Computer Vision

Assignment 8 : Shape Matching

Autumn 2018

Nikita Rudin

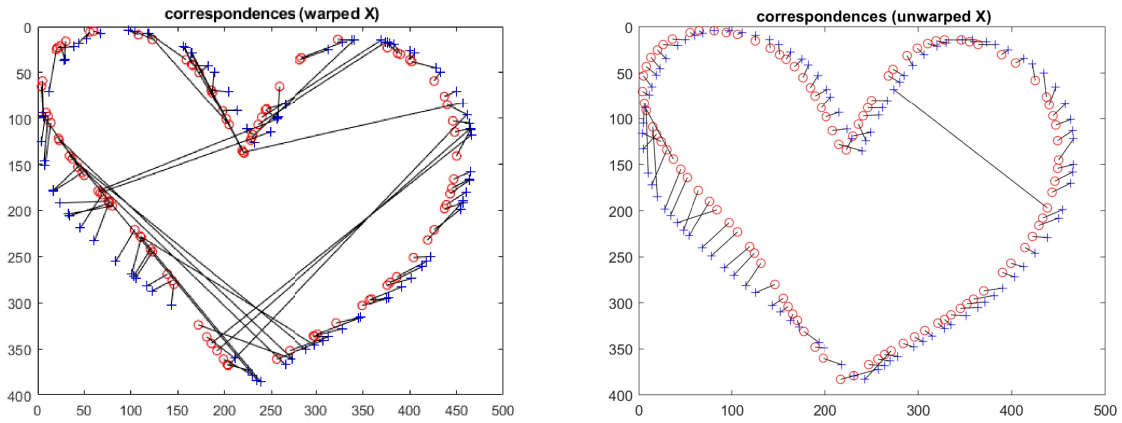
December 3, 2018

1 Shape Matching

In the first part of the assignment we compute shape context descriptors for two sets of points. We then match these sets by computing a cost matrix and using the Hungarian algorithm to get a one to one matching.

In order to compute the descriptors for each point, we get vectors from that point to all other points we then convert these vectors to polar coordinates and finally count the number of occurrences in 2d bins using the hist3() function in MATLAB. To get the wanted logarithmic scale for the radius we use $\log r$ with linear bins ranging from $\log r_{min}$ to $\log r_{max}$.

The cost matrix is computed with the provided formula. For greater efficiency, we avoid having nested for using loops by element wise operations are used. Finally this cost Matrix is sent to the Hungarian algorithm, which returns the matching.



(a) purely random sampling

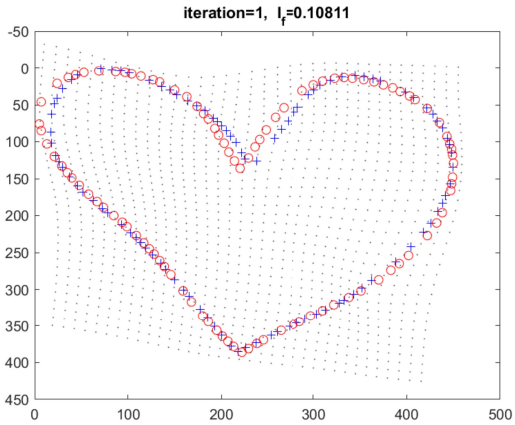
(b) provided uniform sampling method

Figure 1: Matching after 1 iteration of 100 points

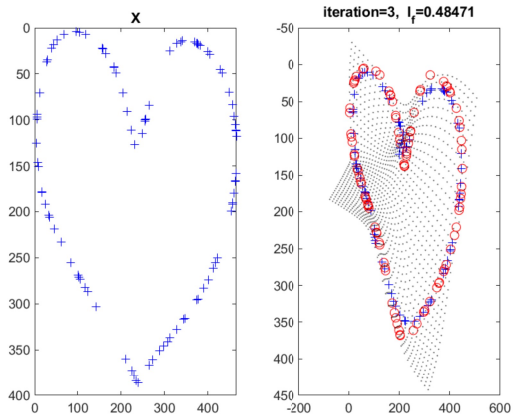
We can see that there is a big difference between the two sampling methods. This is due to the fact that the purely random sampling will select different distributions of points along the shape. The Hungarian algorithm enforces a one to one matching that simply doesn't exist with these different distributions. The provided sampling method mostly solves this issue and we can see a much better matching.

2 Thin Plate Splines

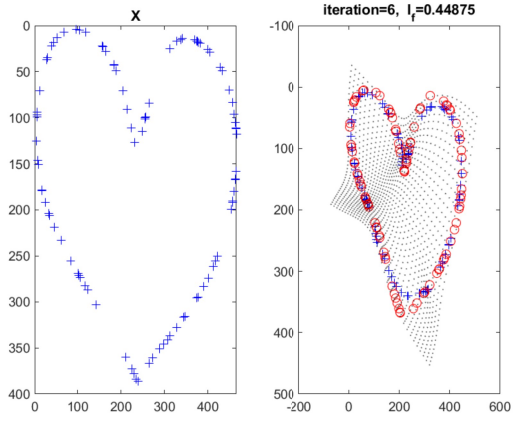
We will now use thin plate splines to model the deformation between the two compared shapes. We do this by directly computing the linear system provided in the description. We compute separate deformation along the x and y axis. The deformation energy, which be used for classification is defined as the sum of the energies for both axis. Once we have an estimate of the deformation model, we can apply it the 1st shape and start the matching again. If everything works correctly we can see how the 1st shape progressively merges into the 2nd.



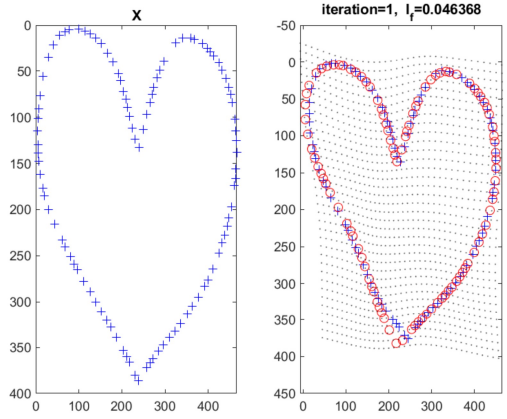
(a) iteration 1 - random sampling



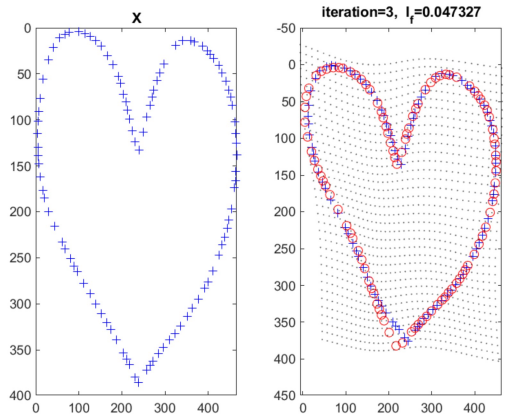
(b) iteration 3 - random sampling



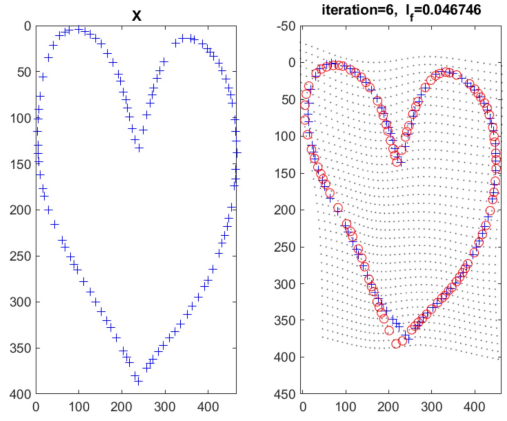
(c) iteration 6 - random sampling



(d) iteration 1 - uniform sampling



(e) iteration 3 - uniform sampling



(f) iteration 6 - uniform sampling

Figure 2: Matching after 6 iteration of 100 points

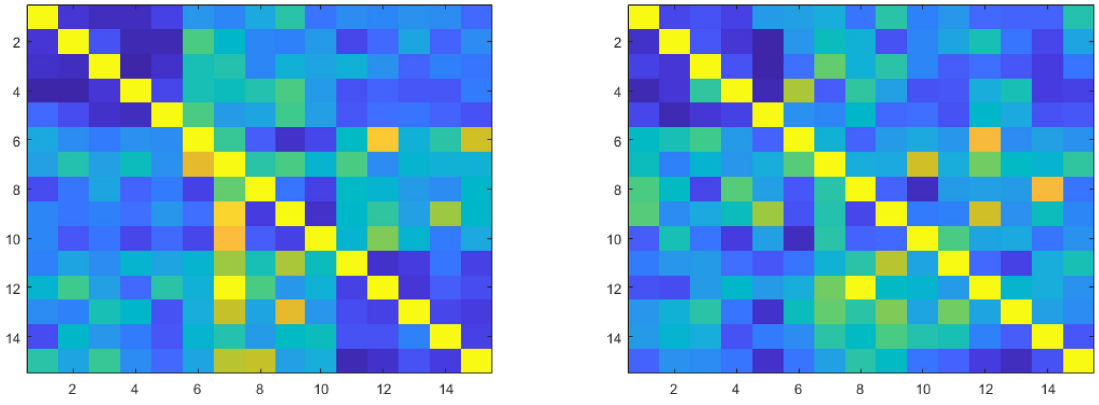
Analysing the results, we can see that despite wrong matches the shapes are matched together. The problem is that the deformation required is much greater in the 1st case (10x more). When we will do classification this can lead us to believe that the difference

between the shapes is much greater. It is not shown here but the matches in the 1st case improve with iterations, but are still not perfect. In the 2nd case only a few matches are incorrect. In the end the provided sampling method provides much more stable result due to its uniform sampling.

3 Shape Classification

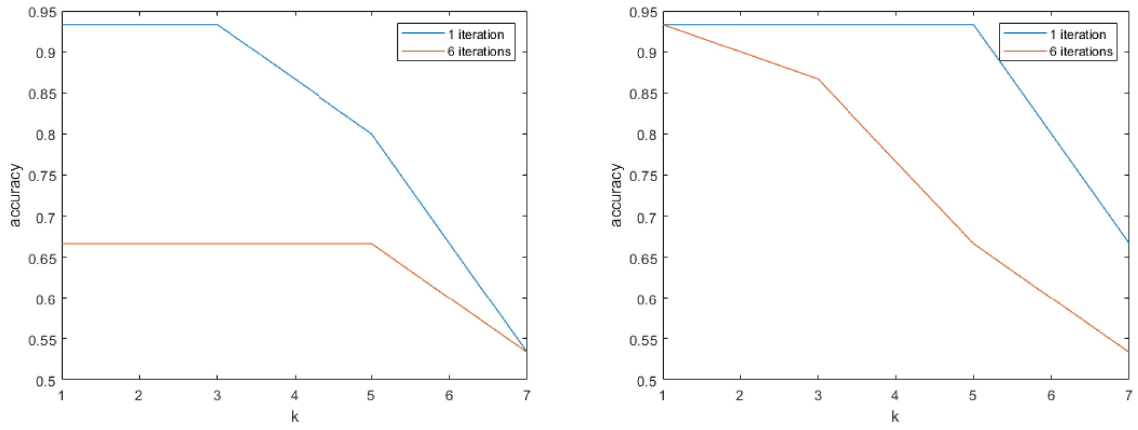
We can now classify shapes into categories. We have 15 images from 3 categories (5 from each). We compare each image to the others and compute the deformation energy. This gives as a 15x15 matrix. The lowest energies are attributed to images with the same shape. This is done with the k-nearest-neighbours algorithm. For each row of the matrix we find the k lowest elements and assign the category most present in these k shapes. Interestingly the best results are obtained for only one iteration of the matching algorithm. This is due to the fact that wrong matches lead to wrong deformations, which only get worse with iterations.

When using the uniform sampling the results become better with iterations as would be expected. None of the tests managed to correctly identify the reversed fork image. It is expected since we did not make our descriptors rotation invariant.



(a) iteration 1 (b) iteration 6

Figure 3: Comparison of cost matrices after 1 and 6 iterations (random sampling of 100 points).



(a) random sampling

(b) uniform sampling

Figure 4: Effect of k and the number of iterations on the accuracy

we see that the accuracy generaly drops with increasing k. Obviously in our case having k larger than 5 doesn't make sense and the results are not great as expected. It is also worth noting that this results and not really stable and can vary up to $+-15\%$. Since one iteration of the algorithm seems to work better in all cases and requires less computation, I would suggest to use as standard.

4 Discussion

Throughout this lab we have seen that the uniform sampling is superior to the random one. As such we should use it extensively.

This matching is not scale independent because we specify fixed r_{min} and r_{max} and as such fixed bins for the descriptors. We can easily make it scale invariant by dividing by the maximum r for example.

An other interesting improvement would be to combine this method with RANSAC to avoid matching points that are not present in the other sampling. This sould give much more robust results.