Medical Image Processing- UE 183.630

Assignment 1:

Principal Component Analysis and Shape Modelling

Computer Vision Lab
Institute of Visual Computing and Human-Centered Technology
georg.langs@tuwien.ac.at
roxane.licandro@tuwien.ac.at

Tutors: Simon Gutwein, Raphael Pruckner

SS 2025

General

This document summarizes the tasks and technical requirements for assignment 1 of the exercise *(UE) Medical Image Processing* at TU Wien in the summer term 2025.

Due: 02.06.2025, 23:59 (CET) - one submission per team via TUWEL
Total achievable points: 45

In the summer term 2025 the tasks of assignment 1 have to be implemented using Python3.9 and Jupyter Notebook, a browser-based code editor. We recommend installing Miniconda, a minimal installer for Anaconda, which installs Python as well. You can find the installer based on your OS on the official Miniconda website[1]. If you already have Anaconda or Miniconda installed on your computer, skip the installation.

Python Virtual Environment Install Instructions

In this exercise a custom Python virtual environment is required to be set up for this task. An environment contains a Python installation of a defined version and a collection of packages. Different virtual environments can be created for different applications and have to be activated before their use. Follow this step-wise guide to make sure your environment is ready and all dependencies are installed correctly:

- Use the .yml file in the setup folder to create the virtual environment. All python libraries, the correct python version and dependencies for the respective task will be installed.
  `conda env create -f [PATH_TO_ENV_FILE].yml`

- Activate the created python environment
  `conda activate [NAME_OF_CREATED_ENVIRONMENT]`

- Start the jupyter notebook file and now you should be able to start coding
  `jupyter notebook [PATH_TO_FILE].ipynb`

Read the instructions and implement the tasks in the notebook file notebook.ipynb. Use the hints and descriptions mentioned in the comments in the code and refrain from installing other libraries, not included in the environment .yml file.

NOTE: We will provide support for issues that might arise with your Miniconda setup, but if you have a valid reason not to use it or if you have problems setting up your environment in general use the TUWEL exercise forums, contact the tutor or attend the tutoring sessions.

****IMPORTANT:****
The use of any Large Language Models (LLMs), including but not limited to ChatGPT, is **strictly prohibited**. **CAUTION: We will check for plagiarisms!**
Detected plagiarization or any evidence of LLM use in completing this assignment will be considered a serious violation of academic integrity and result in extensive point reduction and

[1] https://www.anaconda.com/docs/getting-started/miniconda/main

an invitation to a formal interview to check the submission and knowledge of the group and students.

Deliverables

- Executable code as `.zip`, filename: `Submission-PCA-XX.zip`, where `XX` holds your group number.
- The python notebook `notebook.ipynb` with the requested changes, which computes all results and plots without user interaction.
- The code has to be documented and commented, explaining the reasoning behind your implementation.
- **A PDF report** that contains explanation and interpretation of all exercises (approximately 3 - 5 pages).
    - The report's front page contains group number and team member names and student registration number (Matrikelnummer).
    - The report has to be short and precise (do not include code).
    - The report can be written in German or English.

Assignment 1

Aim of the first assignment is to get familiar with *Python*, the implementation of principal component analysis (PCA) and the investigation of its characteristics. Parts of the code (data, plot functions) are provided so you can focus on the main topics of the assignment.

Helper functions (helper_functions.py)
`plot2DPCA()` and `plot3DPCA()` can be used to plot data points, Eigenvectores & Eigenvalues, ellipses & ellipsoidal and reconstructed data. plotDEMO() demonstrates the usage for the 2D case.

Useful Python Packages for this assignment
matplotlib.pyplot https://matplotlib.org/ imported as `plt` in notebook.ipynb
numpy https://numpy.org/ - imported as `np` in notebook.ipynb

Task
Maximum number of points per subtask are denoted in brackets (**45** points in total). In this exercise, data points are given in a $d \times n$-Matrix X, where n denotes the number of points and d their dimensionality, i.e. $2 \times n$ for a set of 2D points and $3 \times n$ for points in 3D.

1. **Covariance matrix**
    a. Implement a function `ourCov(X)` that computes the covariance matrix C from a data matrix X. You are not allowed to use any built-in function that directly computes the covariance matrix (such as `np.cov` for the actual implementation). However, you may use basic NumPy operations, including mean, matrix multiplication, and transpose. After implementing ourCov,

compare its result to the output of `np.cov` to validate your implementation. Note that `np.cov` expects a matrix of shape (`n_features × n_samples`), so you may need to transpose your input accordingly before using it. **(3 points)**

b. Compute the covariance matrix C for each of the data matrices: `data0`, `data1`, and `data2`. **(3 points)**

i. Visualize each dataset using `plt.scatter()` in separate figures, and ensure that the axis scale is equal by applying `plt.axis('equal')`.

ii. After computing and visualizing the covariance matrices, interpret the results. Consider what the individual entries of the covariance matrix **C** represent and identify which positions in **C** correspond to specific types of information, such as variances and covariances (or correlations) between features.

2. <u>Principal Component Analysis (PCA)</u>

a. Implement a function `pca(X,n_components)` that performs Principal Component Analysis (PCA) on a data matrix X. **(3 points)**

i. The implementation must work for data of any dimensionality (i.e., arbitrary number of features).

ii. The function should return: The Eigenvalues, sorted in descending order. The corresponding normalized Eigenvectors, sorted according to their Eigenvalues.

iii. You may use NumPy's `np.linalg.eig()` function to compute Eigenvalues and Eigenvectors.

b. Use the helper function `plot2DPCA()` to plot results for all matrices (data0, data1, data2) **(2 points)**

c. What do the Eigenvectors represent? **(2 points)**

i. Where can this information be seen in the plot?

d. What do Eigenvalues represent? **(3 points)**

i. Where can this information be seen in the plot?

ii. Is there a connection to the total variance of the data?

e. How does omitting the mean subtraction from the data matrix X affect the computation of PCA? **(2 points)**

i. Explain the consequences.

3. <u>Subspace projection</u>

a. PCA Projection and Reconstruction on 2D Data

<u>First:</u> **(2 points)**

i. Perform PCA on `data1`.

ii. Project the data onto the first principal component (main vector).

iii. Plot the projected data.

iv. What is the dimensionality of the data after projection?

Second: **(3 points)**

v. Reconstruct the data from the projection back into the original 2D space.
vi. Plot the reconstructed data alongside the original data using `plot2DPCA()`.
vii. Describe the effects of the projection and reconstruction: How do the reconstructed points compare to the original ones?
viii. Compute the average reconstruction error (e.g., mean squared error between original and reconstructed points).
ix. How would the average reconstruction error change if you did the same for `data2`.

b. Reconstruction Using the Second Principal Component (2 points)

i. Repeat the steps from part (a), but this time project the data onto the second principal component (side vector).
ii. Reconstruct the data and analyze the results.
iii. Which principal component would you choose to achieve a reconstruction with minimal error, using only one component?
iv. Justify your answer based on error and visual comparison.

4. Investigation in 3D

a. PCA and 3D Visualization (3 points)

i. Perform PCA on the `data3d`.
ii. Plot the original data points in 3D using `plotPCA3D`, along with the Eigenvectors of the covariance matrix.
iii. Describe the relationships between the covariance matrix, the Eigenvalues, the Eigenvectors and the ellipsoidal of standard deviations.

b. Projection and Reconstruction in Subspace (2 points)

i. Project the data onto the subspace spanned by the first two principal components (i.e., the top 2 Eigenvectors).
ii. What is the dimensionality of the data after projection?
iii. Reconstruct the data from the 2D projection back into the original 3D space.
iv. Plot both the original and the reconstructed data in 3D using `plotPCA3D`.
v. Discuss what type of information has been lost due to the projection.

5. Shape modeling

a. Data Exploration: The variable `bones` has the shape (`nPoints`, `nDimensions`, `nShapes`). **(2 Points)**

i. Visualize all bone shapes. Use `plt.scatter` to plot each shape in black with transparency (`alpha=0.5`). Compute and overlay the

mean shape in red. Ensure that all bones are plotted in a single figure for comparison.

b. Principal Component Analysis (2 Points)

i. Reshape the 3D shape array `bones` into a 2D matrix of shape (`nShapes`, `nPoints × nDimensions`).

ii. Perform PCA on this reshaped matrix to obtain: Eigenvectors, Eigenvalues, Mean shape

c. Implement a function `generate_shape(b, mean_shape, eigVec)` **(3 Points):**

i. `b` is a coefficient vector (`length` = number of modes used).

ii. Generate a shape by linearly combining the mean shape and a subset of Eigenvectors.

iii. Reshape the output `new_shape` back to shape (`nPoints`, `nDimensions`) for plotting.

iv. Set `b = [100, 100]` and plot the newly generated shape and the mean shape together in one plot.

d. Visualize Variation Along a Single PCA Mode (4 Points)

i. Write a function `visualize_shape_mode(bones, mode, num_samples=10)` that performs the following steps:

1. Reshapes the 3D shape array bones into 2D form for PCA with shape (`nShapes, nPoints × nDimensions`).
2. Applies PCA to extract the eigenvectors, eigenvalues, and the mean shape.
3. Plots the results: The mean shape in *red*. A series of generated shapes by varying the specified PCA mode in the range ±3λ, where λ is the standard deviation (square root of the eigenvalue) for the selected mode. The shapes should be plotted as transparent black points (`plt.scatter(..., alpha = 0.5)`) in a single figure to visualize the effect of that PCA mode alongside the mean shape.
4. Use the `generate_shape()` function internally to create the individual shapes.
5. Try different values for the mode argument and interpret how each mode affects the overall shape of the bone.

e. Random Shape Sampling and Variance Thresholding (4 Points)

i. Implement a function that generates random shapes from the PCA shape model using a random coefficient vector of the form:

```
b = np.random.randn(1, n_components) * stddevs[:n_components]
```

ii. The number of principal components (i.e., the length of `b`) determines how many PCA modes influence the generated shape.
iii. Your function should include the following steps:
    1. Compute the total variance and cumulative explained variance from the PCA eigenvalues.
    2. Determine how many components are needed to reach the following thresholds of total variance: 100%, 95%, 90%, 80%
    3. For each threshold:
        - Generate a random shape using the corresponding number of PCA components.
        - Plot the generated shape in blue.
        - Overlay the mean shape in red for reference.
        - Use equal axis scaling (e.g., `plt.axis('equal')`) and enable a grid (e.g., `plt.grid(True)`).
        - Add a title indicating how many components were used and what percentage of variance they capture.

Interpret the results:

iv. How does reducing the number of components affect the realism, detail, or variability of the generated shapes?
v. What kind of shape features are preserved versus lost at lower variance thresholds?