CS294-6 (Spring 2004) **Recognizing People, Objects and Actions** Lecture: February 10, 2004

Classifiers, Transformations and Shape Representation

Lecturer: Jitendra Malik *Scribe: Parvez Ahammad*

What we discussed earlier:

- LeNet (Convolutional Neural Nets)
- Tangent Distance approach (Nearest Neighbor Classifier methodology)

Today's topics:

- Idea of an augmented data set
- Broad categories in classifiers
- Shape
- Transformations
- Idea of deformable templates; in particular - shape context method

Idea of an Augmented Dataset:

(This was in response to the question about the augmented dataset used in LeNet paper). Let's say we have 6000 examples for training. For each example, generate ten copies by applying small transformations. So, effectively we now have 60,000 examples with some in-built invariance (vaguely) to the transformation that we used. This augmented dataset can now be used for training purposes.

Couple of observations in relation to data augmentation:

- Local minima related problems still remain
- If we know that there ought to be certain invariances, why not build them into our recognition scheme?

For example, the convolutional-nets used in LeNet paper are shift-invariant. We desire that scale in-variance be built in. In other words, let us say we are given a detector which takes a 32x32 patch as input and finds the digits at that scale. So, the question is: *How do we attempt to build-in the scale-invariance?* .

STRATEGY: *Scanning Strategies using Pyramids*

The idea is to build multiple instances of the image at varying scales by using decimation(down-sampling) schemes. Let's say we will decimate by a factor of 2 (*When decimation is done by 2, the scales are considered to be one octave apart*). One of the things to note while building such pyramids is not to simply take the average of 4 pixels to make the next-level pixel (that will cause aliasing). It is better to first apply a low-pass filter on the image (example: a Gaussian) and then do the down-sampling. Once the pyramid is built, then run the detector at different scales (different levels of the pyramid) and claim success if we succeed in detecting the object any of the scales. There will be a problem if the jump from one scale to the next scale is too large. In that scenario, our detector might miss the scale at which it can recognize the object. To avoid

this, we need to be careful about ensuring the presence reasonable scales- by using $(\sigma/\sqrt{2})$ steps between the standard deviation of the Gaussian kernel, we can create n-octave spacing in the scales. Standard practice in computer vision community (especially in face recognition applications) is to use scales that are one-quarter octave spacing apart.

The downside of this approach is that these schemes are computationally expensive. For naive implementations, this computational cost will increase linearly with the number of objects (assuming that the location and scale are unknown throughout). As we can see, all scanning strategies are *top-down* approaches.

Critique regarding Scanning Strategies: Computation is being wasted in areas that couldn't possibly have any instances of the objects of interest. *So, what is the alternative?* Cascade Classifiers!

Broad categories of classifiers:

- **Cascade Classification Schemes**: A sequence of tests placed in the increasing order of computational cost - such that the simpler (computationally cheap) tests can be carried out first and the more expensive tests can be carried out on the images that passed the simpler tests. The justification for sequential testing schemes is that the *average computational cost is much lower* across the whole dataset. Clearly, a problem with this approach is that decisions are made sequentially - which means - *if a bad decision was made early, it cannot be rectified at later stages.*
- **Attention/Segmentation based Classification Schemes**: These schemes operate by performing some low-level processing that is independent of the detector mechanism - to find regions of interest (*ROI*s), and then run the detector in these *ROI*s. Potentially, one could also figure out the scale of optimal detection by looking at the size of the *ROI*s. An example of this type of scheme is a corner detector algorithm that usually precedes certain algorithms (example: SFM). A milder version of these schemes are the over-segmentation schemes (assuming that all necessary edges or regions are preserved along with some junk).

Majority of today's classification algorithms in computer vision follow the cascade approach.

Shape:

(*Slide 1:*) Biological Shape - D' Arcy Thompson (1917) studied transformations between shapes of organisms. Key insight here is to note that *description of shape is hard, but description of transformations between shapes is easier!* The two steps involved in being able to do this, are:

- Correspondence problem (In nature, correspondences are clear for landmarks and fiducial points)
- Transformation problem

Transformations:

From $R^2 \rightarrow R^2$, here are the kinds of transformations we come across:

- Translation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{1}$$

- Rotation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2}$$

- Linear Transformations:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3}$$

- Affine Transformations:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{4}$$

- Euclidean Transformations:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{5}$$

As we can clearly see, all of the above mentioned transformations are basically affine transformations.

In general, we can classify transformations into two broad categories:

- Continuous Transformation: An equivalence relation and one-to-one correspondence between points in two geometric figures or topological spaces which is continuous in both directions, also called a homeomorphism. A homeomorphism which also preserves distances is called an isometry. Affine transformations are another type of common geometric homeomorphism.
- Smooth transformation: A map between manifolds which is differentiable and has a differentiable inverse (also called diffeomorphism).

Linear transformations are quite useful because they involve solving a set of linear equations. Although a bit unmanageable, most smooth non-linear transformations can be treated as *piecewise linear in the differential sense*. If we consider only the first two terms of the Taylor series expansion of any function, it looks like an affine transformation (of the form $\alpha x + \beta$). So, any non-linear transformation can be approximated up to first-order using affine transformations.

Idea of deformable templates: (*Slide 2:*)

- Fischer & Elschlanger (1973)
- Grenander et. al. (1991)
- van der Malsburg (1993)

(*Slide 3:*) Matching Framework:

- Solve correspondences: correspondences can be points, curves or regions - the usual method of local appearance matching (at point or patch levels) is not going to be much useful.
- At various points, a histogram is constructed by creating radially subdivided concentric circles (2D scenario) with logarithmic spacing between the circles. There are some constraints placed on the radius length, and on the spacing between the circles etc. (Refer: Belongie, Malik, Puzicha - Shape Context paper). By counting the membership in these bins, we create a two dimensional histogram for each point chosen (where these concentric circles were centered). This histogram is the *descriptor* that we use to match various points on the object, in order to determine the correspondences. These descriptors are called ***shape contexts***.

- Chi-squared distance metric is used for comparing the histograms of the shape-contexts of the chosen points:

$$f(x) = \frac{1}{2}\sum_{k=1}^{N}\frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)} \tag{6}$$

Thin-plate theory deals with creating a transformation such that a set of given starting points are morphed toward a set of end points, while ensuring that all the points lying in-between (on the surface) are reasonably interpolated(*Refer Duchon (1977) etc. for a better exposition of thin plate theory*). This transformation is usually achieved by minimizing some physical energies such as bending energy of the surface formed by the starting points.

So, the basic algorithms for matching shapes using shape-contexts is:

1. *find correspondences using the shape descriptors* (as explained earlier)
2. *find the deformation of thin-plate by minimizing the bending energy* (and solving the resultant linear equations involved)
3. *iterate between steps 1 and 2 until convergence*