Computer Vision Assignment 8

Luca Ebner (ebnerl@student.ethz.ch)

December 4, 2020

In this weeks assignment we follow an algorithm proposed in the paper *Shape Matching and Object Recognition using Shape Contexts*.

1. Shape Matching

The Shape Matching algorithm pseudo code looks as follows:

1. Compute shape context descriptors for the points from the template and the target contour sets.
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 a one-to-one matching.
4. Use the solution of the correspondence problem to estimate a transformation from template to target points.
5. Iterate steps 1-4.

1.1 Shape Context Descriptors

The Shape Context descriptors of each data set point can be visualized as a histogram, where each bin in the Shape Context's polar coordinate system is represented by a tile in the histogram, as seen in Figure 1.

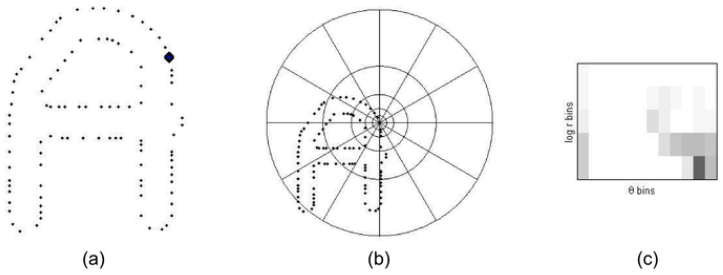

Figure 1: a) data set points, b) polar coordinate system for histogram bins, c) histogram

In our code framework, we create a function called `sc_compute` which takes the data set of point coordinates and bin parameters as inputs and gives the Shape Context Descriptors to all the given points as output. The function counts for each point how many neighbor points there are in each bin sector around it and then creates a histogram. In Figure 2 a visualization of some example histograms resulting from this function can be seen.

It is here to mention that Shape Context descriptors are not invariant to rotation by default. This is due to the fact that for the histogram bins we also consider the angle where other points lie in the polar coordinate system. The descriptor is therefore not the same whether a point lies at radius $r = r^*$ and $\theta = \theta^*$, or at radius $r = r^*$ and $\theta = \theta^* + 180\,\text{deg}$, for example.

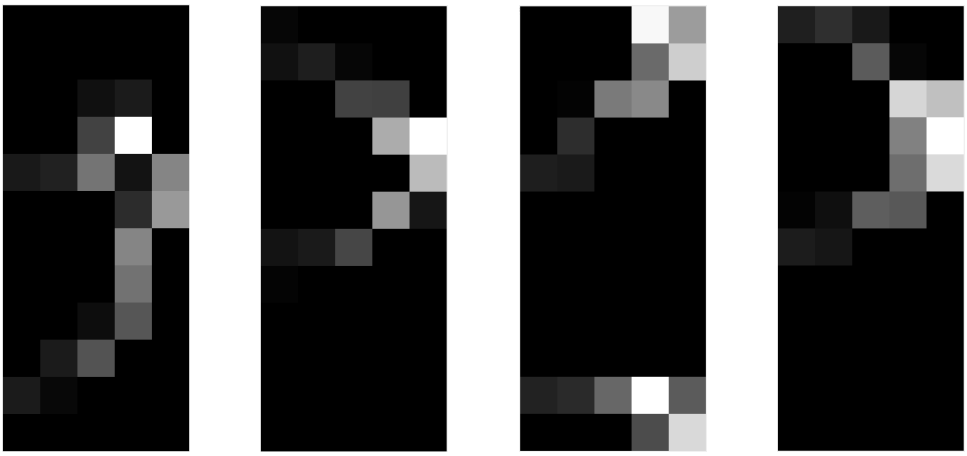

Figure 2: Example histograms restulting from the `sc_compute` function. Each tile represents a bin, where the horizontal axis corresponds to the radius and the vertical axis to the angle. The more points there lie in a bin, the more white the color in the histogram.

1.2 Cost Matrix

The formula for the matching cost between two descriptors g, h is given as

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

With this, we can define another function called `chi2_cost`, which takes two sets of descriptors as inputs and gives the the cost matrix $\mathbf{C}$ as output.

1.3 Hungarian Algorithm

The correspondence problem between the descriptors from the two sets can be solved by using the Hungarian Algorithm. In the exercise code framework, the Hungarian Algorithm is already given. The result of the Hungarian Algorithm is a one-to-one matching between two points, based on minimizing the costs of the previously found cost matrix $\mathbf{C}$. It is to note that the Hungarian Algorithm in general needs a square cost matrix $\mathbf{C}$ as an input, which is why we need to make sure that we use the same amount of data set sample points for the shape pair that we want to match. In our case, we've set the sampling number to `num_samples = 500` for all shape classes.

1.4 Thin Plate Splines

Based on the found point correspondences found in the hungarian algorithm, we can estimate a plane transformation to map the template shape to the target. This estimation can be found by using Thin Plate Spline (TPS) models.

Also for this part of the Shape Matching pipeline we define another function, namely `tps_model`. The inputs for this function are the point correspondences between the two sets and a regularization parameter λ. For each TPS model we have to solve the system given by

$$\begin{pmatrix} K + \lambda I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} w \\ a \end{pmatrix} = \begin{pmatrix} v \\ 0 \end{pmatrix}. \tag{2}$$

The output of the function are the parameters for the two TPS models needed for a 2D warping as well as the bending energy. The latter is an indication for how much energy is needed to match two shapes.

The lower the energy, the better the two shapes match. Figures 3 and 4 show the difference between the unwarped and warped template to two different targets.

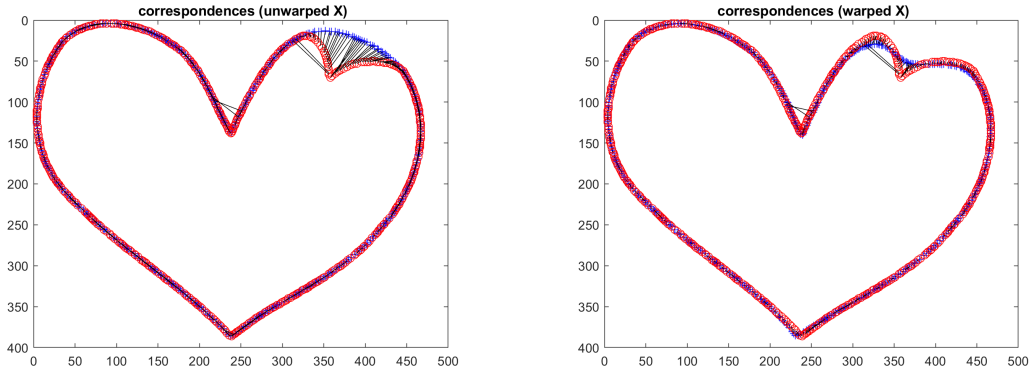

Figure 3: Unwarped (left) and warped (right) template shape to the target shape (data set shapes 1 and 3). Point correspondences between the two shapes are represented by black lines. In this case, the shapes are in fact classified as "Heart" in both cases, and the total bending energy matching cost was computed to be $E = 0.0982$.

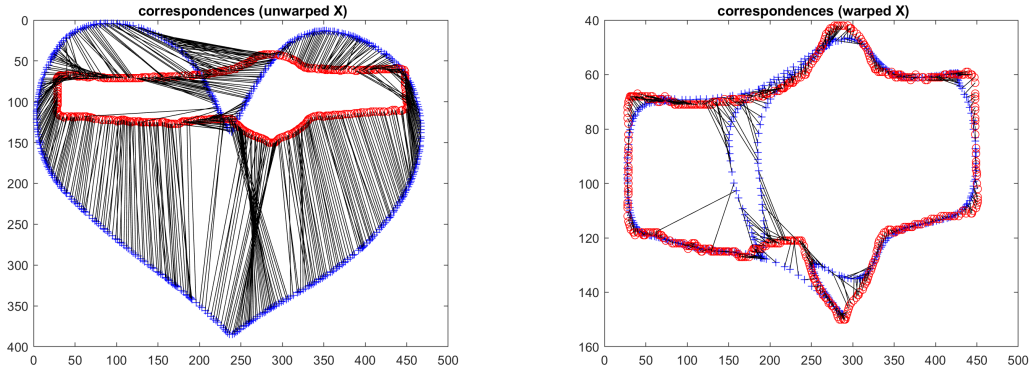

Figure 4: Unwarped (left) and warped (right) template shape to the target shape (data set shapes 1 and 11). Point correspondences between the two shapes are represented by black lines. In this case, the shapes are in fact classified differently for the two shapes. "Heart" for the first one and "Watch" for the second one. The total bending energy matching cost was computed to be $E = 2.0159$.

From Figures 3 and 4 we can see that we can guess whether two shapes belong to the same class by looking at the matching cost E. The lower the bending energy, the more likely it is that the two shapes belong to the same class.
However, it is to mention that shape the algorithm is not perfect. Sometimes descriptor histograms can look very similar but do in reality not belong to correspondent points. Even though we iterate over the algorithm steps several times (6 iterations in this exercise), there can still occur many faulty matches. As a result, the template model can be deformed wrongly and the resulting matching cost is very high, even though the matched shapes actually belong to the same class. Such an example can be seen in Figure 5.

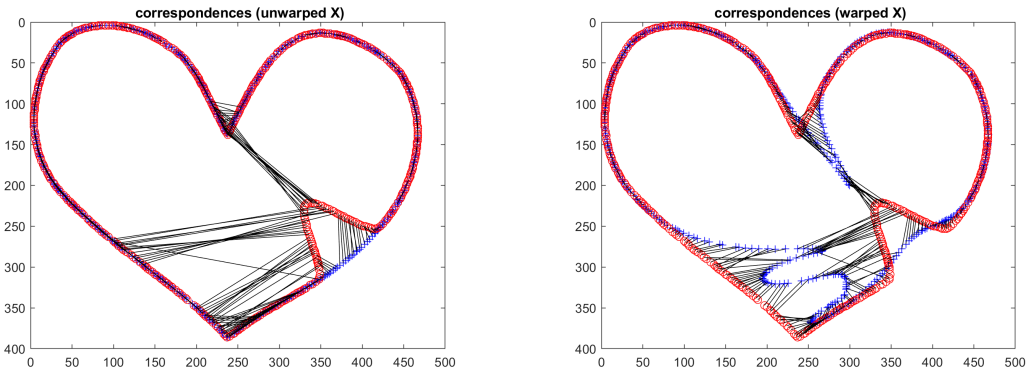

Figure 5: Unwarped (left) and warped (right) template shape to the target shape (data set shapes 1 and 5). Point correspondences between the two shapes are represented by black lines. In this case, the shapes are in fact classified as "Heart" in both cases but the descriptor matching shows multiple wrong matches, what trew off the transformation parameters during the TPS model computation. The total bending energy matching cost was computed to be $E = 2.9854$. Note that this cost is even higher than in Figure 4, while in theory it should be lower. A possible way to cope with this could be to increase the number of sampling points to make the descriptor histograms more unique, but this would also leads to much longer computation times.

Question: Is the shape context descriptor scale-invariant?

The shape descriptor itself is not scale-invariant by default. However, if we use the hint given in the exercise sheet (*"For increased robustness, implement the normalization of all radial distances by the mean distance of the distances between all point pairs in the shape."*) and normalize the radial distances, we can make the algorithm scale-invariant.