Laboratory Sheet – Shape Representations

Computer Vision CMP-6035B / CMP-7026B

Dr. David Greenwood

February 2022

Aims and Objectives

The aim of this lab exercise is to illustrate the steps involved in building the Elliptical Fourier Descriptors (EFDs) presented in the lectures. As you go along this exercise, you will need to refer to (Kuhl and Giardina 1982) and the lecture notes.

During the first exercise you will:

- visualise the contour described by the chain code
- build the EFD from the chain code representation
- see how different parameters affect the shape of the Fourier representation
- analyse the code and compare it with the paper and the lecture content

In the second part of the lab, you will complete the Procrustes Analyses algorithm presented in the lectures.

Exercise 1

The file `EFDtoolbox.zip` contains the Matlab EFD toolbox. Unzip it and make EFDtoolbox the working folder in Matlab. Inside you will find several functions. First, open `example1.m`. The script contains the chain code description of a contour. It also calls the function which plots the contour and then it calls another function which plots its Fourier representation. Run the script and have a look at both contours.

Now let's have a look at what happens in the `plot_chain_code` function. Open the file and look inside. The function does not do much except calling `calc_traversal_dist` function. Have a look inside this function.

The function calculates the x and y distances according to two equations which at the first sight look a bit odd. Note that these equations were not given in the lectures, but they can be found in the paper on the second page (original page 238). Make sure you understand these equations. Talk to the lab demonstrator if necessary.

Now, you understand how the traversal distance is calculated so let's have a look at the `plot_fourier_approx` function. Again, it does not do much except calling `fourier_approx`. Have a look at the code of this function.

It takes four parameters: chain code (ai), number of harmonic elements (n), number of points for reconstruction (m) and whether the shape is to be normalised. We did not discuss normalisation at the

lecture so we will ignore this part of the code for the time being.

The function `fourier_approx` calls the `calc_harmonic_coefficients`. Open this function and see how the harmonics are calculated. This should correspond to what was given to you at the lecture. Have a look how the traversal time is calculated in `calc_traversal_time`. Do you understand the equation it uses? Again, this is given in the paper on page 238. The remaining equations should correspond to what we have derived in the lecture which can be also found in the paper.

Now, experiment with the parameters of the `plot_fourier_approx` function. In particular, see how the shape changes as you change the number of harmonics.

Set the normalisation flag to 1 and see what happens. Note that as we said in the lectures, the contour is translated to the origin by setting the DC terms of the EFD to 0. The scale and rotation normalisation is more complex and this is beyond what we are asking you to analyse, but if you are interested you can read about it in the paper and later analyse the normalisation code in the relevant part of the `fourier_approx` function.

Experiment with the second shape in `example2.m`.

If you have any questions or problems then please ask the demonstrator!

Exercise 2

During the lecture, we have derived the formulae for aligning shapes using Procrustes' method. Matlab has a built-in function called `procrustes`. Read the reference page of this function from Matlab help. If you are inquisitive, you can see the code of this built-in function by typing `edit procrustes` in the command window. Next, execute and analyse the `procrustes_script_fillGaps` script (except the last code cell), which illustrates the use of this function.

In the next step, you will complete the implementation of the version of the procrustes algorithm as derived during the lecture. The skeleton of the algorithm can be found in `procrustes_lecture_fillGaps`. You will need to compare this file with the lecture contents and write the three missing lines of code. Next, you can test this function from the `procrustes_script_fillGaps` by running the code in the final cell. Note that also here you will need to write one line of code which is missing.

Compare the results of the two versions of the algorithm. Despite different implementations, initially they should produce the same result (green crosses will overlap and occlude the blue crosses). This is despite the fact that unlike Matlab implementation, our lecture implementation does not enforce the transformation to be a rotation. Now, increase the level of noise in the 9th line of code in the main script, until the results of the two algorithms start to diverge.

Any questions or problems then please ask the demonstrator!

Kuhl, Frank P, and Charles R Giardina. 1982. "Elliptic Fourier Features of a Closed Contour." *Computer Graphics and Image Processing* 18 (3): 236–58.