Lecture 15: Deformable Templates

A.L. Yuile

March 21, 2012

1 Introduction

This lecture describes two ways to formulate deformable templates. The first formulates the task as a matching problem where the goal is to find a correspondence between a target shape and a source shape. This can be used for tasks like face recognition after a face has been detected. The second formulation is more suitable for detection when the goal is to detect a deformable object, like a hand or a face, from an image that contains one, or more, examples of the object but also cluttered background.

The basic idea of deformable templates is that objects have shape deformation. The classic reference is Fischler and Erschlager (1973) who modeled a face in terms of a small number of features – at the eyes, nose, and mouth – and allowed deformations between them. Another early example used deformable templates to model the structures of eyes and mouths (Yuille, Hallinan, and Cohen 1991). There are many other applications of these ideas – detecting particles in high energy physics experiments, heuristic algorithms for solving the travelling salesman problem.

2 The Matching Problem: Part I

The simplest version of the matching problem is to define the model by a set of attribute features (AFs) $\{(y_a, B_a) : a = 1, ..., M\}$ and the data also consists of a set of attributed features $\{(x_i, A_i) : i = 1, ..., N\}$. Here y_a, x_i denote spatial positions and B_a, A_i are attributes. (e.g. response of Gabor filters, Lowe's SIFT features, orientation of edgelets).

We define a set of correspondence variables $\{V_{ai} : a = 1, ..., M \ i = 1, ..., N\}$ such that $V_{ai} = 1$ if AF (y_a, B_a) of the object model matches AF (x_i, A_i) in the image, $V_{ai} = 0$ otherwise. Often you define a "junk node" $i = 0$ so that $V_{a0} = 1$ means that the AF from the object model is unmatched – to allow for cases where the AF is either occluded in the image or not detected in the image (due to thresholds used in the feature detector). Set a matching constraint $\sum_{i=0}^{N} V_{ai} = 0, \ \forall a$ to ensure that all AFs in the model either have a unique match, or are unmatched.

The simplest (non-trivial) formulation minimizes an energy function of form:

$$E(V) = \sum_{ai} V_{ai} \phi(x_i, y_a, A_i, B_a), \tag{1}$$

where $\phi(x_i, y_a, A_i, B_a)$ is small when the $\{x_i\}$ and $\{A_i\}$ are close to the corresponding $\{y_a\}$ and $\{B_a\}$. If a junk node is used, then there is typically a penalty used $\lambda \sum_a V_{a0}$ which penalizes the number of unmatched points (otherwise it may be energetically favorable to have all points unmatched). This penalty can be interpreted as the probability that a point is unmatched (e.g., a generative model where there is a non-zero probability that an AF of the object model is not detected).

This can be formulated as a generative model:

$$P(\{(x_i, A_i)\}|\{V_{ai}\}, \{(y_a, B_a)\}) = \frac{1}{Z} \exp\{-E(V)\}. \tag{2}$$

but we will not explore this interpretation in this section.

An energy of form given by equation (**??**) can be minimized (or the probability maximized) using polynomial-time algorithms. If $N = M$ and we impose 1-1 matching between the model and the data points (i.e. V_{ia} is restricted to being a permutation matrix) then minimizing $E(V)$ reduces to the *linear assignment* problem. It can solved by the Hungarian algorithm or by Bertsekas's auction algorithm, or by Sinkhorn's algorithm.

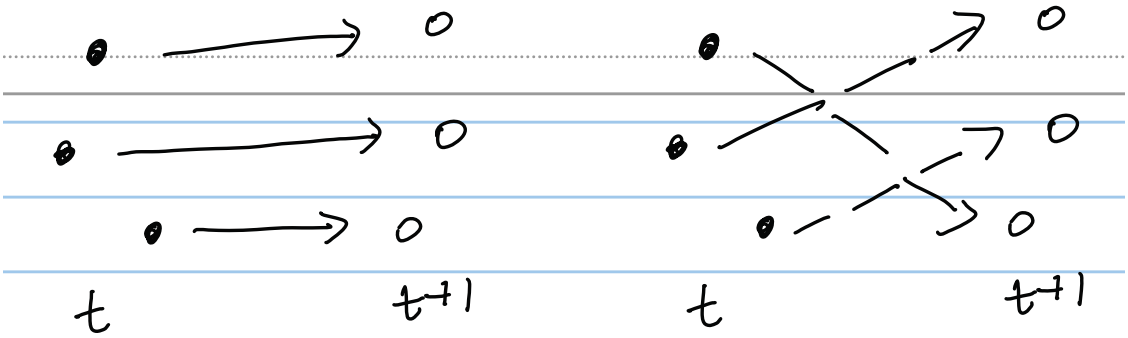

Figure 1: Matching problems require solving the correspondence problem to determine which points match between two images. The form of $E(V)$ specifies the attributes and the relations which are needed to make the correspondence problem unambiguous.

This model can be applied to matching models where the attributes are sufficient to lead to unambiguous matching – e.g., the $\{A_i\}$ and $\{B_a\}$ are distinctive and $\phi(.)$ is chosen to be of form $|A_i - B_a|$. It would be less effective if the data consists simply of un-attributed points. But in this case we can introduce attributes by *shape contexts*, which assign attributes to points based on the histograms of the relative positions of nearby points.

More complex models introduce quadratic terms in the energy function:

$$E(V) = \sum_{ai} V_{ai}\phi(x_i, y_a, A_i, B_a) + \sum_{aibj} V_{ai}V_{bj}\psi(x_i, x_j, A_i, A_j, y_a, y_b, B_a, B_b). \tag{3}$$

This includes relations between the model points which are being matched to the data. This can put encourage constraints so that neighboring points match, and hence make the correspondence unambiguous. This model is typically much harder to minimize unless the relational terms are only non-zero for a simple neighborhood structure. This will happen, for example, if we formulate the travelling salesman problem in this way (it is NP-complete, so clearly no polynomial time algorithm like dynamic programming can solve it). We can, however, use heuristic algorithms like mean field theory (variational methods) or belief propagation or stochastic sampling.

3 The Matching Problem: Part 2

We can augment the previous models by having additional variables. One model (Yuille and Grzywacz, Rangarajan and colleagues) takes the following form:

$$E[V, v] = \sum_{ai} V_{ai}\{x_i - y_a - v(y_a)\}^2 + E_p[v], \tag{4}$$

where $v(\vec{y})$ is a velocity (or matching) field defined over the input space (i.e. all y). The idea is that points y_a matches point x_i after translation $v(\vec{y})$. The prior encourages the velocity field to have properties such as: (i) slowness (i.e., $\vec{v}$ is small), and (ii) smoothness (i.e. the derivatives of $\vec{v}$ are small).

To perform inference on this type of model we can use the EM algorithm (requiring good initial conditions – see later). We formulate a free energy:

$$F[v, q(v)] = \sum_V q(V) \log q(V) - \sum_V q(V) \log P(v, V|x). \tag{5}$$

When is EM practical (apart from the initial conditions)? We need to be able to compute the two steps $q^t(V) = P(V|v^t, x)$ and to solve $v^* = \arg\min\{-\sum_V q^t(V) \log P(v, V|x)\}$ (which pre-requires being able to compute $\sum_V q^t(V) \log P(v, V|x)$). As in previous lectures, these tasks are usually practical when $q(V)$ is a factorizable distribution. This may happen naturally – as a result of the form of $P(v, V|x)$ and $E(v, V)$ – or it can be imposed as an approximation (the mean field theory approximation). Note that it will fail to happen if we impose both sets of constraints $\sum_a V_{ai} = 1 \ \forall i$ and $\sum_i V_{ai} = 1, \ \forall a$. It is impossible to have a factorizable distribution in this case since it requires one-to-one matching between AF's in the image and the object model – a factorizable model would require that each AF in the object model is matched independently, which can allow for several AFs in the object model to be matched to the same AF in the image (forbidden by 1-1 constraint).

There is a problem with imposing one-to-one matching. In practice, it seems easier to impose only half the constraints (e.g., $\sum_i V_{ai} = 1, \ \forall a$) and then rely on the prior to impose the other set of constraints – or, at least, to ensure that they are not violated too badly. For example, violating the constraints will typically lead to a velocity field that is not spatially smooth.

These types of models can also allow global transformations, like affine transformations, see figure (2). See Rangarajan et al.

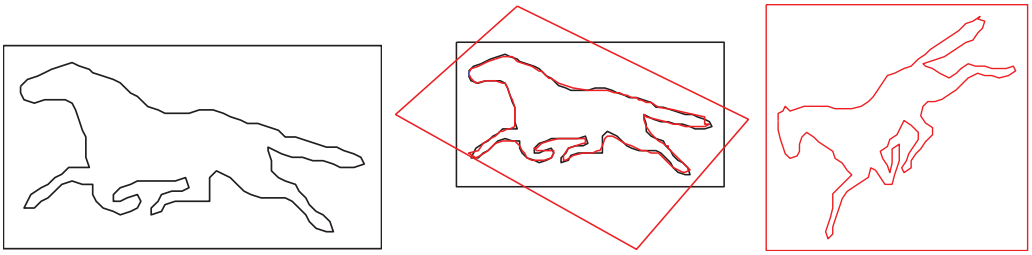

Figure 2: Global variables can be introduced to allow for affine deformations.

These methods can yield good results, see figure (3), but typically require good initialization (which can be performed using methods like shape contexts).

4 Deformable Templates for Detecting Objects

An important class of models can be defined in terms of probability distributions defined over graphs, see figure (5). The formulation is again described on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with state variables $\mathbf{W}$ defined on the nodes $\mathcal{V}$, where the state w_μ of node μ represents the position (and possibly orientation) of a point, or part, of the object. The unary potentials $\lambda^D \cdot \phi(w_\mu, \mathbf{I})$ specify how points/parts of the object relate to the image – e.g., some points on the boundary of objects may correspond to edges in the image intensity, while others may be modeled by interest points such as corners. The edges $\mathcal{E}$ specify which points/parts of the object are directly related and the binary potentials $\lambda^P \cdot \phi(w_\mu, w_\nu)$ model the spatial relations – e.g., capturing the overall shape of the object.

This can be expressed by a similar distribution:

$$P(\mathbf{w}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp\{-\sum_{\mu \in \mathcal{V}} \lambda^D \cdot \phi_\mu(w_\mu, \mathbf{I}) - \sum_{\mu\nu \in \mathcal{E}} \lambda^P \cdot \psi_{\mu\nu}(w_\mu, w_\nu)\}, \tag{6}$$

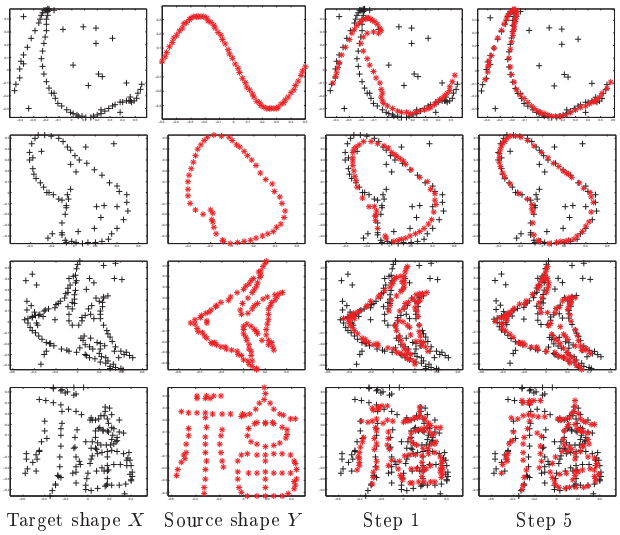

Figure 3: Examples of matching results. The target shape is the data and the source shape is the model. The quality of the matching improves with the number of steps of the algorithm.

where the main differences are the state variables w_μ at the nodes and the graph structure, see figure (5). Dynamic programming can correctly detect the hand, see figure (**??**). The model parameters λ can be learnt as described in the next lecture.

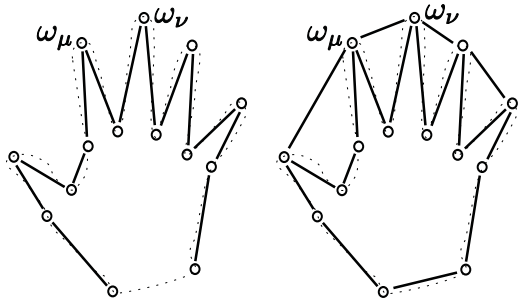

Figure 4: A deformable template model of a hand without closed loops (left) and with closed loops (right).

Inference requires finding the most probable states w. There are several differences to previous probability models we have studied. Firstly, the graph structures are often fairly simple and may even have no closed loops, see figure (5). Or if they do have closed loops we can use algorithms like junction trees which expresses the models in terms of triangles without closed loops (figure needed here – e.g., Amit). In general, the number of closed loops is small and manageable. Secondly, each state variable can take a very large number of values. For example, each node may correspond to an image feature anywhere in the image. This means that even if the graph structure has no closed loops algorithms like dynamic programming can take a long time to converge (convergence is quadratic in the number of possible states. Hence pruning is

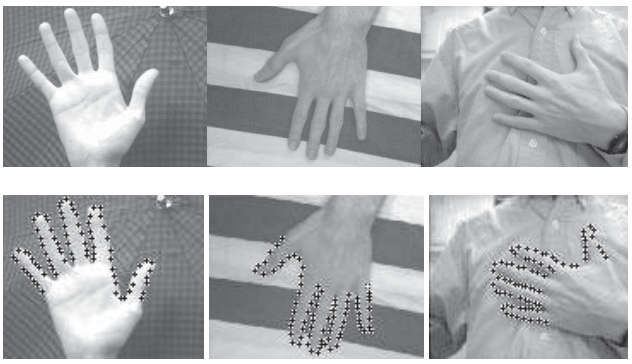

Figure 5: Examples of hand images and the final matches of the model to the image, indicated by the crosses.

sometimes needed to ensure that dynamic programming converges quickly. Curiously the first attempts to use dynamic programming for these types of problems were rejected by conference reviewers – Coughlan and Yuille, Felzenszwalb et al.).

Most of the original models of this type were hand-specified although stochastic sampling was sometimes used to check if the model parameters were set correctly.