Lecture 17: Active Appearance Models and FORMS

A.L. Yuille

March 22, 2012

1 Introduction

This lecture covered several topics. Some of them – active bases, active appearance models, FORMs – are described in more detail in handouts. These notes also describe grammars, which were only partially discussed in lecture 18.

2 Active Appearance Models

AAM's model the intensity properties of objects allowing for spatial warps. The simplest model is:

$$I(x) = T(\phi(x)), \text{ where } \phi(x) = x + \sum_{\mu} \alpha_\mu B_\mu(x). \tag{1}$$

Here $T(.)$ is an intensity template for the object and $\phi(x)$ is a spatial warp. The $\{B_\mu(.)\}$ are basis functions of the warps and the $\{\alpha_\mu\}$ are the coefficients. The basis function are fixed and the α's specify different examples of the object. For example $T(.)$ is the image of a face and the $\phi(.)$ allow warps to get different shapes of faces. Note that the assumption that the warps can be expressed in terms of basis functions is limited. It is plausible for limited deformations of faces but a very large number of eigenvectors are required to model the movement of lips when speaking (Bregler and Omohundro). It is also not a good way to describe the shape changes of an articulated deformable model – like a person or a cow walking – which are better modeled by parts (e.g., lower leg, upper leg, torso, arms, head) joined at joints, see figure (1) and later part of this lecture.

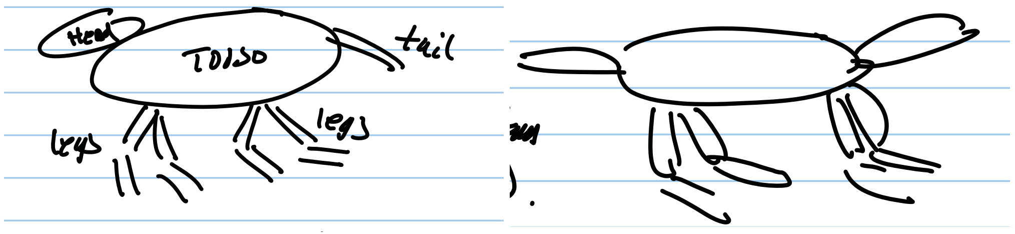

Figure 1: AAMs are not good models for articulated objects that contain parts – head, upper legs, lower legs, torso – that move semi-independently. AAMs are better at describing the variability of each part separately.

The basis functions B_μ can be learnt in several ways from a set of different example images $\{I^a(x) : a \in \mathcal{A}\}$: (I) Fully supervised (e.g., Cootes and Tayler), where the correspondence is specified between points in the images – e.g., we know that points x^a in image $I^a(x)$ correspond to points x^b in image $I^b(x)$ (or we know the correspondence for a set of labeled points and can interpolate the rest). Then we can estimate the basis functions by doing Principal Component Analysis on the $\{x^a : a \in \mathcal{A}\}$. (II) Unsupervised, either by first estimating the correspondence between images by using a generic model (e.g., by minimizing a term like $\sum_x |I^a(x) - I^b(\phi(x))| + K \sum_x |\nabla \phi(x)|$ which attempts to estimate the warp $\hat{\phi}(x)$ by matching points with similar intensities while requiring the warps to spatially smooth – Hallinan), or by the methods described below (Kokkinos and Yuille).

A more advanced model involves a richer model for the image appearance – e.g., that, before warping, the image can be expressed as a linear combination of basis functions.

$$I(x) = \sum_i \beta_i C_i(x + \sum_\mu \alpha_\mu B_{mu}(x)), \tag{2}$$

where the $C_i(.)$ are eigenvectors of appearance (with coefficients β_i) and the $B_\mu(.)$ are the eigenvectors of the spatial warps (with coefficients α_μ).

This can be thought of as a generative model for images with energy function:

$$E(\beta, \alpha : C, B] = \sum_x \{I(x) - \sum_i \beta_i C_i(x + \sum_\mu \alpha_\mu B_\mu(x))\}^2. \tag{3}$$

and a probability distribution:

$$P(I|\beta, \alpha; C, B] = \frac{1}{Z} \exp\{-E(\beta, \alpha : C, B]\}. \tag{4}$$

A prior $P(\beta, \alpha)$ can also be imposed on the model parameters (note, this model assumes that the image is generated with additive gaussian noise, hence the quadratic energy function).

The inference task is to estimate the coefficients β_i, α_μ for a specific image. The learning task for AAMs is to learn the basis functions C_i, B_μ from training data, which will be described below.

The inference task can be performed by an alternating algorithm which minimizes the energy E with respect to the β_i and the α_μ in alternation (with the other fixed). Minimizing with respect to the β_i reduces to solving linear equations. Minimizing with respect to the α_μ can be performed by steepest decent using multi-scale to blur the images and hopefully prevent the algorithm from getting stuck in local minima (multi-scale is commonly used in computer vision – by blurring out the small scale structure in the image it leaves the coarse-scale structure which is likely to be less ambiguous for matching problems). But this procedure requires good initialization to converge to a good minimum.

Learning an AAM model can, in theory, be performed by using the generative model in equation (4) and then selecting β, C, α, B to maximize

$$\prod_{a \in \mathcal{A}} P(I^a | \beta^a, \alpha^a; C, B), \tag{5}$$

with respect to C_i, B_μ and α^a_μ, β^a_i (i.e. the basis functions B, C are learnt for the entire dataset, but the coefficients α, β are estimated for each image).

But performing EM on equation (5) is very difficult due to the large number of local minima (and I do not think anybody has done it). But EM has been successfully applied to a simpler version (Kokkinos and Yuille 2007) which filters the image to extract features like edges and ridges, which are invariant to appearance changes, and then learning the warps basis functions and estimating the coefficients of the warps. This relies on using mean shift methods, see figure (2) to first roughly align the images. By using edges and ridges to remove appearance changes it has only a single appearance basis function $B_1(x)$ with coefficient $\alpha_1 = 1$. For details, see handout.

3 Models with Semantic Parts

For articulated objects it is usually better to represent the object as a set of parts. Each part can be represented by an AAM. An early example of this work is FORMs (Zhu and Yuille) which is applied to binary images of the silhouettes of objects.

Each object is composed of a grammar or parts, see figure (3). Each part is modeled based in its symmetry axis, see figure (4).

FORMS was tested only on binary shapes (cite enormous amount of material on these topics). But similar methods can be applied to real images, see figure (5) (from Kokkinos and Yuille 2007).

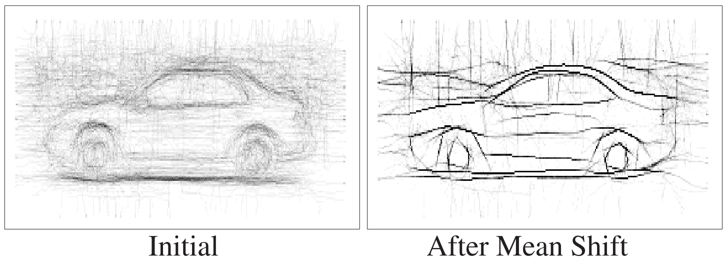

Figure 2: The mean shift algorithm (a variant which moves object edges only in the direction perpendicular to the edges) is used to perform alignment of the images in a pre-processing step.

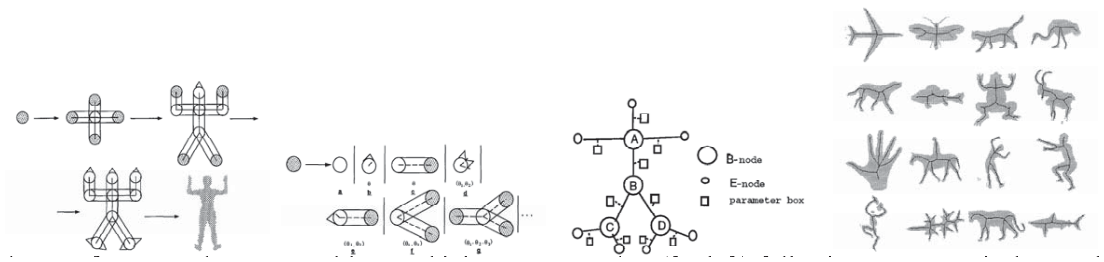

Figure 3: A human figure can be generated by combining parts together (far left) following a grammatical procedure (left). The result can be expressed as a graphical model (right) and the same approach can be applied to a range of other objects (far right).

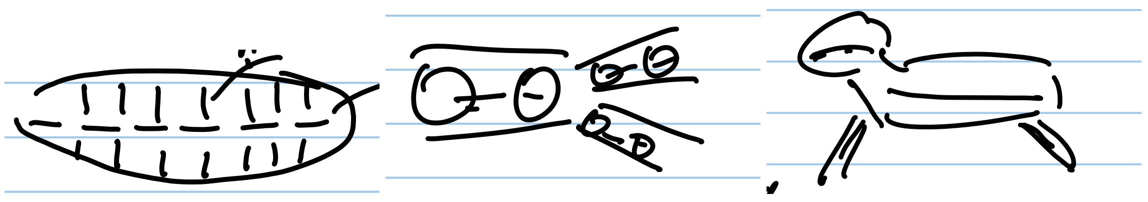

Figure 4: Left Panel: Each part can be represented in terms of basis functions of the radius from the symmetry axis. Center Panel: the symmetry axis is detected by rolling a ball from point of high curvature so that the boundaries of the ball touch the side walls. When the object splits into parts the symmetry axis bifurcates, indicating the presence of a split. Right Panel: this gives a decomposition of the object into parts where each part has one symmetry axis and the splits are at the bifurcations of the symmetry axis.

4 Grammars

Not covered in lecture. A small part of this material was discussed in lecture 18.

Grammars provide a natural way to model many phenomena in vision. Images, objects, and scenes can be thought of as being built out of a vocabulary of elementary components using a set of stochastic rules which allow certain configurations but prohibit others. For example, Biederman's Geon theory (Biederman 1987) can be thought of as providing a grammar of objects in terms of elementary components. The FORMS system (Zhu and Yuille 1996) is one example where a grammar is defined that can build deformable articulated objects, such as people and birds, from elementary components which can yield the heads, arms and torso, see figure (3). The Gestalt rules which enable humans to perceptually group components, see figure (6) may also be modeled in terms of compositional rules which

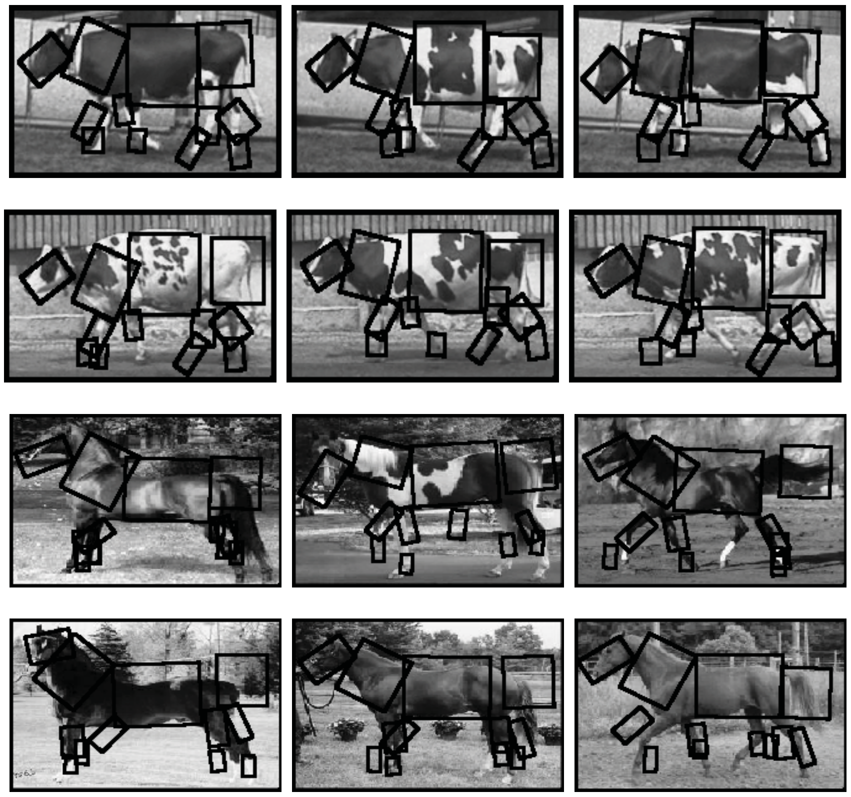

Figure 5: Detecting cows using part-based models.

relate closely to grammars (see discussion in Mumford and Desolneux 2010). For example, tokens in the image (e.g. edge segments) can be grouped by *proximity*, *alignment*, and *parallelism*, while image regions can be grouped based on *proximity*, *similar color or texture*, and *symmetry*. Many Gestalt phenomena are beautifully demonstrated in Kanizsa (1979).

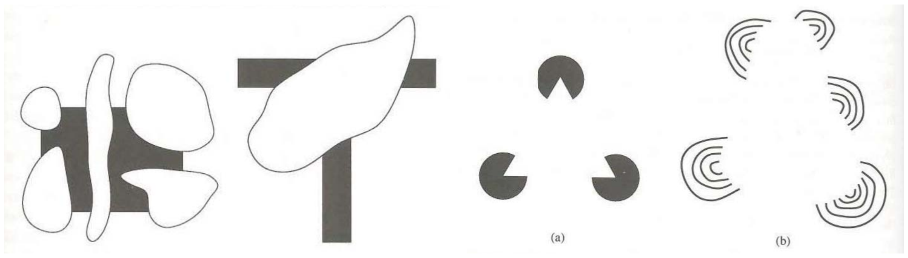

Figure 6: Gestalt laws (left panel) can be used to help detect the letter T and the square using color (i.e. black or white) and alignment despite the present of occluding white surfaces which obscure parts of the objects. Similarly Gestalt laws can be used to explain why we perceive white surfaces in the two images on the right despite the lack of direct visual evidence for them. These examples are from Kanizsa 1979.

There are, however, several differences between grammars used in vision and those which have been successfully applied to natural language. Firstly, language is a human construct which has evolved to enable humans to communicate with each other. Hence it is reasonable to believe that there are considerable syntactic regularities in language which can be modeled by grammars. By contrast, visual stimuli are generated by light rays reflected from objects in the visual scene and so have much more complex structures with less apparent regularities. Hence grammars for vision tend to be more complex and there is less general agreement and which forms they will take. There are, of course, some exceptions where there is some underlying process that leads naturally to certain types of visual structures – e.g, the ”grammar of animal shapes” derives ultimately from the the same underlying physical and biological processes (Zhu and Yuille), the ”grammar of buildings” follows the set of rule invented by architects and engineers, and the ”grammar of text documents” uses basic elements like paragraphs, lines, and figures (cite VIOLA!!). Secondly, language has a one-dimensional structure while vision is two-dimensional. This is not a conceptual barrier but it does mean that it is easier to perform learning and inference for language by exploiting the one-dimensional structure (e.g., by dynamic programming). Thirdly, there is no natural analogy in vision to the distinction between syntax and semantics which occurs in language. Nevertheless there are considerable conceptual similarities between language and vision. As argued by Mumford (Mumford and Desolneux 2010), the key ingredients of grammars are the existence of shared, or re-usable, parts which are often combined hierarchically.

We now proceed more formally by defining a stochastic context free grammar (SCFG) as used in language and modifying it to make it suitable for vision. This will lead to compositional and AND/OR graph models.

SCFGs are defined by a set of rules $R \in \mathcal{R}$, non-terminal nodes $l \in \mathcal{N}$, and terminal nodes $l \in \mathcal{T}$. There is a root node $l_0 \in \mathcal{N}$. A rule R_l is applied to a non-terminal node l to generate a set of k nodes $l_1, ..., l_k \in \mathcal{N} \bigcap \mathcal{T}$. There is a probability distribution $P_l(R_l)$ defined over the rules so that $\sum_{R_l} P_l(R_l) = 1$. Starting at the start node S and applying rules, sampled from the distribution, will produce a parse tree whose leaf nodes are terminals (technical conditions are required to ensure that the parse trees will be almost certainly finite). The SCFG is specified by a set of values $\{\mathcal{T}, \mathcal{N}, S, \mathcal{R}, \{P_l\}\}$. A specific parse tree β consists of a set of nodes $\mathcal{V}$ where each node has a label $L(\nu) \in \mathcal{N} \bigcap \mathcal{T}$. The probability of this tree is given by:

$$P_{tree}(\beta|\text{root } l_0) = \prod_{\nu \in \mathcal{V}} P_{L(\nu)}(R(\nu)). \tag{6}$$

In vision applications the terminal nodes correspond to image pixels or to features, or tokens, extracted from images such as edges. For objects the non-terminal nodes will correspond to parts and subparts of objects. For example, in figure (7)(Kokkinos and Yuille –), the object is composed of three parts of the car (front, middle, and back) while each part is composed from a number of contours, which are themselves composed of tokens in the image. In this case, the root node is the *Object* node and there is a single rule R_{l_0} which is applied to l_0 and which generates three *part* nodes. Other rules are applied to the part nodes to generate *contour* nodes. Finally, rules are applied to the contour nodes to generate *token* nodes in the image. This is a very simple grammar and we will describe a more complex variant later.

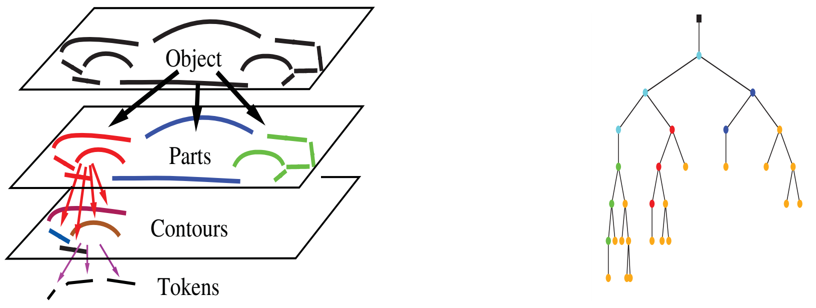

Figure 7: Hierarchical Model for a Car (Left Panel). A parse tree (Right Panel).

There are, however, limitations to the SCFG described in equation (6). It completely ignores the spatial positions of the parts/contours and other attributes which they can have (e.g., their color). This can be addressed by introducing attributes at each node which can represent, for example, properties such as the spatial positions of parts of the car shown in figure (7). These attributes could be more general and include color, texture, or precise specifications about the type of part (e.g. the front of a Prius or of a Mercedes-Benz). In addition, we require that the attributes of nodes may be related – for example the positions of the front, middle, and back parts of the car must obey spatial regularity constraints. These can be imposed by introducing edges between nodes whose attributes are related (e.g., in figure (7) we could put edges connecting the parts in order to impose constraints on their relative positions).

More formally, following the exposition in Mumford and Desolneux (2010) we introduce *attribute* variables $\alpha(l)$ for each node $l \in \mathcal{N} \bigcap \mathcal{T}$. Edges $e(\mu, \nu) \in \mathcal{E}$ are added to the graph in order to connect nodes μ, ν whose attributes we want to relate. These edges will introduce closed loops into the graph structure and so we will now have parse trees instead of parse graphs. We modify the rules $mathcalR$ so that they generate a set of attributed nodes together with a set of edges to specify the relations. In addition, we require that the attributes of a parent node $\alpha(\mu)$ can be expressed as a deterministic function $f(.)$ of the attributes of its child nodes $\mu_1, ..., \mu_n$ – $\alpha(\mu) = f(\alpha(\mu_1), \alpha(\mu_2), ..., \alpha(\mu_n))$. For example, in figure (7) the position of the car is a deterministic function of the position of its parts. For the Gestalt examples shown in figure (6). Finally, we can place probability distributions over these graphs by defining *binding factors* $B_e(\alpha(\mu_i), \alpha(\mu_j), \alpha(\mu))$ where (μ_i, μ_j) is an edge connecting two child nodes of μ (e.g. $B_e(, ., .)$ specifies the likely relative positions of the parts of the car). These are combined with the earlier SCFG in equation (6) to yield a distribution for the entire parse graph ω (which include the nodes $\mathcal{V}$, the edges $\mathcal{E}$, the labels and attributes of the nodes $(L(\nu), \alpha(\nu)) : \nu \in \mathcal{V}$:

$$P_{tree}(\omega) \propto \prod_{\nu \in \mathcal{V}} P_{L(\nu)}(R(\nu)) \prod_{(\nu_i, \nu_j) \in \mathcal{E}} B_e(\alpha(\mu_i), \alpha(\mu_j), \alpha(\mu)). \tag{7}$$

Mumford and Desolneux (2010) sketch how probabilistic grammars of this type can be applied to model simple Gestalt phenomena. For example to grouping together two edge contours, Γ_1 and Γ_2 to form a bigger contour Γ. In this case, a natural choice for the binding factor is the likelihood ratio $\frac{P(\Gamma)}{P(\Gamma_1)P(\Gamma_2)}$ of the probability of modeling the two contours jointly (i.e. by $P(\Gamma)$) compared to the probabilities of modeling them separately by $P(\Gamma_1)$ and $P(\Gamma_2)$. This has a natural interpretation in terms of minimal length encoding theory (Rissanen 1989) where you group two elements, Γ_1, Γ_2, provided the cost $-\log P(\Gamma)$ of encoding them jointly is less than the cost $-\log P(\Gamma_1) - \log P(\Gamma_2)$ of encoding them separately. These ideas have been explored much further in Geman's work on compositional modeling (Geman et al. 2002). In many applications the attributes of the nodes, and the relationships between the attributes of nodes connected by edges, carries more useful information than the nature of the terminal nodes themselves. For example, in the car example the parts are the child nodes of the root (car) node. In other words, they are generated by only one rule (e.g., the one applied to the root) and their most important properties are their positions (i.e. their attributes).

These modified SCFG's have considerable similarities to AND/OR graphs (Zhu and Mumford 2006) which represent objects by graphical models where the nodes are of two types – AND's and OR's. The AND nodes build extra nodes by composition. For example, see figure (8)(top left) from (L. Zhu et al. 2010), the root node of a model of a Baseball player is the AND of the head, the torso, and the legs (the arms are not modeled in this application).

To define these classes of models we must introduce some additional notation. We consider probability models formulated over graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the vertices and $\mathcal{E}$ denotes edges. State variables y_μ are defined at the nodes $\mu \in \mathcal{V}$. The variable $\mathbf{y} = \{y_\mu : \mu \in \mathcal{V}\}$ describes the state of the entire graph. The edges $\mathcal{E}$ in the graph specify which nodes are directly connected and define the cliques $\mathcal{C}l$ – i.e. for all $\mu_1, \mu_2 \in \mathcal{C}l$ then $(\mu_1, \mu_2) \in \mathcal{E}$. We index $\mathcal{C}l$ by γ and let $\mathbf{y}_\gamma$ represent the state of all variable in clique γ. Potential functions $\boldsymbol{\phi}_\gamma(\mathbf{y}_\gamma, \mathbf{I})$ and parameters $\mathbf{w}_\gamma$ are defined over the cliques $\gamma \in \mathcal{C}l$ (note that these potentials allow direct input from the data $\mathbf{I}$). And and Or nodes will have different types of potential functions. We also define potentials which depend on the states of individual graph nodes and are dependent only on the input $\mathbf{I}$. An example of the probability models are given in figure (8). The nodes $\mu \in \mathcal{V}$ represent subparts of the object where states y_μ specify whether the part is present/absent and, if it is present, its position and other attributes (e.g., orientation and size). For our vision applications the graphs are typically organized in layers where a node at one layer is connected to a subset of nodes at the lower level forming a clique. These cliques are of two types: (i) AND-cliques where the upper node represent the position/attributes of a part which is composed from subparts represented by the lower nodes, and (ii) Or-cliques where the upper node takes the same state as one of the lower nodes – i.e. it chooses between them.

How do these AND/OR models relate to the stochastic grammars defined in equations (6,7)? The state variables y_μ correspond to the attributes of the nodes in the stochastic grammars. The labels correspond to "Head", "Torso", "Leg" as shown in figure (8). The forms of the AND-cliques and the OR-cliques show that the states (i.e. attributes) of

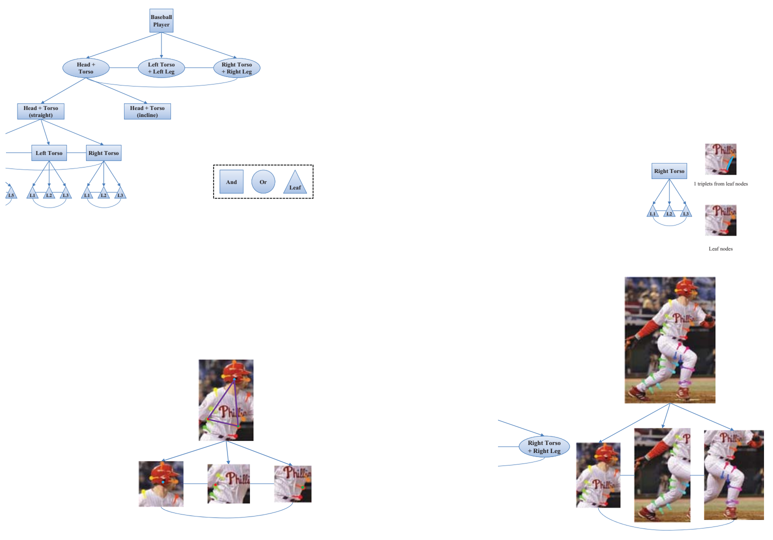

Figure 8: The AND/OR Graph Model. The Baseball player is an AND of the head and torso, and left and right legs, but the head is an OR of straight head and torso or an inclined head and torso (top left).

the parent nodes are indeed deterministic functions of the states of the child nodes. Hence the potential terms $\phi(y_\gamma)$ correspond to the binding factors.

More formally, we specify a conditional distribution on the probability of the state variables $\mathbf{y}$ conditioned on the input image $\mathbf{I}$:

$$P(\mathbf{y}|\mathbf{I};\mathbf{w}) = \frac{1}{Z[\mathbf{w},\mathbf{I}]} \times \exp\{\sum_{\mu\in\mathcal{V}} \mathbf{w}_\mu \cdot \boldsymbol{\phi}(y_\mu,\mathbf{I}) + \sum_{\gamma\in\mathcal{C}l} \mathbf{w}_\gamma \cdot \boldsymbol{\phi}_\gamma(\mathbf{y}_\gamma,\mathbf{I})\}. \tag{8}$$

To use this distribution, for detecting the poses of baseball players, requires learning the parameters $\mathbf{w}$ of the probability model and also estimating the most probable state $\mathbf{y}^* = \arg\max P(\mathbf{y}|\mathbf{I};\mathbf{w})$. As discussed in L. Zhu et al. (2011) the most probable state can be estimated by using the junction trees learning algorithm (which is an extension of dynamic programming – has this, or message passing been mentioned in the book?). The parameters $\mathbf{w}$ can be estimated by standard methods (do we mention them?) although some approximations are required (L. Zhu et al. 2011).

One application is to estimate the pose of baseball players, see figure (8). The AND/OR graphs are designed to model multiple poses of the object using a single graphical model. The object is constructed by composition which involves AND-ing and OR-ing elementary parts together to form the object. The OR-ing operation allows the model to deal with different poses of the object. This is formulated as probability defined on a graph with state variables $y_\mu = (p_\mu, t_\mu)$, where p_μ represents the pose (i.e. position, orientation, and scale) of an object part, and t_μ is a binary variable which indicates if the part is selected or not. The layers of the graph alternate between AND nodes, for which the part is a composition of the sub-parts of its child nodes (see figure (8), and OR nodes which requires a node to select one of its child nodes (i.e. a 'head' must be 'head-up' or 'head-down'). The t variables perform the selection for the OR nodes, which can be thought of as switch variables. The graph structure can change topology due to the state of the t variables (i.e. the selections made at the OR nodes). This means that a graph containing only 40 (check !!) nodes can have 100 different topologies, which correspond to 100 different poses. An alternative strategy would be to have 100 different models – one for each pose of the object – and perform pose estimation by selection between these different models. The AND/OR graph is more compact and efficient, because it is able to *share parts* (i.e. the most elementary components) between different poses. The potentials $\boldsymbol{\phi}_\gamma(\mathbf{y}_\gamma)$ impose spatial relations on the relative positions of parts, their composition from more elementary parts, and the selection choices made at OR nodes. The data potentials $\boldsymbol{\phi}^D(y_\mu,\mathbf{I})$ relate the node at level $l = 1$ to the image (e.g., by encouraging the boundaries of parts to correspond to edge-type features in the image).

The graph structure for this model contains some closed loops because of the graph edges connecting siblings (e.g. the spatial representations between parts at the same scale). But the numbers of closed loops is small enough that we can perform *inference* use dynamic programming with the junction tree algorithm. The size of the state space, however, is very large because there are many possible values for the pose p_μ of each part. So we perform pruning based on the energy and by surround suppression (penalizing states which are too similar to each other). We emphasize that we are performing inference over the state variables *including* the topology (i.e. the t_μ's), which means that we do inference over 100 different models efficiently be exploiting the re-use of the elementary parts. For more details see (L. Zhu et al. 2011).

The graph structure of the graphical model of the AND/OR graph was specified by hand. But it is more desirable to learn these types of models automatically from data. This is a very challenging problem because it requires a search over the enormous space of models as well as over the model parameters.