Shape Context

Chao Ni

December 4, 2020

1 Shape Matching

In this exercise, we are aimed at allow the compute to be able to recognize shapes the way that a human observer does. A shape comparison may similar to us but for a computer, it is difficult to recognize purely the intensity-based pixel values. In the following, we are going to provide a feature descriptor called shape context and use this to match two shapes.

1.1 Shape Context Descriptors

We create a function to compute the shape context descriptors here. We are given a set of points, and need to provide for each point in this set a histogram that shows the distribution of other points around it. In order to measure the distance, we use polar coordinate and normalize all distance with the mean distance of the distances between all point pairs in this point set.

Some keypoints when creating the histogram:

- We take the log to the distance, such that the descriptor is more sensitive to positions of nearby sample points than to those of points farther away.
- We specify the number of bins in the angular dimension and the number of bins in the radial dimension. The histogram is built on this two-dimension plane.

1.2 Cost Matrix

To measure the distance two shape context descriptors, we use the χ^2 test static:

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

where C_{gh} is measures the cost between two points from two shape context descriptors.

1.3 Hungarian Algorithm

To minimize the pair-to-pair cost based on the cost matrix above, we use Hungarian Algorithm, which is already provided in the exercise.

1.4 Thin Plate Splines

From the point correspondences, we can estimate the transform by a map $T : \mathcal{R}^2 \longrightarrow \mathcal{R}^2$. This map function can be parameterized by $T(x, y) = (f_x(x, y), f_y(x, y)$ and

$$f_{x(y)}(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} \omega_i U(||(x_i, y_i) - (x, y)||), \tag{2}$$

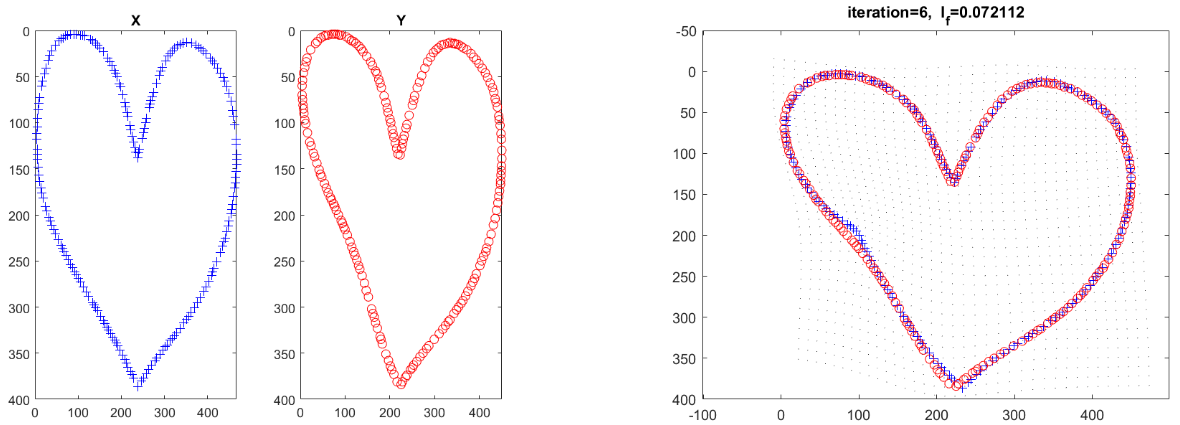
where $U(t) = t^2 log(t^2)$ and $U(0) = 0$ and it holds that $\sum_{i=1}^{n} \omega_i = \sum_{i=1}^{n} \omega_i x_i = \sum_{i=1}^{n} \omega_i y_i = 0$. Minimizing the cost boils down to solve a linear equation and this could be done efficiently. Details can be seen from the paper handed over with the report.

2 Results

2.1 Figure plotting

For each type of items, we pick two images each and sample 200 points. The separate figure comparison and the entangled images are shown below, respectively, in Fig. 1, 3 and 5. We also provide the warped and unwarped image for each pair, as shown in Fig. 2, 4 and 6. We can see from the figures, the results are generally as expected, but the algorithm still miss-matches some pairs. This might be due to following reasons:

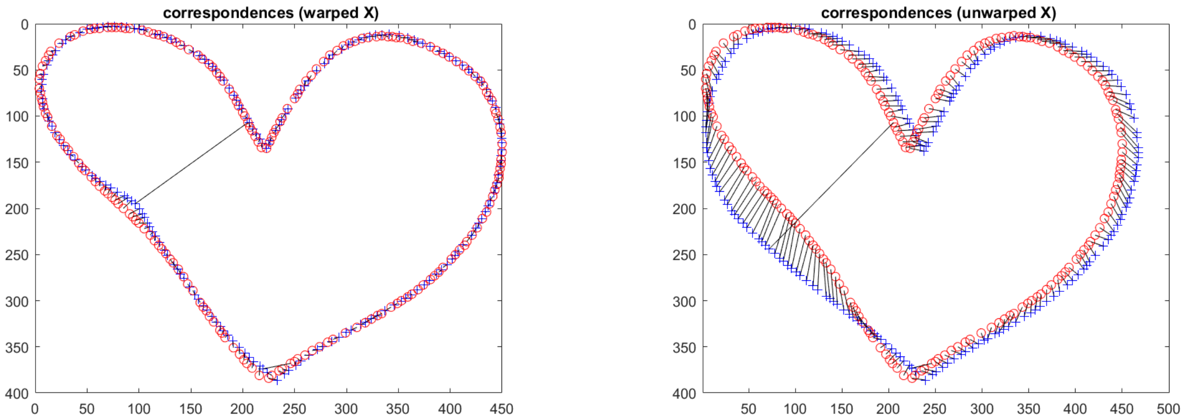
- Not enough iteration time: more iterations we run, more close the matching is;
- Lack of sampling points: more sampling points can give more detailed information for each descriptor, but also means heavier computation burden and sometimes would cause singular issues.



(a) initial comparison

(b) entangled after six iterations.

Figure 1: Heart: comparison of initial images and entangled image



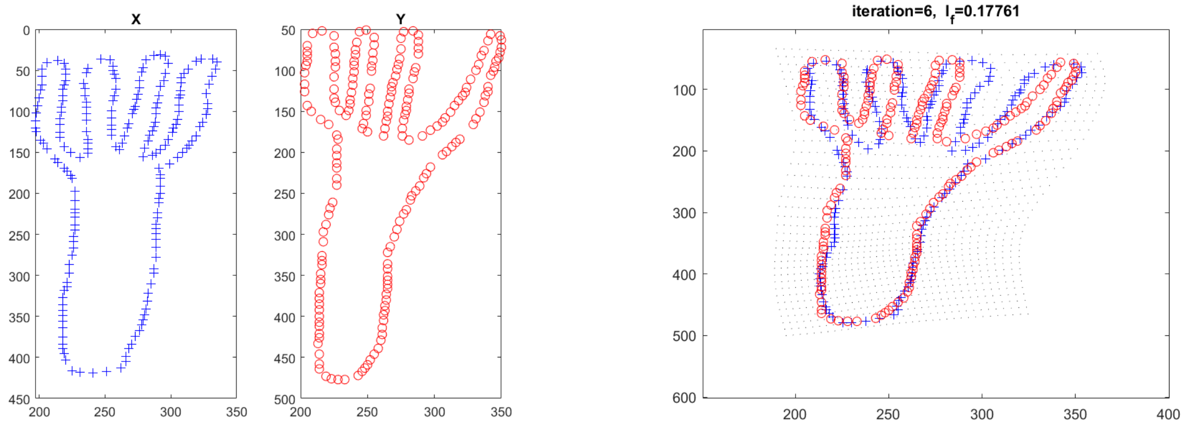
(a) warped image

(b) unwarped image.

Figure 2: Heart: comparison of warped images and unwarped image

2.2 Short questions

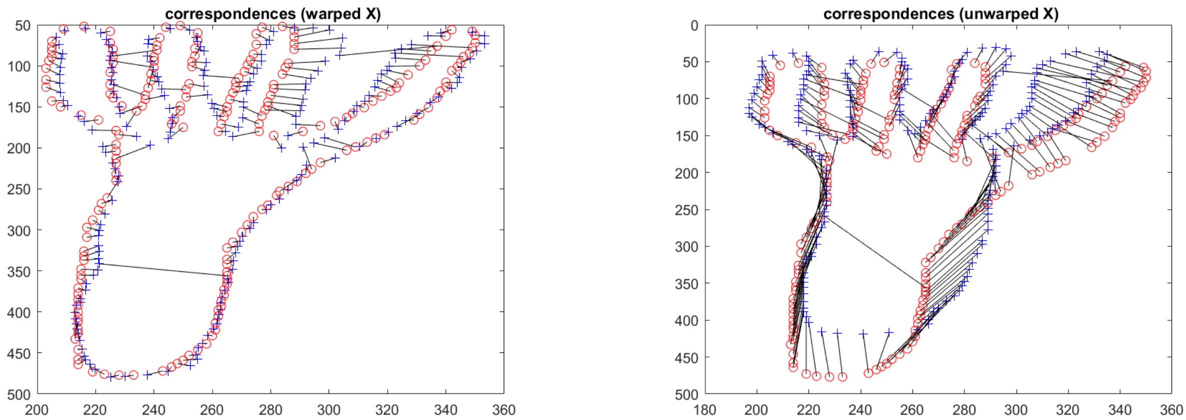
Yes, the shape context descriptor is scale-invariant. Because all radical distances are normalized by the mean distance between all point pairs in the shape.



(a) initial comparison

(b) entangled after six iterations.

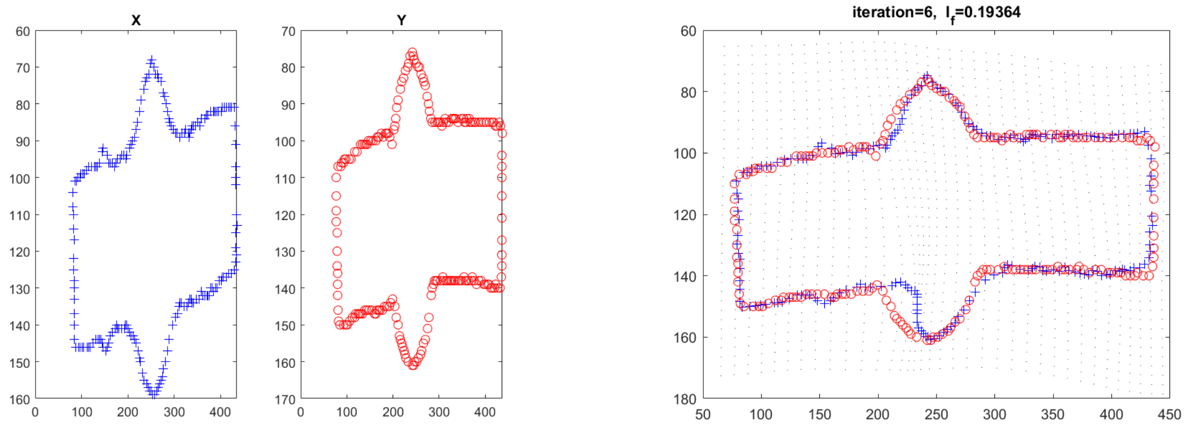
Figure 3: Folk: comparison of initial images and entangled image



(a) warped image

(b) unwarped image.

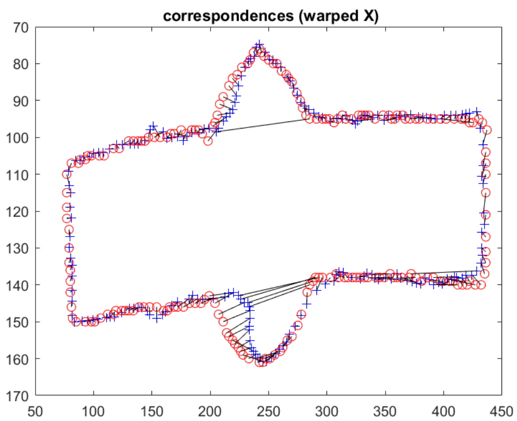
Figure 4: Folk: comparison of warped images and unwarped image



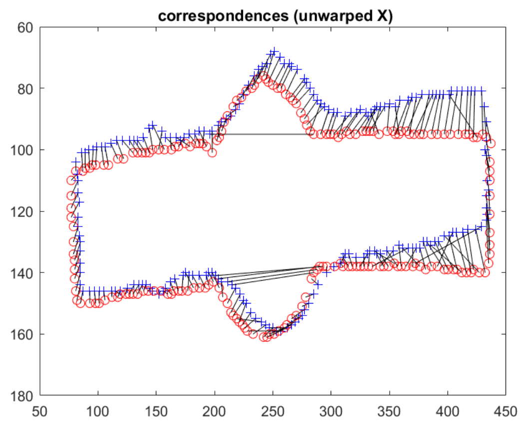
(a) initial comparison

(b) entangled after six iterations.

Figure 5: Watch: comparison of initial images and entangled image



(a) warped image



(b) unwarped image.

Figure 6: Watch: comparison of warped images and unwarped image