Medical Image Processing- UE 183.630

Assignment 2:
Bone Segmentation

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 2 of the exercise *(UE) Medical Image Processing* at TU Wien in the summer term 2025.

Due: 25.06.2025, 23:59 (CET) - one submission per team via TUWEL
Total achievable points: 55

In the summer term 2025 the tasks of assignment 2 have to be implemented using Python3.9 and Jupyter Notebook as well. Create a new virtual Python environment using the yml file with the new dependencies provided (see description assignment 1 for installation instructions of virtual environments).

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

Deliverables

- Executable code as `.zip`, filename: `Submission-BoneSeg-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 5-8 pages, including figures and references).
    - 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.

NOTE: We will provide support for issues that might arise with your Miniconda setup or questions regarding the assignment. In general, use the TUWEL exercise forums, contact the tutor or attend the tutoring sessions (dates are listed in TUWEL).

Assignment 2

Aim of the second assignment is the implementation of a segmentation algorithm for bone contours. We will investigate image features extraction and selection methods as bases for image segmentation. Three different approaches are investigated, (1) a classifier based approach using Random Forests, (2) a Deep Learning approach using a UNet and (3) a bone contour Shape Model based approach that uses PCA resulting in a simplified version of Particle Filters[1].

Relevant topics within this exercise:

- Feature extraction
- Classification and feature selection using Random Forests
- Implementation and optimization of a cost function for segmentation

Data and helper functions are provided within the provided source code and documentation:

Helper functions (helper_functions.py)

- `get_data` loads all the necessary variables that you will be working on
- `plot_shape` can be used to plot generated shapes and the given mean shape
- `plot_convolutions` plots images along with their convolutions
- `plot_prediction_triplets` plots triplets of images, segmentation predictions and segmentation ground truths
- `show_feature_importance` visualizes the importance of a Random Forest's features
- `evaluate_binary_segmentation` computes quantitative measures between a predicted and target segmentation for evaluating binary segmentation performance using a Confusion Matrix, Dice Score, Precision and Recall.
- `optimize` is used to optimize parameters with a given cost function *f*
- `plot_fitted_shapes` plots fitted shapes together with a segmentation and optionally with the ground truth landmarks

Additional provided functions - you do not need to change these

- ***training.py containing*** `train_unet_model` trains a U-Net and returns the trained model
- ***model.py*** containing the model to be trained
- ***optimize_demo.py*** containing a small demonstration of how the `optimize` function can be used

Dataset **(`handdata.mat`)**
The dataset contains the following sets:

- `images` - contains the images visualising bones
- `masks` – contains corresponding image contour masks of the bone
- `landmarks` – contains sets of landmark coordinates
- `aligned` - contains already aligned landmarks of each bone of the PCA

Useful Python Packages for this assignment
matplotlib.pyplot – imported as `plt` in notebook.ipynb
numpy – imported as `np` in notebook.ipynb
scikit-learn and scikit-image – imported as `sklearn` and `skimage` in notebook.ipynb

[1] see PDF included in the files “Particle Filters MICCAI 2004.pdf”

Task

Maximum number of points of subtasks are denoted in brackets (55 in total). Images 1-30 are part of the training data (PCA model and classification training), images 31-50 are part of the test data to evaluate your implementation.

1. **Data Exploration: (10 points)**
   - **1.1.** **(2 points)** Plot the first 5 bone images stored in `images` and the respective landmarks stored in `landmarks` on top.
   - **1.2.** **(1 point)** Plot one set of landmarks together with the corresponding set of aligned landmarks.
   - **1.3.** **(4 points)** Extend your function `generate_shape` from Assignment 1 so that it allows rotation, scaling and translation of shapes according to the parameters `scaling, rotation, x_t, y_t`(hint: ***rotation matrix***). Next to the parameters needed for PCA, the extended function should have 4 additional parameters:
     `p = (b, eigen_vectors, mean_shape, scaling, rotation, x-translation, y-translation).`
   - **1.4.** **(3 points)** Perform PCA on the first 30 landmark sets in `aligned`. Use your *group number* assigned in TUWEL for parameterizing the following transformations, apply these to the bone shapes and plot them with the mean shape using `plot_shape`.
     - rotate by ([90 + 2 * [*group number*]) degrees
     - move +(90+[*group number*]) pixels in x direction and -50 in y direction
     - scale the bone by a factor 1.5
     - use a PCA weight vector of your choice

2. **Edge Detection via Convolutions and Feature Extraction (20 Points)**
   The aim of this task is to discover filter based and non-filter based image feature extraction methods.

   2.1. Edge Detection via Convolutions (10 points)

(a) **(2 point)** Create 2 edge detection kernels (image filters) represented as matrices (`Fx`, `Fy`) of dimension `[i,j]` where i and j are 3. We will focus on the filters called Prewitt Operator. These have the aim to identify edges in images by focusing on detecting changes in intensity levels. You should implement a horizontal and vertical edge detection filter (1 point):

Horizontal Prewitt Operator `Fx`	**Vertical** Prewitt Operator `Fy`
$\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline\end{array}$	$\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$

You may also create additional kernels for edge detection, if you want to experiment with these.

(b) **(5 points)** Implement the convolution operation on your own as function `conv2d`(., .) with two input parameters (1) `image` representing a 2D image of size `[m,n]` and (2) `kernel` representing a `3x3` filter kernel. The output should be the convolved image of size `[m,n]`. For convolving the kernel at the image boundaries, image padding with 0 values should be used.

The following equation should be implemented.

$$O(m,n) = (I \oplus F)(m,n) = \sum_i \sum_j I(m-i, n-j)\, F(i,j)$$

where $\oplus$ denotes the convolution operator, I the input image of size [m,n], F the filter kernel of size [i,j] and O the output image of size [m,n]. A schematic illustration of the convolution process is also illustrated in Figure 1.

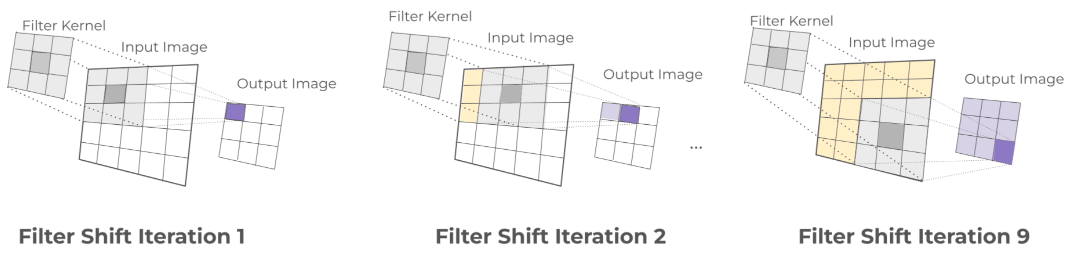

Figure 1 Schematic illustration of image convolution

(c) **(3 points)** Select 3 random bone images of the dataset `images` loaded and convolve each of them with the filters you created. Visualise the resulting images using the function `plot_convolutions`. Please discuss what you observe in the output images (2-3 sentences).

2.2. Image Feature Computation (10 points) Implement a function `computeFeatures(image)` that returns a feature matrix of size `[nfeatures × npixels]` of an image, including the following n features:

- Grey value of an image
- Gradient in x- and y- direction
- Magnitude of the gradient
- x- und y- coordinates of a pixel

Illustrate the features of the first image in `images`. Feel free to implement and evaluate other additional features!
Visualize the features for the first image in `images`.

3. Classification and Feature Selection (25 points)

In this task we investigate how image features are selected to train a classifier to solve a binary classification task - the detection of edges (bone contours) in an image.

The classification result provides per image pixel a label, where 1 denotes an edge and 0 background/no edge. In total in this tasks we will compare and explore three bone contour extraction methods: (1) Random Forest[2], (2) a U-Net[3] based Deep Learning framework (with residual connections instead of concatenations in the decoder stream) and (3) Particle Filters using a defined training and testing set.

3.1. **Data preparation:** Split the image set of 50 images and corresponding masks into a training set of 30 images (`I_train`) and masks (`y_train`) and a test set of 20 images (`I_test`) and corresponding masks (`y_test`). Compute the features of all images in store them in `X_train` and `X_test`, respectively.

3.2. **Random Forest (RF) (10 points):** Train and evaluate a Random Forest Classifier

(a) Implement the function `train_rf(I_train, y_train)` taking as input a training image dataset (`I_train`) and corresponding set of masks encoding the location of bone contours - target labels (`y_train`). The function returns the trained Random Forest classifier. Use the Random Forest classifier (`sklearn.ensemble.RandomForestClassifier`) for this purpose which is implemented in the Python package `scikit-learn`[4]. Get familiar with the parameters and select suitable ones for the training.
The RF classifier should be trained on ***extracted image features*** and not the images directly. Thus, compute image features for all input images in `I_train` and `I_test` and store the resulting features in `X_train` and `X_test`.
Hint: To speed up the training process you should use all pixels of the bone contours but only a randomly sampled subset of the background pixels (equal amount of fore- (bone contours) and background (non- bone contours)).

(b) Use the image features `X_train` as input to train a Random Forest classifier and the corresponding image masks `y_train` as class labels.

(c) Implement the function `predict_segmentation_rf` which takes the Random Forest classifier as well as an image and returns the prediction from the classifier.

---

[2] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm [last accessed 2025-05-27]
[3] https://doi.org/10.1007/978-3-319-24574-4_28 [last accessed 2025-06-03]
[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Visualize the prediction of the first test image (from I_test) and compare it to the corresponding target label from y_test.

(d) Use the trained classifier and predict bone contours for the feature test set X_test and store the predictions in y_pred_RF. Compare the bone contour predictions y_pred_RF to the target labels y_test using the helper function evaluate_binary_segmentation to get Dice Score, Precision and Recall as similarity metric.

(i) Qualitatively compare the images to the bone contour predictions and the ground truth using the helper function plot_prediction_triplets.

(ii) Evaluate and interpret the importance of different features using the helper function show_feature_importance.

- Where is the model struggling, where is it performing well?
- Alternate the input features and compare the results - leave features in the training out and discuss the feature importance of the random forest classifier features.

3.3. U-Net (7 points): Train two U-Nets – one with data augmentation and one without. Evaluate the results and compare them to the Random Forest.

(a) Train two U-Net bone contour classifiers with the helper function train_unet_model – one with the flag augment=False and the other with augment=True) **(1 point)**

Note: The U-Net classifier trains on the ***training images*** directly with the corresponding image masks as labels. The U-Net approach learns the relevant image features for classification in the training process directly, thus no precomputation of image features is required here.

The training function requires as input all images but splits them into training, validation and testing (30-10-10). After training, the function will plot training and validation loss. Compare the visualized loss-graphs with each other and discuss these. What is affecting them? Why are they different?

(b) Instead of using a predefined set of filters (as we used in Task 2.1 for edge detection), a convolutional neural network – learns the filter weights during training to extract image features in combination with convolution. In this task we want to analyse and visualize which image features are extracted by the first layer of one of the trained U-Net classifiers. Therefore, use the trained U-Net model's method show_first_layer_outputs and a test image of your choice.

Interpret what image features the learned filters are extracting (e.g. are these focusing on detecting edges or shapes.). **(2 points)**

(c) Use the trained UNet classifiers and predict bone contours for the image test set I_test and store the predictions in y_pred_UNet1 and y_pred_UNet2. Compare the bone contour predictions y_pred_UNet[1,2] to the target labels y_test the same way you did for the Random Forest. **(2 points)**

(i) Evaluate and interpret the impact of (not) using data augmentation in the training process qualitatively (visualizations – you may use `plot_prediction_triplets`) and quantitatively (similarity metrics) for the test set.

(d) Compare the performance of the U-Net to the Random Forest using the same training and testing data split for each method. Which method performs better? Are there advantages or disadvantages? Discuss it qualitatively and quantitatively and accurately describe the experimental setup, in such a way it is reproducible. **Hint:** you can use again the helper functions `plot_prediction_triplets` and `evaluate_binary_segmentation`. **(2 points)**

3.4. Shape Particle Filters (PF) (8 points): Shape particle filters are sequential Monte Carlo Methods used for solving a segmentation task. Therefore a shape model is required, which subsequently is fitted to the image to be segmented. For this task we will be using the PCA shape model of the first 30 images (created in Task 1.4 of this assignment) and create a fitting routine including the definition of a cost function used in the optimization process of the fitting. The shape model will be fitted to the output of the Random Forest segmentations.

(a) As the first step we formulate a function that models costs of fitting a shape to a target image. We are looking for a point in a parameter space (described by shape parameters, rotation, scaling, and translation) which describes an optimal fitted shape that segments the contours of a target object.
Implement the function `fit_shape_model` which should take a target image segmentation (in our case a predicted segmentation by the RF classifier in `y_pred_RF`) and the parameters `p` of the shape model as input. The implemented function should fit the shape model to the target segmentation.
- For running the optimization, use the helper function `optimize`, which takes a cost function (`cost_function`) and the upper and lower boundaries for the parameters to optimize.
- Implement the cost function which returns a scalar value that describes how well the (from p) generated shapes fits the classification result. (The better the shape fits the classification result, the lower the returned value). Describe the implemented cost function in your report. The cost function will be used by `fit_shape_model` in order to run the optimization for fitting the shape model to the segmentation.

Both `fit_shape_model` and `cost_function` will use `generate_shape` to generate shapes based on the shape parameters of the shape model and the transformation parameters used for scaling, rotation and translation. **(6 points)**

(b) Optimize this function for all segmentations in `y_pred_RF` (20). We are using a stochastic optimization approach called ***Differential Evolution***[5] for this purpose. This method is very simple, robust and converges fast. We provide an implementation of

[5] http://en.wikipedia.org/wiki/Differential_evolution

this approach in `optimize`. An example to create and use a cost function for an optimization process can be found in `optimize_demo.py`. You can simply run it to get an example demonstration of the optimization.
Visualize your results using the helper function `plot_fitted_shapes`. You may also plot the ground truth landmarks using this function.
Describe the results. Where is the optimization struggling? What could be the underlying issue? **(2 points)**