EEE3032 – Computer Vision and Pattern Recognition

Lab sheet for Week 7 –Shape Recognition

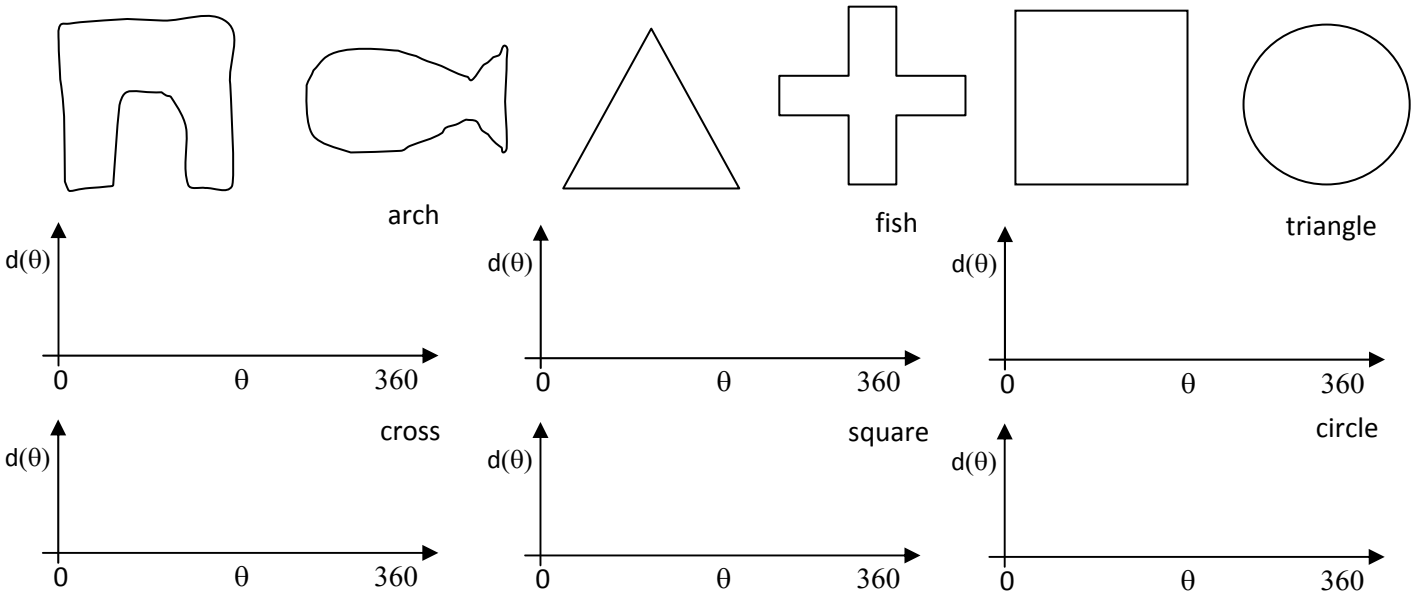
Miroslaw Bober (m.bober@surrey.ac.uk)

Task: In today's lab you will experiment with some of the methods for shape recognition covered in lectures. The learning objectives of this lab are to gain practical experience using **shape descriptors** for shape recognition, and more generally, experience in performing **supervised pattern classification** tasks using Eigenmodels.

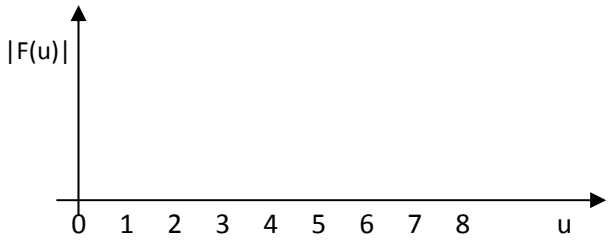
Resources: The exercises will be performed using Matlab. Download the Matlab code for the labs, if you have not done so before, and unzip the code into a separate folder. Ensure the code folder is in the Matlab search path (File -> Set Path.. -> Add with subfolders). Also download the Shape dataset (`cvprlab_shape_data.zip`). Unzip this into a separate folder and make a note of the path (e.g. `~\shapeimages`).

Ex1: Paper exercise – predict central distance Fourier Descriptors *(5 minutes)*

Consider the shapes below. **Sketch** the position of the centroid. **Sketch** a ray, at an angle θ degrees from vertical, extending from the centroid to the exterior edge of the shape. **Consider** how the length of the ray (written d(θ)) varies as θ increases from 0 to 360 degrees; **sketch** the resulting periodic signal on the axes below.



Based on your sketches above, what do you think the frequency spectrum F(u) looks like for each signal d(θ)? **Sketch** your ideas below for a few of the simpler shapes e.g. triangle, square, cross, circle. (u=frequency).



Ex2: Compute central distance Fourier Descriptors *(5 minutes)*

Run the program `FourierDescriptorDemo`. Sketch one of the above shapes by left-clicking the mouse to draw a polygon. Right-click the mouse to specify the final point on the polygon.

The system will compute the first 16 central distance Fourier descriptors of the shape you sketch. Compare the results to the graph you sketched in Ex1.

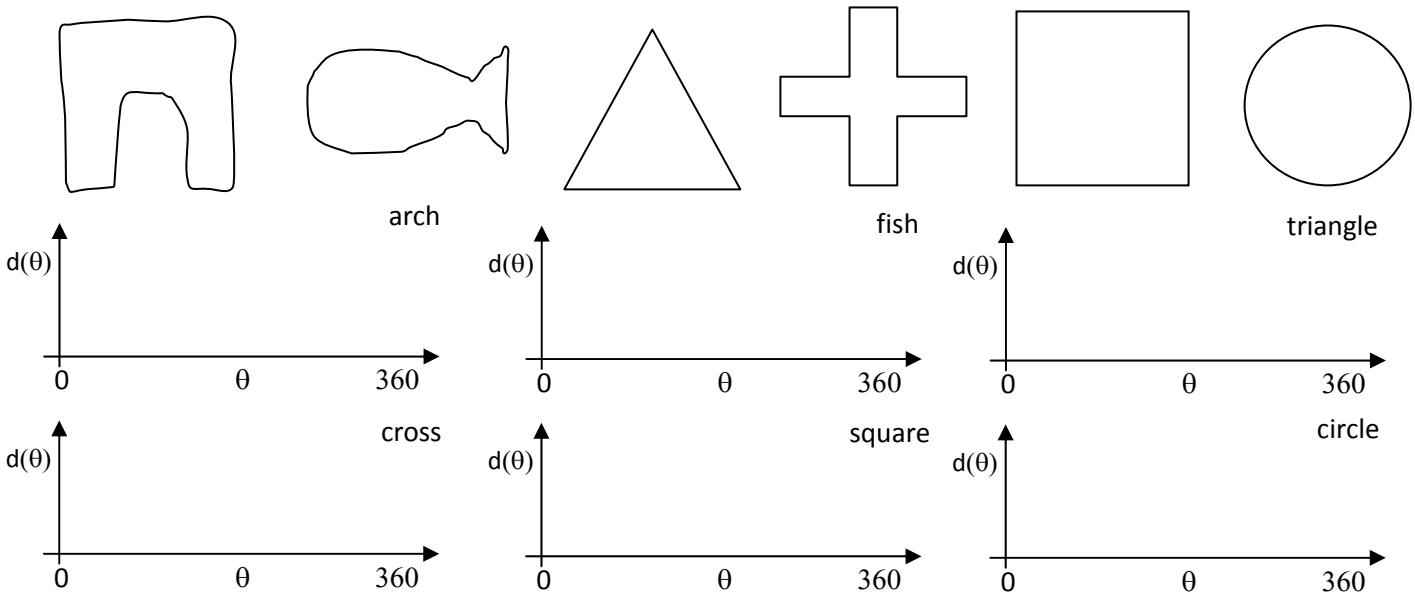
Ex3: Understanding the code *(10 minutes)*

Inspect the code for `FourierDescriptorDemo.m` and the function it calls to compute the Fourier Descriptors (`ComputeFD.m`). **Ensure you understand** how the Fourier descriptor code implements the maths in lectures for the central distance Fourier Descriptor. *Ask your tutors about any Matlab functions you are unfamiliar with.*

Ex4: Angular Fourier Descriptors *(10 minutes)*

Modify `FourierDescriptorDemo.m` so that it computes ***angular*** Fourier Descriptors (make edits as indicated on lines 31-34 of that file).

Sketch the signal for each of the six shapes that you think will be generated for this type of Fourier Descriptor.



Run your modified `FourierDescriptorDemo.m` and **inspect the signal and frequency response** for each shape you sketch.

Ex5: Shape Recognition *(10 minutes)*

Modify the program `ShapeDemoInteractive.m` changing the line `PATH='...'` to point to the folder into which you unzipped the shape example images (e.g. ~/shapeimages). Run **the program and draw a triangle** The program should identify your sketch as a triangle. **Try** a few examples the other shapes. If it fails, look at the probabilities – was the second most likely candidate correct? Note in your lab book any common misclassifications.

This program is loading 30 examples of the 6 shape categories, and computing the angular Fourier Descriptor for each example. These Fourier Descriptors describe each shape; they each a shape descriptor. The shape descriptors for each shape category are used to build an Eigenmodel. The Eigenmodels for each category are then used to classify new shapes (drawn by the user) using the standard Mahalanobis distance approach covered in lectures. This is an example of supervised classification; the example shapes are "expert" provided training data (exemplars).

Inspect the code and ensure you understand the steps taken to 1) load the files, 2) compute the descriptors, 3) build the Eigenmodels, 4) input the sketched shape, and 5) perform classification of the sketched shape.

Ex6: Batch Shape Recognition *(10 minutes)*

Modify the program `ShapeDemoBatch.m` changing the line `PATH='...'` to point to the folder into which you unzipped the shape example images. This program runs an automated test to evaluate the performance of the classification system. The set of shape examples is split into a "training" and "test" set. The training set is used to train the classifier as in Ex6. Each shape in the test is then classified, and the result noted. The TESTSET variable defines the training/test split –it is set to 0.25 i.e. one quarter of the data is used to test, the rest to train. The actual examples that make up each side of the split are picked randomly. This evaluation method is called **cross-validation**.

Run the code. The results of the classification are tabulated in a "confusion matrix". This should be read row by row. For example, row 1 refers to category 1 (the arch). The columns of row 1 show the fraction of arches classified as category 1 (arch), 2 (fish), and so on. A perfect classifier would have zeros everywhere and ones down the diagonal. You can come up with a figure describing the overall precision of the system by averaging the numbers on the diagonal; the so called **Average Precision** of the system.

Infact this program runs the cross-validation 10 times to remove potential bias in the random selection of examples. This is a standard evaluation practice –so called **10-fold cross-validation**.

Based on the results of your evaluation, note the following in your lab book:

1) **What is the average precision of the system?**
2) **Which object classes tend to be easily confused with one another?**
3) **Looking at the shapes, can you hypothesise why this confusion might occur?**
4) **How do the results change if you vary the proportion of train/test data so that fewer examples are used for training (i.e. you increase TESTSET)?**
5) **Why do you think the accuracy is affected in this way?**

OPTIONAL Ex.7: Batch recognition using central distance Fourier Descriptors *(10 minutes)*

Outside the lab, or if any time remains, modify the code `ShapeDemoBatch.m` to use classical Fourier Descriptors (based on ray length). You need to modify the code (approx location line 51) to call the function `ComputeFD`, which will now return a different set of descriptors. **Re-run the evaluation. Did the average precision change? Did the confusion matrix change? Based on these indicators, which shape descriptor (central FD or angular FD) is the best performer for this dataset?**