Transformations and Shape
Signal and Image Processing

February 22, 2019

This is an individual assignment.

1. Transformations on point clouds

1.1. 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 objects and intend to find the 2D Procrustes transformation (translation, rotation and scaling) that maps the source points on the target ones. How many free parameters are there in this problem and what should be the minimum value of N that one would need to determine these parameters?

1.2. The `diatoms` dataset, which you find in the files `SIPdiatomsTrain.txt` and `SIPdiatomsTest.txt`, contains a set 780 of *diatom outlines* in the form of closed curves. Each diatom is represented by 90 landmark points (x_i, y_i), which are stored in a vector of the form $(x_1, y_1, x_2, y_2, \ldots, x_{90}, y_{90})$. Using the first diatom from the training set as a target, use Procrustes transformations to optimally align the remaining diatoms in both the training and test set to the target. Plot a few examples before and after alignment, along with the target diatom, to verify your alignment.

1.3. The diatoms belong to 37 different species (taxa), which you find in the files `SIPdiatomsTrain_classes.txt` and `SIPdiatomsTest_classes.txt`. 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 un-aligned diatoms:

- Train your classifier on the training set (found in `SIPdiatomsTrain.txt` with class labels in `SIPdiatomsTrain_classes.txt`)
- Use the trained classifier to predict species for the diatoms in the test set (found in `SIPdiatomsTest.txt` and `SIPdiatomsTest_classes.txt`).
- Compute and report the classification accuracy.

Next, repeat the experiment for the *aligned* diatoms from the previous exercise, and report the classification accuracy. Comment on the result.

1.4. In the above experiment, you have removed some information from the diatoms and used the remaining information for prediction. What did you remove, and what did you keep?

You could have removed a different set of information by aligning using a different class of transformations. What would the effect be of using a larger class of transformations, including for instance nonlinear transformations? What would the effect be of using a smaller class of transformations?

2. Transformations on images

2.1. **Translations.** Let $I(i, j)$ be a 2D image. Consider the image $\tilde{I}$ obtained by translation of I by one pixel to the right.

2.1.1. Express what is $\tilde{I}(i, j)$ with respect to the pixels in image I.

2.1.2. Write the matrix corresponding to this transformation.

2.1.3. It is possible to formulate this translation using a linear filter. Write the linear filter kernel corresponding to this transformation.

Create a zero-valued image of odd size in x and y containing a centered white square.

2.1.1. Write a Python function to translate this image from any specified integer number of pixels in both directions (you are not allowed to use a built-in Python function or package to do the translation). Discuss the different boundary conditions that you can put and what they do.

2.1.2. Extend your function to handle non-integer translations, using nearest neighbor interpolation. Include an illustration of your original image, as well as the image you get by translating by $(0.6, 1.2)$.

2.2. **Procrustes transformations.** As above, let $I(i, j)$ be a 2D image. Consider the image $\tilde{I}$ obtained by scaling $s \in (0, \infty)$, rotation $t \in [0, 2\pi)$ and translation $t \in \mathbb{R}^2$. Using nearest neighbor interpolation,

2.2.1. Express what is $\tilde{I}(i, j)$ with respect to the pixels in image I.

2.2.2. Write a Python function to perform the Procrustes transformation above. Include in your report an example where you have rotated the image from the previous exercise 45 degrees, translated it by $(4.6, 2.7)$ and scaled it by 0.7.

2.3. Explain briefly and in your own words the interest of using homogeneous coordinates for general Procrustes, affine or perspective alignment.

A Training and evaluating kNN

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 = np.loadtxt('SIPdiatomsTrain.txt', delimiter=',')
XTest = np.loadtxt('SIPdiatomsTest.txt', delimiter=',')
XTrainL = np.loadtxt('SIPdiatomsTrain_classes.txt', delimiter=',')
XTestL = np.loadtxt('SIPdiatomsTest_classes.txt', delimiter=',')
```

- 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 = sum(Pred_labels==XTestL)/len(XTestL)
```