ETH zürich | DINFK

December 6, 2018

Computer Vision

Lab 08: Shape Context

Sumeet Gyanchandani

1 Shape Matching

The first task was to implement a shape matching algorithm using MATLAB. I made use of a descriptor called the shape context descriptor, described in *Shape Matching and Object Recognition Using Shape Contexts*, by Serge Belongie, Jitendra Malik, and Jan Puzicha.

The algorithm used was:

1. Compute shape context descriptors for the points from both sets, the template and the target contour
2. Estimate the cost matrix between the two sets of descriptors
3. Use the cost matrix to solve the correspondence problem between the two sets of descriptors, finding the one-to-one matching that minimizes the total cost using Hungarian algorithm
4. Use the solution of the correspondence problem to estimate a transformation from template to target points with Thin Plate Splines and perform this transformation on the template points
5. Iterate steps (1-4)

1.1 Shape Context Descriptors

Function *sc_compute* computes the shape context descriptors for a set of points. I started with normalizing the smallest and the biggest radius. Then, I simply calculated the theta for pairs of points. Then, I the normalized of all radial distances by the mean distance of the distances between all point pairs in the shape.

1.2 Cost Matrix

Function *chi2_cost* computes a cost matrix between two sets of shape context descriptors. The cost matrix should be an $n \times m$ matrix giving the cost of matching two sets of points based on their shape context descriptors. It calculates:

$$C_{gh} = \frac{1}{2}\sum_{k=1}^{K}\frac{[g(k)-h(k)]^2}{g(k)+h(k)}$$

I also added 0.00001 to the denominator to avoid the division by zero error.

1.3 Hungarian Algorithm

hungarian.m was pre-implemented

1.4 Thin Plate Splines

Function *tps_model* computes the weights ω_i and a_1, a_x, a_y for both f_x and f_y. I solved the linear system by simply doing $x = A\backslash b$. For regularization, I set *lambda* to the square of the mean distance between two target points.

2 Shape Classification

I carried out the following steps for the k-nearest neighbour classification:

1. Determine shape matching costs between a test shape and all training shapes
2. Classify the test shape based on the labels of the k-nearest neighbour training shapes

2.1 Shape Matching

Function *compute_matching_costs* calls *shape_matching* on every object in both the images.

2.2 Nearest-Neighbour Classifier

Function *nn_classify* takes as its input the shape matching costs obtained by matching the test shape to all training shapes *matchingCostVector*, the class labels of the training shapes *trainClasses*, and the number of neighbours to consider k. It returns the class label of the test shape as its output *testClass*.

3 Results

Yes, the shape context descriptor is scale-invariant. In the beginning of the algorithm itself we assume normalized distances. The shape context descriptor is not rotation-invariant though.

The average the classification accurary that I observed was between 73.33% to 80.00%.

I ran the *shape_classification* function for value of k ranging from 1 to 7, for both the sampling scripts *get_samples.m* and *get_samples_1.m*. This demo code can be found in *shape_context_demo.m* and the results are given in Table 1.

Classification Accuracy for K values	using my get_samples.m	using get_samples_1.m
1	0.8000	1.0000
2	0.7333	0.9333
3	0.8000	0.8000
4	0.7333	0.7333
5	0.7333	0.8667
6	0.5333	0.4000
7	0.2667	0.3333

Table 1. Classification accuracies in various scenarios

I observed that the accuracies generally decreased with the increase in the value of K. There was a sharp decrease from value K=6 onwards, that can to attributed to the number of images and the number of classes being lower than the value of K. The *get_samples_1.m* performed better than my *get_samples.m* because rather than random sampling, Jitendra Malik's algorithm samples a more uniform sample from the distribution by eliminating 1 point from the closest pair in the mix at every iteration.