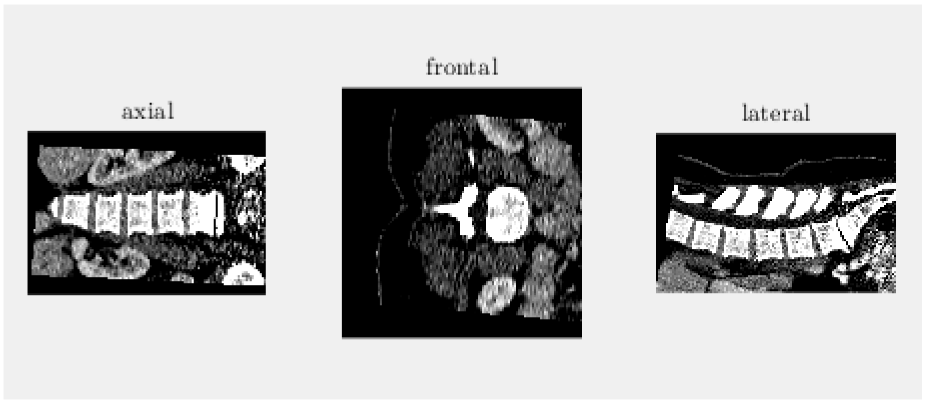
In The Name OF God

Medical Image Analysis and Processing Project Report

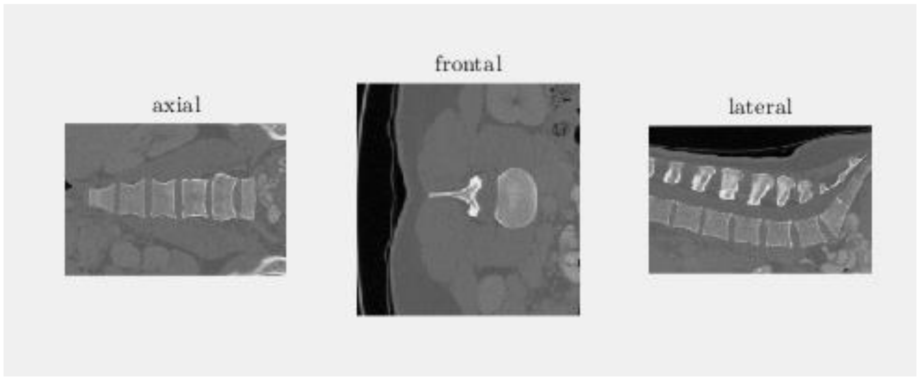
Negin Esmaeilzade 97104034

PART 1: Reading the Data and Preparing the Images

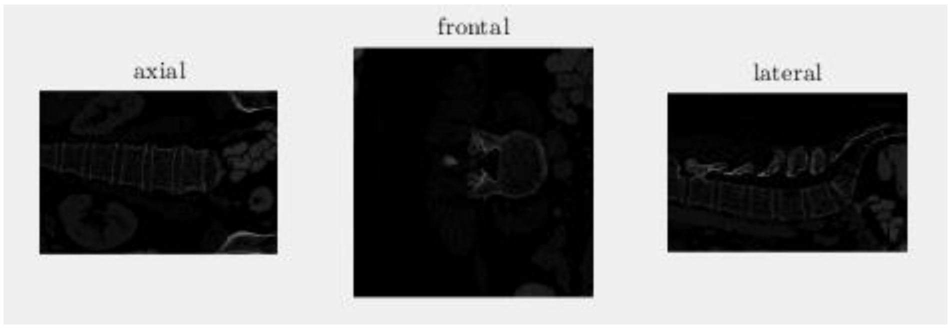
1.A) Images for 3 cases (healthy mask,pat2, pat10, pat21) are shown below:



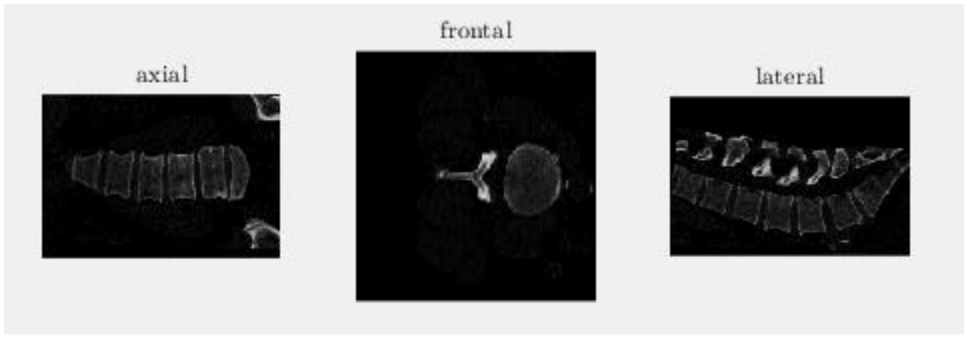
axial, lateral and frontal look at the healthy mask



axial, lateral and frontal look at the patient number 2 image



axial, lateral and frontal look at the patient number 10 image

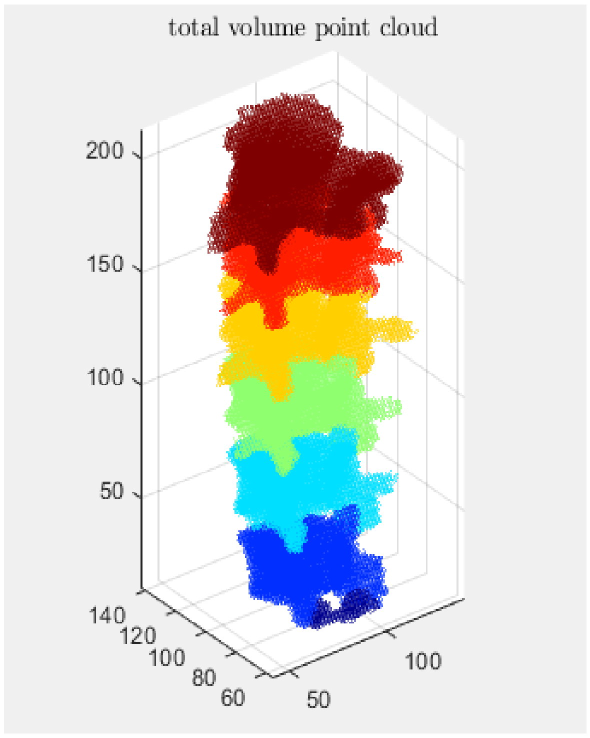


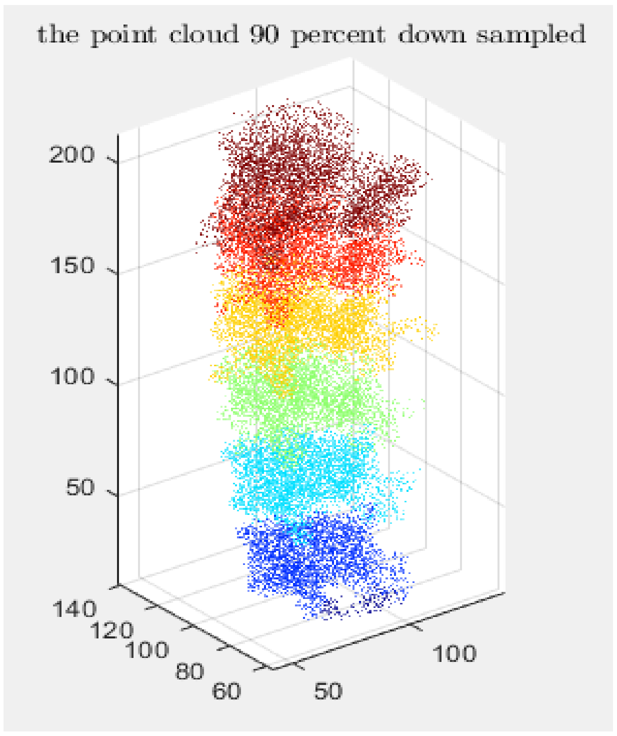
axial, lateral and frontal look at the patient number 21 image

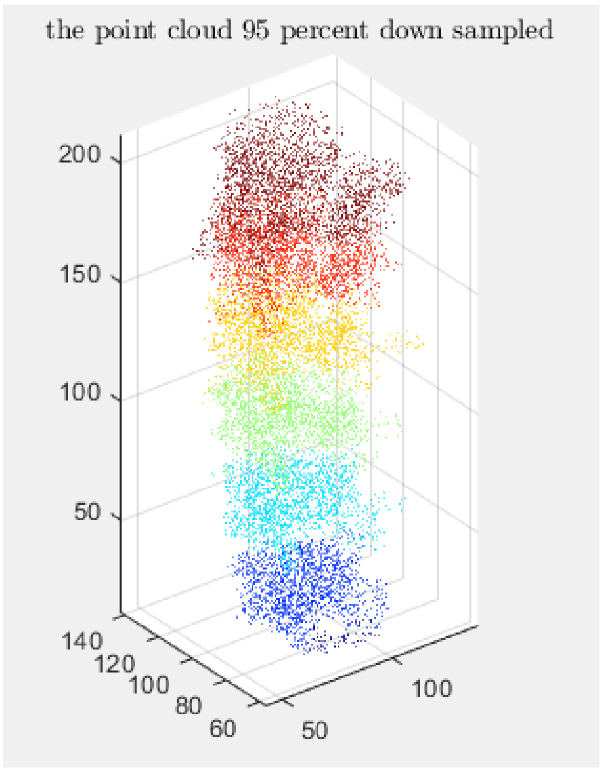
The volumetric images and labels can be better shown (with colorful labeled vertebras) by the Volume Viewer Matlab toolbax , through APPS/VolumeViewer and importing the background image and the labeled ones. But, unfortunately my Matlab version is 2018 while this toolbox can work for the version 2020 of Matlab.

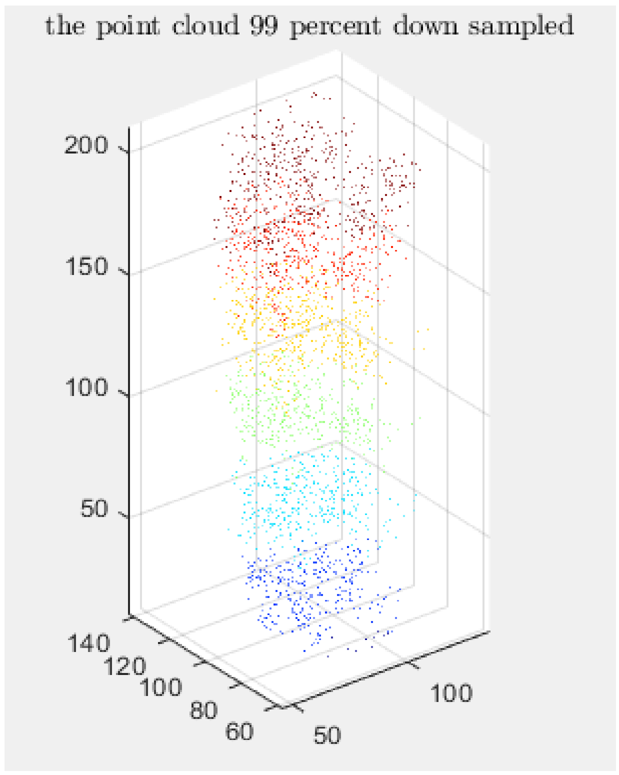
1.B) A point cloud is a set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z). Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. As the output of 3D scanning processes, point clouds are used for many purposes, including to create 3D CAD models for manufactured parts, for metrology and quality inspection, and for a multitude of visualization, animation, rendering and mass customization applications. Point clouds are often aligned with 3D models or with other point clouds, a process known as point set registration. We prefer working with point clouds instead of 3D images because in this way we can reduce memory usage and speed up the computations. Also point clouds can make an easy way to show a 3D image in only two dimensions. In fact, if we had an easy way to see a 3D image on paper and work with, making point clouds weren’t this much important.

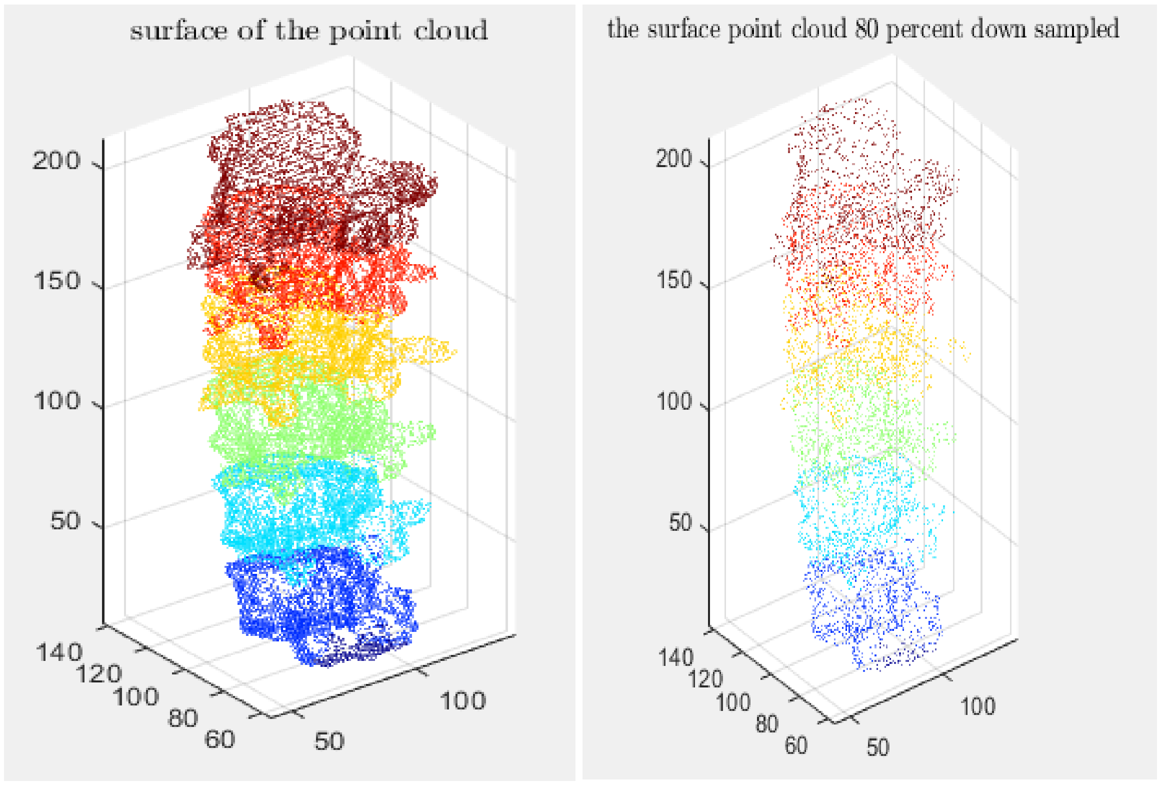
I plotted the point cloud of the healthy mask label in 4 states: total samples, down sampled by 90%, down sampled by 95%, down sampled by 99%, boundary, boundary down sampled by 90%. The results are presented below:



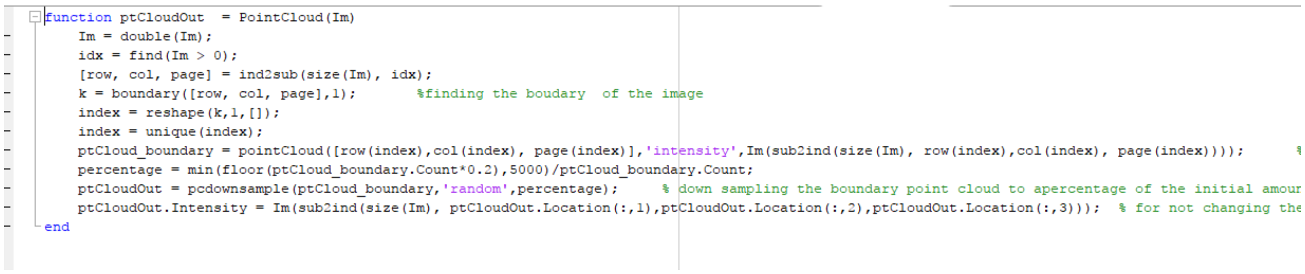








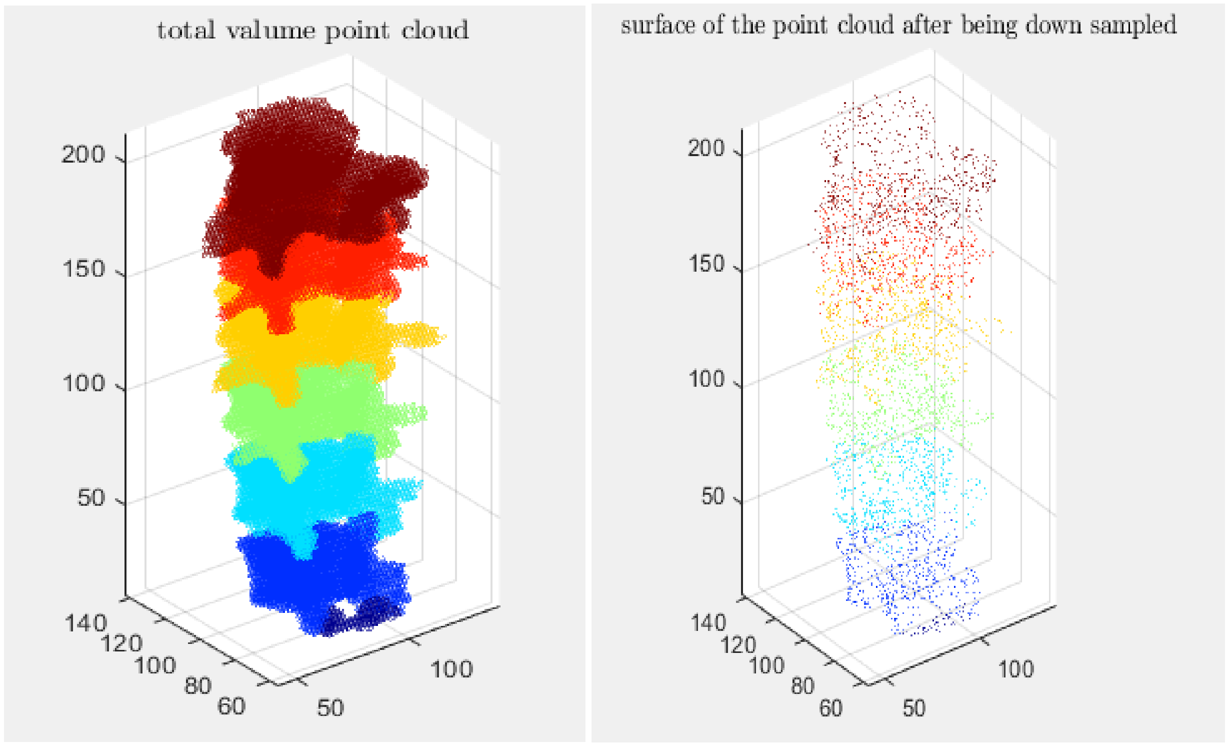
As we can see in these results, although it is better to reduce the samples in order to have lower computational cost, more than 95% down sampling in the main point cloud can cause some main information to be lost. However, if we extract the boundary which has almost all of the main information and then down sample we can efficiently reduce the sample as much as possible. I decided to keep the boundary with 80% down sampling, but I also kept all samples for boundary point clouds with less than 5000 sample in order not to lose the main information. I finalized this formula and turned it into a function in “PointCloud.m”.



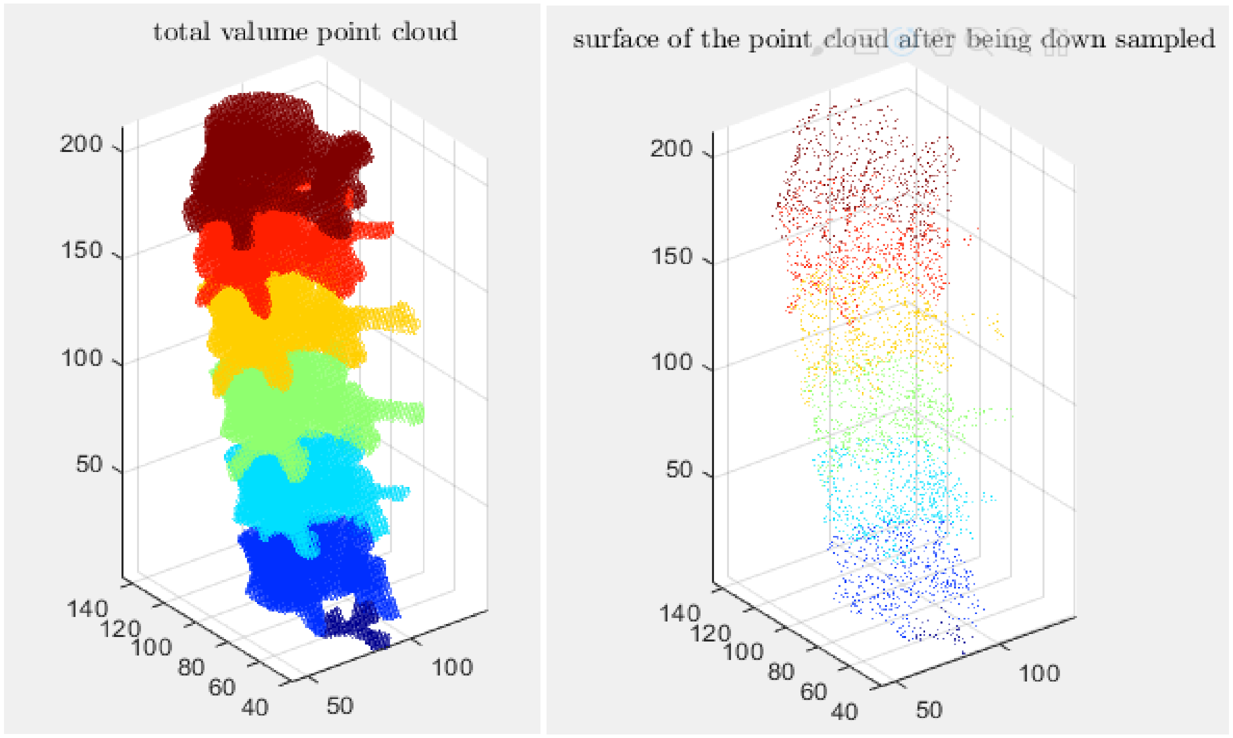
```
function ptCloudOut  = PointCloud(Im)
    Im = double(Im);
    idx = find(Im > 0);
    [row, col, page] = ind2sub(size(Im), idx);
    k = boundary([row, col, page],1);       %finding the boudary  of the image
    index = reshape(k,1,[]);
    index = unique(index);
    ptCloud_boundary = pointCloud([row(index),col(index), page(index)],'intensity',Im(sub2ind(size(Im), row(index),col(index), page(index))));      %
    percentage = min(floor(ptCloud_boundary.Count*0.2),5000)/ptCloud_boundary.Count;
    ptCloudOut = pcdownsample(ptCloud_boundary,'random',percentage);     % down sampling the boundary point cloud to apercentage of the initial amour
    ptCloudOut.Intensity = Im(sub2ind(size(Im), ptCloudOut.Location(:,1),ptCloudOut.Location(:,2),ptCloudOut.Location(:,3)));  % for not changing the
end
```

Here are the point clouds of the 4 cases in basic form vs. with the implemented PointCloud function:

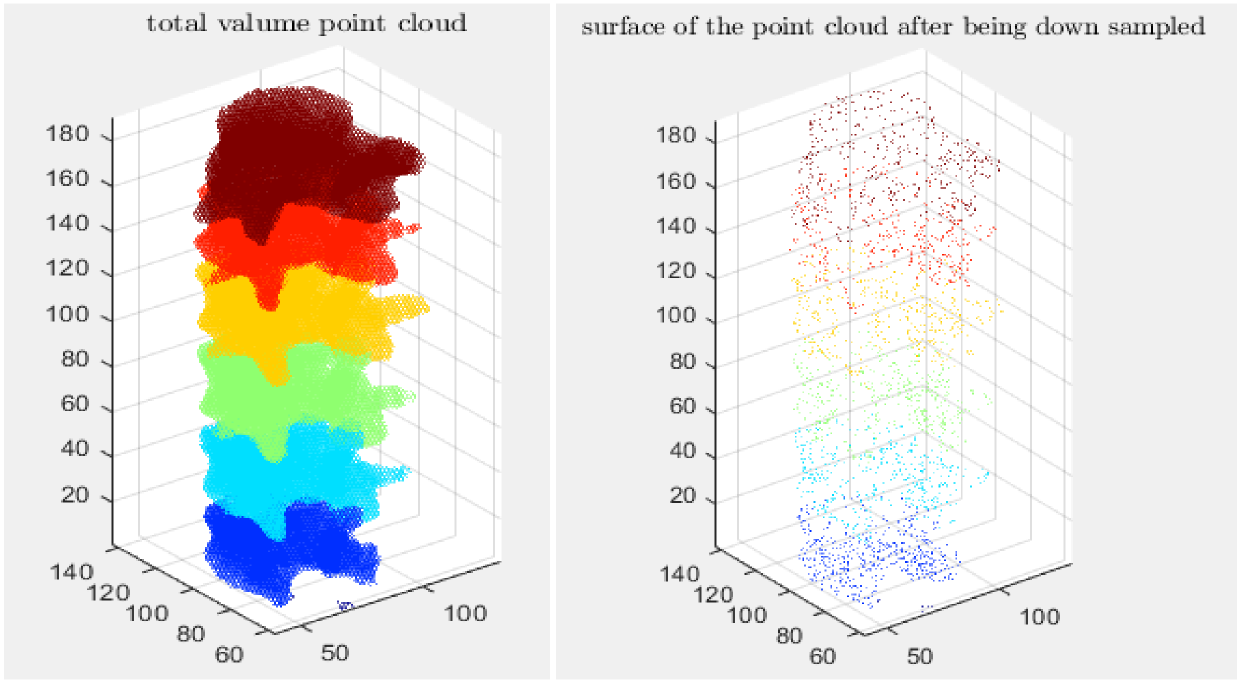
Healthy mask:



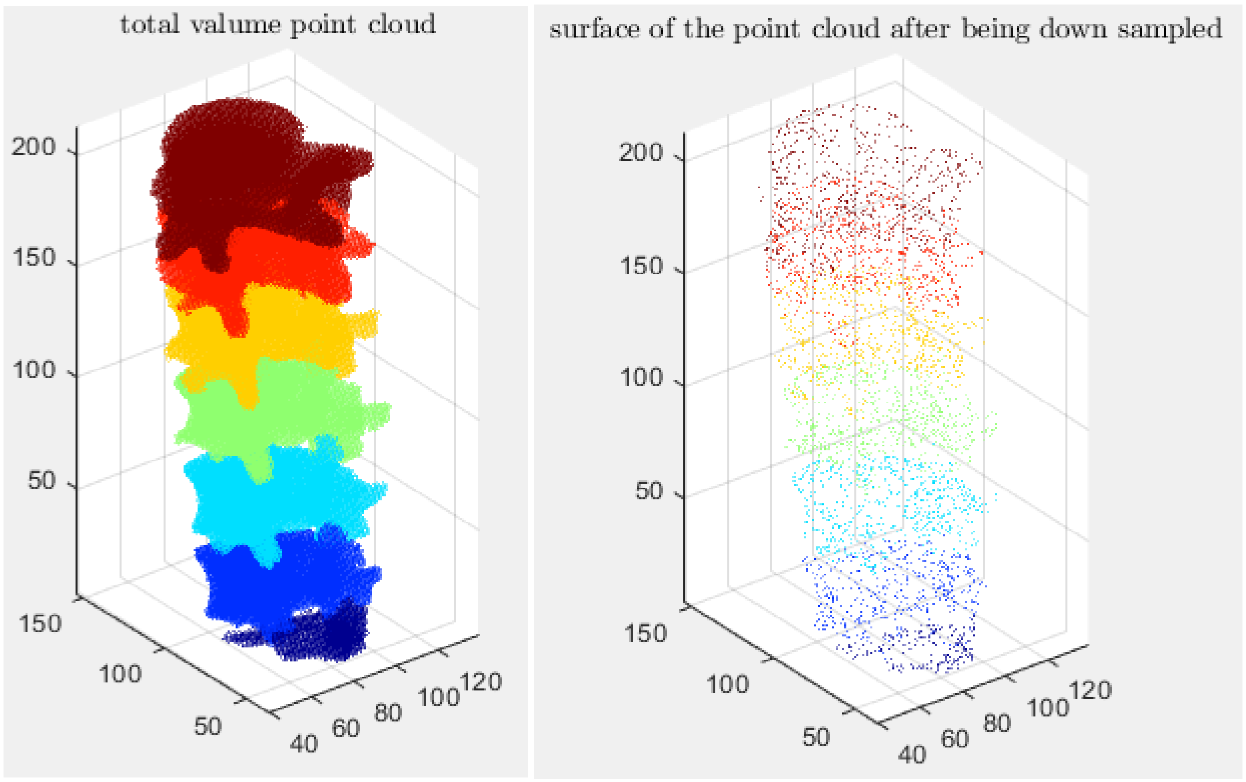
Patient number 2:



Patient number 10:



Patient number 21:



PART 2: Conformity Assessment

2.A)

Dice Score:

Dice score is computed as a measure of overlap between the surface volume (V) from automatic method and the reference surface volume (V_{ref}), giving a measurement value between 0 (full overlap) and 1 (no overlap).

$$D^* = 1 - \frac{2\left(V \cap V_{ref}\right)}{V + V_{ref}}$$

Average Surface Distance:

The mean surface distance, d_{mean}, between the surface (S) from automatic method and the reference surface (S_{ref}) defined as:

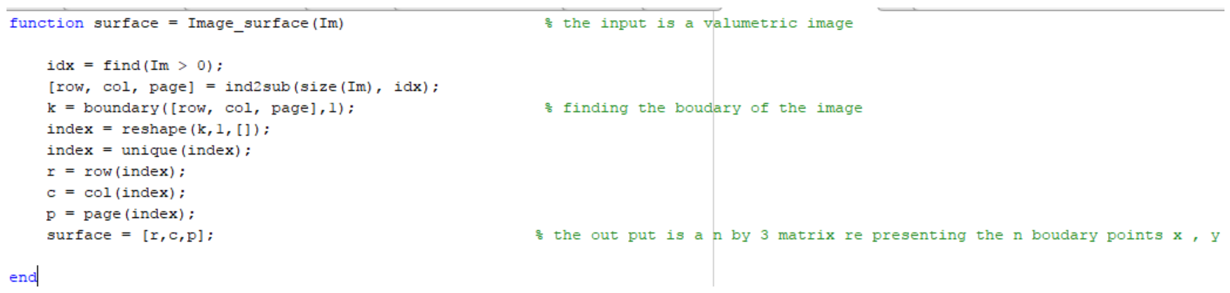
$$d_{mean} = \frac{1}{2}\left[\bar{d}\left(S, S_{ref}\right) + \bar{d}\left(S_{ref}, S\right)\right]$$

where $d(S,S_{ref})$ is the mean of distances between every surface voxel in S and the closest surface voxel in S_{ref}, while $d(S_{ref},S)$ is computed in a similar way. This value will be computed for both End Diastolic $d_{mean,ED}$ and End systolic $d_{mean,ES}$ instances.

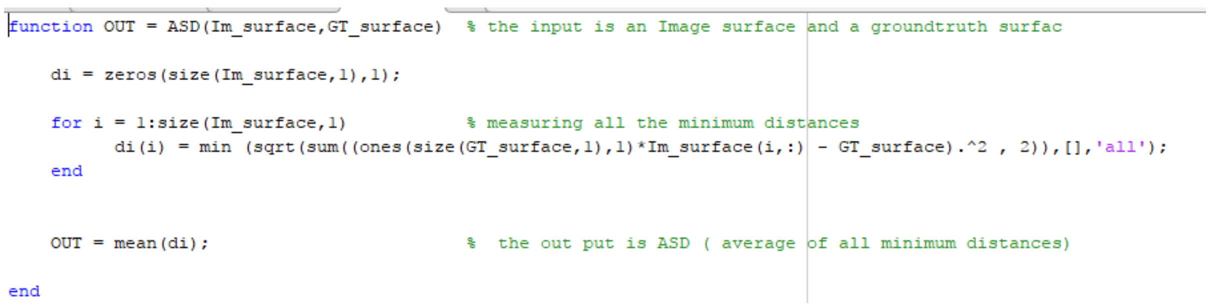
Hausdorff Distance:

The Hausdroff distance, d_H, measures the local maximum distance between the two surfaces S and S_{ref}. This value will be computed for both End Diastolic $d_{H,ED}$ and End systolic $d_{H,ES}$ instances.

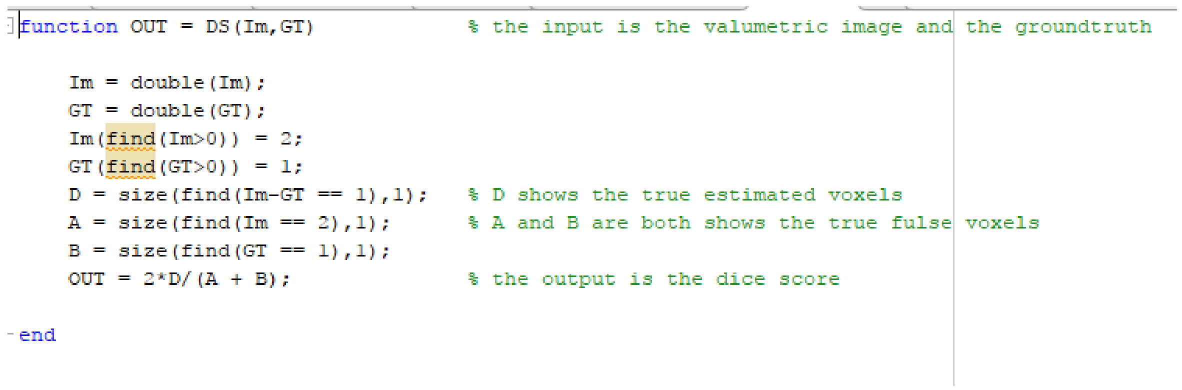
2.B) To evaluate these criteria I first a function named “Image_surface.m” that gets the image and gives the external surface of the input image using Matlab boundary function and I gave the output to the self-implemented “DS.m”, “ASD.m” and “HD.m” functions to calculate the corresponding criteria respectively.



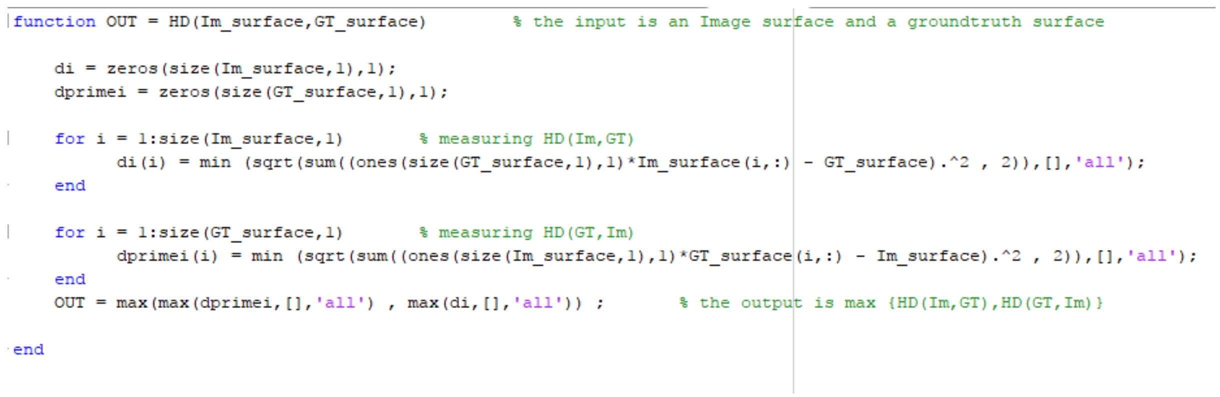
```
function surface = Image_surface(Im)                    % the input is a valumetric image

    idx = find(Im > 0);
    [row, col, page] = ind2sub(size(Im), idx);
    k = boundary([row, col, page],1);                   % finding the boudary of the image
    index = reshape(k,1,[]);
    index = unique(index);
    r = row(index);
    c = col(index);
    p = page(index);
    surface = [r,c,p];                                 % the out put is a n by 3 matrix re presenting the n boudary points x , y

end
```



```
function OUT = ASD(Im_surface,GT_surface)  % the input is an Image surface and a groundtruth surfac

    di = zeros(size(Im_surface,1),1);

    for i = 1:size(Im_surface,1)           % measuring all the minimum distances
          di(i) = min (sqrt(sum((ones(size(GT_surface,1),1)*Im_surface(i,:) - GT_surface).^2 , 2)),[],'all');
    end


    OUT = mean(di);                        %  the out put is ASD ( average of all minimum distances)

end
```

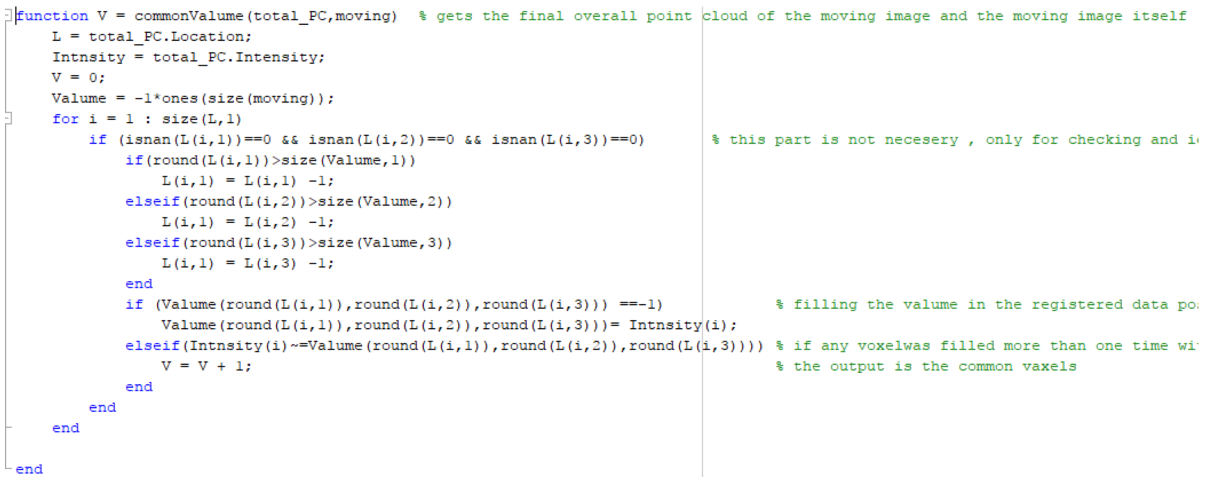


```
function OUT = DS(Im,GT)             % the input is the valumetric image and the groundtruth

     Im = double(Im);
     GT = double(GT);
     Im(find(Im>0)) = 2;
     GT(find(GT>0)) = 1;
     D = size(find(Im-GT == 1),1);   % D shows the true estimated voxels
     A = size(find(Im == 2),1);      % A and B are both shows the true fulse voxels
     B = size(find(GT == 1),1);
     OUT = 2*D/(A + B);              % the output is the dice score

end
```



```
function OUT = HD(Im_surface,GT_surface)        % the input is an Image surface and a groundtruth surface

    di = zeros(size(Im_surface,1),1);
    dprimei = zeros(size(GT_surface,1),1);

    for i = 1:size(Im_surface,1)        % measuring HD(Im,GT)
          di(i) = min (sqrt(sum((ones(size(GT_surface,1),1)*Im_surface(i,:) - GT_surface).^2 , 2)),[],'all');
    end

    for i = 1:size(GT_surface,1)        % measuring HD(GT,Im)
          dprimei(i) = min (sqrt(sum((ones(size(Im_surface,1),1)*GT_surface(i,:) - Im_surface).^2 , 2)),[],'all');
    end
    OUT = max(max(dprimei,[],'all') , max(di,[],'all')) ;       % the output is max {HD(Im,GT),HD(GT,Im)}

end
```

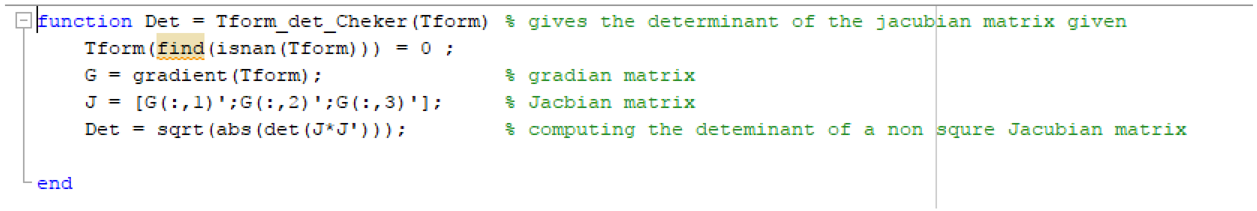
3.C) To measure the common volume I implemented the “commonValume.m” function which gets the interpolated point cloud and tries to reconstruct a volumetric image and simultaneously checks if in any final position 2 differently labeled voxels are mapped.



```
function V = commonValume(total_PC,moving)  % gets the final overall point cloud of the moving image and the moving image itself
    L = total_PC.Location;
    Intnsity = total_PC.Intensity;
    V = 0;
    Valume = -1*ones(size(moving));
    for i = 1 : size(L,1)
        if (isnan(L(i,1))==0 && isnan(L(i,2))==0 && isnan(L(i,3))==0)      % this part is not necesery , only for checking and i
            if(round(L(i,1))>size(Valume,1))
                L(i,1) = L(i,1) -1;
            elseif(round(L(i,2))>size(Valume,2))
                L(i,1) = L(i,2) -1;
            elseif(round(L(i,3))>size(Valume,3))
                L(i,1) = L(i,3) -1;
            end
            if (Valume(round(L(i,1)),round(L(i,2)),round(L(i,3))) ==-1)          % filling the valume in the registered data po
                Valume(round(L(i,1)),round(L(i,2)),round(L(i,3)))= Intnsity(i);
            elseif(Intnsity(i)~=Valume(round(L(i,1)),round(L(i,2)),round(L(i,3)))) % if any voxelwas filled more than one time wi
                V = V + 1;                                                          % the output is the common vaxels
            end
        end
    end

end
```

3.D) to measure this criterion I implemented the “Tform_det_Checker.m“ function that gets the transform function, calculates its gradian and makes the Jacobian matrix. Then the determinant of this non-square matrix is calculated by this formula:

$$\sqrt{\det \mathbf{J}_F^{\mathrm{T}} \mathbf{J}_F}$$



```
function Det = Tform_det_Cheker(Tform) % gives the determinant of the jacubian matrix given
    Tform(find(isnan(Tform))) = 0 ;
    G = gradient(Tform);                % gradian matrix
    J = [G(:,1)';G(:,2)';G(:,3)'];      % Jacbian matrix
    Det = sqrt(abs(det(J*J')));         % computing the deteminant of a non squre Jacubian matrix

end
```

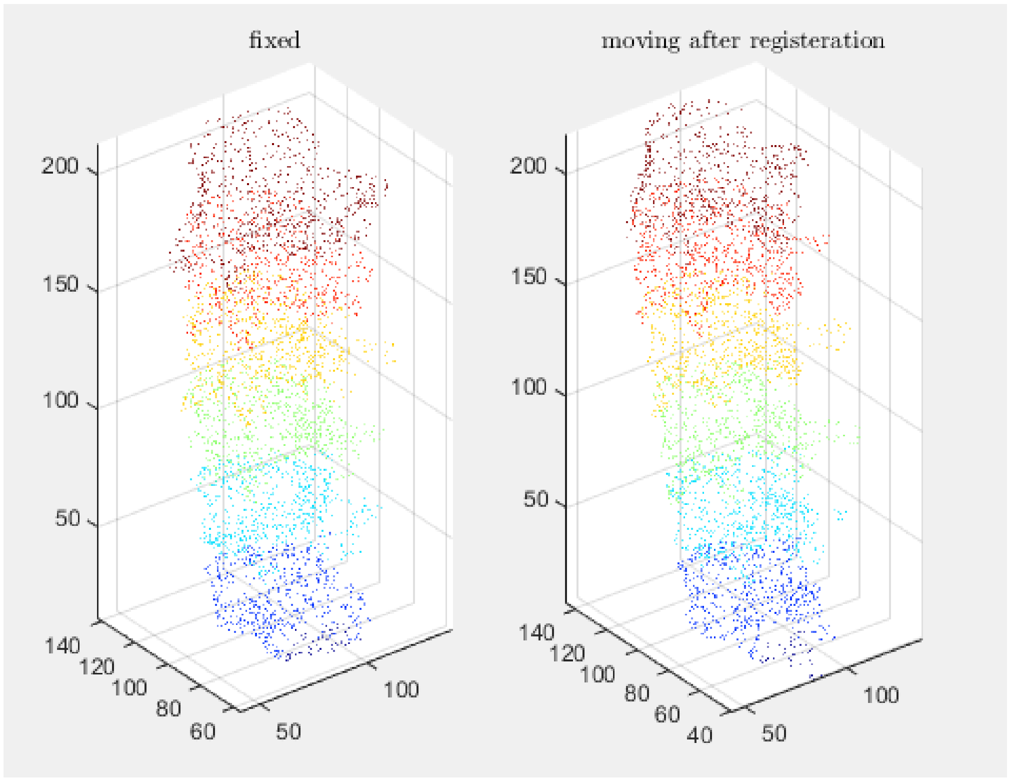
PART 3: Implementation of Image Registration

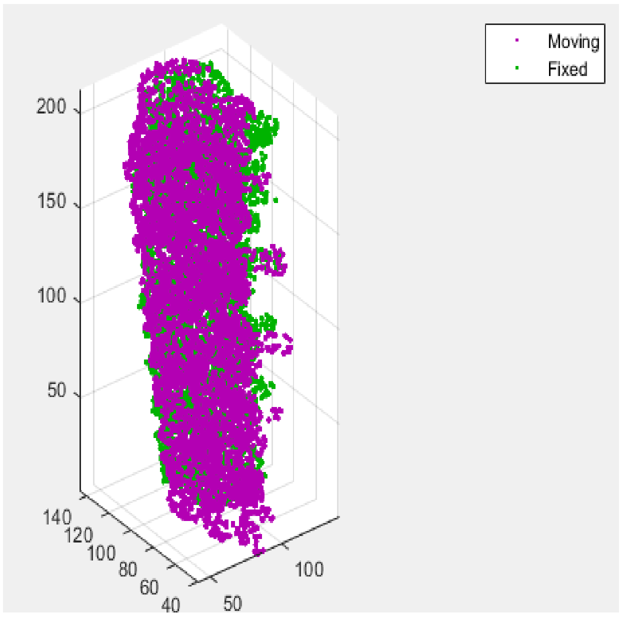
3. A)

In order to register the mask and a patient labeled image point cloud Non rigid CPD algorithm is used, cause as we know a rigid transform cannot handle the small rotations curvature caused by small gesture differences.

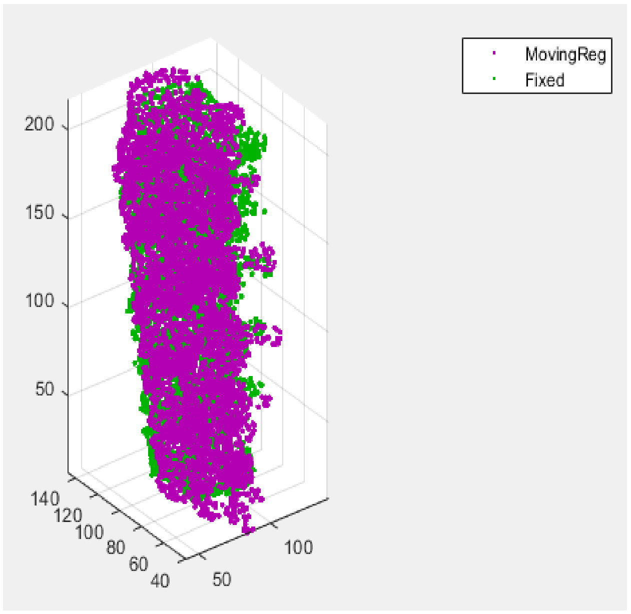
Here are given the examples of those 3 patients image reduced point clouds before and after they were registered to the healthy mask vs. the mask (note that the moving image is the patient image and the fixed one is the mask).

Case 1: Patient number 2



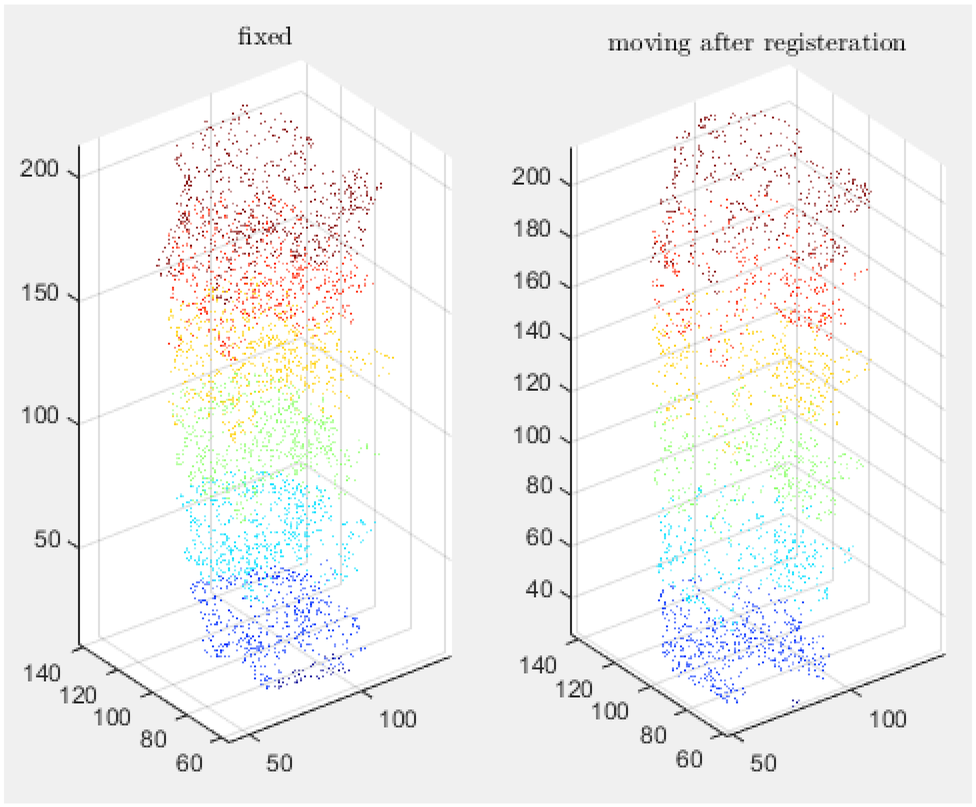


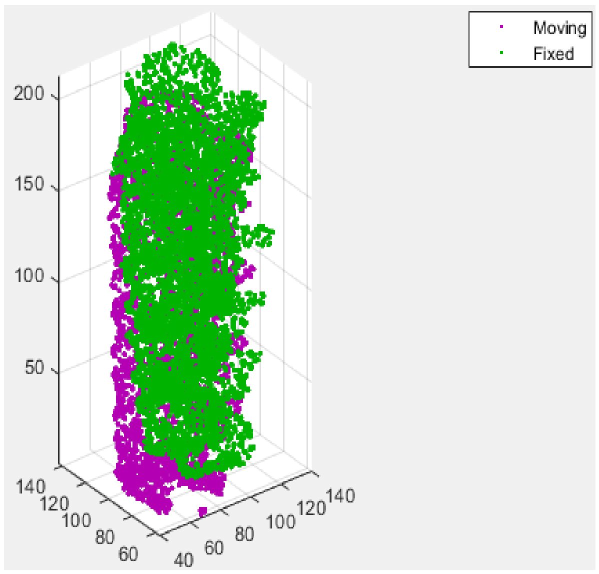
moving before registration vs. fixed



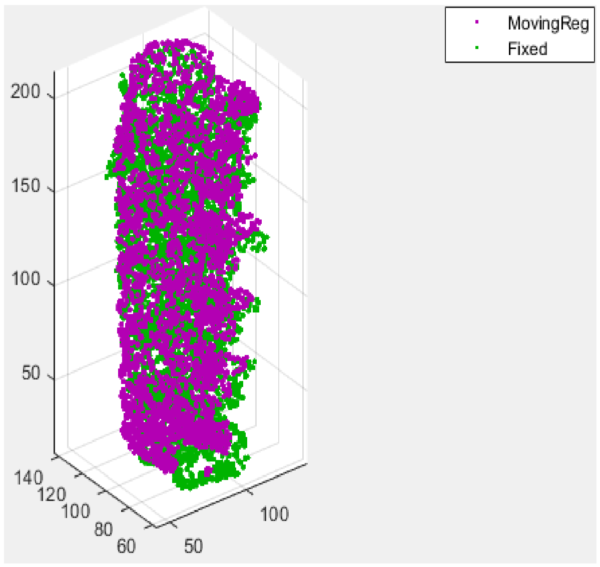
moving after registration vs. fixed

Case 2: Patient number 10



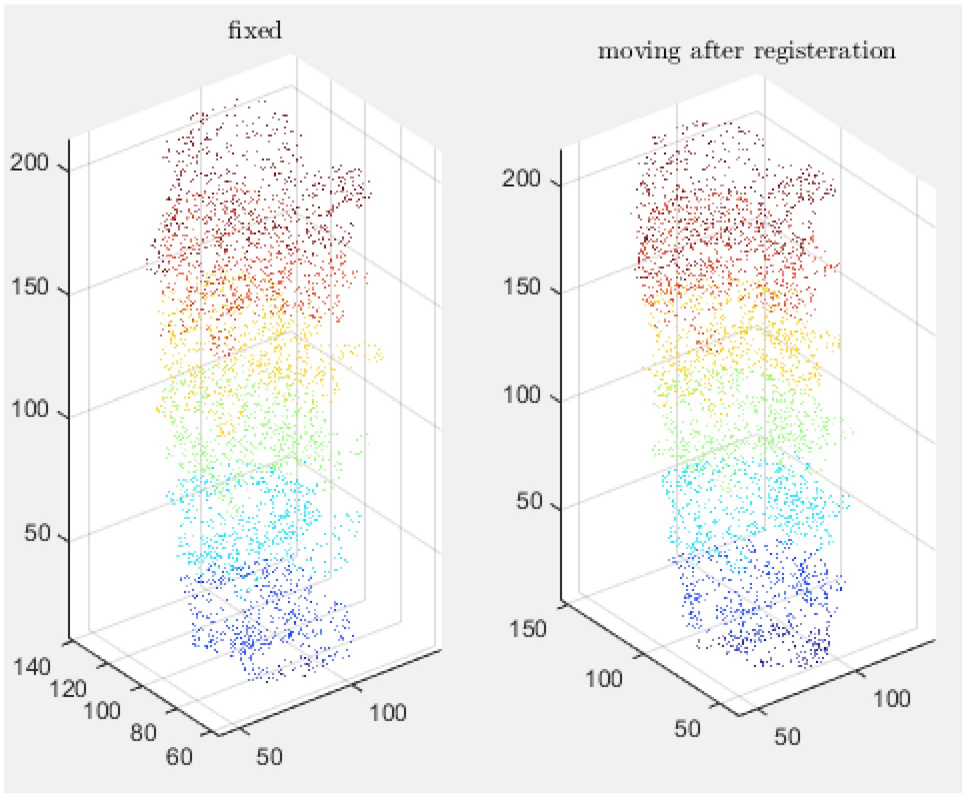


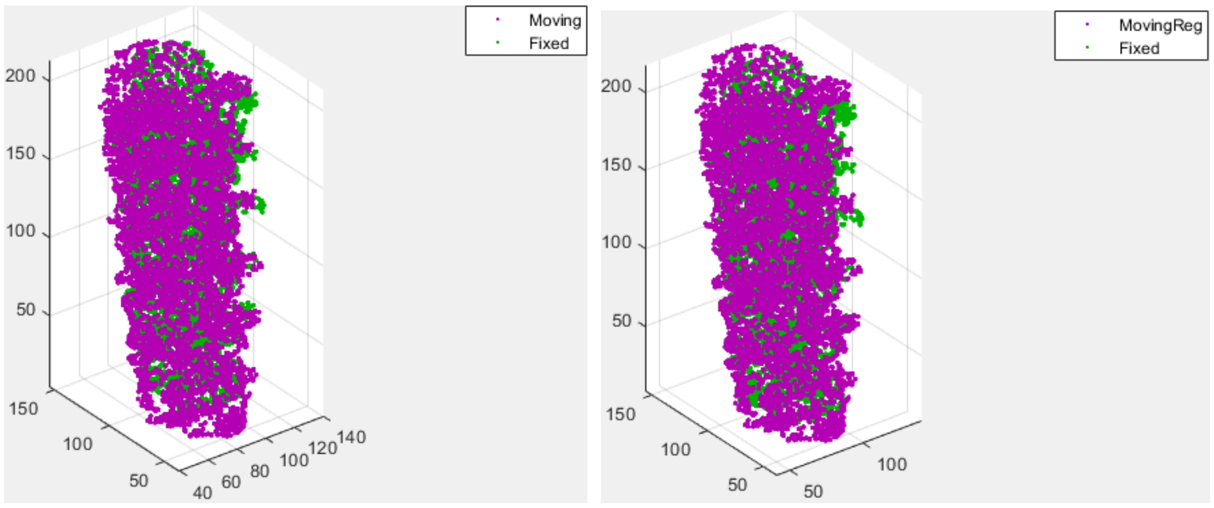
moving before registration vs. fixed



moving after registration vs. fixed

Case 3: Patient number 21

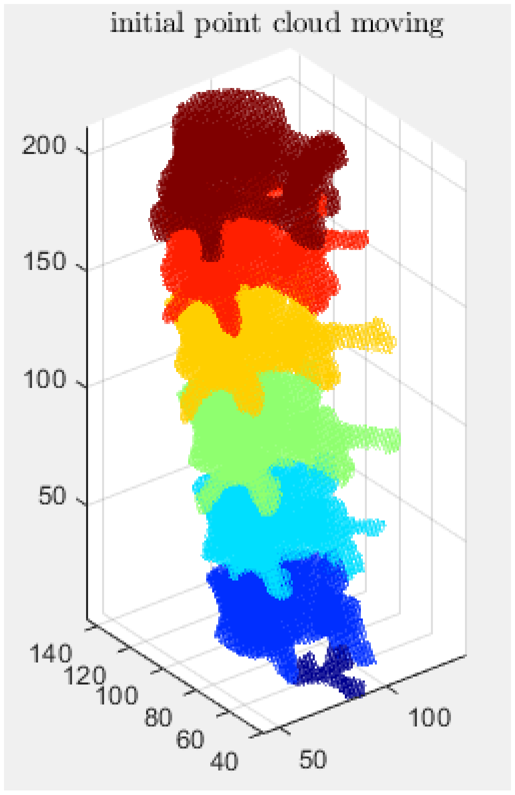


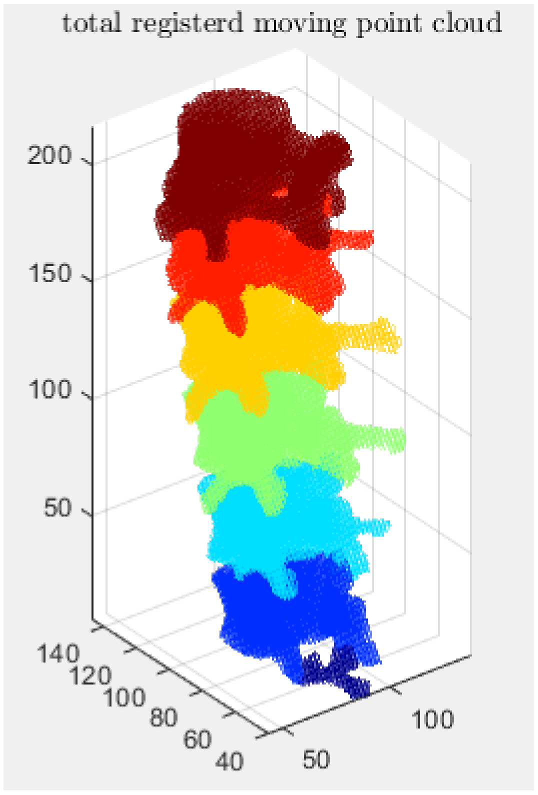


moving before registration vs. fixed moving after registration vs. fixed

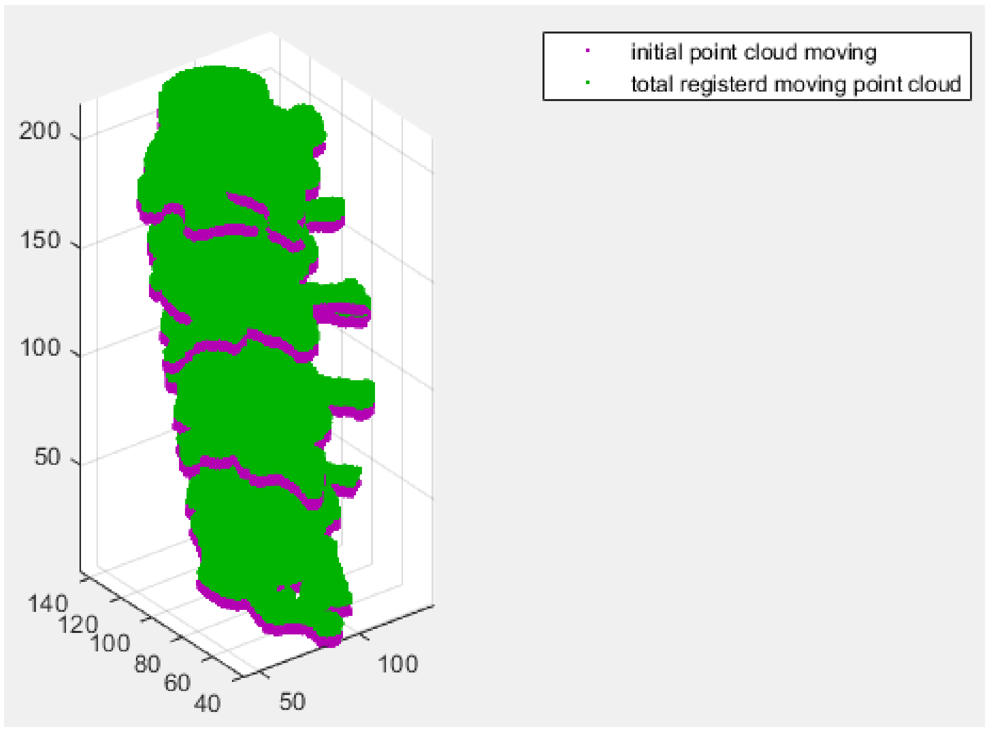
After registering the reduced point clouds, I interpolated the output transformation matrix (DDF) in order to have the transformation matrix at all voxels of the image which are labeled as a vertebra. I used the Matlab function “griddata” for this interpolation. “griddata” is used for a one-dimensional interpolation, so I used it 3 times to calculate the interpolation in the direction of the three axes. In the last step I applied the transformation on the total (not reduced) point cloud of the patient's images to reach the total point clouds after the registration. Here are the results for the 3 patients:

Case 1: Patient number 2

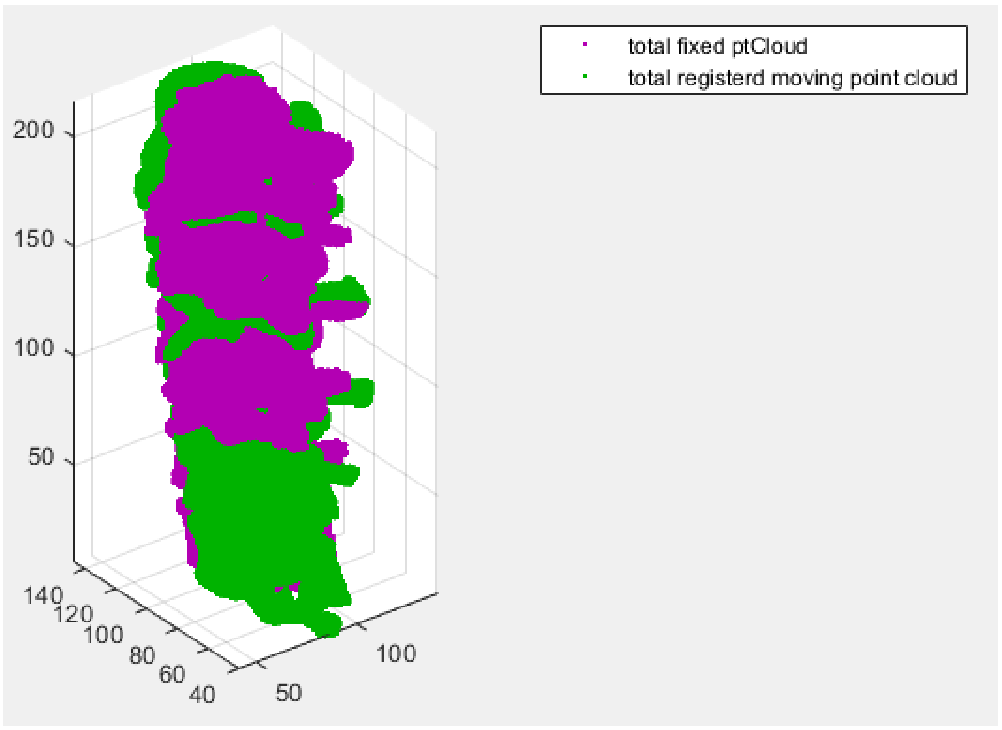




the total moving point cloud before registration the total moving point cloud after registration

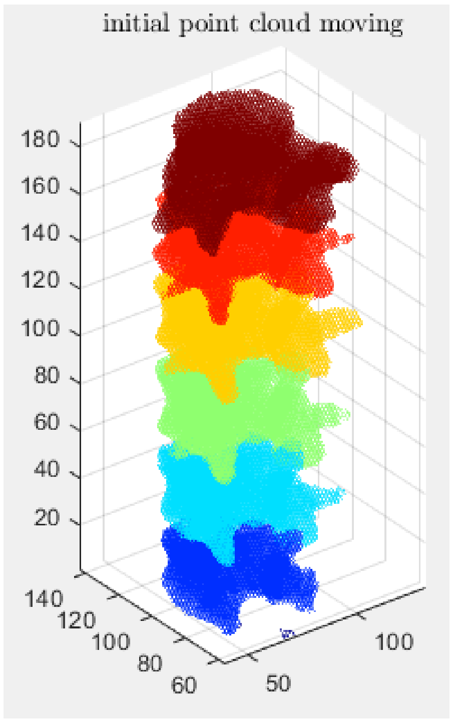


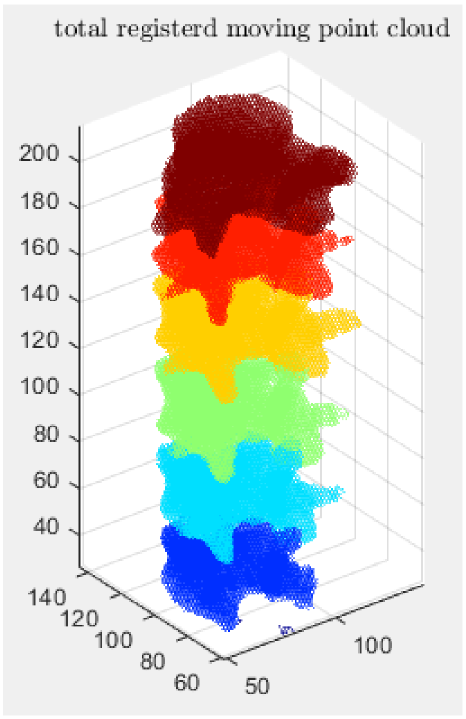
the total moving point cloud before registration vs. the total fixed point cloud



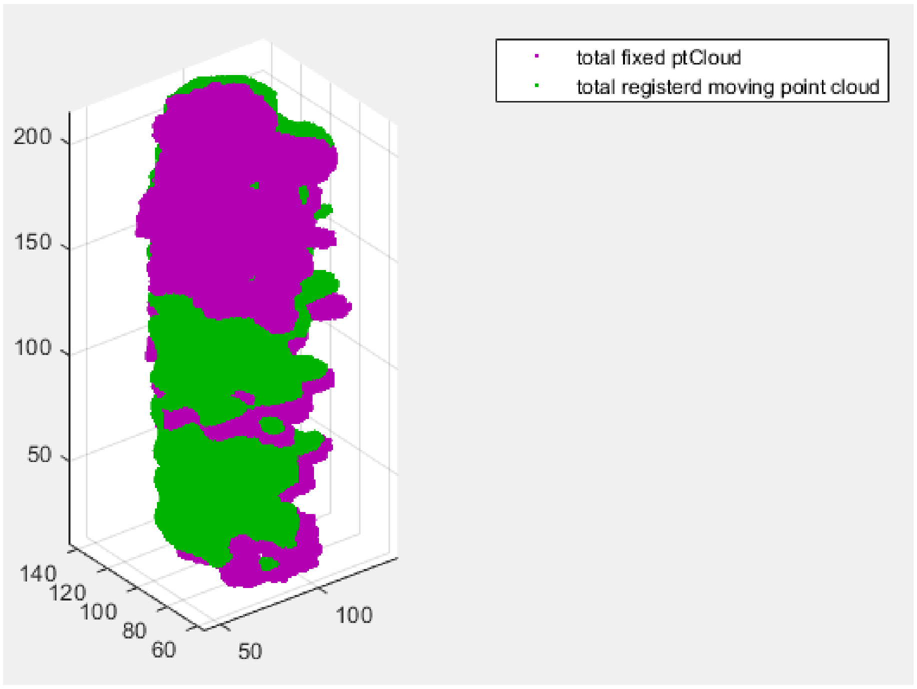
the total moving point cloud after registration vs. the total fixed point cloud

Case 2: Patient number 10

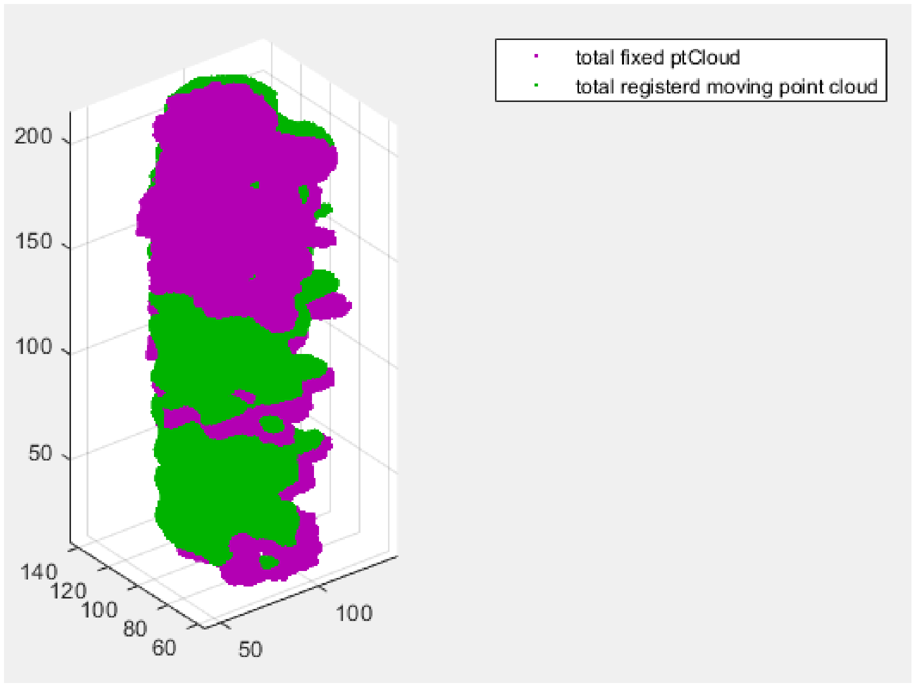




the total moving point cloud before registration the total moving point cloud after registration

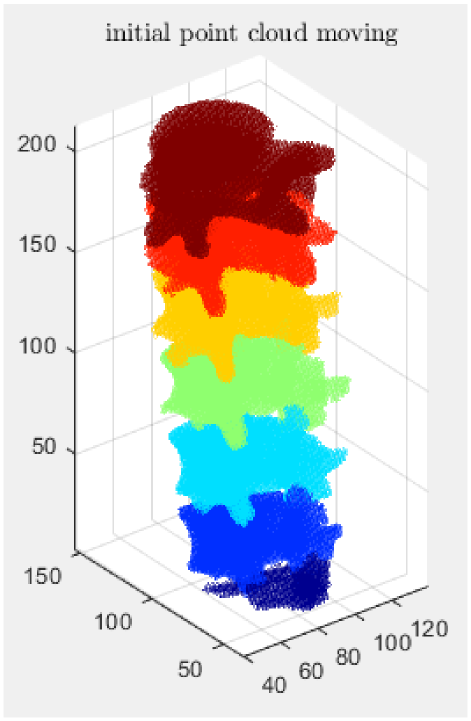


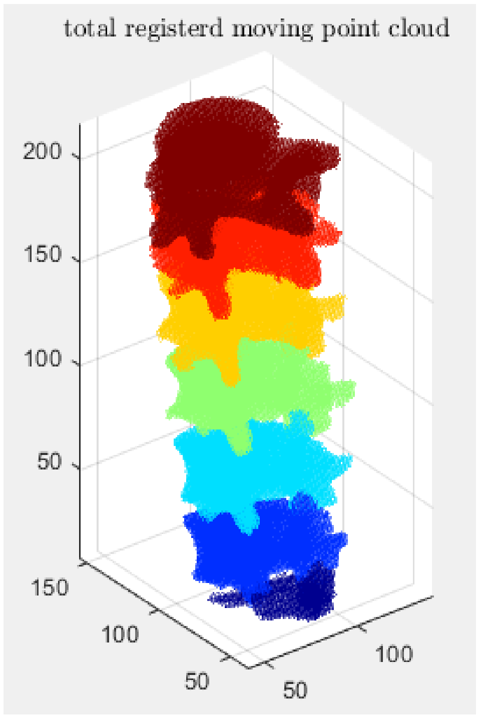
the total moving point cloud before registration vs. the total fixed point cloud



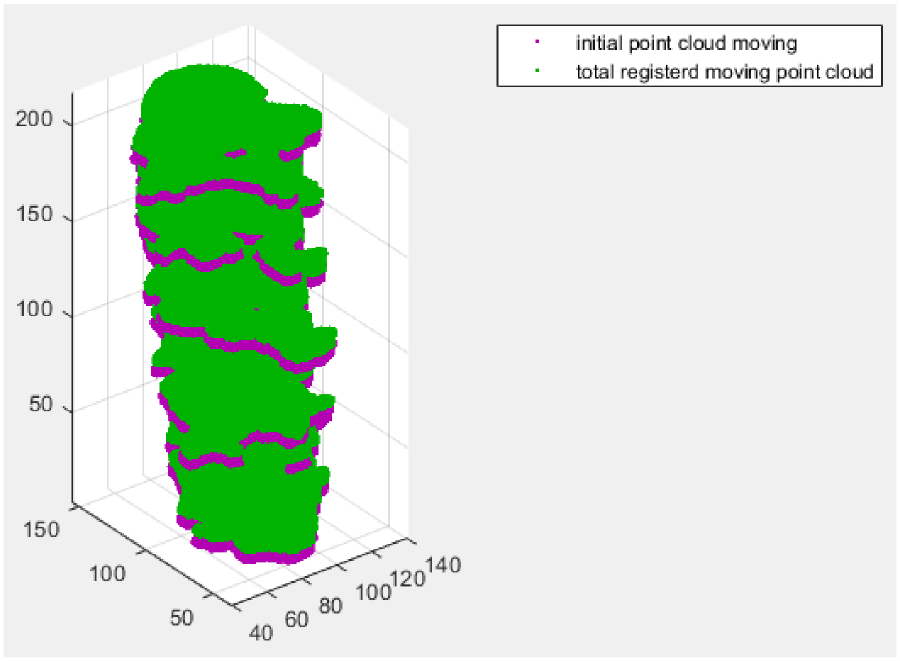
the total moving point cloud after registration vs. the total fixed point cloud

Case 3: Patient number 21

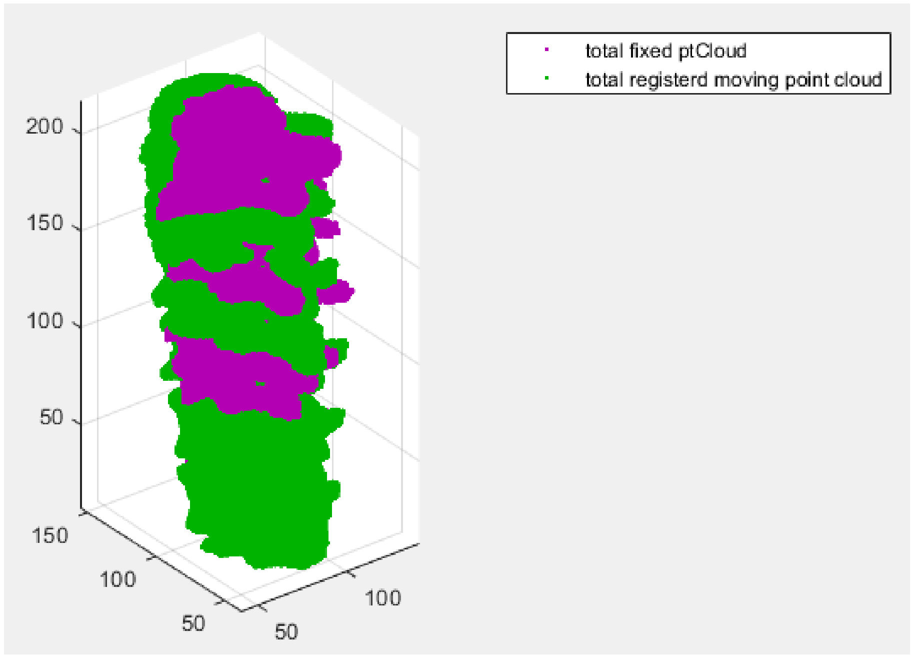




the total moving point cloud before registration the total moving point cloud after registration

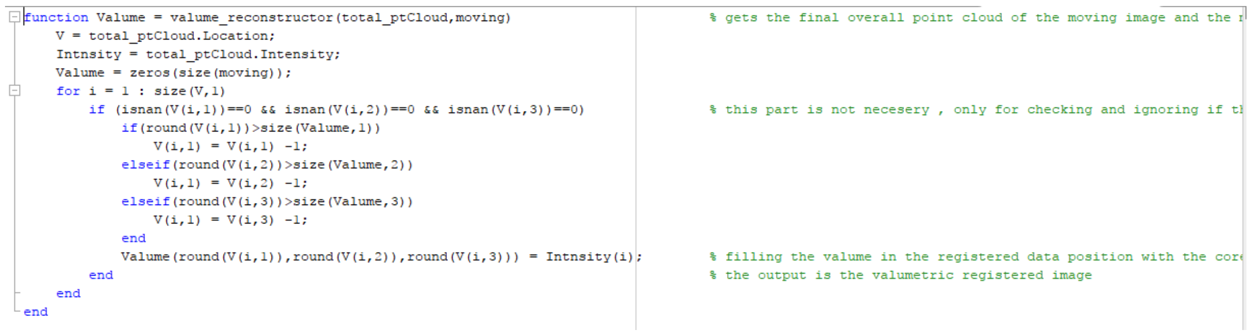


the total moving point cloud before registration vs. the total fixed point cloud



the total moving point cloud after registration vs. the total fixed point cloud

At last, I reconstructed the 3D volumetric image from its point cloud with the self-implemented function “valume_reconstructor.m” after registration that can be shown shown by Volume Vieer in Matlab version 2020.

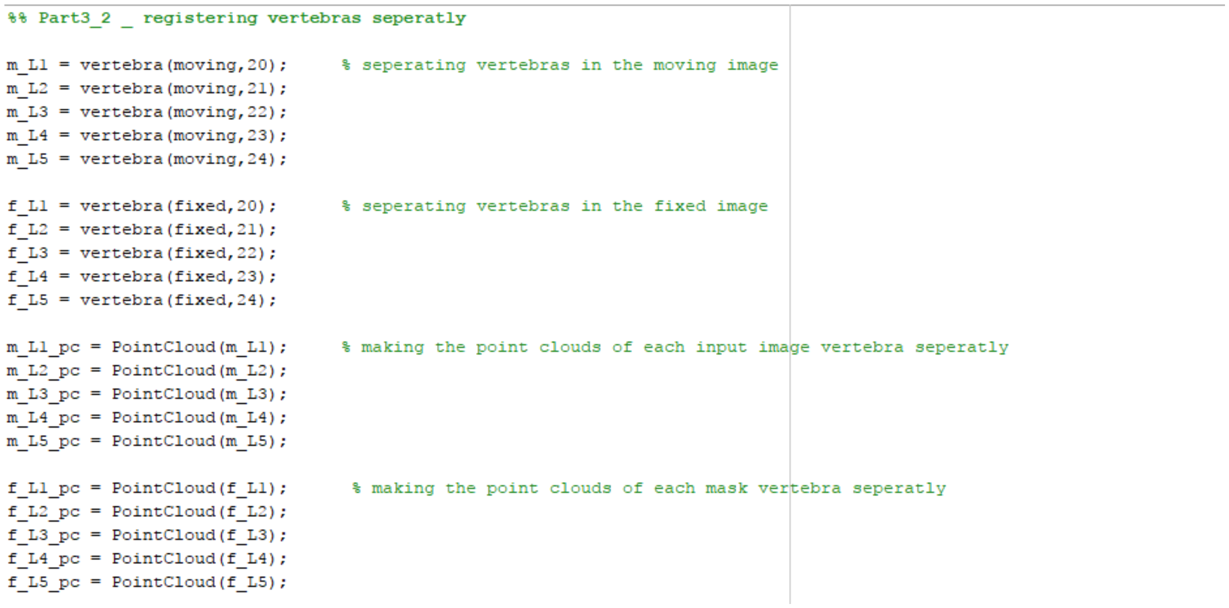


```
function Valume = valume_reconstructor(total_ptCloud,moving)                     % gets the final overall point cloud of the moving image and the r
    V = total_ptCloud.Location;
    Intnsity = total_ptCloud.Intensity;
    Valume = zeros(size(moving));
    for i = 1 : size(V,1)
        if (isnan(V(i,1))==0 && isnan(V(i,2))==0 && isnan(V(i,3))==0)           % this part is not necesery , only for checking and ignoring if th
            if(round(V(i,1))>size(Valume,1))
                V(i,1) = V(i,1) -1;
            elseif(round(V(i,2))>size(Valume,2))
                V(i,1) = V(i,2) -1;
            elseif(round(V(i,3))>size(Valume,3))
                V(i,1) = V(i,3) -1;
            end
            Valume(round(V(i,1)),round(V(i,2)),round(V(i,3))) = Intnsity(i);     % filling the valume in the registered data position with the corr
        end                                                                       % the output is the valumetric registered image
    end
end
```

The results of this registration are investigated by measuring the criteria mentioned in part 2 and here is the results table:

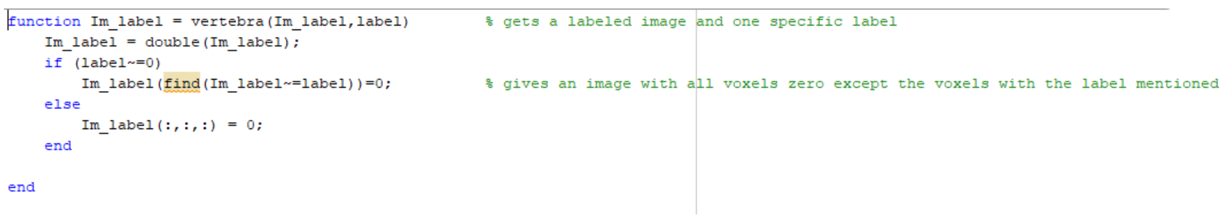
	DS	ASD	HD	Det(J(Tform))	CV (common volume)
Patient 2	0.5937	3.6080	22.3383	0.2242	0 voxels
Patient 10	0.5956	3.1024	16.0935	5.0680	0 voxels
Patient 21	0.6178	4.2536	25.9808	0.0805	0 voxels

3. B) In this step I decided to register the L1 to L5 vertebras seperatly and then make a DDF from these 5 DDFs. Therefor, first I changed the moving and fixed image each into 5 images that only the labeled patrs for the vertebra Li remained non zero in them. Then I made 5 coresponding point clouds of these images. Here is the code for this part:



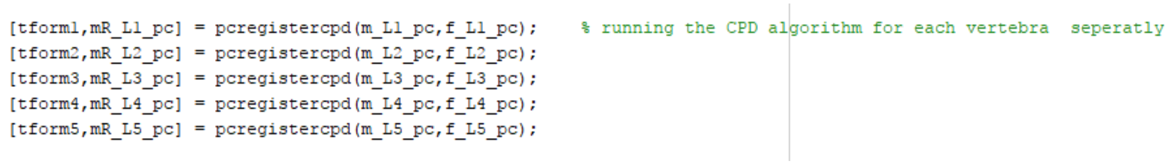
```
%% Part3_2 _ registering vertebras seperatly

m_L1 = vertebra(moving,20);      % seperating vertebras in the moving image
m_L2 = vertebra(moving,21);
m_L3 = vertebra(moving,22);
m_L4 = vertebra(moving,23);
m_L5 = vertebra(moving,24);

f_L1 = vertebra(fixed,20);       % seperating vertebras in the fixed image
f_L2 = vertebra(fixed,21);
f_L3 = vertebra(fixed,22);
f_L4 = vertebra(fixed,23);
f_L5 = vertebra(fixed,24);

m_L1_pc = PointCloud(m_L1);      % making the point clouds of each input image vertebra seperatly
m_L2_pc = PointCloud(m_L2);
m_L3_pc = PointCloud(m_L3);
m_L4_pc = PointCloud(m_L4);
m_L5_pc = PointCloud(m_L5);

f_L1_pc = PointCloud(f_L1);       % making the point clouds of each mask vertebra seperatly
f_L2_pc = PointCloud(f_L2);
f_L3_pc = PointCloud(f_L3);
f_L4_pc = PointCloud(f_L4);
f_L5_pc = PointCloud(f_L5);
```

The “vertebra.m” function gets and image and a label and gives an image with all voxels zero except the ones that have the given label.



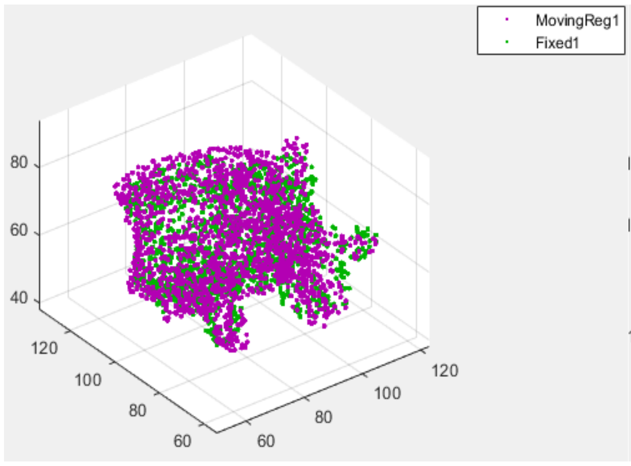
```
function Im_label = vertebra(Im_label,label)          % gets a labeled image and one specific label
    Im_label = double(Im_label);
    if (label~=0)
        Im_label(find(Im_label~=label))=0;            % gives an image with all voxels zero except the voxels with the label mentioned
    else
        Im_label(:,:,:) = 0;
    end

end
```

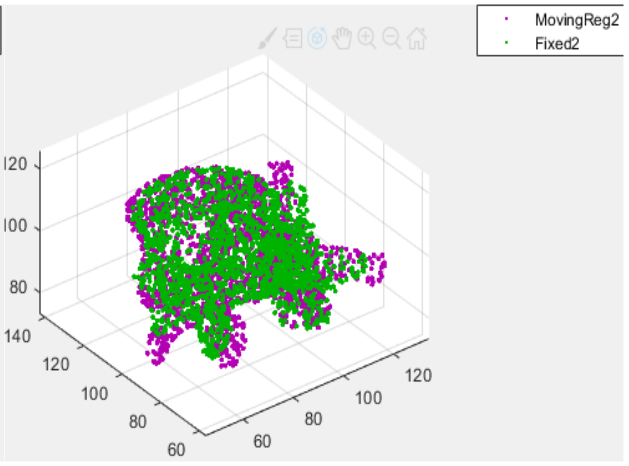
Then I ran the CPD algorithm for each vertebra(each of the five images) and got 5 coresponding transform functions.



```
[tform1,mR_L1_pc] = pcregistercpd(m_L1_pc,f_L1_pc);    % running the CPD algorithm for each vertebra  seperatly
[tform2,mR_L2_pc] = pcregistercpd(m_L2_pc,f_L2_pc);
[tform3,mR_L3_pc] = pcregistercpd(m_L3_pc,f_L3_pc);
[tform4,mR_L4_pc] = pcregistercpd(m_L4_pc,f_L4_pc);
[tform5,mR_L5_pc] = pcregistercpd(m_L5_pc,f_L5_pc);
```

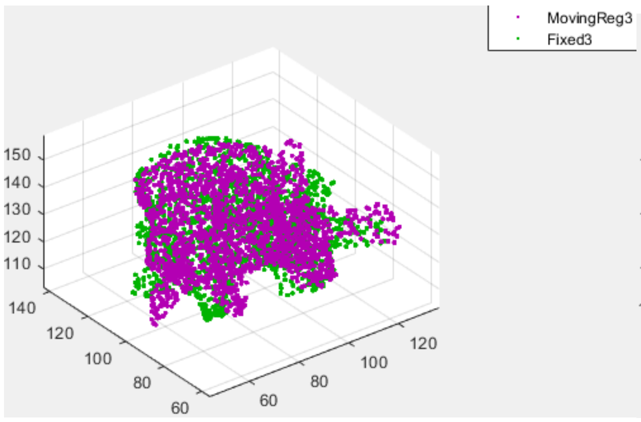
The picture of each moving vertebra’s registered point cloud is plotted with the coresponding fixed one’s just for the patient number 2 as an example:

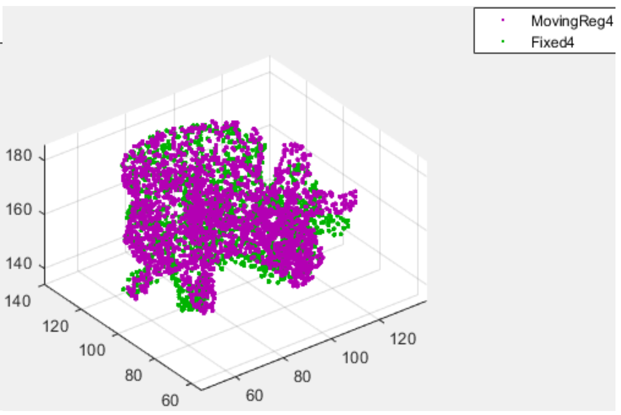




vertebra L1, fixed vs. registered moving

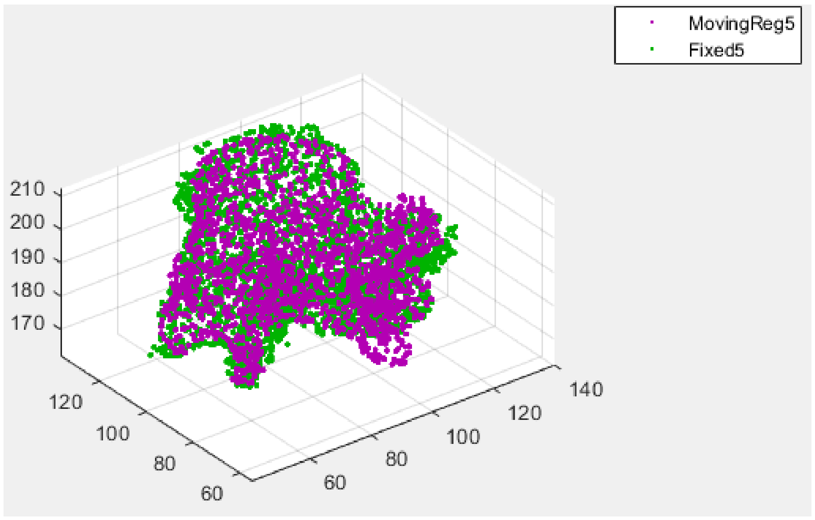
vertebra L2, fixed vs. registered moving





vertebra L3, fixed vs. registered moving

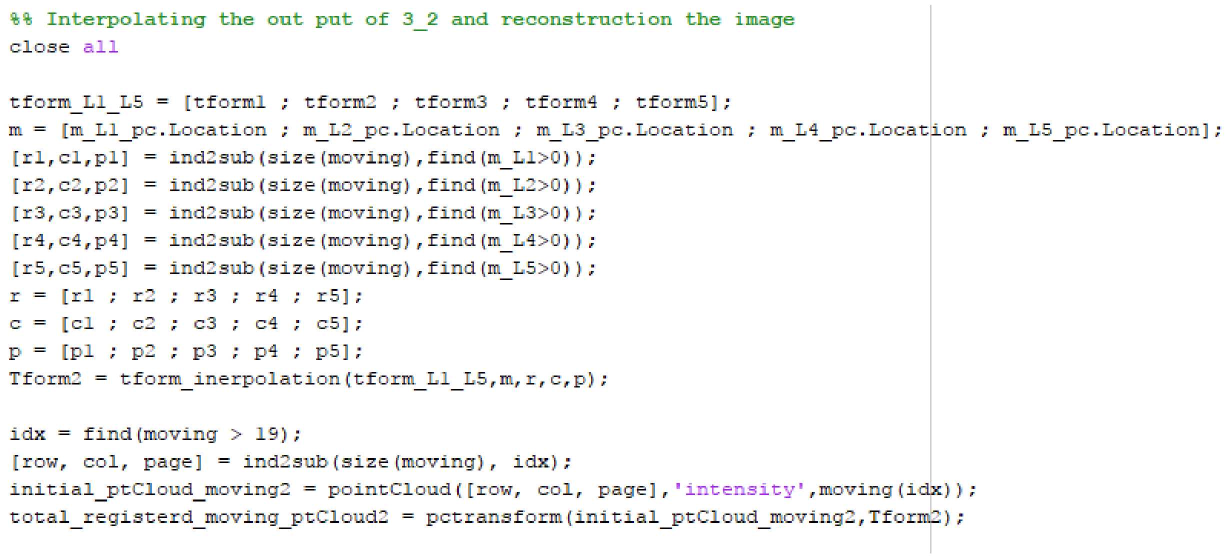
vertebra L4 , fixed vs. registered moving



vertebra L5, fixed vs. registered moving

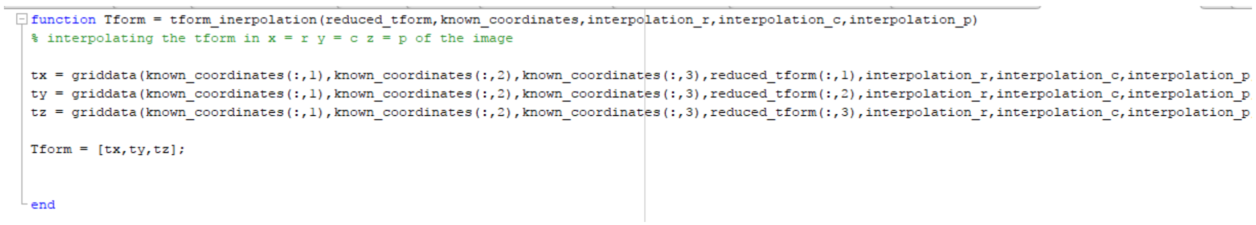
Interpolation

In order to aggregate these 5 transformations (DDFs) and come into a single transformation for all labeled vertebras I used intepolation. We know that each DDF is applied to a single and different set of locations in the image so I put these five set of point cloud locations one under another and do this to the tranformations matrix too, and took them inputs and outputs of a total DDF, respectively.



```
%% Interpolating the out put of 3_2 and reconstruction the image
close all

tform_L1_L5 = [tform1 ; tform2 ; tform3 ; tform4 ; tform5];
m = [m_L1_pc.Location ; m_L2_pc.Location ; m_L3_pc.Location ; m_L4_pc.Location ; m_L5_pc.Location];
[r1,c1,p1] = ind2sub(size(moving),find(m_L1>0));
[r2,c2,p2] = ind2sub(size(moving),find(m_L2>0));
[r3,c3,p3] = ind2sub(size(moving),find(m_L3>0));
[r4,c4,p4] = ind2sub(size(moving),find(m_L4>0));
[r5,c5,p5] = ind2sub(size(moving),find(m_L5>0));
r = [r1 ; r2 ; r3 ; r4 ; r5];
c = [c1 ; c2 ; c3 ; c4 ; c5];
p = [p1 ; p2 ; p3 ; p4 ; p5];
Tform2 = tform_inerpolation(tform_L1_L5,m,r,c,p);

idx = find(moving > 19);
[row, col, page] = ind2sub(size(moving), idx);
initial_ptCloud_moving2 = pointCloud([row, col, page],'intensity',moving(idx));
total_registerd_moving_ptCloud2 = pctransform(initial_ptCloud_moving2,Tform2);
```

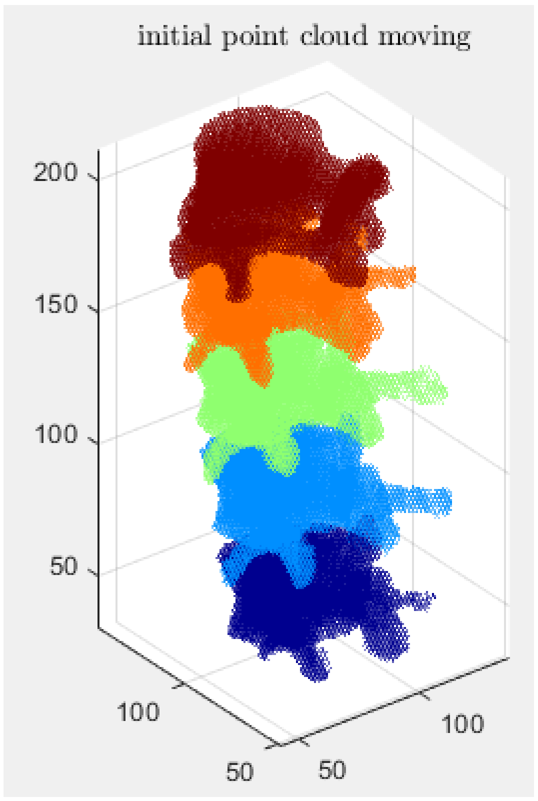
Then I interpolated this total DDF in all locations of the patient image total point cloud(not reduced) through the self-implemented “tform-interpolation” function made with matlab “griddata” function.



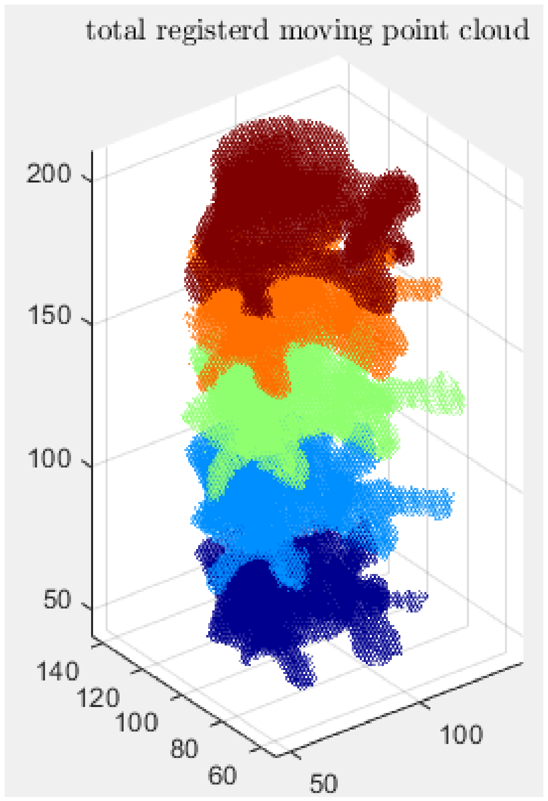
```
function Tform = tform_inerpolation(reduced_tform,known_coordinates,interpolation_r,interpolation_c,interpolation_p)
% interpolating the tform in x = r y = c z = p of the image

tx = griddata(known_coordinates(:,1),known_coordinates(:,2),known_coordinates(:,3),reduced_tform(:,1),interpolation_r,interpolation_c,interpolation_p
ty = griddata(known_coordinates(:,1),known_coordinates(:,2),known_coordinates(:,3),reduced_tform(:,2),interpolation_r,interpolation_c,interpolation_p
tz = griddata(known_coordinates(:,1),known_coordinates(:,2),known_coordinates(:,3),reduced_tform(:,3),interpolation_r,interpolation_c,interpolation_p

Tform = [tx,ty,tz];


end
```

In the last step I applied the interpolated transformation on the total (not reduced) point cloud of the patient's images to reach the total point clouds after the registration. Here are the results for the 3 patients:

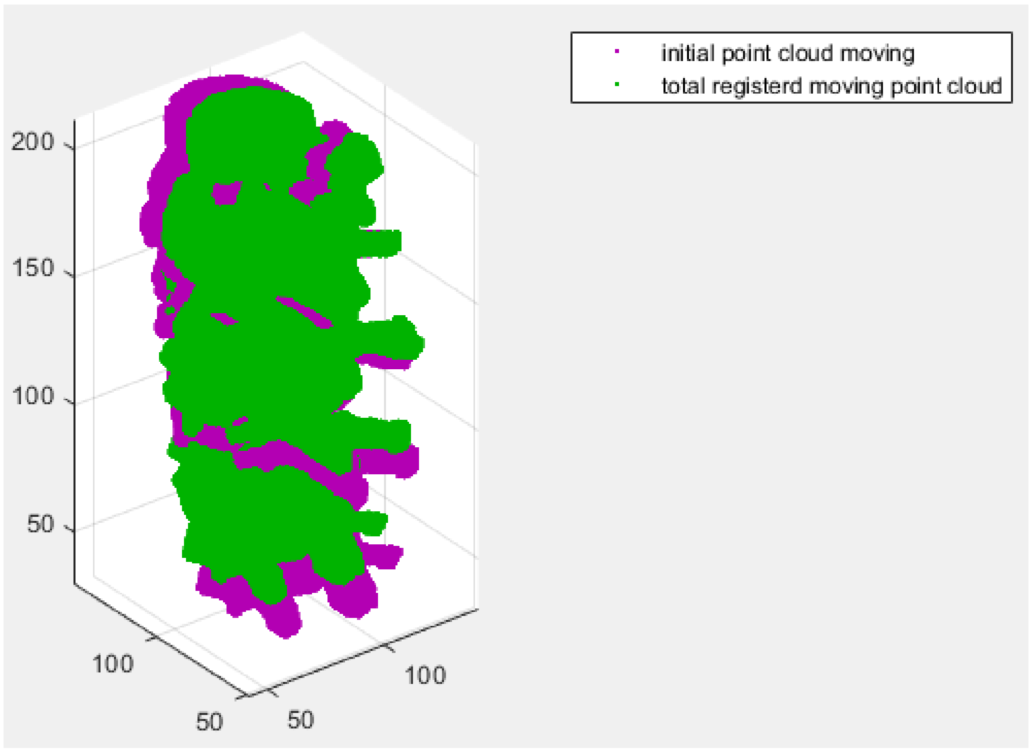
Case 1: Patient number 2



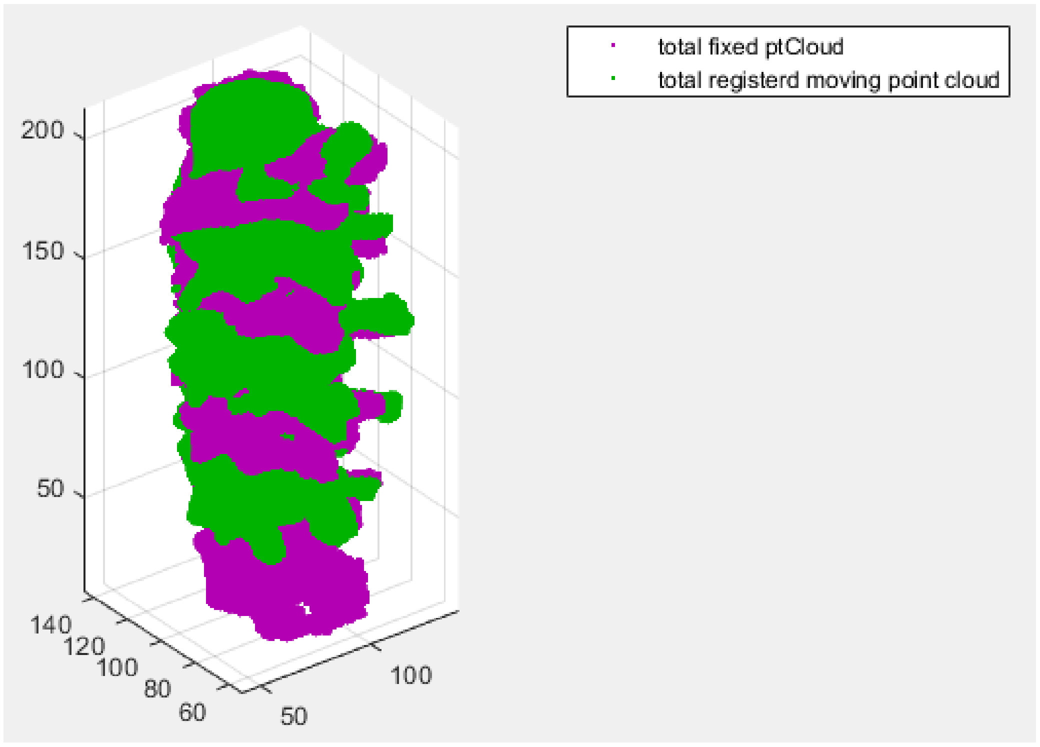
the total moving point cloud before registration



the total moving point cloud after registration

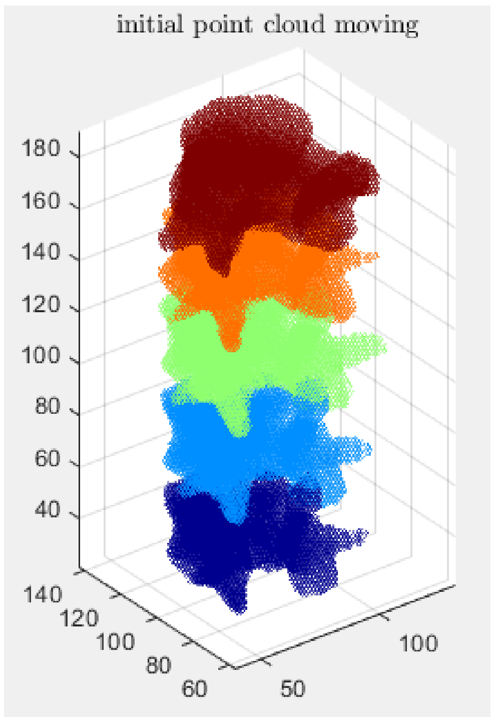


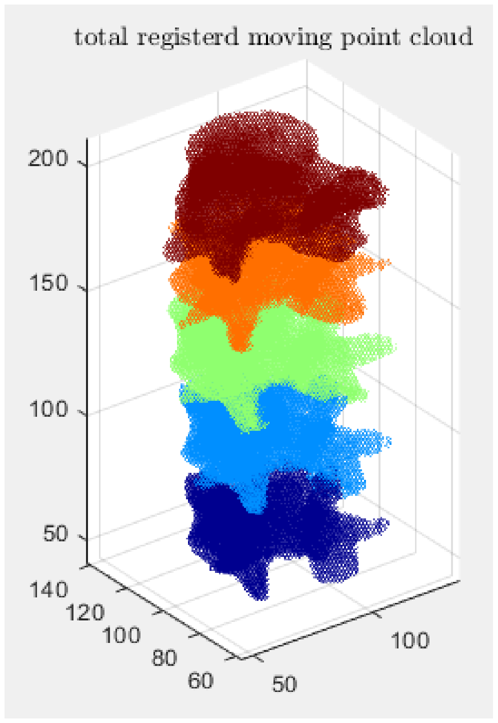
the total moving point cloud before registration vs. the total fixed point cloud



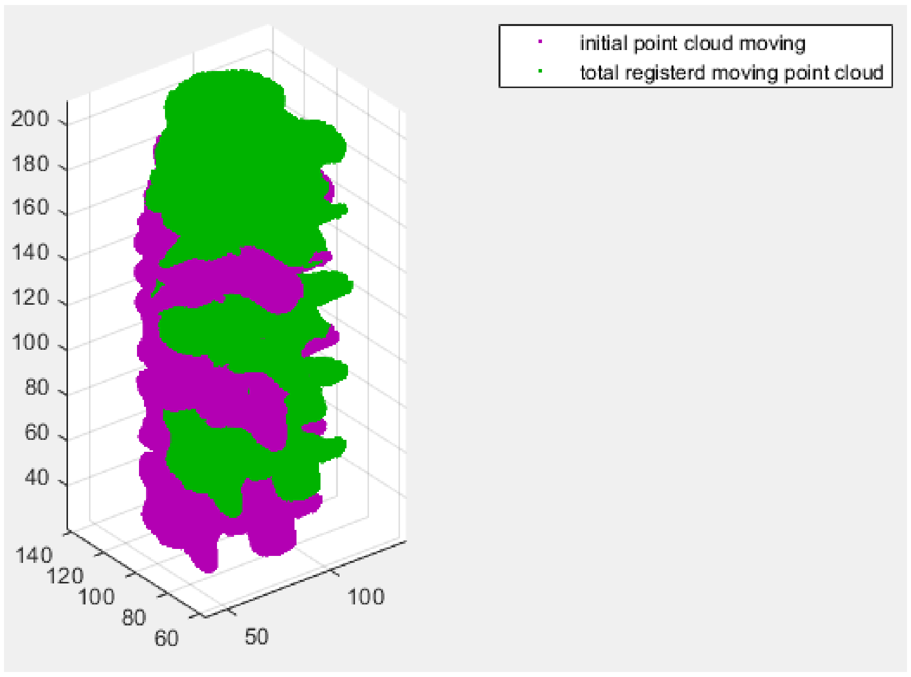
the total moving point cloud after registration vs. the total fixed point cloud

Case 2: Patient number 10

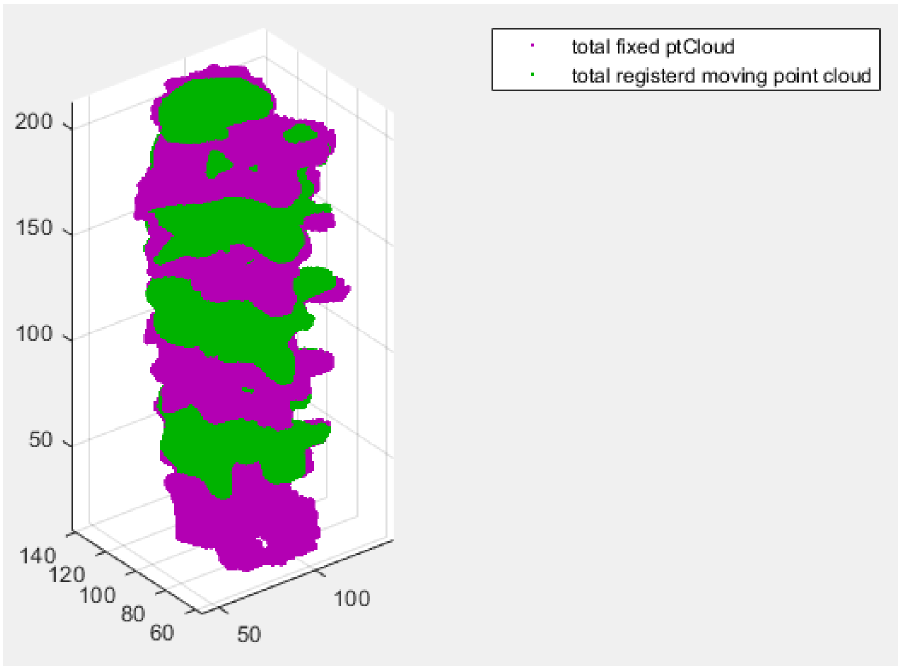




the total moving point cloud before registration the total moving point cloud after registration

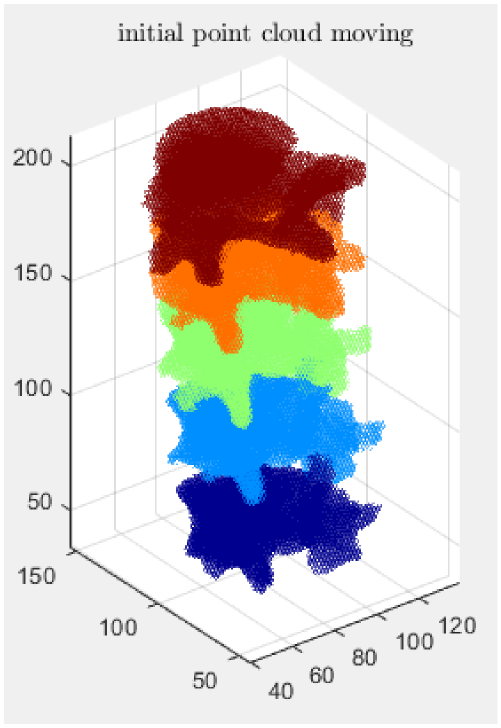


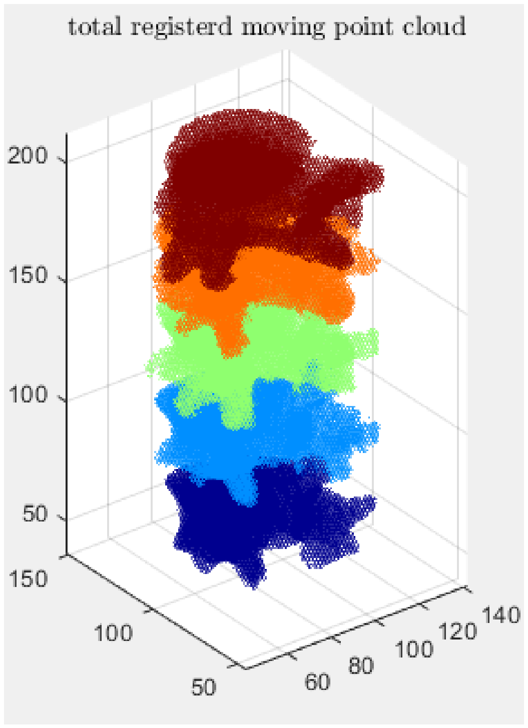
the total moving point cloud before registration vs. the total fixed point cloud



the total moving point cloud after registration vs. the total fixed point cloud

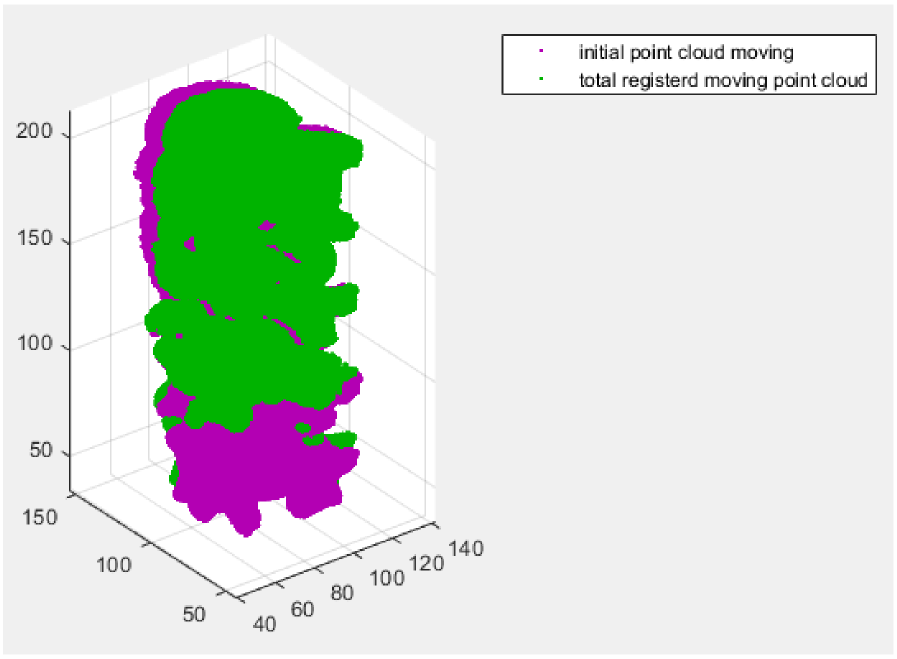
Case 3: Patient number 21



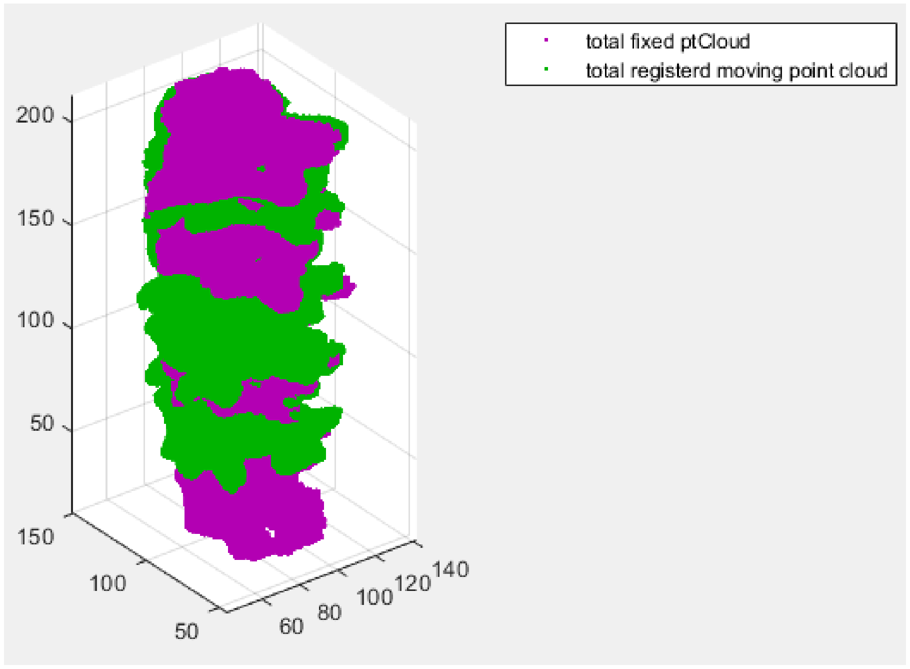


the total moving point cloud before registration

the total moving point cloud after registration



the total moving point cloud before registration vs. the total fixed point cloud



the total moving point cloud after registration vs. the total fixed point cloud

At last, I reconstucted the 3D volumetric image from itts point cloud with the "valume_reconstructor" function after registration that can be shown by Volume Vieer in Matlab version 2020.

The results of this registration are investigated by measuring the criteria mentioned in part 2 and here is the results table:

	DS	ASD	HD	Det(J(Tform))	CV (common volume)
Patient 2	0.7088	2.1342	33.0606	508.9998	362 voxels
Patient 10	0.7516	1.6543	36.4555	3.1642e3	21 voxels
Patient 21	0.6847	3.0893	35.2278	213.5846	164 voxels

4. Final Evaluation

The part 3.A and part 3.B algorithms are tested in 3 patiants of learning data and 4 patients of testing data. Here is the aggregation of the results:

Total results of part 3.A

	DS	ASD	HD	Det(J(Tform))	CV (common volume)
Patient 2	0.5937	3.6080	22.3383	0.2242	0 voxels
Patient 10	0.5956	3.1024	16.0935	5.0680	0 voxels
Patient 21	0.6178	4.2536	25.9808	0.0805	0 voxels
Patient 561	0.5297	3.3488	19.3907	4.5171	0 voxels
Patient 593	0.4709	4.5934	25.9422	3.2028	0 voxelss
Patient 605	0.6452	3.1050	19.3391	0.1400	0 voxels
Patient 631	0.5186	4.1513	24.5357	3.4487	0 voxels
AVG	0.5674	3.7375	3.5191	2.383	0
STD	0.0576	0.5531	21.9458	2.0201	0

Total results of part 3.B

	DS	ASD	HD	Det(J(Tform))	CV (common volume)
Patient 2	0.7088	2.1342	33.0606	508.9998	362 voxels
Patient 10	0.7516	1.6543	36.4555	3.1642e3	21 voxels
Patient 21	0.6847	3.0893	35.2278	213.5846	164 voxels
Patient 561	0.7186	1.9299	33.8969	1.8525e4	282 voxels
Patient 593	0.6406	3.5853	33.3766	4.4956e4	635 voxels
Patient 605	0.7091	2.2685	35.5106	229.6822	251 voxels
Patient 631	0.6637	3.1442	33	2.4794e4	180 voxels
AVG	0.6967	2.5437	34.0754	1.59e4	178.6966
SD	0.0342	0.6713	1.1963	1.3199e4	270.7143

As we can see there is no strong reason to say one of the mathods over comes the other, cause as an example, although the first method resulted into less DS and higher ASD, it compensated in HD and CV results. Both methods approximatle lead to little common volumes (note that even 300 voxels when our image size is approximately about 200x200x200 our images has the same voxel size of 1 mm^3, means this much error occurs in nearly 0.00375% of total voxels).

The Jacobian Determinant in the first method is acceptable(nearly zero), but in the second method is totally wrong. Hoever I think that this problem is beacause of the element movements in the matrix rows (maybe I should have followed the order of locations) and its not atrustorthy result.

The first method as we see some how loses in evaluation, but the second method also cannot totally make improvments. I think this failure comes from the way I aggregated the 5 vertebras DDF, may be a better interpolation method , a conditional interpolation or an other method to make a single DDF from thos five could rech to better results.

Some References

https://www.mathworks.com/

https://simpleitk.org/doxygen/latest/html/classitk_1_1simple_1_1DisplacementFieldJacobianDeterminantFilter.html

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0148483

https://www.creatis.insa-lyon.fr/Challenge/CETUS/evaluation.html

https://en.wikipedia.org/wiki/Point_cloud#:~:text=A%20point%20cloud%20is%20a,a%203D%20shape%20or%20object.&text=Point%20clouds%20are%20generally%20produced,surfaces%20of%20objects%20around%20them.