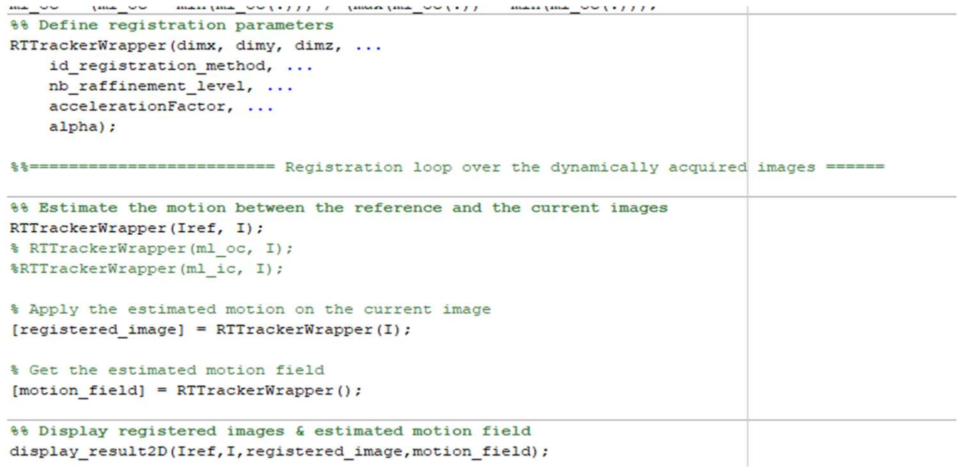
```
clc;
close all;
clear all;

addpath(pwd);

%%=========================  Read data file =============================================

% Read reference image
info = dicominfo('./P01-0108.dcm');
Iref = dicomread('./P01-0108.dcm');
% figure();
% imshow(Iref,[]);
Iref = im2double(Iref);

% Read template image
info2 = dicominfo('./P01-0100.dcm');
I = dicomread('./P01-0100.dcm');
% figure();
% imshow(I,[]);
I = im2double(I);

% Read manual label
ml_oc = imread('C:/Users/Shuo/Desktop/Assignment3/RTTracker_v02/Data1/ManualLabel/P01-0100-ocontour-manual.png');
ml_ic = imread('C:/Users/Shuo/Desktop/Assignment3/RTTracker_v02/Data1/ManualLabel/P01-0100-icontour-manual.png');

% Read evaluation label
gt_oc = imread('C:/Users/Shuo/Desktop/Assignment3/RTTracker_v02/Data1/GroundTruth/P01-0108-ocontour-manual.png');
```



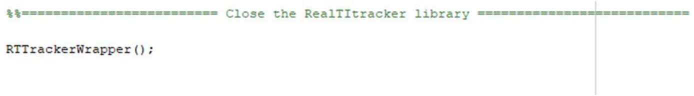
```
% Read evaluation label
gt_oc = imread('C:/Users/Shuo/Desktop/Assignment3/RTTracker_v02/Data1/GroundTruth/P01-0108-ocontour-manual.png');
gt_ic = imread('C:/Users/Shuo/Desktop/Assignment3/RTTracker_v02/Data1/GroundTruth/P01-0108-icontour-manual.png');

%%========================== Configuration parameters for the motion estimation library =====

[dimx dimy] = size(Iref);
dimz = 1;

%% Define registration method
%% 0: No motion estimation
%% 1: L2L2 optical flow algorithm
%% 2: L2L1 optical flow algorithm
id_registration_method = input('Please choose L2L2 (1) or L2L1 (2):\n');
if id_registration_method == 1
    id_registration_method = 1;
elseif id_registration_method == 2
    id_registration_method = 2;
else
    print('Please choose again');
end

% Dynamic image used as the reference position
reference_dynamic = 0;
```



```
%% Weighting factor (between 0 and 1) Close to 0: motion is highly sensitive to grey level intensi
alpha = 0.3;
if (id_registration_method == 2)
    alpha = 0.6;
end

%% Computation of the highest resolution level to perform
%% (accelerationFactor=0 => computation is done on all resolution levels,
%%  accelerationFactor=1 => computation is done on all resolution levels except the highest one)
accelerationFactor = 1;

%% Number of iterative raffinement within each resolution level
nb_raffinement_level = 1;

%% Normalize the reference image
Iref = (Iref - min(Iref(:)))/(max(Iref(:)) - min(Iref(:)));

%% Normalize the template image
I = (I - min(I(:))) / (max(I(:)) - min(I(:)));

%% Normalize the icontour image
ml_ic = (ml_ic - min(ml_ic(:))) / (max(ml_ic(:)) - min(ml_ic(:)));

%% Normalize the ocontour image
ml_oc = (ml_oc - min(ml_oc(:))) / (max(ml_oc(:)) - min(ml_oc(:)));
```

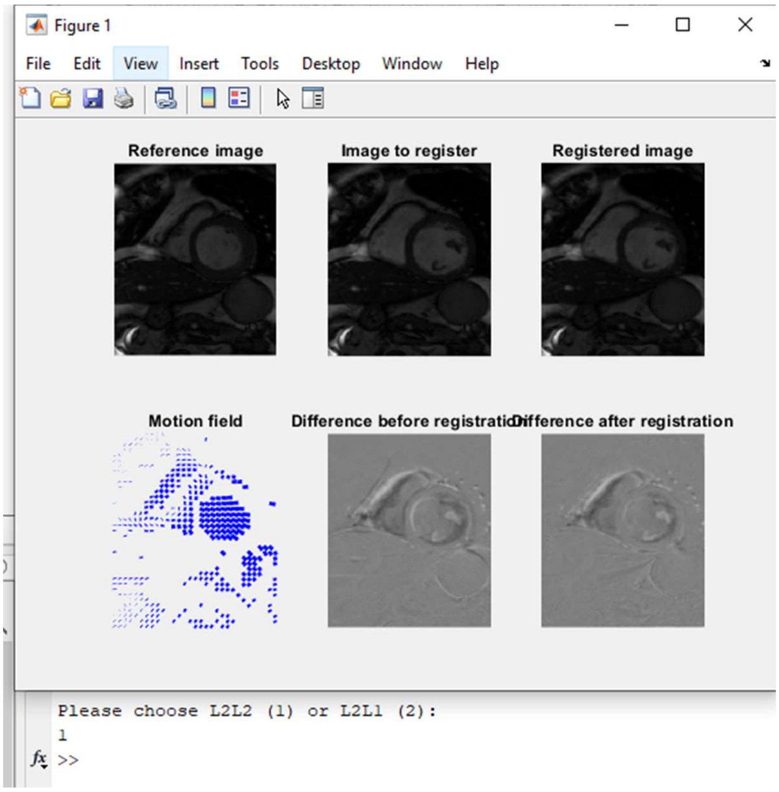


```
ml_oc = (ml_oc - min(ml_oc(:))) / (max(ml_oc(:)) - min(ml_oc(:)));
%% Define registration parameters
RTTrackerWrapper(dimx, dimy, dimz, ...
    id_registration_method, ...
    nb_raffinement_level, ...
    accelerationFactor, ...
    alpha);

%%========================== Registration loop over the dynamically acquired images ======

%% Estimate the motion between the reference and the current images
RTTrackerWrapper(Iref, I);
% RTTrackerWrapper(ml_oc, I);
%RTTrackerWrapper(ml_ic, I);

% Apply the estimated motion on the current image
[registered_image] = RTTrackerWrapper(I);

% Get the estimated motion field
[motion_field] = RTTrackerWrapper();

%% Display registered images & estimated motion field
display_result2D(Iref,I,registered_image,motion_field);
```



```
%%======================== Close the RealTItracker library ==========================

RTTrackerWrapper();
```

L2L2:



```
Please choose L2L2 (1) or L2L1 (2):
1
>>
```

L2L1:

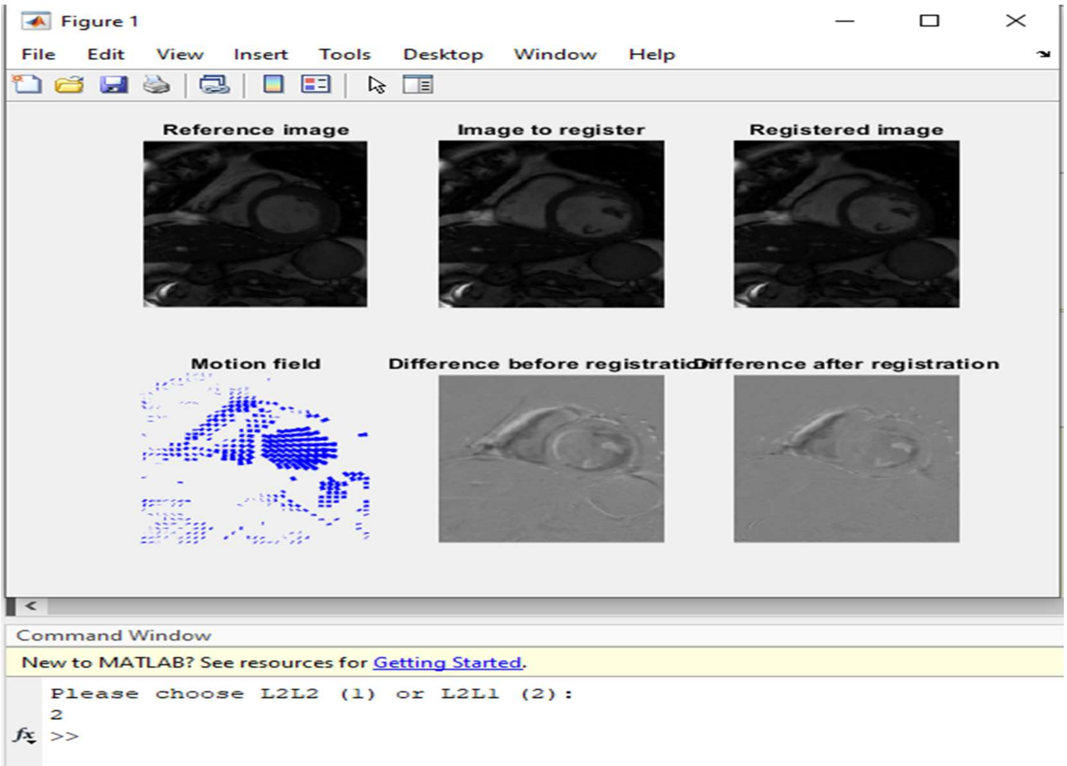
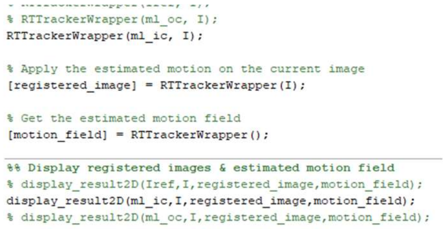


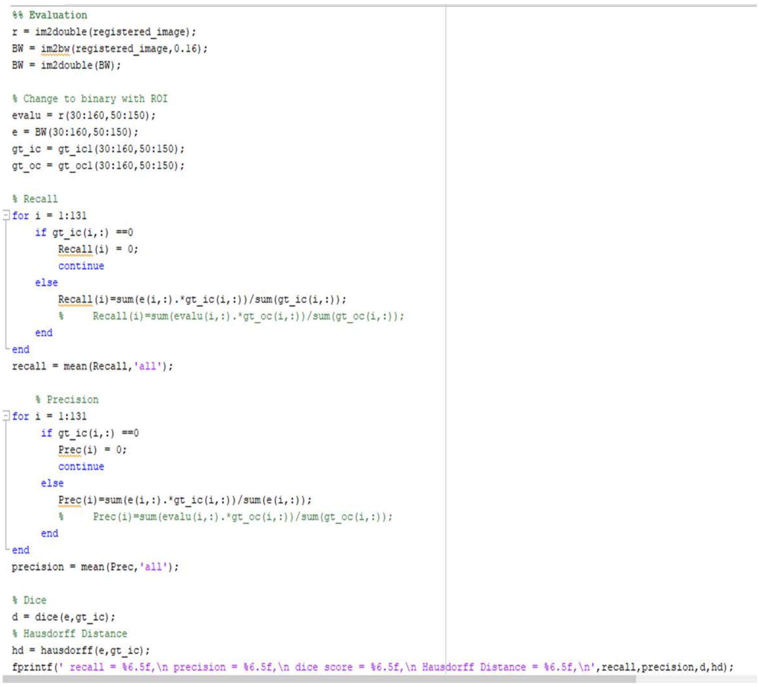
Figure 1
File Edit View Insert Tools Desktop Window Help
Reference image
Image to register
Registered image
Motion field
Difference before registratioDifference after registration
Command Window
New to MATLAB? See resources for Getting Started.

```
Please choose L2L2 (1) or L2L1 (2):
2
>>
```

Icontour L2L2:



```
% RTTrackerWrapper(ml_oc, I);
RTTrackerWrapper(ml_ic, I);

% Apply the estimated motion on the current image
[registered_image] = RTTrackerWrapper(I);

% Get the estimated motion field
[motion_field] = RTTrackerWrapper();

%% Display registered images & estimated motion field
% display_result2D(Iref,I,registered_image,motion_field);
display_result2D(ml_ic,I,registered_image,motion_field);
% display_result2D(ml_oc,I,registered_image,motion_field);
```

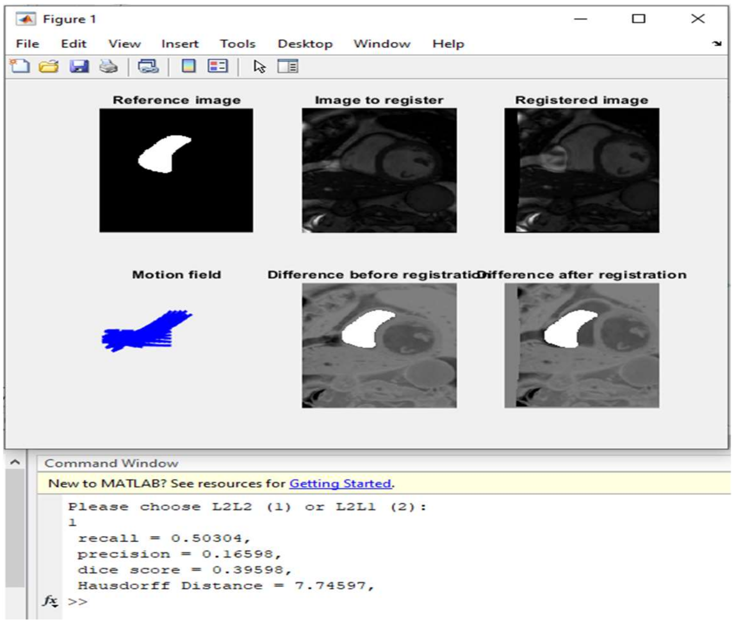


```
%% Evaluation
r = im2double(registered_image);
BW = im2bw(registered_image,0.16);
BW = im2double(BW);

% Change to binary with ROI
evalu = r(30:160,50:150);
e = BW(30:160,50:150);
gt_ic = gt_ic1(30:160,50:150);
gt_oc = gt_oc1(30:160,50:150);

% Recall
for i = 1:131
    if gt_ic(i,:) ==0
        Recall(i) = 0;
        continue
    else
        Recall(i)=sum(e(i,:).*gt_ic(i,:))/sum(gt_ic(i,:));
        %     Recall(i)=sum(evalu(i,:).*gt_oc(i,:))/sum(gt_oc(i,:));
    end
end
recall = mean(Recall,'all');

    % Precision
for i = 1:131
     if gt_ic(i,:) ==0
        Prec(i) = 0;
        continue
     else
        Prec(i)=sum(e(i,:).*gt_ic(i,:))/sum(e(i,:));
        %     Prec(i)=sum(evalu(i,:).*gt_oc(i,:))/sum(gt_oc(i,:));
     end
end
precision = mean(Prec,'all');

% Dice
d = dice(e,gt_ic);
% Hausdorff Distance
hd = hausdorff(e,gt_ic);
fprintf(' recall = %6.5f,\n precision = %6.5f,\n dice score = %6.5f,\n Hausdorff Distance = %6.5f,\n',recall,precision,d,hd);
```

```
function [D,idx] = hausdorff(A,B,method)
```

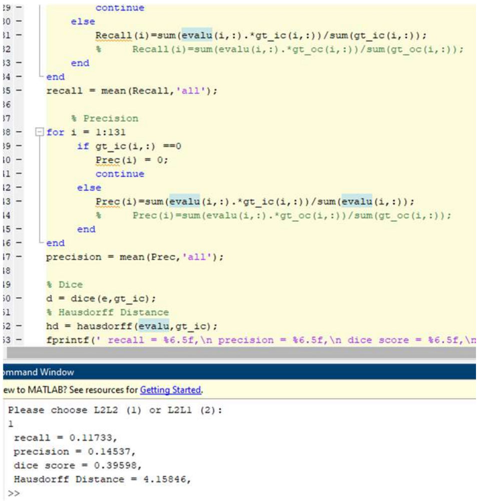
```
%HAUSDORFF distance for point clouds.
%
%function [D,IDX]=hausdorff(a,b,method='euclidean')
%
%   D = HAUSDORFF(A,B) computes the Hausdorff distance between
%   point sets A and B. Rows of A and B correspond to observations,
%   and columns correspond to variables. A and B must have same number
%   of columns.
%
%   D = HAUSDORFF(A,B,METHOD) lets you compute the Hausdorff distance
%   with an alternate point-to-point distance.  METHOD can be any
%   method supported by PDIST2.  METHOD defaults to 'euclidean' if not
%   specified.
%
%   D = HAUSDORFF(A,B,DISTFUN) lets you compute the Hausdorff distance
%   with a distance function specified using a function handle @
%
%   [D,IDX] = HAUSDORFF(...) also returns the indices of the farthest
%   points contributing to the distance.
%
%   Notes
%   -----
%   HAUSDORFF uses PDIST2 for computation. For gridded image data
%   it is often preferred to use IMHAUSDORFF.
%
%   HAUSDORFF(A,[]) = inf
%   HAUSDORFF([],[]) = 0
%
%   Example 1
%   ---------
%   Compute the Hausdorff distance for a binary segmentation.
%   This is much slower than using IMHAUSDORFF.
%
%     % Read in an image with an object we wish to segment.
%     A = imread('hands1.jpg');
%
%     % Convert the image to grayscale.
%     I = rgb2gray(A);
%
%     % Use active contours to segment the hand.
%     mask = false(size(I));
%     mask(25:end-25,25:end-25) = true;
%     BW = activecontour(I, mask, 300);
%
%     % Read in the ground truth against which to compare the segmentation.
%    BW_groundTruth = imread('hands1-mask.png');
%
%    % Extract object point coordinates
%    A=regionprops(BW,'PixelList');
%    B=regionprops(BW_groundTruth,'PixelList');
%
%    % Compute the Hausdorff distance of this segmentation.
%    [distance,idx] = hausdorff(A.PixelList,B.PixelList);
%
%    % Display both masks on top of one another.
%    figure
%    imshowpair(BW, BW_groundTruth)
%    title(['Hausdorff distance = ' num2str(distance)])
%
%    % Display a line indicating the farthest points
%    p=[A.PixelList(idx(1),:);B.PixelList(idx(2),:)];
%    hold on;
%    plot(p(:,1),p(:,2),'rx-','linewidth',2');
%    hold off
%
%   See also IMHAUSDORFF, PDIST2.
%
%Author: Joakim Lindblad
% Copyright (c) 2019, Joakim Lindblad
if size(A,2) ~= size(B,2)
   error('A and B must have the same number of columns.');
end
if nargin < 3
   method = 'euclidean';
else
   if strcmp(method,'chessboard')
      method = 'chebychev'; % synonymous
   end
end
if isempty(A) || isempty(B)
   if isempty(A) && isempty(B)
      HD=0;
   else
      HD=inf;
   end
   return
end
if strcmp(method,'euclidean')
   method='squaredeuclidean'; % faster
   apply_root=true;
```

```
else
    apply_root=false;
end
% Max of dist from A to B and B to A
if (size(A,1)*size(B,1) < 1e8)
    D = pdist2(A,B,method);
    [D1,idxA1] = min(D,[],1);
    [D2,idxB1] = min(D,[],2);
    clear D;
else
    % Less memory hungry version
    [D1,idxA1] = pdist2(A,B,method,'Smallest',1);
    [D2,idxB1] = pdist2(B,A,method,'Smallest',1);
end
[D1,idxB2]=max(D1);
[D2,idxA2]=max(D2);
if (D1>D2)
    D=D1;
    idx=[idxA1(idxB2),idxB2];
else
    D=D2;
    idx=[idxA2,idxB1(idxA2)];
end
if apply_root
    D = sqrt(D);
end
if (nargout < 2)
    clear idx;
end
```



```
Please choose L2L2 (1) or L2L1 (2):
1
 recall = 0.50304,
 precision = 0.16598,
 dice score = 0.39598,
 Hausdorff Distance = 7.74597,
>>
```

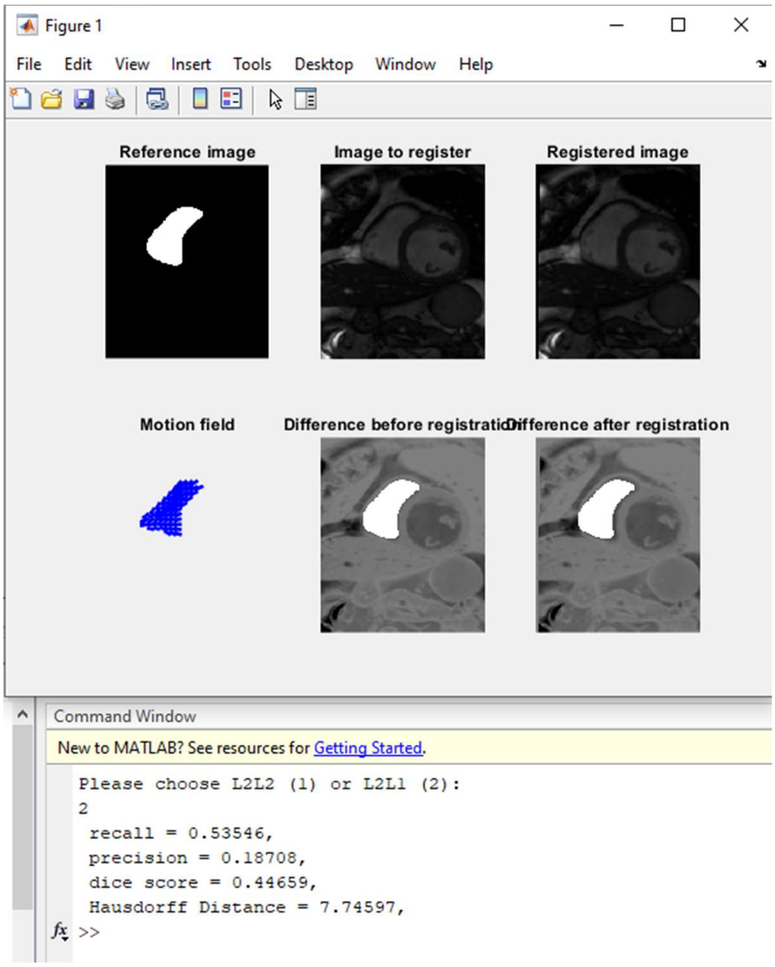
Without binary image converting:



```
            continue
        else
            Recall(i)=sum(evalu(i,:).*gt_ic(i,:))/sum(gt_ic(i,:));
            %      Recall(i)=sum(evalu(i,:).*gt_oc(i,:))/sum(gt_oc(i,:));
        end
end
recall = mean(Recall,'all');

    % Precision
for i = 1:131
     if gt_ic(i,:) ==0
        Prec(i) = 0;
        continue
     else
        Prec(i)=sum(evalu(i,:).*gt_ic(i,:))/sum(evalu(i,:));
        %      Prec(i)=sum(evalu(i,:).*gt_oc(i,:))/sum(gt_oc(i,:));
     end
end
precision = mean(Prec,'all');

% Dice
d = dice(e,gt_ic);
% Hausdorff Distance
hd = hausdorff(evalu,gt_ic);
fprintf(' recall = %6.5f,\n precision = %6.5f,\n dice score = %6.5f,\n
```

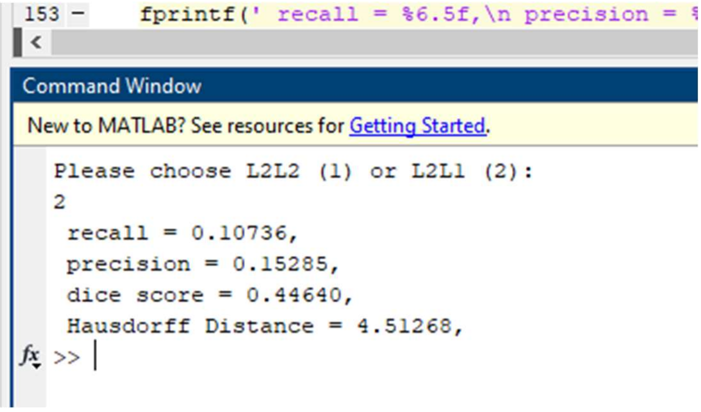
```
Please choose L2L2 (1) or L2L1 (2):
1
 recall = 0.11733,
 precision = 0.14537,
 dice score = 0.39598,
 Hausdorff Distance = 4.15846,
>>
```

Icontour L2L1:



```
Please choose L2L2 (1) or L2L1 (2):
2
 recall = 0.53546,
 precision = 0.18708,
 dice score = 0.44659,
 Hausdorff Distance = 7.74597,
>>
```

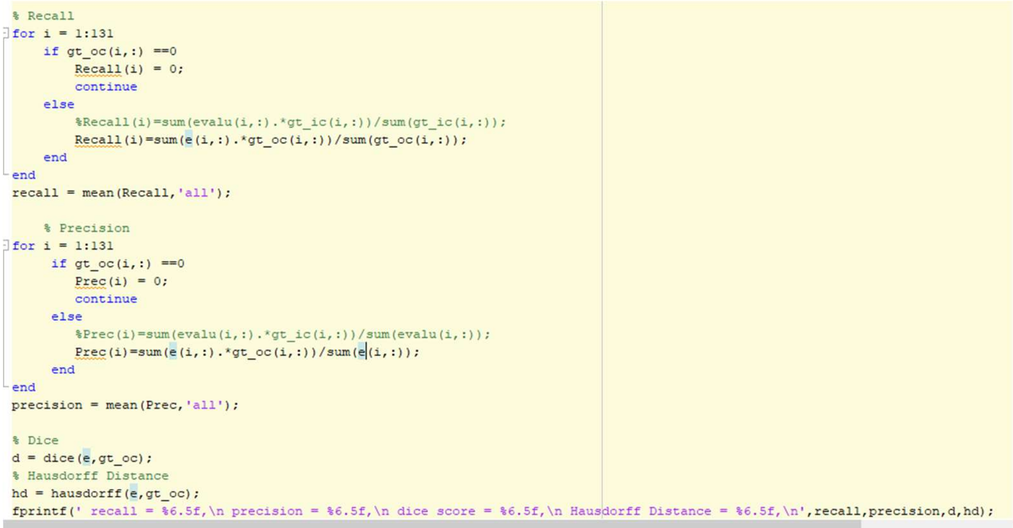
Without binary image converting:



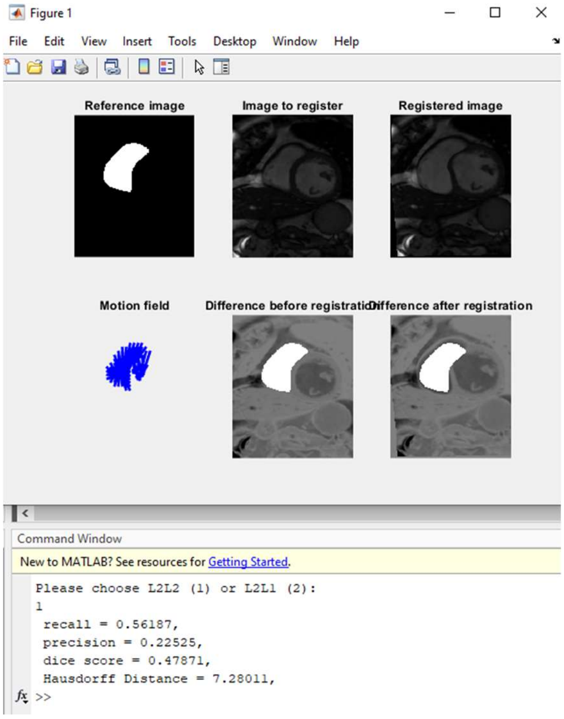
```
    fprintf(' recall = %6.5f,\n precision = %
```

```
Please choose L2L2 (1) or L2L1 (2):
2
 recall = 0.10736,
 precision = 0.15285,
 dice score = 0.44640,
 Hausdorff Distance = 4.51268,
>>
```

Ocontour L2L2:

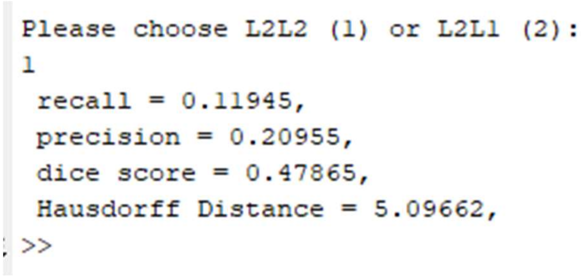


```
% Recall
for i = 1:131
    if gt_oc(i,:) ==0
        Recall(i) = 0;
        continue
    else
        %Recall(i)=sum(evalu(i,:).*gt_ic(i,:))/sum(gt_ic(i,:));
        Recall(i)=sum(e(i,:).*gt_oc(i,:))/sum(gt_oc(i,:));
    end
end
recall = mean(Recall,'all');

    % Precision
for i = 1:131
     if gt_oc(i,:) ==0
        Prec(i) = 0;
        continue
     else
        %Prec(i)=sum(evalu(i,:).*gt_ic(i,:))/sum(evalu(i,:));
        Prec(i)=sum(e(i,:).*gt_oc(i,:))/sum(e(i,:));
     end
end
precision = mean(Prec,'all');

% Dice
d = dice(e,gt_oc);
% Hausdorff Distance
hd = hausdorff(e,gt_oc);
fprintf(' recall = %6.5f,\n precision = %6.5f,\n dice score = %6.5f,\n Hausdorff Distance = %6.5f,\n',recall,precision,d,hd);
```



```
Please choose L2L2 (1) or L2L1 (2):
1
 recall = 0.56187,
 precision = 0.22525,
 dice score = 0.47871,
 Hausdorff Distance = 7.28011,
>>
```

Without binary image convert:



```
Please choose L2L2 (1) or L2L1 (2):
1
 recall = 0.11945,
 precision = 0.20955,
 dice score = 0.47865,
 Hausdorff Distance = 5.09662,
>>
```

Ocontour L2L1:

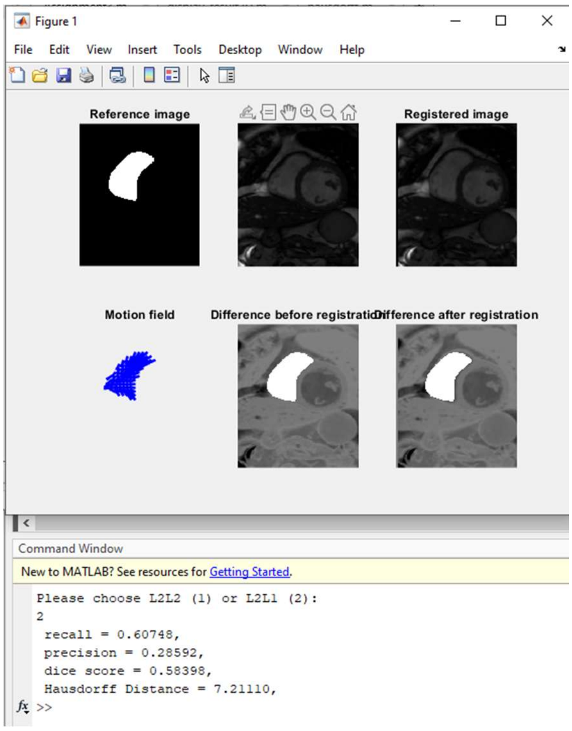
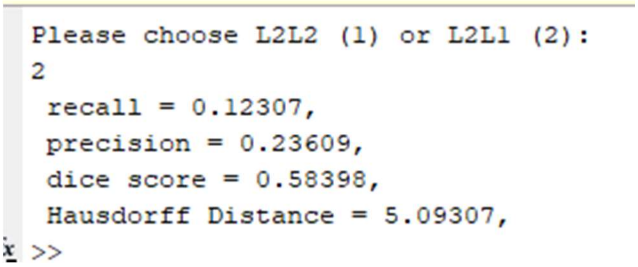


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Reference image
Registered image
Motion field
Difference before registratioDifference after registration
Command Window
New to MATLAB? See resources for Getting Started.

```
Please choose L2L2 (1) or L2L1 (2):
2
 recall = 0.60748,
 precision = 0.28592,
 dice score = 0.58398,
 Hausdorff Distance = 7.21110,
fx >>
```

Without binary image conver:



```
Please choose L2L2 (1) or L2L1 (2):
2
 recall = 0.12307,
 precision = 0.23609,
 dice score = 0.58398,
 Hausdorff Distance = 5.09307,
x >>
```