MIAP Project Report

Ali Ghavampour 97102293

1 Data Preparation

1.1 Knowing the dataset

Here are some examples from the dataset.

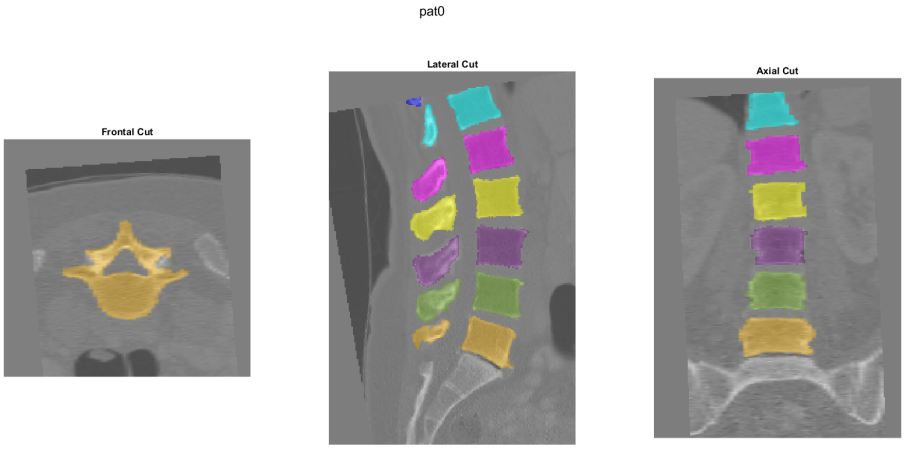

Figure 1: Atlas

Three slices can be seen. The slices are extracted based on the number of the non-zero elements in segmentaion mask.

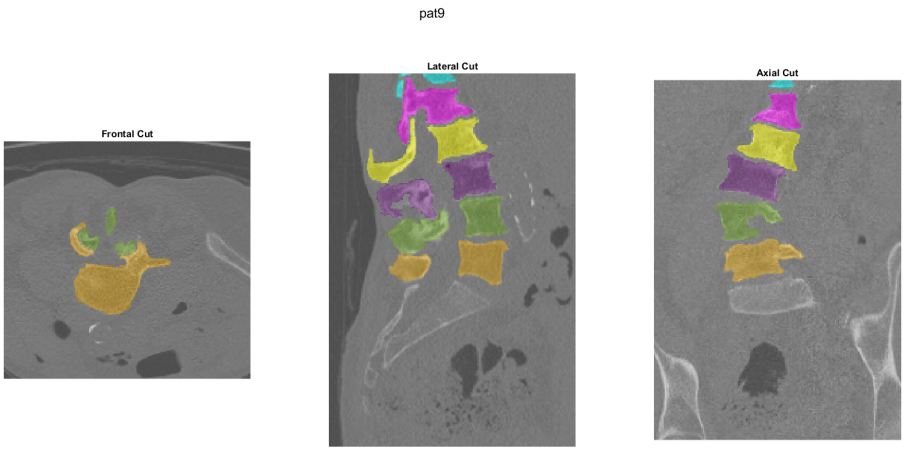

Figure 2: Patient 9

3D view of L1 to L5:

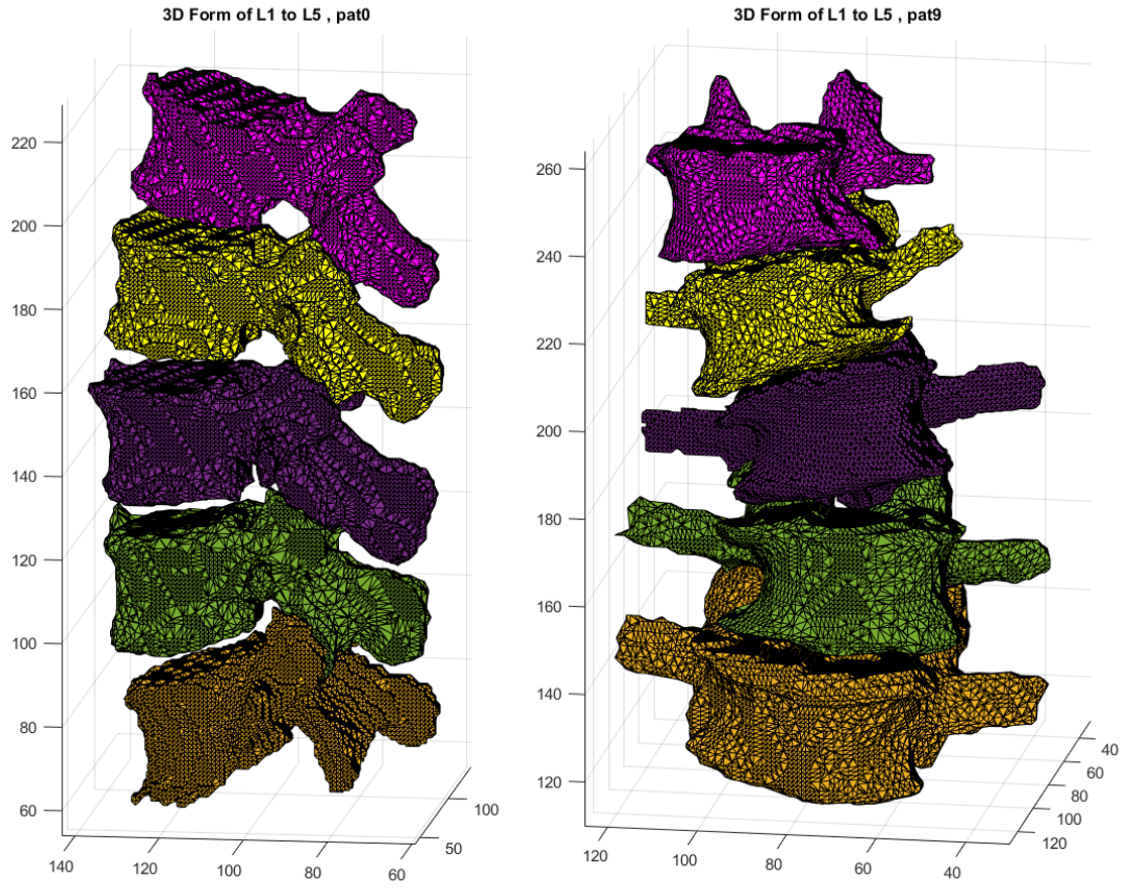

Figure 3: 3D view

For each patient different vertebras are available. Because we only want to register L1 to L5, other vertebras are removed from now on.

1.2 Generating Point Cloud

Point cloud generation in MATLAB is as simple as a function `pointCloud()`. The CPD non-rigid registeration process is cmputationally heavy and we must choose a subset of points in the point cloud. In a fast implementation of the algorithm, the calculation are reduced from O(MN) and O(M^3) to O(M+N) and O(K^3) which M and N are the number of points in the point sets.
These are related to FGT and low-rank matrix approximation respectively. We can decide our points based on the second one. If the number of points is large but they are well clustered, only a small number of eigen values are needed for the approximation. In this dataset the points are clustered beacuse of the spacing between vertebras. So, as long as we do not corrupt the general shape of a vertebra (e.g. the tails and other physiological features) we can reduce the number of points. By inspection even as low as 1024 points from the boundary and using non-uniform resampling is quite enough for the algorithm. But based on the method, most of the time 4000 or 8000 points from the boundary was used.

2 Registeration Quality Measures

2.1 Dice Score

Dice score can be defined as,

$$Dice = \frac{2|X \cap Y|}{|X| + |Y|}$$

where X and Y are binary images and $|X|$ is the cardinality of set. Also, the method can be defined as,

$$Dice = \frac{2TP}{2TP + FP + FN}$$

where TP is true positive, FP is false positive and FN is false negative.

2.2 Surface Distance

In order to define ASD and Hausdorff we must first define surface distance. The distance of one point P on surface S from surface S' is defined as,

$$d(p, S') = \min_{p' \in S'} ||p - p'||_2$$

Doing this for all the pixels we get surface distance, $d(S, S')$.
We use boundaries (edges) of the slices to calculate the surface distance between atlas and registered image.

2.3 Huasdorff Distance

We define hausdorff distance as,

$$HD = max[d(S, S'), d(S', S)]$$

2.4 ASD

We define ASD as,

$$ASD = \frac{1}{n_S + n_{S'}}[\sum_{p=1}^{n_S} d(p, S') + \sum_{p'=1}^{n_{S'}} d(p', S)]$$

where n_S and $n_{S'}$ are the cardinalities of S and S'

2.5 Intersection Volume Between Vertebras

Using MATLAB `alphaShape()` each vertebra is reconstructed from the registered points. Then the intersection voulumes are calculated. Figure 4 shows an example of the intersection volume denoted by red.

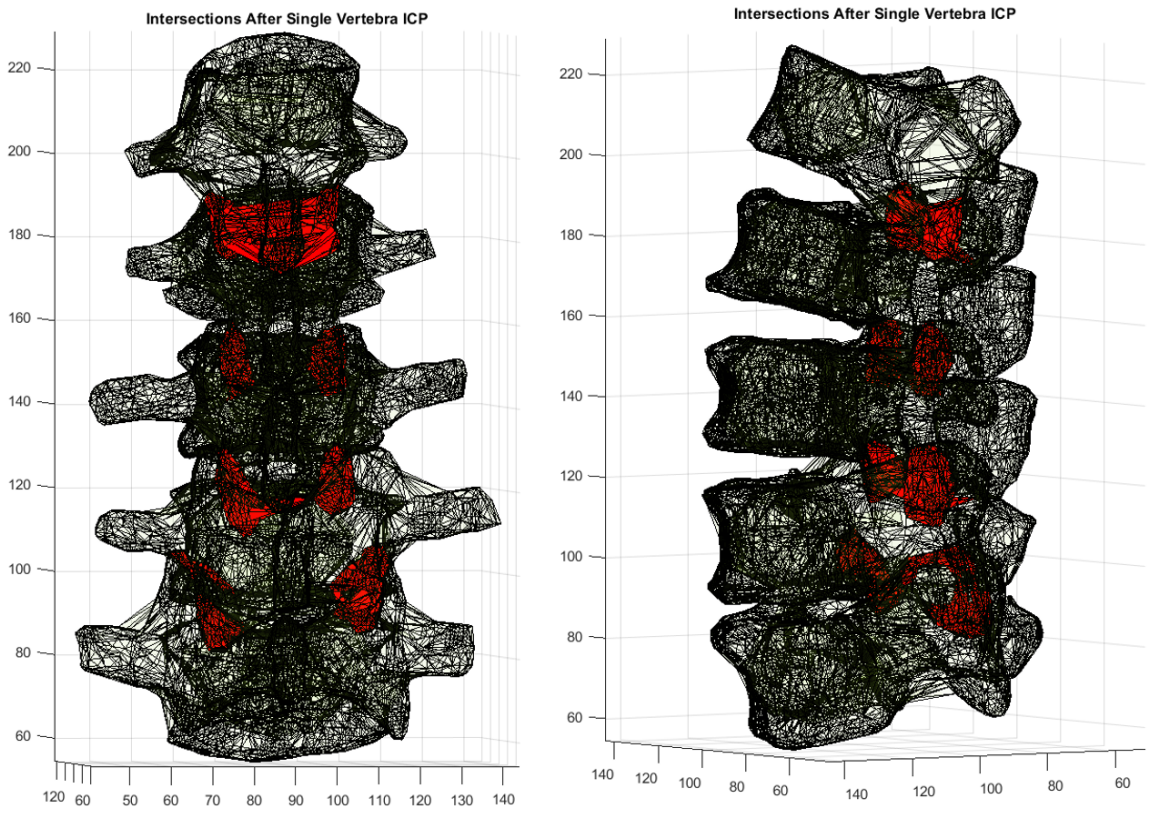

Figure 4: Red areas are the intersections

2.6 Jacobian Determinant

Assume that the transforms in each dimension is like,

$$x_t = f_1(x, y, z) \qquad y_t = f_2(x, y, z) \qquad z_t = f_3(x, y, z)$$

where, x, y, z are the original coordinates and x_t, y_t, z_t are the transformed ones. The Jacobian determinant of these functions is a measure of smoothness of the transform. What we do in this project is a different but has the same idea. As said before the algorithms are applied on a subset of points. e.g. 4096 points. It means that we only have the transformed points for those 4096 points. In order to find all of the transformed points we have to interpolate other tranformed points. After doing these steps, we either may want to fit three function f_1, f_2, f_3, or find the Jacobian numerically. Second method is used here.
After the interpolation we have an m by 3 matrix that hold the transformation vectors where m is the total number of points that are transformed. We make three matrices S_x, S_y, S_z that consist of x,y and z values of the transformation vectors in the form of original mask size. For each point in the original mask, we have a value of vector in dimensions x, y, z. Then the spatial gradients of these matrices are calculated and the numerical determinant is found. (See function `myJacobian()`)
Then the number of negative determinant values are found.

3 Registeration

6 methods are applied to the data. The methods are:
1) Whole ICP
2) Whole CPD(non-rigid)
3) Whole ICP then CPD(non-rigid)
4) Single Vertebra ICP
5) Single Vertebra CPD(non-rigid)
6) Single Vertebra ICP then CPD(non-rigid)

In the first three methods, "whole" means that the fixed and moving data consist of all the L1 to L5 points. These are the most basic ways we can register two data.
The last three are the new methods that was implemented in the project. "Single Vertebra" refers to the fact that in these methods, first L1 to L5 are seperated, then the registeration are applied to all the 5. Then, the registered vertebras are added together again.

The motivation for applying each method was:
1) A simple ICP to compare the other methods with it.
2) Same as method 1.
3) The ICP registeration is a rigid transform and sjust rotates and shifts the shapes. It actually alignes the shapes in way that rmse is minimized. On the other hand non-rigid CPD is used in this project which is able to deform the shapes. So, a combination of an alignment and then deformation may have a better result than only doing one of them.
4) The motivation for single vertebra is that we can take the labels into account and register the vertebras distinctively.
5) Same as 4 but for CPD.
6) The single vertebra approach of method 3 may seem like to be the best approah. (Although, in the results we see that it is not the case)

Here we take a look at the results from each method for different subjects.
The results are from all 4 test subjects and also patient 9 in the train dataset because its spine is deformed significantly.

3.1 Method 1 - Whole ICP

As mentioned before, it is a rigid transform. First, the outlines of the data and atlas are extracted then, without any downsampling, the data is registered to the atlas using MATLAB function `pcregistericp()`. Then, the output affine transform is applied to all the points.
The problem with this method is that first, it looks at the the whole L1 to L5 data as a single shape and does not change its shap. Only moves and rotates it. So, if there are any spinal deformations, the method will not fix it. The other problem is that if the spine shape is as same as the atlas but if single vertebras are deformed, they won't be fixed either.
Basically this method is just a simple alignment.

3.2 Method 2 - Whole CPD

Implementation is the same as method 1 but with two differences. First, `pcregistercpd()` function is used and second, the transformed version of all poitns is calculated using interpolation. By using MATLAB `griddata()` function and `'natural'` method, we interpolate the transformed points.
This method must be better than method 1 becuaes it deformes the points. But, as we will see they are not much different because it does not take single vertebras into the account.

3.3 Method 3 - Whole ICP then CPD

The implementation is a series impelemntation of ICP and CPD that the output transform points of the ICP are the inputs of CPD. The only different scheme here is the interpolation. Since we have the rotation and translation matrix of ICP, first we apply these on all the points. Then we interpolate the CPD output based on these points.
The problem that happens here is that if we want to reconstruct the transformed label matrix, we have to round the transformed coordinates to decimal values. This rounding causes lots of points to be exactly equal. So, first the intersection volume will not be precise. Second, The points in the intersection between two vertebra will have a probability of belonging to each vertebra.
All of these problems could be fixed by upsampling the data grid. (which we did not do so the problems remain)

3.4 Outputs of method 1 to 3

Method 1 to 3 are applied to patient 9. Here are the results.

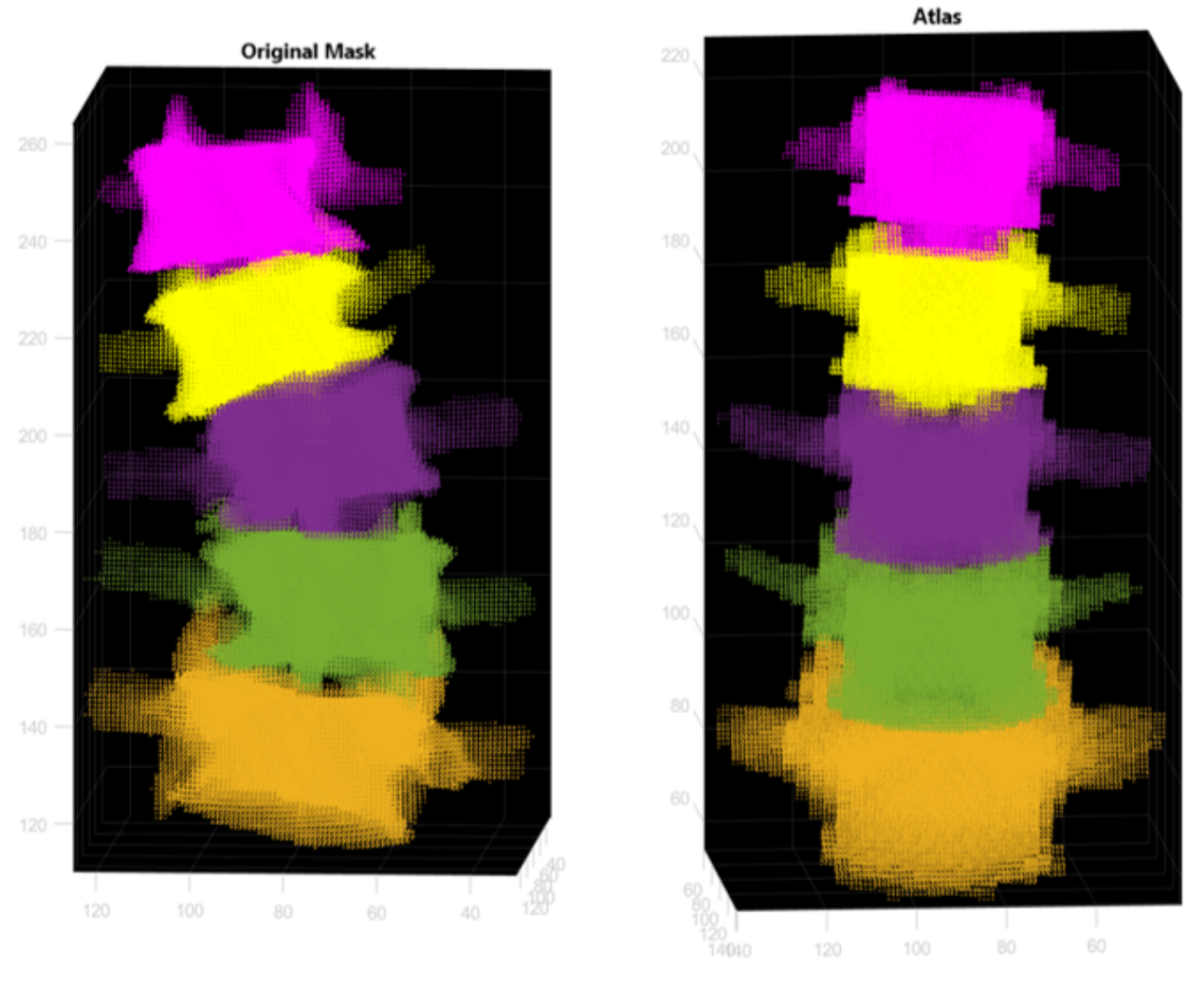

Figure 5: Point cloud with segment colors - left: Patient 9 , Right: Atlas

Patient 9 was choosed from the train dataset because of the abnormal spine shape. (In fact to show the important advantages of the "Single Vertebra" methods)

Figure 6 is the registered point clouds using the three methods.

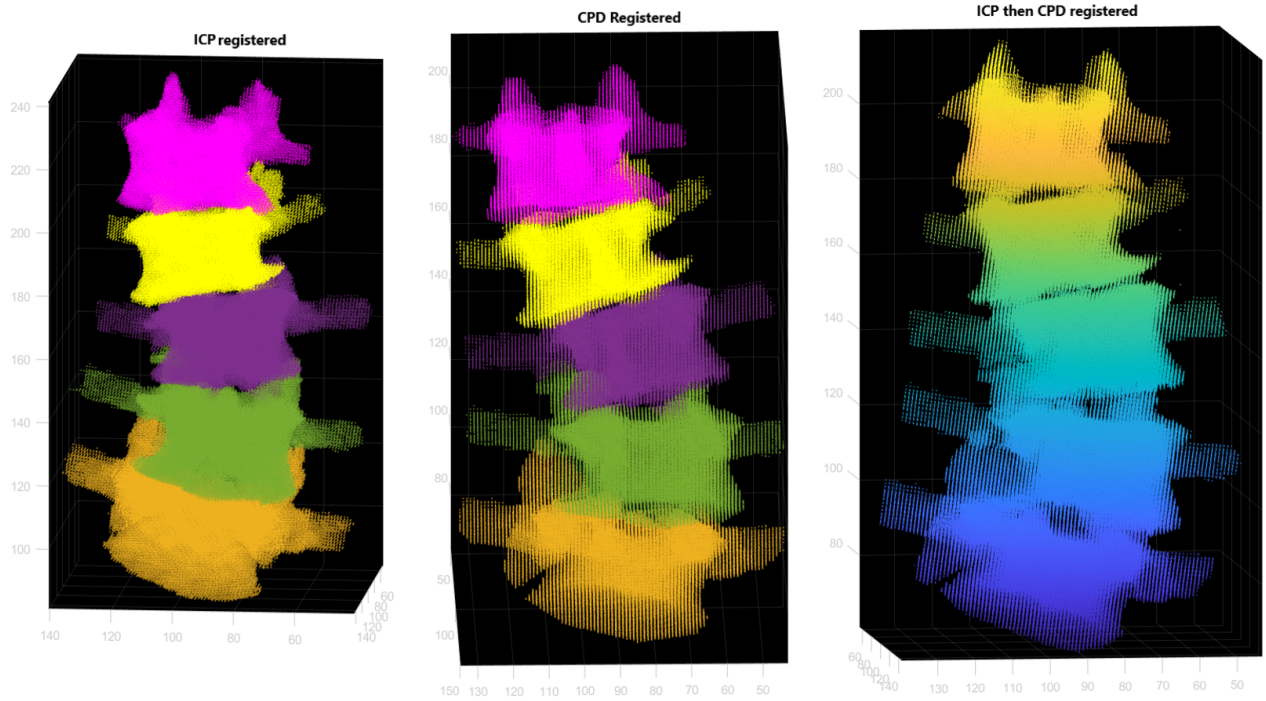

Figure 6

Because of the rounding problems of "ICP then CPD" method, it was not straight forward to find the transformed point colors. We can see that none of the methods fixed the spine abnormality.
The 2D slices are like figure 7. The slice indices are the same for all of the images and they are the best slices from atlas. (See function `extractSlice`) for more information.

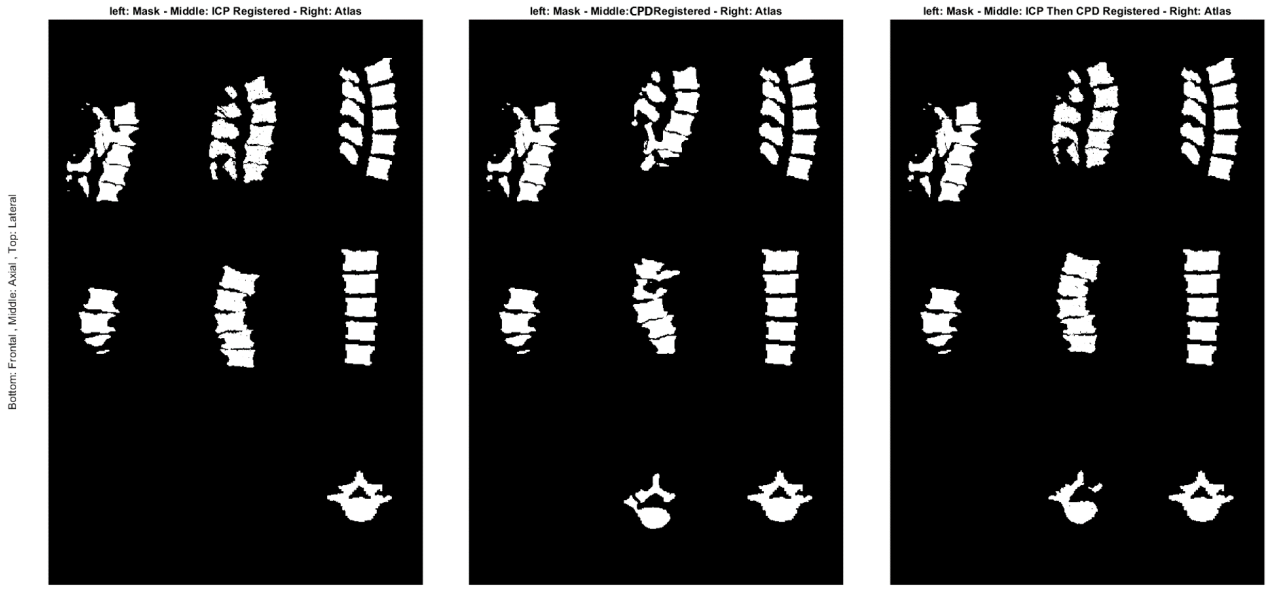

Figure 7

As can be seen from the figure, the registeration was not much helpful and it is just a simple alignment.
The fully black images in bottom left of figures are because, the extracted plane does not intersect with any of the vertebras. This tend to be fixed after registeration. Scores of each registeration are available in Results section.

3.5 Method 4 - Single Vertebra (SV) ICP

As told in the method motivation section, when we have spinal abnormalities such as in patient 9, "single vertebra" methods are expected to result better. Vertebras are better rotated and aligned when we register each vertebra to its coresponding one in atlas. This is because our problem would be simpler. We are actually breaking one problem into five smaller ones.

But now lets talk about, "what is our problem?". Is the best registeration the one that completely equals two images? Or in our case it is the registeration that only alignes the coresponding vertebras and does not change the form of vertebras themselves?

I believe the answer could be either one. Imagine the case we want to use the registeration to atlas image for the segmentation of other spine images. In this case a possible best transform would be a revertable one that compeletly fits all the moving points to the atlas. Since PCD deforms the vertebras it is able to fully fit each vertebra to its corresponding one in atlas. (Altough in the runs I did, it did not happen which could be because of the choice of method hyperparameters)

Another case could be examination of abnormal spine of a patient. In this case we may want to only align the spine with the atlas spine so that the doctor would be more comfortable and sure what he is looking at. But if we deform the vertebras, the examination could result incorrectly.

All of the discussion above was to say that both "SV ICP" and "SV CPD" could be usefull for our needs. (Also dice and surface distance scores are the best for the first problem and does not measure the quality for the second one)
In "SV ICP", L1 to L5 are first extracted using function `extractVertebra()` in both moving data and atlas. Then we register the corresponding vertebras. The output of the algorithm is 5 different Rotation and Translation matirces. So, in order to find the total displacement field we must somehow combine these transformations and find a general transformation. Because the result could be insufficient, we used a more general approach that is interpolating the transformed coordinates for all the points and then calculate the displacement field as,

displacement field = trans coord - original coord

Another way would be calculating the transformed points for all the points of each vertebra, then calculating the displacement field like above. Also, for "Single Vertebra" methods, intersection volume is calculated.

3.6 Method 5 - Single Vertebra (SV) CPD

Almost every steps even the interpolation part are the same in this method and method 4. The only difference is use of CPD algorithm instead of ICP.
It worths mentioning that CPD algorithm has three free parameters, ω, λ and β. I believe the choice of these prameters could be important in the output. But in this project, MATLAB default parameters were used.

3.7 Method 6 - Single Vertebra (SV) ICP then CPD

This method is the series combination of "SV ICP" and "SV CPD". The interpolations are also series. This method is expected to be the best overall method because it consists of a fast prealignemnt and a non-rigid transform. But as we will see it is not the case. This could happen because of the rounding problems discussed in method 3.

3.8 Outputs of method 4 to 6

Atlas and patient 9 vertebras are shown in figure 8.

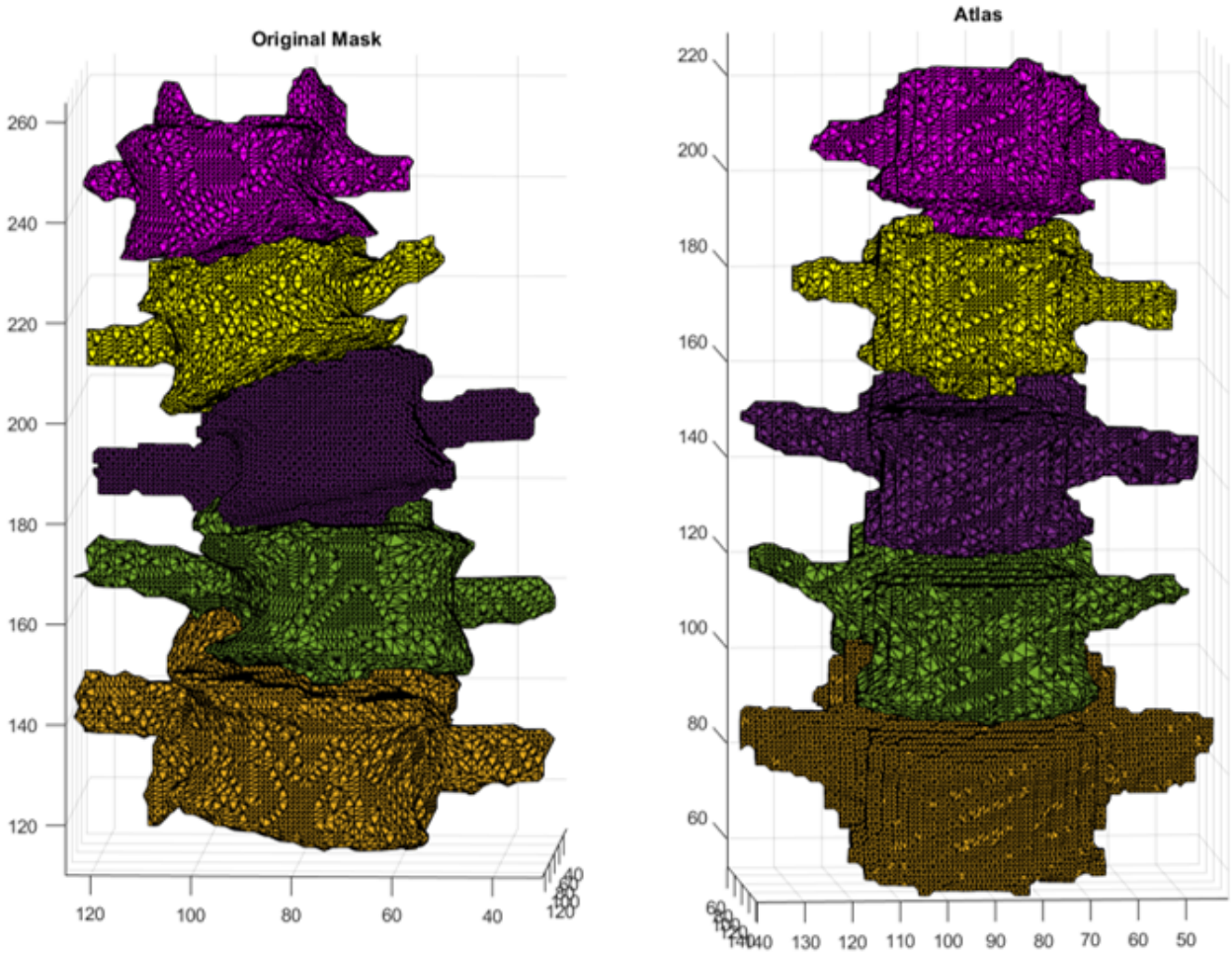

Figure 8: Left: patient 9 , Right: Atlas

This figure is made using MATLBA `alphaShape()` function.

The transformed images can be seen in figure 9

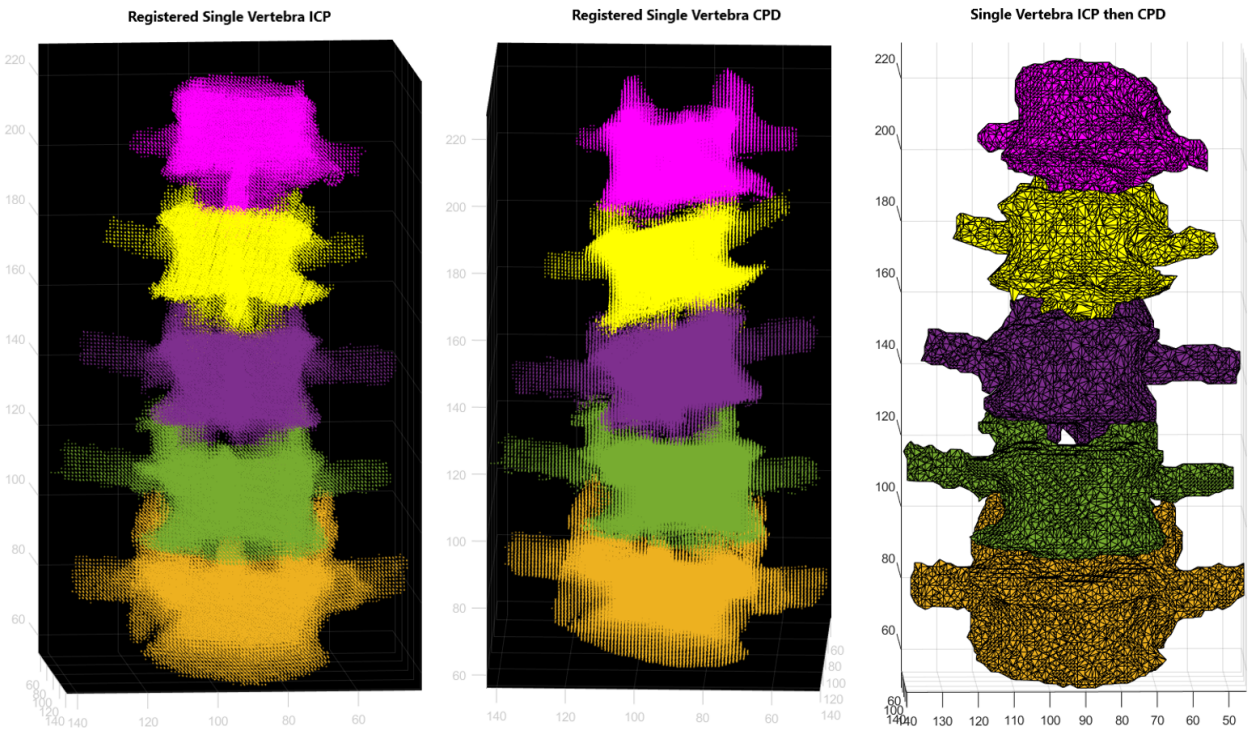

Figure 9

As told before, the spinal abnormality in the single vertebra methods has been almost removed. But, the CPD does not seem like to be more similar to the atlas. This was not what I expected. But, still it could be because of the bad algorithm parameter choice or not sufficient number of registeration points and needs more investigations.

The 2D slices are shown in figure 10.

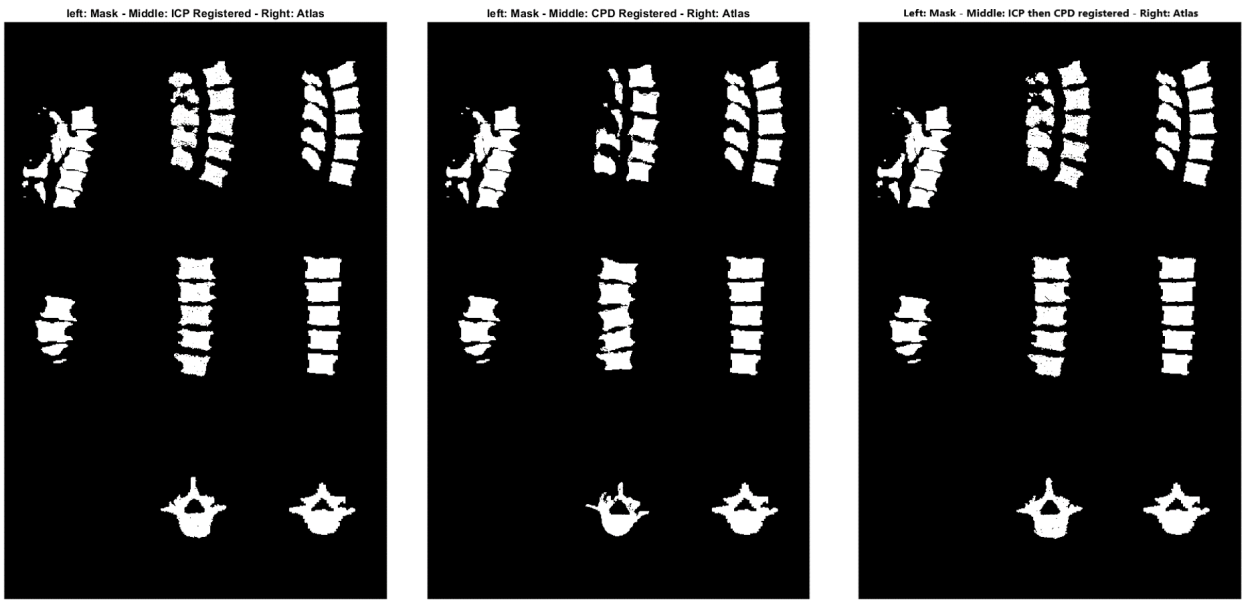

Figure 10

The small empty holes in the "ICP" and "ICP then CPD" are because of the problems in interpolation. Sometimes the output of the interpolation does not converge and we will not have any corresponding transformed coordinates for some points. This could be easilly fixed using small hole filling kernels.

Again the bottom left of the images are totally black. When this happens dice value becomes zero and surface distance can not be defined. So, we set the values of Hausdorff and ASD to 9999 when this happens.

Since we register the vertebras distinctively, they could intersect with each other. The intersection plots of the three algorithms (figure 11) show that in the case of patient 9, the most intersection belongs to "ICP" and the least to "ICP then CPD". In the result section, we will see that for all the test subjects the "ICP" has the most intersection voluem.

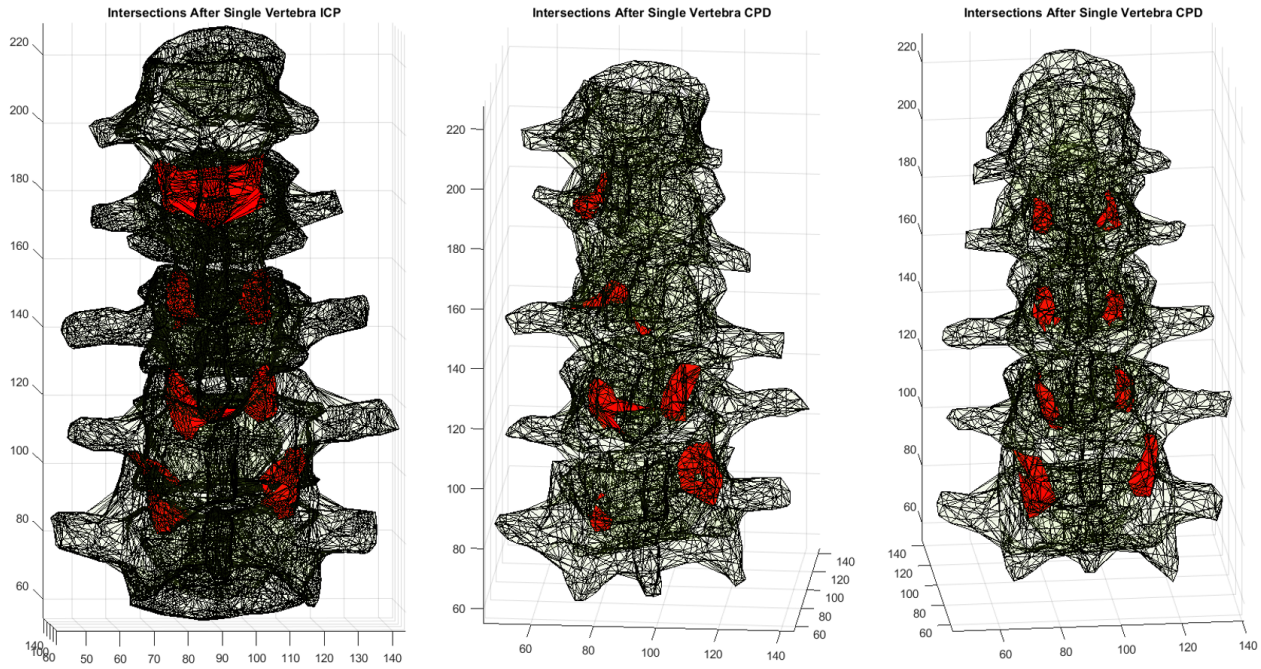

Figure 11

4 Using the code

4.1 Preparations

It is essential to follow these steps accurately.

1)
Change the name of the healthy subject data and label to ”pat0.nii” and ”pat0_label.nii”.
”00.nii” → ”pat0.nii”.
”00_mask.nii” → ”pat0_label.nii”.

2)
These folders are available with the uploaded code:
”Functions”
”Train”
Copy all the ”.nii” files to the ”Train” folder. You have to see something like figure 12 in the ”Train” folder.

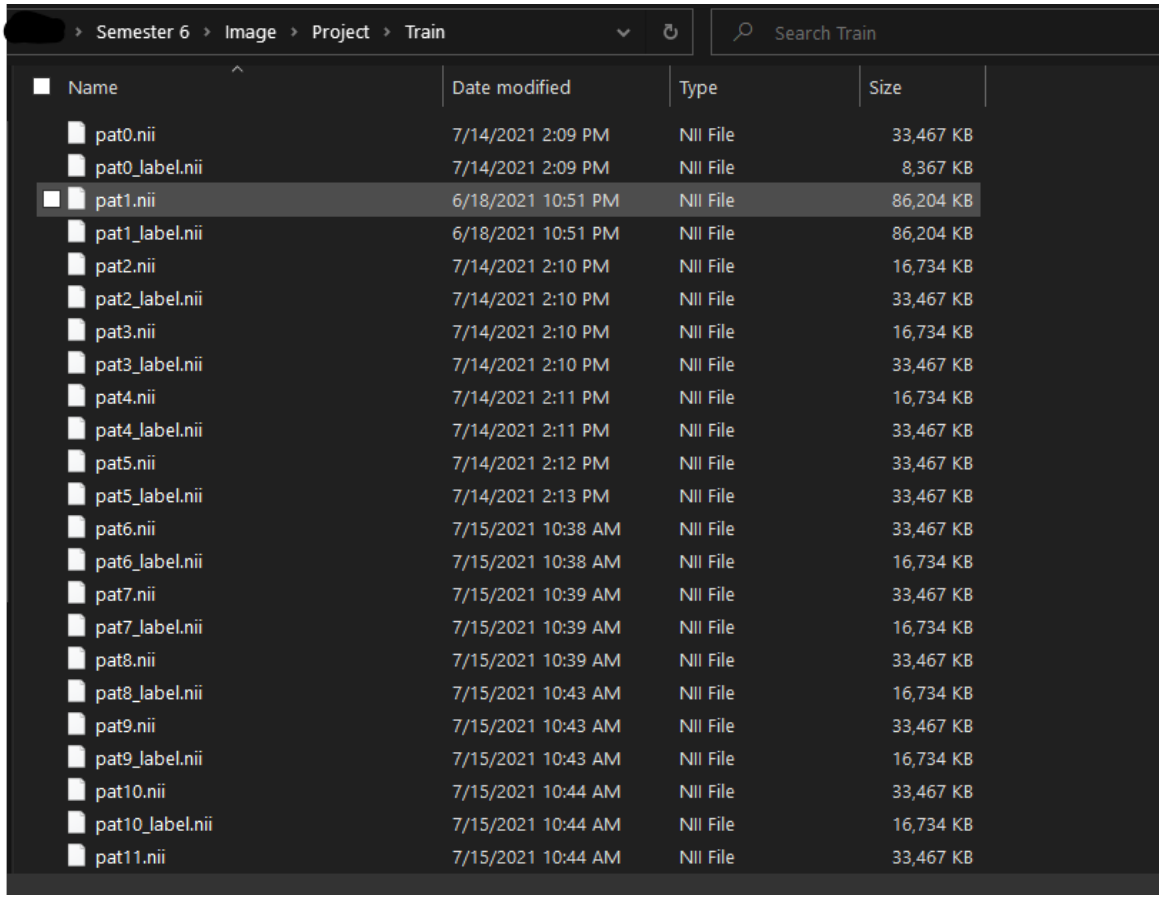

Figure 12

Also, do not change the contents of ”Functions” folder.

3)
When files are ready, change the MATLAB currnet folder to the project folder. It should look like this:

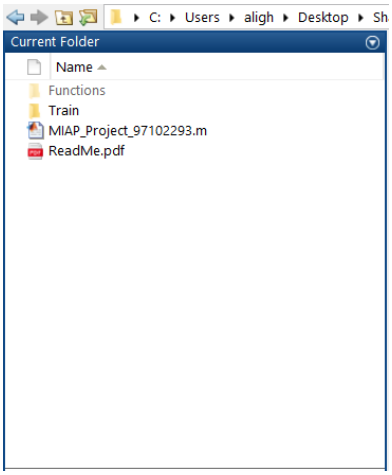

Figure 13

4.2 Running The Code

First you have to run the very first section of the code with the "addpath" functions.
Then you can run the sections one by one to see the resuslts. The only variable you have to change in the code is "fileName". You can see it in several parts of the code. It is the name of the patient you want to register to the atlas. ('pat0', 'pat2', ...)
Note that 'pat1' and 'pat20' data are not available.

Multiple methods are implemented in this project. You have to run the sections one by one. Each section may take a while to fully run. Each method has a "Score" section that evaluates the performance of the registeration.

4.3 Functions

```
[data,mask,sz] = loadData(fileName)
```

Input: e.g. 'pat9'
Output: MRI data, Segmentation mask and size of the matrices

This function loads the patient data in the workspace. Also, it will modify the value of labels. L1 to L5 will be labeled as 5,6,7,8,9 respectively.

```
verteb = extractVertebra(fileName)
```

Input: e.g. 'pat9'
Output: a cell that holds the segmentation mask of L1 to L5 vertebras.

```
[dataSlice,labelSlice,inds] = extractSlice(mask,data))
```

Input: segmentation mask and MRI data.
Output: The best slice of each cuts (frontal, axial and latera). *dataSlice* and *labelSlice* are cells that hold the slices. *inds* is the index of slices in each dimension.

```
[dataSlice,labelSlice,inds] = extractSlice_ver2(mask,data,inds)
```

Input: segmentation *mask* and MRI *data*. *inds* forces the slice indices it is not empty.
Output: The best slice of each cuts (frontal, axial and latera). *dataSlice* and *labelSlice* are cells that hold the slices. *inds* is the index of slices in each dimension.

```
[x,y,z,boundary] = outerLayer(mask)
```

Input: segmentation *mask*.
Output: *x,y,z* are vertical vectors that contain the coordinates of the boundary of the input mask. *boundary* is the 3D boundary of *mask*.

```
labelMat = ReconstructInterpMat(x_tformed,y_tformed,z_tformed,xq,yq,zq,vertebMask)
```

Input:
verterbMask: segmentation mask of only L1 to L5. Could be obtained using `extractVertebra()`.
xq,yq,zq: The coordinates of the non zero points in *verterbMask*.
x_tformed,y_tformed,z_tformed: The coordinates of interpolated transformed points. The length of the vectores must equal to *xq,yq,zq*.
Output: transformed segmentation matrix.

```
[x_tformed,y_tformed,z_tformed] = cpdInterpolation(x,y,z,regPoints,xq,yq,zq,method)
```

Input:
x,y,z: Coordinates of the points in patient mask that used in registeration.
regPoints: The registered points. Length must equal to *x,y,z*.
xq,yq,zq: The points that we want to interpolate the values in.
method: The interpolation mathod.
Output: The interpolated coordinates coresponding to *xq,yq,zq*.

```
[patFlag,pat] = isPat(pat)
```

Input: e.g. 'pat2'
Output: puts condition on some patients.

This function is not needed after TA provided the corrected data.

```
color = colorPC(mask)
```

Input: segmentation maks.
Output: a color vector that is sutiable for plotting point cloud. The color are: L1: pink , L2: Yellow , L3: Purple , L4: Green , L5: Orange

```
[d12,d21] = surfd(img1,img2,searchWin,plot)
```

Input: *searchWin* is the search window around each point. *plot* is the plot handle.
Output: caculates the surface distance $d(s, s')$ and $d(s', s)$ between two images.

```
asdScore = asd(img1,img2,searchWin)
```

Input: similar to `surfd()`
Output: caculates the Average Surface Distance measure.

```
hdScore = hd(img1,img2,searchWin)
```

Input: similar to `surfd()`
Output: caculates the Huasdorff Distance of two images.

```
det_J = myJacobian(sx,sy,sz)
```

Input: Displacement field matrices in each dimension x,y,z. Must be 3D matrices.
Output: Calculates the Jacobian determinant of displacement field.

```
jDetDisp(sx,sy,sz)
```

Input: Displacement field matrices in each dimension x,y,z. Must be 3D matrices.
Output: Figures that show the sign of the Jacobian determinant. If negative, red and if positive, blue.

* The idea of the algorithm -ICP then CPD- was developed with cooperation with Maryam Maghsoudi.