ETH Zürich
Computer Science Department

Prof. Marc Pollefeys
Prof. Vittorio Ferrari
Prof. Luc Van Gool
HS 10

Computer Vision - Exercise 8

Shape Context

Hand-out: 21-11-2019
Hand-in: 29-11-2019 23:00
Yawei Li: `yawei.li@vision.ee.ethz.ch`

Shape Context Descriptors and Shape Matching

How can we enable a computer to recognize shapes the way that a human observer does? Looking at two handwritten letters, they appear very similar as images but will be very different if one compares the pixel intensity values. This exercise will introduce a feature descriptor called *shape context* [1] and explore their use in matching two shapes.

8.1 Shape Matching (100%)

Your task is to implement a shape matching algorithm using MATLAB. We will make use of a descriptor called the *shape context descriptor*, described in detail in [1], which is available in the released files as *BelongiePAMI02.pdf*.

Suppose we are given a set of points from a template contour for which we want to match to a set of points on a target contour. One possible algorithm to achieve this is:

a) Compute shape context descriptors for the points from both sets, the template and the target contour.

b) Estimate the cost matrix between the two sets of descriptors.

c) 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 (e.g. with the provided Hungarian algorithm).

d) Use the solution of the correspondence problem to estimate a transformation from template to target points (e.g. with Thin Plate Splines) and perform this transformation on the template points.

e) Iterate steps (a-d).

We provide parts of the algorithm in `shape_matching.m`. Your task is to provide the missing code marked in `shape_matching.m` by comments. The MATLAB functions to be written are described below.

a) Shape Context Descriptors (40%)

Write a function which computes the shape context descriptors for a set of points. Your function should have the following form:

```
d = sc_compute(X,nbBins_theta,nbBins_r,smallest_r,biggest_r)
```

with the output `d` containing the shape context descriptors for all input points, and the inputs given by:

- set of points, `X`
- number of bins in the angular dimension, `nbBins_theta`
- number of bins in the radial dimension, `nbBins_r`
- the length of the smallest radius, `smallest_r`
- the length of the biggest radius, `biggest_r`

Hint: The shape context descriptor is described in detail in [1] in pages 511-513. For increased robustness, implement the normalization of all radial distances by the mean distance of the distances between all point pairs in the shape.

b) Cost Matrix (20%)

Write a function which 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. One possibility is to use the χ^2 test statistic:

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

where C_{gh} is the shape matching costs between two points with shape context descriptors g and h, each made up of $K = nbBins\_theta \times nbBins\_r$ bins.

The function should have the following form:

```
C = chi2_cost(s1,s2)
```

where the output `C` is the cost matrix for matching two sets of shape context descriptors `s1` and `s2`.

c) Hungarian Algorithm

We provide the code for the Hungarian algorithm which performs a one-to-one matching of the points based on the cost matrix, minimizing the total cost. This code is provided in the released file:

`hungarian.m`

d) Thin Plate Splines (40%)

From the point correspondences, we can estimate a plane transformation $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that maps any point (not only those originally sampled) from one shape to the other. We will use a thin plate spline (TPS) model:

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

where $U(t) = t^2 log(t^2)$ and $U(0) = 0$ and it holds that $\sum_{i=1}^{n} \omega_i = \sum_{i=1}^{n} \omega_i x_i = \sum_{i=1}^{n} \omega_i y_i = 0$.

This model gives a 1D output. Since we are interested in a 2D warping, we will use **two independent TPS models**, which we call f_x and f_y, to model the x and y coordinate transformations respectively. These combine to give the full transformation T:

$$T(x,y) = (f_x(x,y), f_y(x,y)) \tag{3}$$

For both of these models, (x_i, y_i) are given by the points coordinates in the **original** template shape, and the appropriate coordinates of the corresponding points on the target shape give the values v_i used to solve for the TPS model.

For each TPS model, you will have to solve a system of the form

$$\begin{pmatrix} K + \lambda I & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \omega \\ a \end{pmatrix} = \begin{pmatrix} v \\ 0 \end{pmatrix} \tag{4}$$

where $K_{ij} = U(\| (x_i, y_i) - (x_j, y_j) \|)$, the i-th row of P is $(1, x_i, y_i)$, ω and v are column vectors formed from ω_i and v_i, respectively, a is the column vector with elements a_1, a_x, a_y, and λ is a regularizer (see [1] for details).

Solving this system allows not only the transformation to be determined, but the bending energy of the transformation to be computed. This bending energy can be used as a measure of the cost of the shape matching. It is given by:

$$E = \omega^T K \omega \tag{5}$$

Your task is to implement a MATLAB function that computes the weights ω_i and a_1, a_x, a_y for both f_x and f_y. The function should have the following form:

```
[w_x w_y E] = tps_model(X,Y,lambda)
```

where the outputs `w_x` and `w_y` are the parameters (ω_i and a_i) in the two TPS models, E is the total bending energy and the inputs are as following:

- points in the template shape, `X`
- corresponding points in the target shape, `Y`
- regularization parameter, `lambda`

You can use a MATLAB solver for the linear system (if you have to find x, in $Ax = b$, you can find the solution in MATLAB with `x = A\b`). Having w_x and w_y you are now able to perform the transformation (warping) on the template points.

Hint: For regularization, set *lambda* to the square of the mean distance between two target points.

8.2 Hand In

Write up a short report explaining the main steps of your implementation and discussing the results of the methods. Make sure to include answers to the following questions:

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

Send the report together with the source code for your implementation (including the following functions `shape_matching`, `sc_compute`, `chi2_cost`, `tps_model`) to the Moodle exercise page. If you have any questions about this exercise, please email to `yawei.li@vision.ee.ethz.ch`.

References

[1] Belongie, S., Malik, J. and Puzicha, J., 2002, 'Shape Matching and Object Recognition Using Shape Contexts', IEEE Trans. PAMI