Computer Vision

Shape Context

Name: hanxue Liang

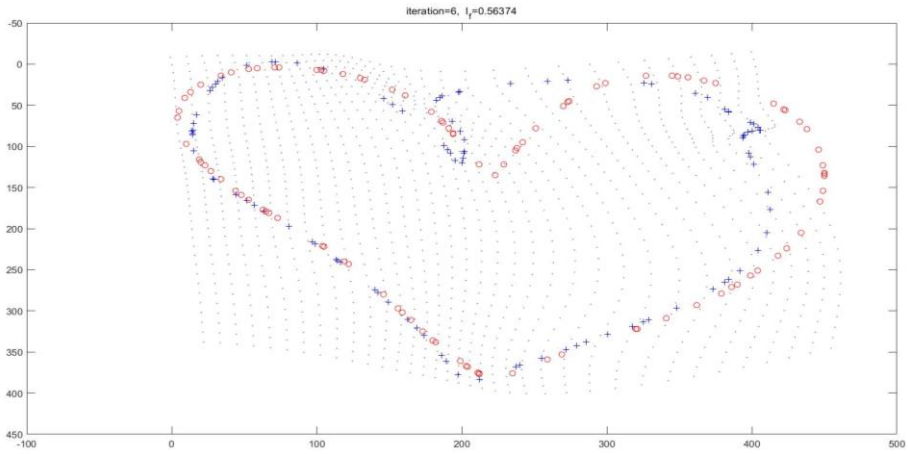
Student ID: 16-912-156

In this exercise, I have implemented a shape matching algorithm using shape context descriptor. Then I have realized shape classification using k-nearest-neighbour classifier.

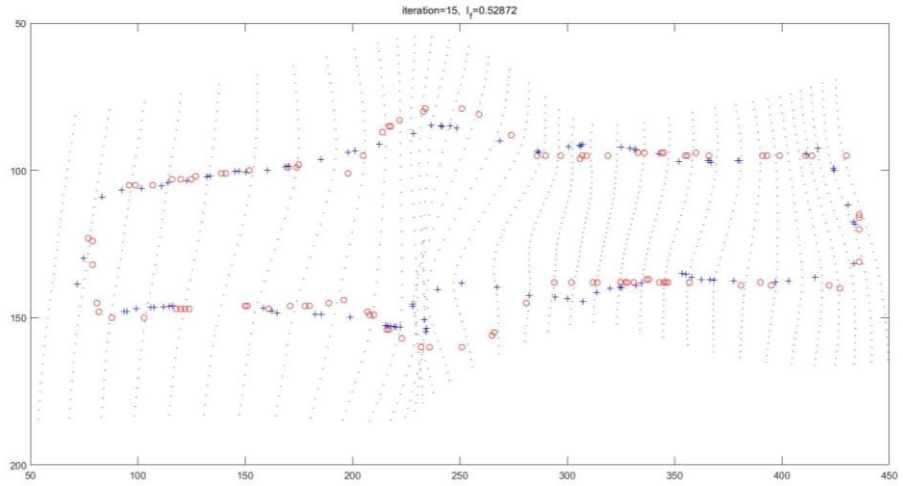
1. Shape Matching

In this part, I am given a set of points from a template contour for which I will match to a set of points on a target contour. The algorithm goes as follows:

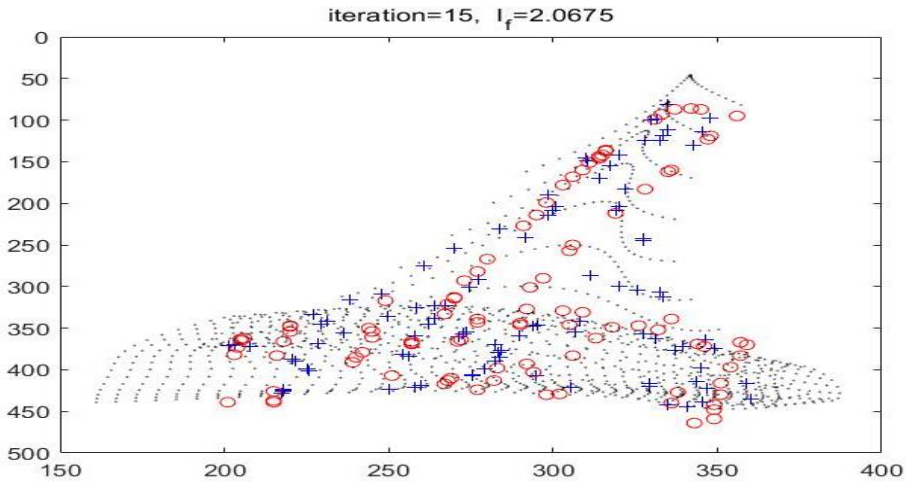
a) Compute shape context descriptors for both sets of points, this is implemented by the function *sc_compute*. The inputs of the function are set of points, number of bins in the angular dimension, number of bins in the radial dimension, the length of the smallest radius, the length of the biggest radius. The output is the descriptor.
b) On second step, I write the function *chi2_cost* to compute a cost matrix between two sets of shape context descriptors. The inputs of the function are two descriptors and output is the cost matrix.
c) On third step, I use the provided *Hungarian* function to perform one-to-one matching of the points by minimizing the total cost.
d) In this step, I use two independent plate spline (TPS) model to compute plan transformation. The parameters of the model are obtained by my implemented function *tps_model*.
e) Steps a-d are iterated for several times and my results are shown in the figure below.



a



b



c

Figure1:a) shape matching for heart, b) shape matching for watch, c) shape matching for fork

Shape context descriptor is scale-invariant, because when construct the descriptor, I normalize all radial distances by dividing the mean distance between all the point pairs in the shape.

2. Shape Classification

In this part, I will realize shape classification using shape context descriptor. Specifically, I use the implemented shape matching algorithm to compute the cost of matching each test shape to all training shapes. Then I find the k-nearest neighbour matches in the training shapes, and use these to assign a label to a test shape. The whole method is implemented in the function *shape_classification.m*.

I chose k to be 3, and the whole procedure is launched for 10 times. The average accuracy is:

a) What is the average accuracy of your classifier?
   When k=3
   average accuracy= 0.8333

b) How does your classification accuracy vary with the number of neighbours *k*?
   When k=2, average accuracy= 0.9200
   When k=4, average accuracy= 0.8667
   When k=5, average accuracy= 0.8467

I could find that when I increase k, the accuracy becomes lower.

c) If instead of your own sampling function, you use the one that we provide (get samples 1.m), do you get better classification results? Why or why not?
   When k=1, average accuracy= 0.9600
   When k=2, average accuracy= 0.9400
   When k=3, average accuracy= 0.9400
   When k=4, average accuracy= 0.9333
   sample_1.m gives us better result because after sampling points, it will remove the samples which are very close to each other. So this method tends to give us a more uniform sampling results.