Individual Assignment : Week 8
Signal and Image Processing

March 25, 2023

This assignment must be solved and submitted **individually** and you are not allowed to work in groups. This assignment count towards your final grade and have to be submitted in order to pass the course. You must follow the report guidelines found in `guidelines.pdf`. The assignment contains questions on the topics we have covered the past weeks. The page limit for this assignment is **15 pages** including everything, i.e. illustrations and code snippets.

1. **Transformations on point clouds**

   1.1. (1 point) **Deliverables:** Explain briefly in your own words the interest of using homogeneous coordinates for general Procrustes, affine or perspective alignment.

   1.2. (1 point) Consider two sets of points $(x_i, y_i)_{i=1,..,N}$ and $(x'_i, y'_i)_{i=1,..,N}$ belonging to two different images. We assume that for all i, these points are located on corresponding shapes in the images, and intend to find the 2D Procrustes transformation (translation, rotation and scaling) that aligns the points on the source shape on to the corresponding points on the target shape.
   Deliverables: How many degrees of freedom (free parameters) are there in this problem and what should the minimum number N of point pairs be that one would need to determine these parameters? Provide an answer with arguments.

2. **Transformations on images**
   Let $I(x, y)$ be a 2D gray scale image. Consider the image $\tilde{I}$ obtained by transforming I by the combined transform **TRS** expressed in homogeneous coordinates by a uniform scaling matrix $\mathbf{S}(s)$ with $s \in (0, \infty)$, counter-clockwise rotation matrix $\mathbf{R}(\theta)$ with $\theta \in [0, 2\pi)$, and a translation matrix $\mathbf{T}(\mathbf{t})$ with $\mathbf{t} \in \mathbb{R}^2$. Using nearest neighbor interpolation do the following:

   2.1. (1 point) Express mathematically what $\tilde{I}(x, y)$ is with respect to the pixels in the image I.
   Deliverables: A mathematical explanation.

   2.2. (2 point) Write a Python function to perform the transformation $\mathbf{T_t T_c R S T_c}^{-1}$ to an input image, where $\mathbf{T_t} = \mathbf{T}(\mathbf{t})$ and $\mathbf{T_c} = \mathbf{T}(\mathbf{c})$ with $\mathbf{c} \in \mathbb{R}^2$ being the center point of the input image. We recommend that you use `skimage.transform.warp` as part of your solution, but notice that we ask for nearest neighbor interpolation and remember that we want the rotation to be counter-clockwise in the image (hint: remember that the positive y-axis points downwards in an image).
   Deliverables: Include in your report a code snippet with your function and an

example where you have transformed the image of a centred white square from the previous Assignment Week 5, Q 3.4, by scaling by $s = 2$, counter-clockwise rotation by $\theta = \pi/10$, and translation by $\mathbf{t} = (10.4, 15.7)$.

3. Transformations on point clouds

3.1. (1 point) The `BioSCAN` dataset, which you find in the files `BioSCAN_dataset_Train.csv` and `BioSCAN_dataset_Test.csv`, contains a set of 261 *fly wings with anatomical landmark points*. An example of the landmarks overlayed on top of an image of a fly wing can be see in Fig. 1 – notice that here we do not use the image for anything. The landmark points are ordered and can be visualised as closed curves, which we will do in these questions. Each wing is represented by 32 anatomical landmark points (x_i, y_i), which are stored in a vector of the form $(x_1, y_1, x_2, y_2, \ldots, x_{32}, y_{32})$. Using the first wing from the training set as a target, apply Procrustes transformations to optimally align the remaining wings in both the training and test sets to the target wing. See appendix below for how to load the datasets.
Deliverables: Include code snippets for your procrustes alignment in the report. Plot the first ten wings, which includes the target wing, before and after alignment to verify your alignment.

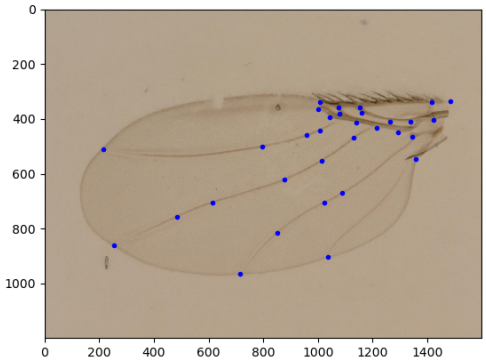

Figure 1: The first fly wing in the `BioSCAN_dataset_Train.csv` dataset with the landmarks overlayed on top.

3.2. (1 point) The fly wings belong to 15 different species (taxa) – see the appendix below for how to get the species class label. You will now use your favorite classifier to predict species from shapes as follows (if you have never used classification algorithms in Python, see the Appendix for how to run the kNN classifier from `scikit-learn`). For the *unaligned* fly wings:

- Train your classifier on the training set (found in `BioSCAN_dataset_Train.csv`)
- Use the trained classifier to predict species for the fly wings in the test set (found in `BioSCAN_dataset_Test.csv`).
- Compute and report the classification accuracy on the test set.

Next, repeat the experiment for the *aligned* fly wings from the previous question, and report the classification accuracy on the test set.
Deliverables: Explain briefly how you put together your solution including

choice of classifier and relevant parameter settings. Report the classification accuracy on the test set for the unaligned and the aligned fly wings and comment on the results.

3.3. (1 point) In the above experiment, you have removed some information from the fly wing dataset and used the remaining information for prediction.
Deliverables: What did you remove, and what did you keep?

4. **Scale-space feature detectors**

Let us use the following notation for linear scale-space partial derivatives of order $m+n$ at scale σ of a 2-dimensional image $I(x, y)$,

$$L_{x^m y^n}(x, y, \sigma) = \frac{\partial^{m+n} L(x, y, \sigma)}{\partial x^m \partial y^n} = \frac{\partial^{m+n} G(x, y, \sigma)}{\partial x^m \partial y^n} * I(x, y) \ , \tag{1}$$

where $G(x, y, \sigma)$ denotes the 2 dimensional Gaussian kernel and $*$ as usual denotes convolution.

In the following, you will implement the multi-scale Harris corner detector using linear scale-space theory and scale normalized derivatives. Recall from the lecture that the scale normalized Harris corner measure (setting $\gamma = 1$ in the γ scale normalized derivatives) is defined by

$$R(x, y, \sigma) = \sigma^4 \left(\det(\mathbf{A}) - \alpha \mathrm{trace}(\mathbf{A})^2 \right) \tag{2}$$

for $\alpha > 0$ and the structure tensor $\mathbf{A}$ is defined by

$$\mathbf{A}(x, y, \sigma) = \begin{pmatrix} G(x, y, k\sigma) * L_x^2(x, y, \sigma) & G(x, y, k\sigma) * (L_x(x, y, \sigma) L_y(x, y, \sigma)) \\ G(x, y, k\sigma) * (L_x(x, y, \sigma) L_y(x, y, \sigma)) & G(x, y, k\sigma) * L_y^2(x, y, \sigma) \end{pmatrix} \tag{3}$$

for $k > 0$. This means that for a fixed scale σ, you can think of $\mathbf{A}(x, y, \sigma)$ as a 2 dimensional image where each pixel has a 2×2 matrix as its value, or as a 2 dimensional image with 4 channels (actually we only need to store 3 channels due to symmetry of $\mathbf{A}(x, y, \sigma)$).

Detecting Harris corner points in scale-space is done by finding local maxima in scale-space (x, y, σ) of the scale normalized Harris corner measure eq. (2).

4.1. (2 point) Write a Python function that implements the multi-scale Harris corner detector using the scale normalized Harris corner measure eq. (2). You are to implement this yourself - not use `skimage.feature.corner_harris` or similar implementation of the Harris corner measure or detector. You are allowed to use parts of your solution or the reference solution from Assignment 6. Hint: You can consider using `skimage.feature.peak_local_max` to find spatial maxima, but remember that you need to find points that are simultaneously maxima across space (x, y) and scale σ. Apply your function to the `modelhouses.png` image for different choices of the parameters α and k in your implementation (Hint: Try $k \geq 1$, $\alpha = 0.05$, and other values $0 < \alpha \leq 1$). You also need to choose an appropriate range of scales to sample the scale space with and we recommend using the following python code to generate appropriate scales for this image

```
# Logarithmically spaced scale levels
scale_levels=30
sigma=np.logspace(0, 5, scale_levels, base=2)
```

Deliverables: Provide code snippets, an explanation of your solution, including your choice of range for and sampling of the scale σ, and illustrate your solution for different parameter settings by plotting points and circles indicating detection location and scale (as we did in Assignment 6). Including the 350 points with the largest $R(x, y, \sigma)$ value in the illustrations should give a good result, but you are free to apply other thresholds as long as you clearly state your choice in the report.

5. Histogram based segmentation

5.1. (1 point) **Deliverables:** Give 3 reasons why simple intensity-based segmentation (i.e. global thresholding of image intensity) is problematic. Sketch examples, using both text and illustrations, where problems can occur.

5.2. (1 point) **Deliverables:** Explain briefly in which situations edge-based segmentation is likely to perform better than intensity-based (global threshold) segmentation. In what situations would the performance worsen?

A Training and evaluating kNN

For loading the dataset we use the Python package called Pandas which can be installed in either way depending on your Python installation.
For an Anaconda installation:

```
conda install pandas
```

For a pip based installation:

```
pip install pandas
```

To load the datasets containing both coordinates and species class labels you can use the following function:

```
def loadData(filename):
    """Read the data in filename and returns a tuple with first element
       being an 261 x 32 array with the all the fly wings along the
       first axis and landmark coordinates along the second axis and
       second element being an array with class labels.
    """
    X = pd.read_csv(filename, usecols=list(range(2, 66))).values
    XL = pd.read_csv(filename, usecols=[1]).values.flatten()
    return X, XL

# Lets read the training set and plot the first wing shape
XTrain, XTrainL = loadData("BioSCAN_dataset_Train.csv")
plt.figure()
plt.plot(XTrain[0, 0::2], XTrain[0, 1::2], 'b.')
plt.figure()
plt.plot(XTrain[0, 0::2], XTrain[0, 1::2], 'b')
```

If you have never used a classifier before, you can test a kNN classifier using sckikit-learn using the code snippets below.

- First, load the necessary modules and the data.

  ```
  import numpy as np
  from sklearn.neighbors import KNeighborsClassifier

  XTrain, XTrainL = loadData("BioSCAN_dataset_Train.csv")
  XTest, XTestL = loadData("../BioSCAN_dataset_Test.csv")
  ```

- Next, train the classifier on the training data:

  ```
  knn = KNeighborsClassifier()
  knn.fit(XTrain, XTrainL)
  ```

- Next, run the trained classifier on the test data and compute the test accuracy

  ```
  Pred_labels = knn.predict(XTest)
  acc = np.mean(Pred_labels==XTestL)
  ```