Computer Vision

Alessia Paccagnella

November 28, 2019

1 Shape Matching

1.1 Shape Context Descriptors

For this task I first read the paper *Shape Matching and Object Recognition Using Shape Contexts*. This helped me implementing the function `sc_compute`.
The inputs of the function are:
- X, a 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

For each point of X I computed an histogram of the radial and angular distances with the remaining points.
I initialized a vector of `edges`, to spread uniformly the given number of radial and angular bins between the smallest and biggest radius and the smallest and biggest angle. Just as described in the paper, we need a diagram of log-polar histogram bins so I initialized the edges vector with the functions `logspace` and `linspace`.
After this, I started iterating through all the points in the given set, and for every point I calculated the relative radial and angular distance with all the others. I used that as the input for the function `hist3`.
Moreover, inside the loop I divided the radial distance for the norm, which is the mean distance of the distances between all point pairs in the shape.

1.2 Cost Matrix

I computed the cost matrix, which is the cost of matching two sets of points based on their shape context descriptors.
I wrote the matrix `chi2_cost` with the following χ^2 test statistic formula. I added `eps` to the denominator not to divide for zero.

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

where g and h are the two descriptors valuated for every k bin.

1.3 Hungarian Algorithm

The code for Hungarian algorithm is already provided in the file `hungarian.m`, it performs a one-to-one matching of points minimizing the total cost.

1.4 Thin Plate Splines

The aim of this task is to estimate a transformation T : R2 $\rightarrow$ R2 that maps any point from one shape to the other.
I wrote the code inside the file `tps_model.m`.
I filled the matrixes A and b to solve the systems for x.
We had to systems, one solved for wx and one for wy.

$$\underbrace{\begin{pmatrix} K+\lambda * I & P \\ \mathrm{P}^T & 0 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} wx \\ a \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} vx \\ 0 \end{pmatrix}}_{\mathbf{b}}$$

$$\underbrace{\begin{pmatrix} K+\lambda * I & P \\ \mathrm{P}^T & 0 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} wy \\ a \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} vy \\ 0 \end{pmatrix}}_{\mathbf{b}}$$

Then $\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$.
In this system:
- **U** is a function defined as

$$U(t) = t2log(t2)$$
$$U(0) = 0$$

- **K** is defined as the fuction U of the euclidean distance between all the points in the set of points.

$$K_{ij} = U(\| (x_i, y_i) - (x_j, y_j) \|).$$

- **P** is a matrix whose rows are (1,xi,yi), where xi and yi are coordinates of points of X.
- $\boldsymbol{\omega}$ and **v** are column vectors formed from ω i and vi.
- **a** is the column vector with elements a1, ax, ay.
- $\boldsymbol{\lambda}$ is the regularizer, given as input.

After solving the system, I calculated the energy function with the formula on the assignment, by using the solutions of the system.

$$E = wx^\top * K * wx + wy^\top * K * wy \quad (1)$$

1.5 Results

I loaded the dataset given. I sampled the two sets of points, but with the simple datasample function given by matlab the samples were not very uniformly distributed so the result was very bad.

After that I wrote a different function that makes sure that the sampled points are not too close to each other, in order to have higher variety. For every pair distance, the function deletes the point at minimum distance from the one we are considering. It is also written on the paper that this works better for sampling [reference 45].
These were some of the results that I obtained. Due to the randomness of the samples, sometimes I had to run the program again to obtain better results.

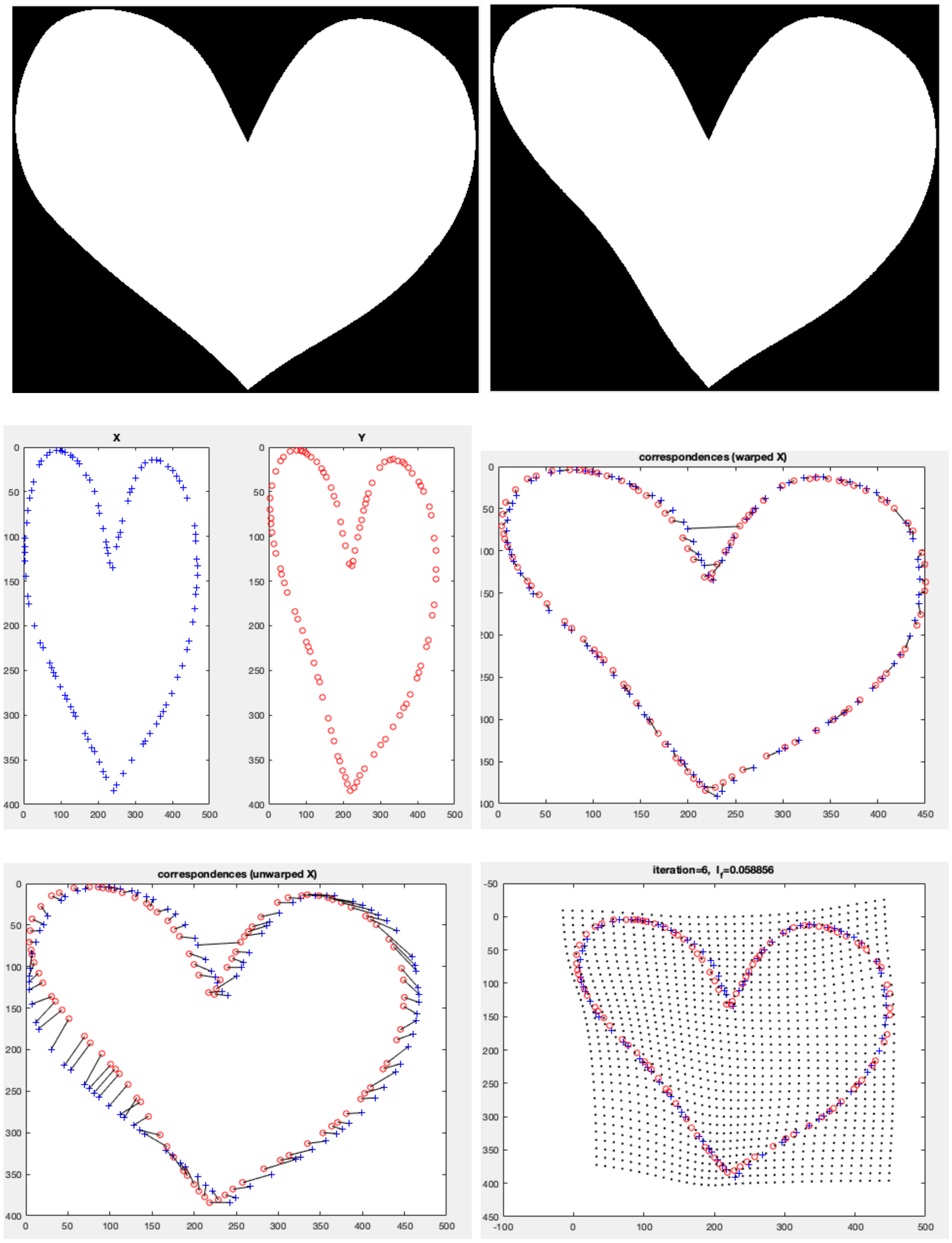

Figure 1: Taking the first two heart shapes

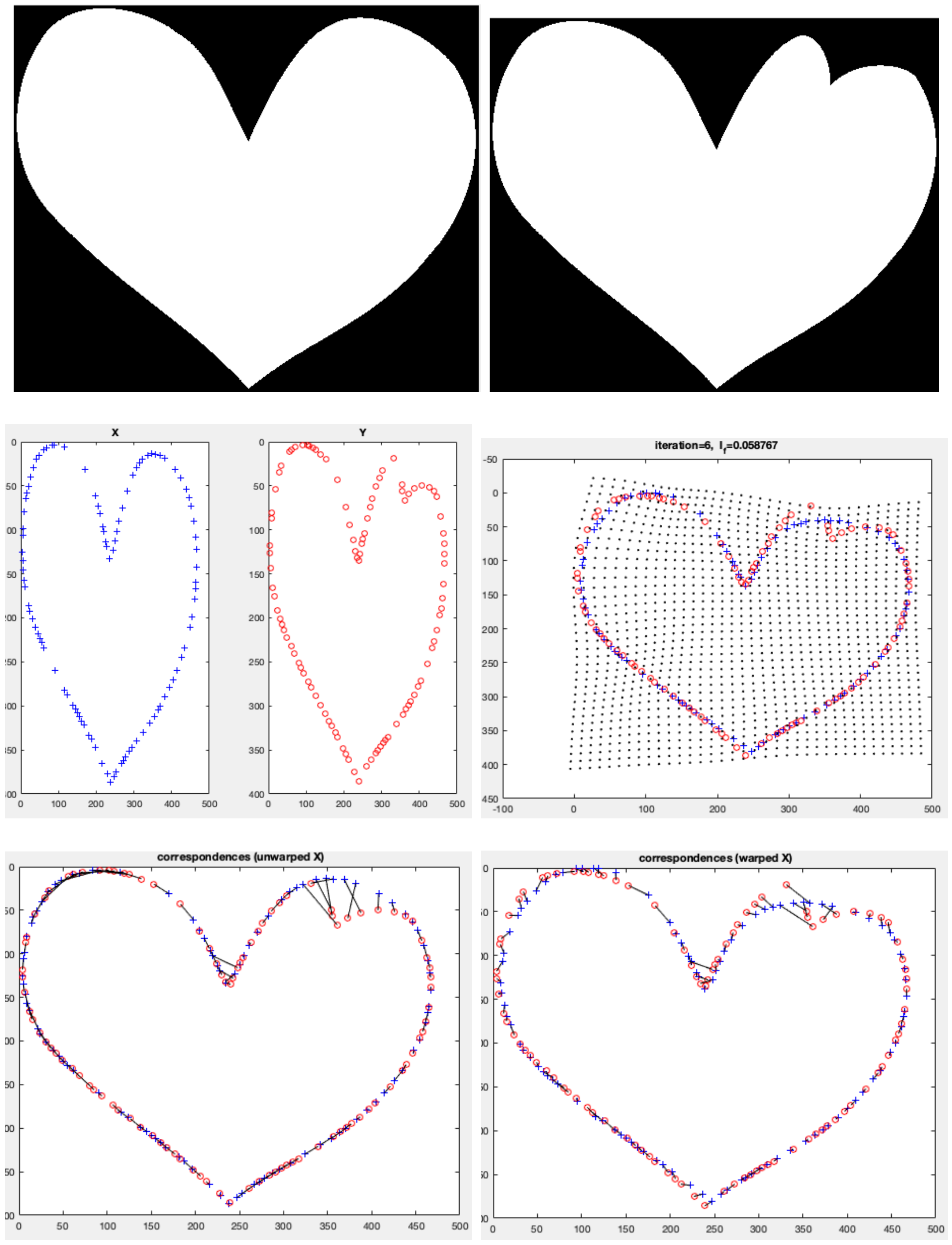

Figure 2: Taking the first and the third hearts shapes

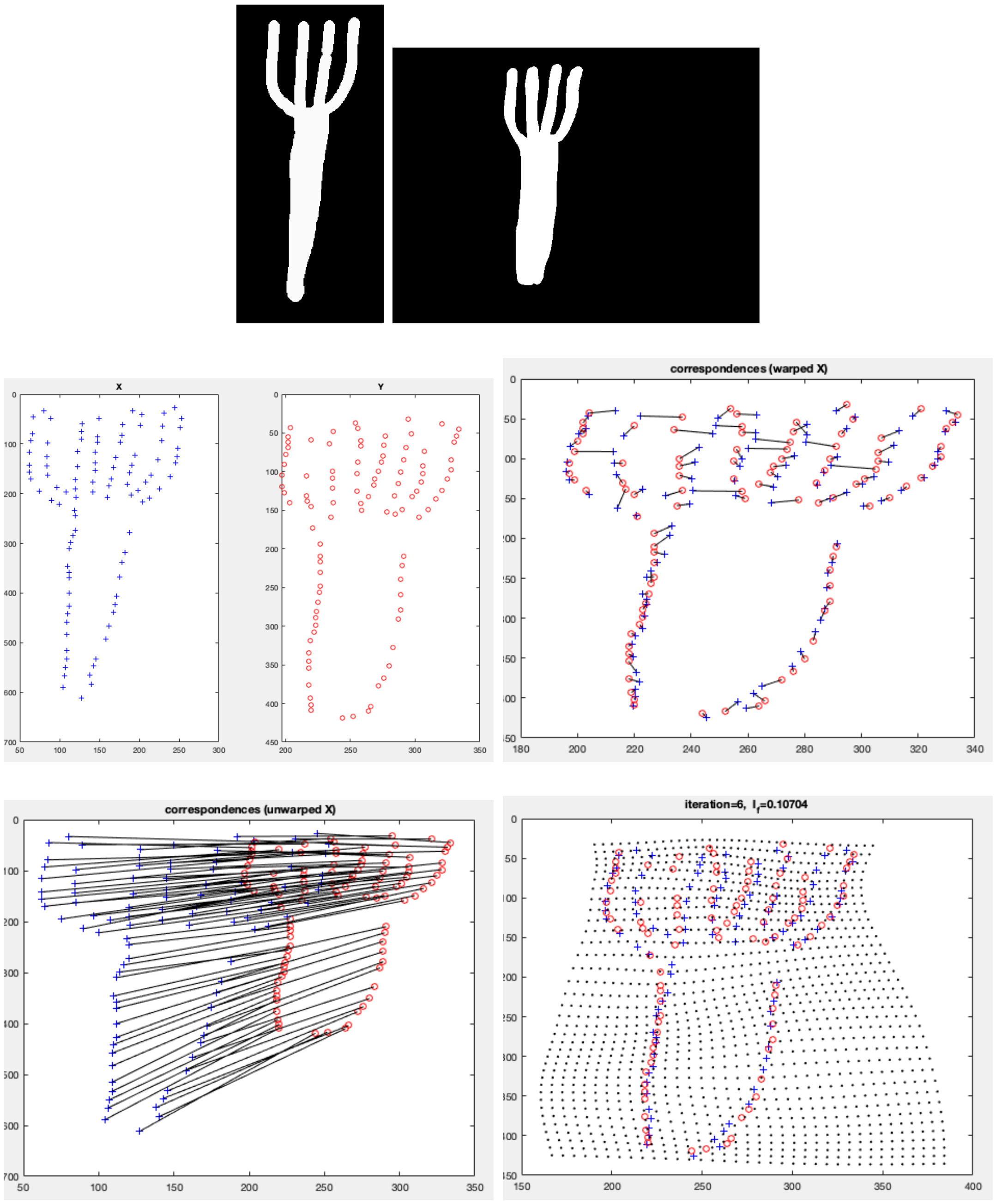

Figure 3: Taking the first and the third fork shapes

1.6 Question: Is the shape context descriptor scale-invariant? Explain why or why not.

The shape context descriptor is scale-invariant. We can also read it in the paper.
We achieve scale invariance by normalizing all radial distances by the mean distance. In fact, by scaling the shape by a factor, the distances and their mean would also be scaled. The scaling is cancelled by the division by the mean distance.