Multidimensional Signal Processing
Second Assignment Report
January/2024

NAIST
Ferreira da Silva LUCAS - 2311328 (OMI Lab)

Contents

1 Assignment 1: Statistical Shape Model – Can we guess age and sex from pelvic shape?

1.1 Background and Dataset

The provided dataset consists of 400 patient pelvis mesh data stored in individual .vtk files. The first 280 files also have a label for gender and age stored in a csv file (PatientID_list_training.csv), while the remaining 120 will be used for testing. Python will be used for this assignment, therefore, PyVista(1) was selected as a library for handling the VTK data. As instructed beforehand, the data is already aligned. The code starts with the reading of the dataset in the following way:

1. Reads patient data in the .csv file using Pandas Library
2. For each file path read, load the .vtk file
3. Create a dictionary with keys: 'id', 'mesh', 'sex', 'age', 'image'
    - 'id': Unique patient ID, ranging from 0 to 280.
        - from id 0 to id 140, patients are all male.
        - from id 141 to id 280, patients are all female.
    - 'sex': String 'M' for male and 'F' for female.
    - 'mesh': Combination of vertices and edges connecting them, making a triangle that represents the whole surface of the pelvis.
        - There are 9996 vertices stored for each patient. This is an array of shape (9996,3), where each point has a XYZ coordinates.
        - There are 19999 edges stored for each patient.
    - 'img': An image captured from the y-plane viewing direction used later for training a machine learning model.
4. Plot and save the image for a sample patient.

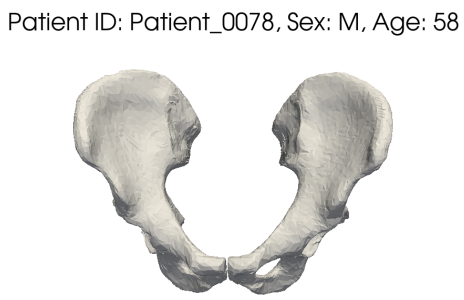

Figure 1: Sample pelvis data for one patient

1.2 1.1: Shape visualization and calculation of average shape

It is expected that male and female pelvis shape / size / features differ in some way. Since the data provided is large, we will group patients into age groups (20s, 30s, 40s, ..., 80s), each age group with 20 patients, 10 male and 10 female.

Firstly, we averaged the patient data for each age group and plotted the average pelvis for visual inspection. To compute the mean shape across the dataset, we use the following formula for each vertex coordinate:

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{1}$$

where N is the number of samples (in our case, $N = 10$), and X_i is the vertex coordinate vector for the i-th sample, represented as:

$$X_i = \begin{pmatrix} x_{1,i} & y_{1,i} & z_{1,i} \\ x_{2,i} & y_{2,i} & z_{2,i} \\ \vdots & \vdots & \vdots \\ x_{n,i} & y_{n,i} & z_{n,i} \end{pmatrix}^T \tag{2}$$

Here, $x_{j,i}$, $y_{j,i}$, and $z_{j,i}$ are the coordinates of the j-th vertex in the i-th mesh, and n is the total number of vertices. The code that computes the average shape works in the following way:

1. Reads the patient data list created before.
2. Group patients based on gender (M/F) and age group (20s, 30s, ..., 80s).
3. For all patients with same gender and age group, use NumPy to average the coordinates.
4. Store data in a dictionary with keys gender ('M', 'F') and age group ('20s', '30s', ...).

The resulting plot:

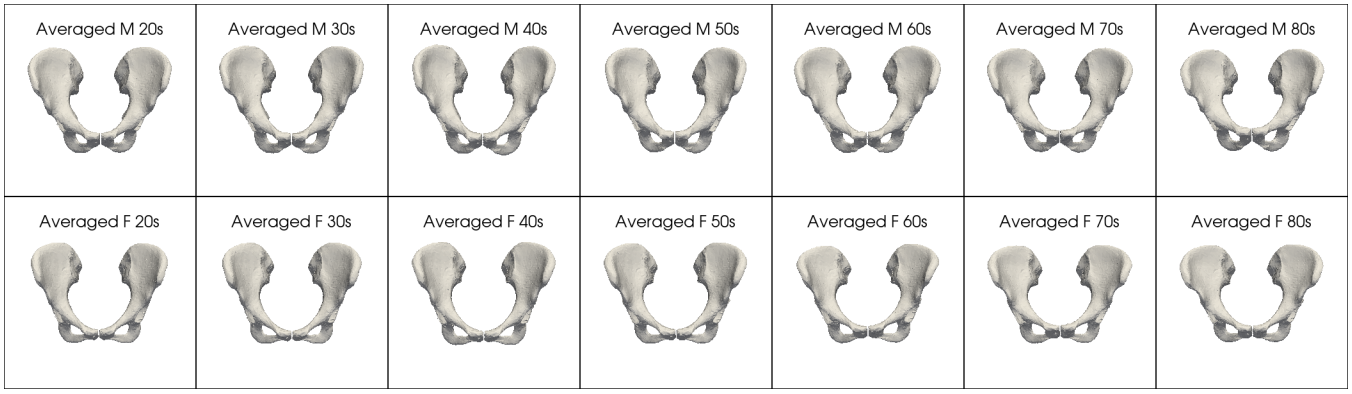

Figure 2: Averaged shape (view direction: y)

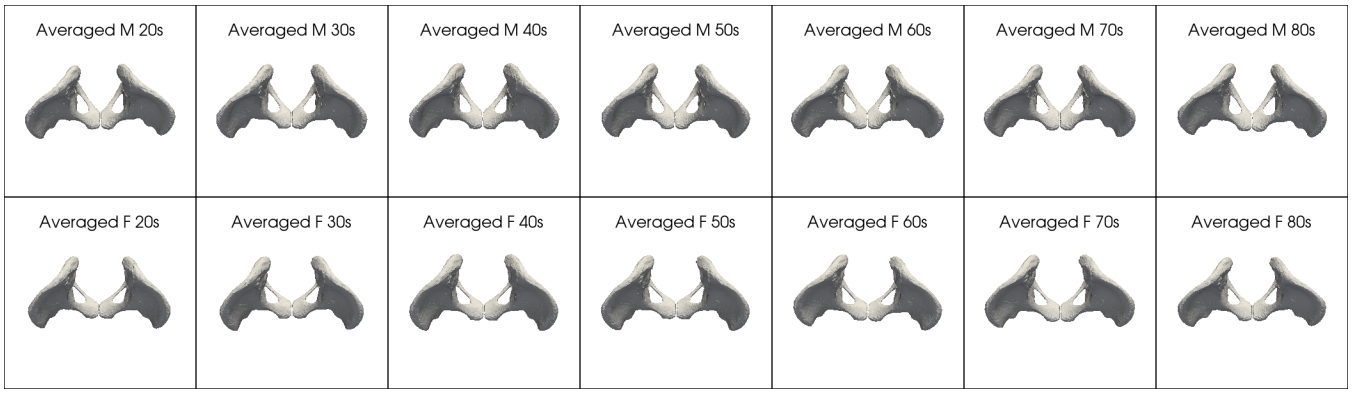

Figure 3: Averaged shape (view direction: z)

We can observe some points from these plots:

- From Figure 2, we notice that the angle in the back part of the bone is wider for female patients.
- From Figure 3, we notice that the pelvis is also wider for female subjects, as expected because of pregnancy. Furthermore, the pubic arc is also wider for female patients (in men this part reassembles a triangle).
- From Figure 4, the angle made by the back part of the bone is more acute for man.

We can also display an overlay of shapes from different ages for a gender to get a grasp at how the bone shape changes over time, as seen below:

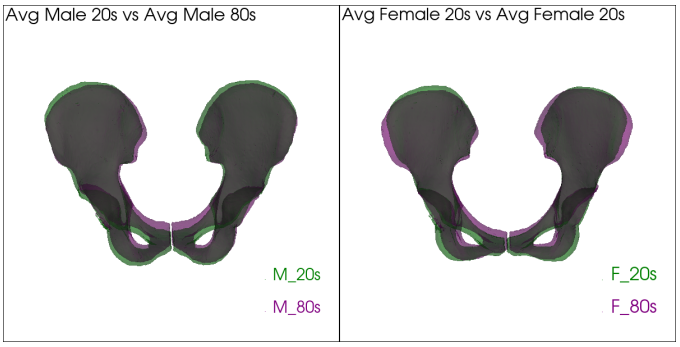

Figure 4: Averaged shape overlay (view direction: y)

From figures 6, 7, and 8, it seems that the shape get narrower with age, with more drastic changes for woman than for men(2).

To get more detailed information, we can also try to make some quantitative analysis. It is possible to compute the total area/volume of a surface directly in PyVista. We can use this information to check if is there any correlation with gender and age (what would be reasonable). The results per age group and gender are below:

From above figure we notice that both area and volume is higher for men, as expected due to physical characteristics. Also, after 40s average volume decreases continuously for woman, mostly because of menopause and osteoporosis (more pronounced than in men).

All qualitative and quantitative differences mentioned before can be explored for prediction tasks in the following sections.

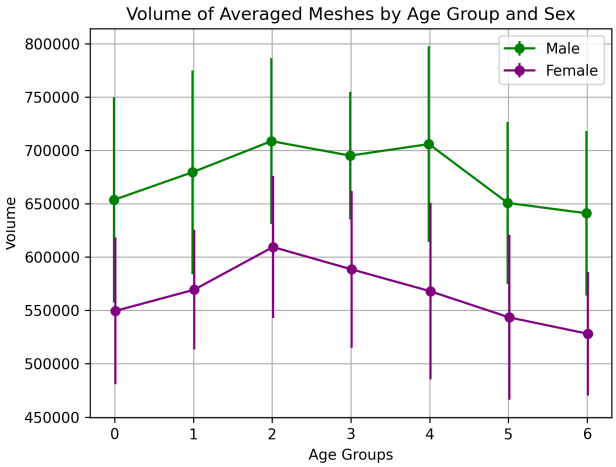

Figure 5: Volume of pelvic bone vs. age

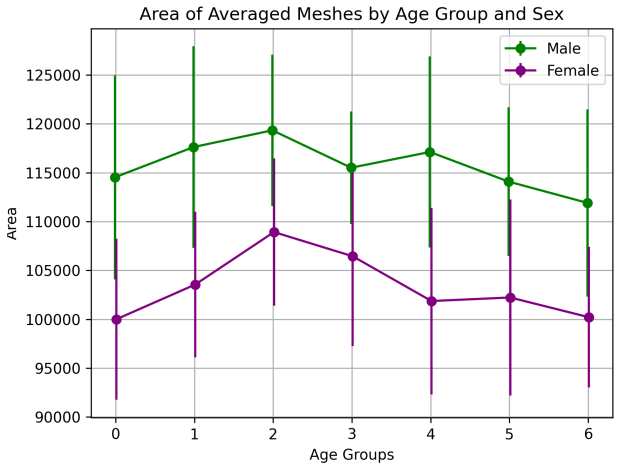

Figure 6: Area of pelvic bone vs. age

1.3 1.2: Principal Component Analysis (PCA) of shape and visualization of variation

Since our dataset is high-dimensional and dense (280 3D meshes), to better analyse and use it need some prepossessing. When working with high-dimensional data PCA (Principal Component Analysis) is often used. PCA is used to identify the primary axes of variation in high-dimensional mesh data. It reduces complexity while preserving the most significant features of the shape variation across the dataset.(3)

In the context of high-dimensional mesh data, such as the shape of a pelvic bone, each mesh can be represented as a point in high-dimensional space. PCA can reduce the dimensionality of this space while preserving the shape characteristics that contribute most to its variance.

Let $\mathbf{A}$ be the matrix that combines the column vectors X_i of the given N shapes.

$$\mathbf{A} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_N \end{bmatrix}$$

Perform PCA on $\mathbf{A}$ to obtain the principal components, visualizing the first two to analyze shape variation. In Python, this is can done using scikit-learn library by:

```
pca = PCA(n_components=num_components)
pca.fit(A.T)  # Transpose to have samples as rows
mean_shape = pca.mean_
components = pca.components_
variances = pca.explained_variance_
std_dev = np.sqrt(variances[0])
```

The matrix of principal components $\mathbf{e}_i$ and variances λ_i are computed as follows:

$$\mathbf{X}_q = \mathbf{X} + w_1\sqrt{\lambda_1}\mathbf{e}_1 + w_2\sqrt{\lambda_2}\mathbf{e}_2$$

where $\mathbf{X}_q$ represents the query shape formed by varying the weights w_1 and w_2 within the standard deviation spectrum. This means that the number of components we are selecting are the buttons we can 'rotate' to alter the global shape of the desired mesh $\mathbf{X}_q$. By selecting just two

components we have lower degree of freedom, but the relevance of these two parameters is higher. If we add more components, we can fine tune more details, at the cost of having to process more data. In the code:

```
std_dev = np.sqrt(variances[w]) # get std from w0 and w1
new_shape = mean_shape +
            sigma * std_dev[w0] * components[w0] +
            sigma * std_dev[w1] * components[w1]
```

After computing the new_shape, we just need to rebuild the mesh using the same edges provided in a sample in the dataset, since the data is aligned.

In the figures below, we can see that the higher the standard variation interval we add to the average shape, the more the resulting shape changes. Since we are using two most relevant components, it is expected that they are gender and age. In figure 10, seems like that when we move away from the average by incriminating, we move towards a more masculine bone. On the other hand, decreasing the std from the average shape, the bone tends to be more feminine (pubic arch has wider angle). From image 11, seems like when we move away from the average shape the pelvis is more narrow and deformed, maybe because of age impact.

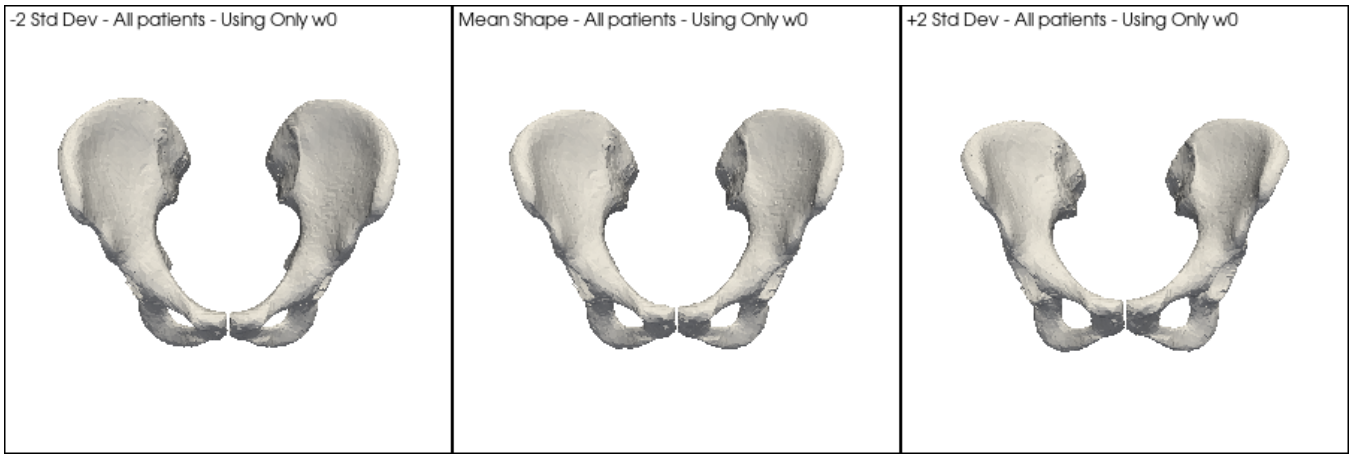

Figure 7: PCA with only component w0 applied to average shape (sigma=2)

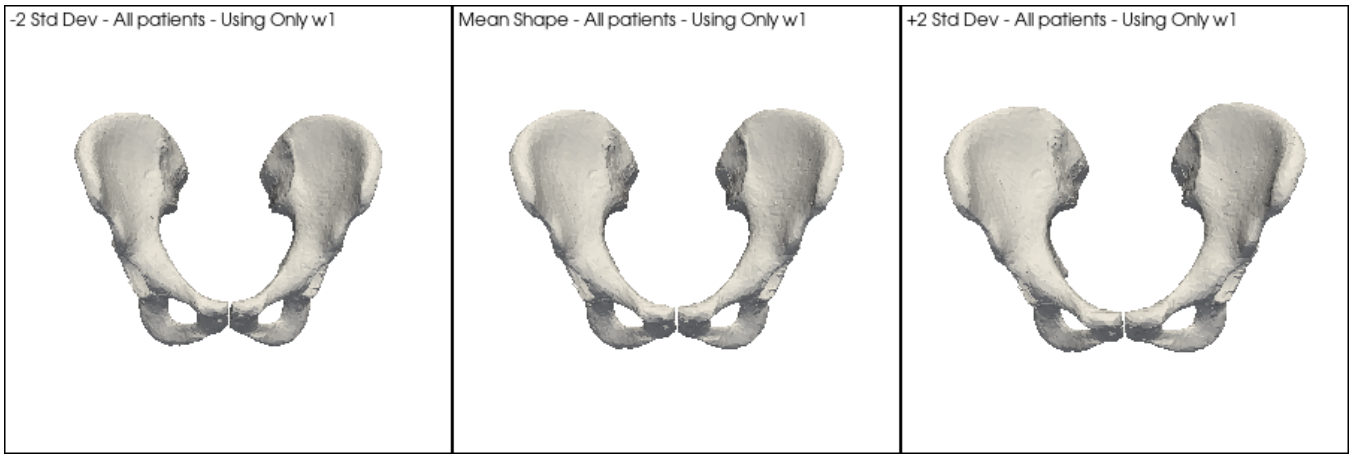

Figure 8: PCA with only component w1 applied to average shape (sigma=2)

The overall logic of the code that implements PCA is the following:

1. Reads all patient data meshes and store them in a stacked array.
2. Feed this array with 280 meshes to the PCA object from scikit-learn and get the mean_shape, components (w1 and w2, since we used only two components) and variances.
3. Select w1 and alter the average shape by a sigma factor.
4. Rebuild the mesh using the new altered shape.

1.4 1.3: Estimation of age and gender from pelvic shape

There are many features available to implement a machine learning model. Trying to fit all mesh data and cues (volume, area, etc.) into a convolutional neural network started to become computationally expensive, so a simpler approach was tried. After experimenting with Random Forest, K-Nearest Neighbors (KNN) and Support Vector Regression (SVR), the best result was achieved with a combination of algorithms following this idea:

1. Firstly, we take the mesh data (vertices data + mesh volume + mesh area) and predict the gender using a RandomForestClassifier();
2. After that, we use again the mesh data (vertices data + mesh volume + mesh area) and the predicted gender from the previous step to predict the age group the patient data belongs using another RandomForestClassifier();
3. Lastly, we regress the age using the mesh data + gender prediction + age group prediction using a Support Vector Regressor (SVR) model;

When experimenting with different models, predicting the age directly turned to be difficult and the error was high. So we tried to break the prediction into steps with specialized models for each step, using the previous predictions as cues for the next model. The idea is that it might be easier to predict the age within an age group (20s, 30s, ..., 90s) instead of doing so within the whole age interval (20 to 90 years).

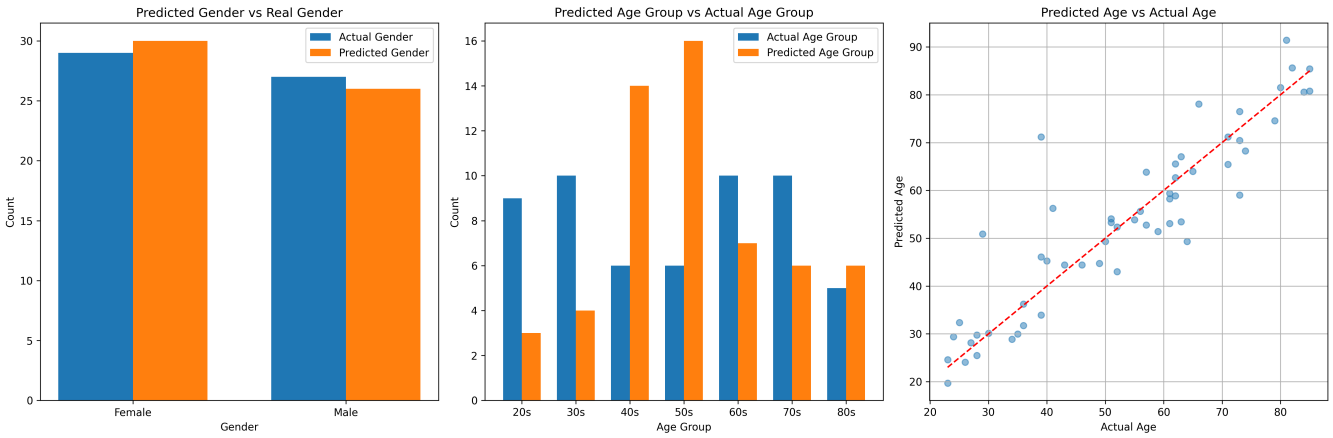

Figure 9: Training results for each step of the architecture used

In the figure above, the gender prediction has good results. Even though the age group is not too accurate, it still helps the final model predict age, with an output that is more closely related to a straight line.

All models have a similar workflow: load data, preprocess and create train-validation-test datasets, train the model and make evaluation on the test data. After training the models were used to make predictions in the unlabeled data and the results were stored in a .csv file.

1.5 1.4: Estimation of pelvic shape from partial measurement

Before working on the estimation, the incomplete data was analyzed. There seemed to be some misalignment between the incomplete shape and the provided data for training, therefore, we performed some translation of the incomplete mesh so it would be fit "inside" the training data (complete meshes) as shown in the figure below. This was done by estimating a point (picking within the image) and using this point to make one mesh be above the other so we can perform the distance calculation between each mesh to compute the estimation.

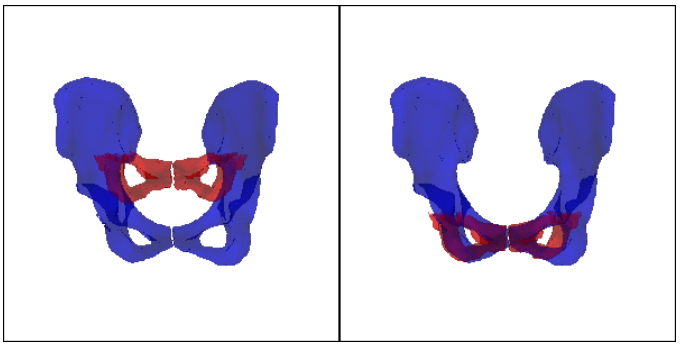

Figure 10: Aligning incomplete and complete averaged meshes

After that, we can proceed to the estimation procedure. It is divided into the following steps:

1. Firstly, we perform PCA in the dataset (280 samples) and selected 50 components;
2. After that, we use a PyTorch implementation to compute the loss function and update the 50 weights during training. The loss function takes the incomplete mesh and compares it with the average mesh produced by the PCA. In each interaction, the model updates the weights accordingly to the direction of the gradient that decreases the loss;
3. After training for several epochs, it is possible to use the optimized weights to reconstruct a complete shape from the incomplete one. The result is plotted in the figure below;

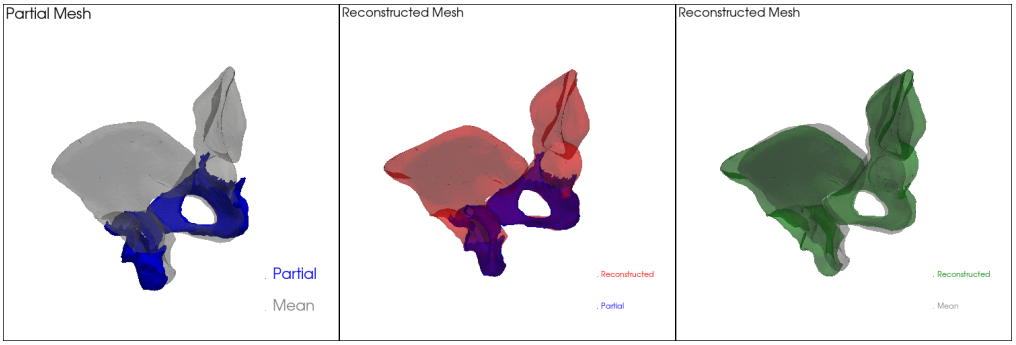

Figure 11: Reconstruction of incomplete data

2 Assignment 2: Interpolation of measured signals

2.1 Exercise 2.1: Interpolation of one-dimensional signal

For this first part, we will use Bernstein polynomials to interpolate points between known data. In this specific task, we will infer new distorted points given the 5 distorted measurements provided. This can be done in the following steps:

1. Define the function to compute the k-th Bernstein polynomial of degree n for a given point x:

$$B_{n,k}(x) = \binom{n}{k} x^k (1-x)^{n-k} \tag{3}$$

   In Python this can be implemented as:

```
def bernstein_polynomial(n, k, x):
    return comb(n, k)  * np.power(x, k) * np.power((1 - x), (n - k))

for k in range(n + 1):
bernstein_poly = bernstein_polynomial(n, k, point_normalized)
term = bernstein_poly * distorted_readings[k]
interpolated_value += term
```

2. Set the degree to be equal to the number of points (5) and calculate the polynomial for each normalized x (new point) over all k (k ranges from 0 to n);
3. Use the previous result as the weight over the the distorted measurement
4. Accumulate all weighed contributions, the result is the interpolated value
5. Repeat for all points. In our case, we will define an array with 50 points, from 0 to 5 with increments of 0.1.

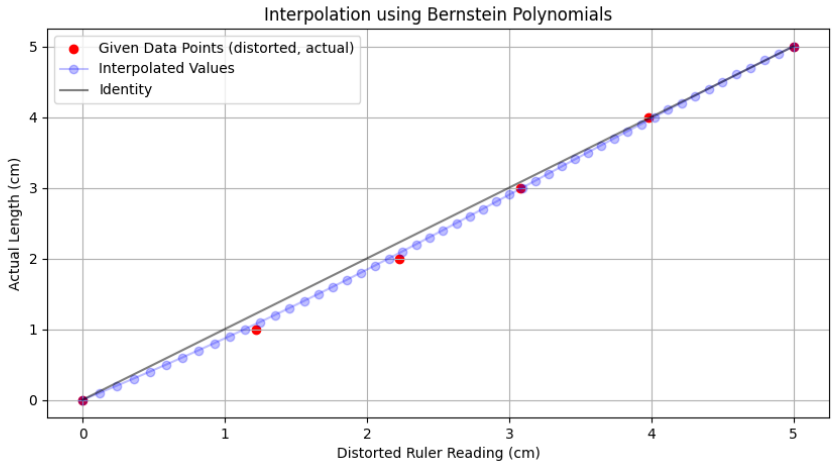

Figure 12: Interpolation result (distorted vs. actual data)

2.2 Exercise 2.2: Interpolation of two-dimensional signal (image)

Bernstein polynomials can be used as weights to account for interpolation of points (in our case, deformation) among control points (in our case, landmarks)(4; 5; 6; 7; 8). For this section, the code is structured in the following way:

1. Firstly, we load the landmarks from the CorrespondingPoint.txt file and compute the deformation:

```
deformation_vectors = []
for (x_b, y_b), (x_a, y_a) in zip(landmark_points_b, landmark_points_a):
    dx = x_a - x_b
    dy = y_a - y_b
    deformation_vectors.append((dx,dy))
```

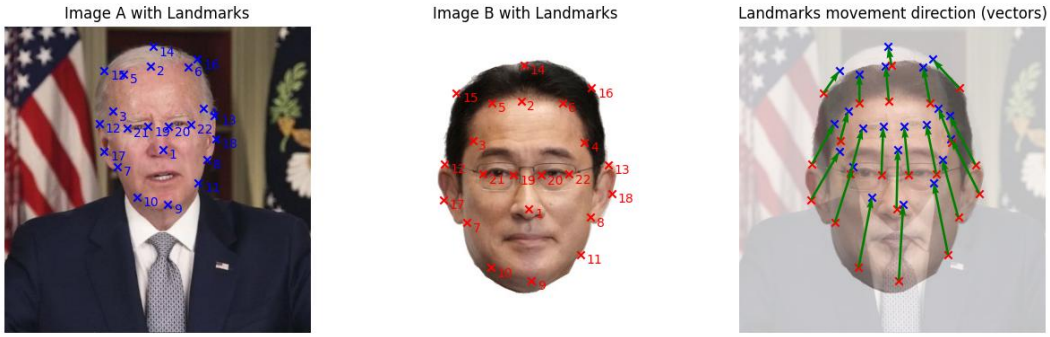

Figure 13: Images with landmarks

2. Next, we use Z-score normalization (subtract mean and divide by standard deviation) for the deformation vectors. This improved the results. We also tried different normalization, but this one seemed to best fit our problem;

3. We then define the Bernstein 2D polynomial as:

$$B_{i,j}^{n}(x,y) = \binom{n}{i}\binom{n}{j} x^{i}(1-x)^{n-i} y^{j}(1-y)^{n-j} \tag{4}$$

4. Next, we iterate over all pixels in the image and:

   - Fix x (and repeat later for y)
   - Compute the 2D Bernstein Polynomial coefficients for the current pixel position
   - Use the computed coefficient as weight to estimate the deformation in the pixel position

```
for x in tqdm(range(image_width)):
    x_value = x_normalized[0, x]
    for i in range(degree + 1):
        for j in range(degree + 1):
            b_poly = bernstein_polynomial_2d(degree,
                                             i, j, x_value,
```

```
                                                    y_normalized[:,0])
deformation_field[:, x,1] += b_poly * deformation_vectors[i][1]
```

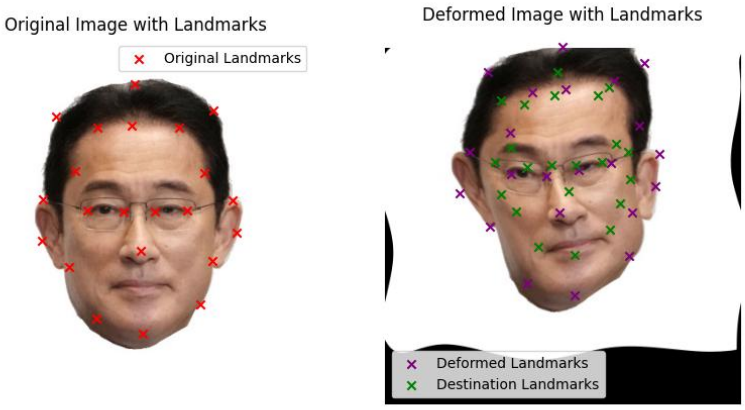

Figure 14: Deformed image b

5. With the deformation field computed above, apply each stored deformation to map the new deformed pixel position;

6. Lastly, we overlay the deformed image over the Biden image;

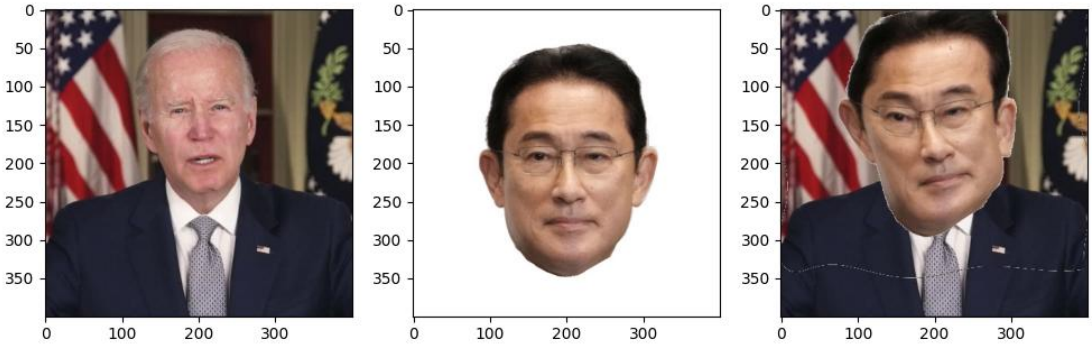

Figure 15: Images a, b and c (deformed b interpolated in a)

3 Exercise 3: Pivot calibration

3.1 Exercise 3.1: Basics of pivot calibration

For this part, we will follow the assignment instructions and some auxiliary tutorials(9). The code is divided as follows:

1. Define a rotation function that can be applied to a point in 3D space along with a translation operation to transform a point position;

2. Compute the rotation matrix and translation vector for all the data provided (.txt files) to convert points from object coordinate to camera coordinate;

3. Estimate the tip position using least squared method;

4. Check the error over different levels of angulation and noise;

The following table presents the estimation errors for various tracker files with different pivot angles and noise levels:

File Name	Pivot Angle	Noise Level	Estimation Error
pa1-1-tracker-30	30°	0mm	3.718×10^{-7}
pa1-1-tracker-05	5°	0mm	3.246×10^{-7}
pa1-1-tracker-30-noisy1	30°	0.1mm	0.07696
pa1-1-tracker-05-noisy1	5°	0.1mm	0.3967
pa1-1-tracker-30-noisy2	30°	1mm	0.3338
pa1-1-tracker-05-noisy2	5°	1mm	6.627

Table 1: Estimation errors for tracker files with various pivot angles and noise levels (4 LED's).

As expected, as the noise raises, the error also raises. Interestingly, for the angle of 5° and noise of 1mm the error is maximum (not for 30°).

3.2 Exercise 3.2: Practical pivot calibration

For this part, we will follow the assignment instructions. The code is divided as follows:

1. Read the Pj data from the provided .txt file;

2. Align both Pj and Aj data using quaternions method and recover R and t;

3. Estimate the tip position using least squared method given all the R and t(10);

4. Check the error over different levels of angulation and noise;

The following table presents the estimation errors for various tracker files with different pivot angles and noise levels:

File Name	Pivot Angle	Noise Level	Estimation Error
pa1-2-tracker-30	30°	0mm	4.308×10^{-7}
pa1-2-tracker-05	5°	0mm	3.704×10^{-7}
pa1-2-tracker-30-noisy1	30°	0.1mm	0.03039
pa1-2-tracker-05-noisy1	5°	0.1mm	0.01332
pa1-2-tracker-30-noisy2	30°	1mm	0.21969
pa1-2-tracker-05-noisy2	5°	1mm	0.38260

Table 2: Estimation errors for tracker files with various pivot angles and noise levels (20 LED's).

As expected, as the noise raises, the error also raises. Interestingly, for the angle of 5° and noise of 1mm the error is maximum (nor for 30°). Also, compared to the case of only 4 LED's, the maximum error is much lower (around 20 times loser).

References

[1] PyVista, "3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk)," https://docs.pyvista.org/, stable version, accessed: [2023/12/22].

[2] R. Slagter and M. C. DeRuiter, "Leiden - Drawing Differences between male and female pelvis - English labels," LUMC, 2019, license: CC BY-NC-SA. [Online]. Available: https://anatomytool.org/content/leiden-drawing-differences-between-male-and-female-pelvis-english-labels

[3] Carnegie Mellon University, "Principal components analysis," Chapter 18, Accessed: 2023, pDF document. [Online]. Available: https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch18.pdf

[4] R. V. Kadison, "Bernstein Polynomials and Approximation," Department of Mathematics, The Trustees of the University of Pennsylvania, 2024, joint work with Zhe Liu, accessed on 13 Jan 2024. [Online]. Available: https://www2.math.upenn.edu/~kadison/bernstein.pdf

[5] E. Weisstein, "Bernstein Polynomial," Wolfram Research, created, developed and nurtured by Eric Weisstein at Wolfram Research. [Online]. Available: https://mathworld.wolfram.com/BernsteinPolynomial.html

[6] "Bernstein polynomial," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Bernstein_polynomial

[7] G. Timmerman, "Approximating Continuous Functions and Curves using Bernstein Polynomials," Department of Mathematics, University of Washington, June 2014. [Online]. Available: https://sites.math.washington.edu/~morrow/336_14/papers/grant.pdf

[8] F. L. Martinez, "Some Properties of Two-Dimensional Bernstein Polynomials," *Journal of Approximation Theory*, vol. 59, pp. 296–306, 1989, communicated by P. L. Butzer, received June 18, 1987; revised March 9, 1988. [Online]. Available: https://core.ac.uk/download/pdf/81940855.pdf

[9] M. Clarkson, S. Thompson, E. Bonmati *et al.*, "MPHY0026: Computer Assisted Surgery and Therapy - Course Repository," GitHub repository, 2024, accessed on January 18, 2024. [Online]. Available: https://github.com/UCL/MPHY00262

[10] Washington University in St. Louis, "Lecture 7: Alignment," Online, 2024, [Accessed: January 18, 2024]. [Online]. Available: https://www.cse.wustl.edu/~taoju/cse554/lectures/lect07_Alignment.pdf