Mixtures of Trees for Object Recognition

Sergey Ioffe David Forsyth

Abstract

Efficient detection of objects in images is complicated by variations of object appearance due to intra-class object differences, articulation, lighting, occlusions, and aspect variations. To reduce the search required for detection, we employ the bottom-up approach where we find candidate image features and associate some of them with parts of the object model. We represent objects as collections of local features, and would like to allow any of them to be absent, with only a small subset sufficient for detection; furthermore, our model should allow efficient correspondence search. We propose a model, Mixture of Trees, that achieves these goals. With a mixture of trees, we can model the individual appearances of the features, relationships among them, and the aspect, and handle occlusions. Independences captured in the model make efficient inference possible. In our earlier work, we have shown that mixtures of trees can be used to model objects with a natural tree structure, in the context of human tracking. Now we show that a natural tree structure is not required, and use a mixture of trees for both frontal and view-invariant face detection. We also show that by modeling faces as collections of features we can establish an intrinsic coordinate frame for a face, and estimate the out-of-plane rotation of a face.

1. Introduction

One of the main difficulties in object recognition is being able to represent the variations in object appearance and to detect objects efficiently. Template-based approaches (e.g., to detect frontal views of faces [8, 11] and pedestrians [7]) are not general because they do not allow object parts to move with respect to each other. An alternative is to use a model that, instead of regarding an object as rigid, models local appearance of parts and the relationships among the parts. Such representations have been used extensively to represent people (e.g. [1, 3]) and have been applied to faces [10, 13, 14]. Detecting articulated objects requires a search of a very large configuration space, which, in the context of tracking, is often made possible by constraining the configuration of the object in one of the frames. However, if our object detector is to be entirely automatic, we need a method that allows us to explore the search space efficiently.

Among the ways to make the search efficient is the *bottom-up* approach, where the candidate object parts are first detected and then grouped into arrangements obeying the constraints imposed by the object model. Examples in face detection include [13] and [14], who model faces as flexible arrangements of local features. However, if many features are used to represent an object, and many candidate features of each type are found in the image, it is impractical to evaluate each feature arrangement, due to the overwhelming number of such arrangements. The correspondence search, where a part of the object model is associated with some of the candidate features, can be made more efficient by pruning arrangements of a few features before proceeding to bigger ones [5]. Alternatively, the model can be constrained to allow efficient search. One example of such a model is a *tree*, in which correspondence search can be performed efficiently with dynamic programming (e.g. [3, 4]).

Representing an object with a fixed number of features makes recognition vulnerable to occlusions, aspect variations, and failures of local feature detectors. Instead, we would like to model objects with a large number of features, only several of which may be enough for recognition. To avoid the combinatorial complexity of the correspondence search, we propose a novel model that uses a *mixture of trees* to represent the *aspect* (which features are present and which are not) as well as the relationships among the features; by capturing conditional independences among the features composing an object, mixtures of trees allow efficient inference using a Viterbi algorithm.

Some objects, such as human bodies, have a natural tree representation (with the torso as the root, for example), and we have shown [4] that mixtures of trees can be used to represent, detect and track such objects. However, our model is not limited to articulated objects, and, because we learn the tree structure automatically, can be used for objects without an intuitive tree representation. We illustrate this by applying our model to face detection. By using a large number of features only a few of which are sufficient for detection, we can model the variations of appearance due to different individuals, facial expressions, lighting, and pose.

In section 2, we describe mixtures of trees, and show how to model faces with a mixture of trees in section 3. We use our model for frontal (section 4) and view-invariant (section 5) face detection. The feature arrangements representing faces carry implicit orientation information. We illustrate this in section 6, where we use the automatically extracted feature representation of faces to infer the angle of out-of-plane rotation.

2. Modeling with mixtures of trees

Let us suppose that an object is a collection of K primitives, $\{X_1 \ldots X_K\}$, each of which can be treated as a vector representing its *configuration* (e.g., the position in the image).

Given an image, the local detectors will provide us with a finite set of possible configurations for each primitive X_k. These are *candidate primitives*; the objective is to build an *assembly* by choosing an element from each candidate set, so that the resulting set of primitives satisfies some global constraints.

The global constraints can be captured in a distribution $P(X_1 \dots X_K)$, which will be high when the assembly looks like the object of interest, and low when it doesn't. Assuming exactly one object present in the image, we can localize the object by finding the assembly maximizing the value of P. In general, this maximization requires a combinatorial correspondence search. However, if $P(X_1 \dots X_K)$ is represented with a tree, correspondence search is efficiently accomplished with a Viterbi algorithm. If there are M candidate configurations for each of the K primitives, then the search takes $O(KM^2)$ time, whereas for a general distribution P the complexity would be $O(M^K)$.

2.1. Learning the tree model

In addition to making correspondence search efficient, the conditional independences captured in the tree model simplify learning, by reducing the number of parameters to be estimated, due to the factorized form of the distribution:

$$P(X_1 \dots X_K) = P(X_{\text{root}}) \prod_{k \neq \text{root}} P(X_k \mid \text{Pa}_k),$$

where X_{root} is the node at the root of the tree, and Pa_k denotes the parent of the node X_k. Learning the model involves learning the structure (i.e., the tree edges) as well as the parameters of the prior $P(X_{\text{root}})$ and conditionals $P(X_k \mid \text{Pa}_k)$. We learn the model by maximizing the log-likelihood of the training data, which can be shown to be equivalent to minimizing the entropy of the distribution, subject to the prior $P(X_{\text{root}})$ and conditionals $P(X_k \mid \text{Pa}_k)$ being set to their MAP estimates. The entropy can be minimized efficiently [2, 12] by finding the minimum spanning tree in the directed graph, whose edge weights are the appropriate conditional entropies.

2.2. Mixtures of trees

It is difficult to use a tree to model cases where some of the primitives constituting an object are missing – due to occlusions, variations in aspect or failures of the local detectors. Mixtures of trees, introduced in [6], provide a solution. In particular, we can think of assemblies as being generated by a mixture model, whose class variable specifies what set S of primitives will constitute an object, while conditional class distributions $P_S(\{X_k : k \in S\})$ generate the configurations of those primitives. The mixture distribution is

$$P(\{X_k : k \in S\}) = \pi(S) P_S(\{X_k : k \in S\})$$

where $\pi(S)$ is the probability that a random view of an object consists of those primitives. This mixture has 2^K components – one for each possible subset S of primitive types. Learning a mixture of trees involves estimating the mixture weights $\pi(S)$, as well the structure and the model parameters for each of the component trees.

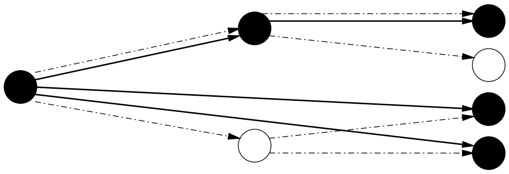

Figure 1: *Using a generating tree to derive the structure for a mixture component. The dashed lines are the edges in the generating tree, which spans all of the nodes. The nodes of the mixture component are shaded, and its edges (shown as solid) are obtained by making a grandparent "adopt" a node if its parent is not present in this tree (i.e., is not shaded). Thus mixture components are encoded implicitly, which allows efficient representation, learning and inference for mixtures with a large number of components. The structure of the generating tree is learned by entropy minimization.*

2.3. Mixtures of trees with shared structure

Explicitly representing 2^K mixture components is unacceptable if the number of object parts K is large. Instead, we use a single *generating tree* which is used to generate the structures of all of the mixture components.

A *generating tree* is a directed tree T whose nodes are $X_1 \dots X_K$, with X_{root} at the root. It provides the structure of the graphical model representing $\pi(S)$: $\pi(S) = P([X_{\text{root}}]) \prod_{k \neq \text{root}} P([X_k] \mid [\text{Pa}_k])$, where $[X_k]$ denotes the event that X_k is one of the primitives constituting a random view of the object, and the distributions are learned by counting occurrences of each primitive and pairs of primitives in the training data. For a subset S of object part types, the mixture component P_S contains all the edges $(X_j \to X_k)$ such that X_j is an ancestor of X_k in the generating tree, and none of the nodes on the path from X_j to X_k is in the set $\{X_k : k \in S\}$. This means that, if the parent of node X_k is not present in a view of the object, then X_k is "adopted" by its grandparent, or, if that one is absent as well, a great-grandparent, etc. If we assume that the root X_{root} is *always* a part of the object, then P_S will be a tree, since X_{root} will ensure that the graphical model is connected. An example of obtaining the structure of a mixture component is shown in figure 1. We ensure connectivity by using a "dummy" feature as the root X_0, representing the rough position of the assembly; candidate root features are added to test images at the nodes of a sparse grid.

The distribution P_S is the product of the prior $P(X_{\text{root}})$ and conditionals $P(X_k \mid X_j)$ corresponding to the edges of the tree representing P_S. We learn the structure of the generating tree T that minimizes the entropy of the distribution. We are not aware of an efficient algorithm that produces the minimum; instead, we obtain a local minimum by iteratively applying entropy-reducing local changes (such as replacing a node's parent with another node) to T until convergence.

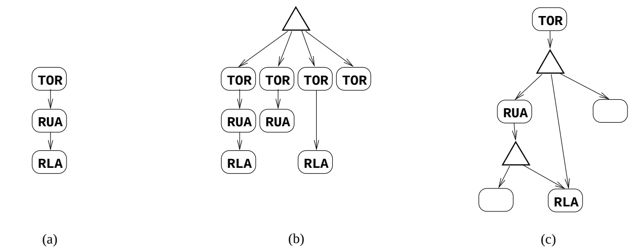

Figure 2: *Converting a mixture of trees into a graph with* choice nodes*, on which correspondence search is performed using dynamic programming.* (a) *A fragment of the generating tree for a person, containing the torso, right upper arm, and right lower arm.* (b) *The mixture of 4 trees that results if we require the torso to be always present. The triangle represents a* choice node*; only one of its children is selected, and the mixture weights are given by the model of the aspect.* (c) *The shared structure of the mixture components is captured using extra choice nodes. The empty nodes correspond to adding no extra segments; the mixture weight corresponding to each child of a choice node is derived from the prior for an aspect.*

2.4. Grouping using mixtures of trees

To localize an object in an image, we find the assembly that maximizes the posterior $\Pr(\text{object} \mid \text{assembly})$ or, equivalently, the Bayes Factor $B = P(\{X_k\})/P_{neg}(\{X_k\})$ where the numerator is the probability of a configuration in a random view of the object, and the denominator is the probability of seeing it in the background. We model the background as a Poisson process: $P_{neg}(\{X_k : k \in S\}) = \prod_{k \in S} \alpha_k$ where α_k is the rate (or density) of the Poisson process according to which the primitives of type X_k are distributed in the background. Because of the independent structure of P_{neg}, the Bayes factor can be obtained by associating the term α_k^{-1} with each member of X_k's candidate set, and multiplying those terms into the likelihood $P(\{X_k\})$.

We perform the correspondence search using a Viterbi algorithm on tree T; at each node, we select not only the best primitives to choose from the children's candidate sets, but also the edges to be included in the tree (i.e., which parts constitute an object instance). This is equivalent to dynamic programming on a graph with *choice nodes*, illustrated in figure 2. This algorithm runs in time $O(KhM^2) = O(K^2M^2)$ where M is the number of primitives in each candidate set, K is the number of object parts, and h is the depth of the generating tree.

3. Learning the model of a face

Representing a face as an assembly of local features allows us to model both the relative rigidity of the facial features and the flexibility of their arrangements. Other approaches modeling faces with local feature arrangements (e.g. [13]) usually rely on a specific, small set of features, because they cannot handle missing features and lack an efficient grouping mechanism. With a mixture of trees, we can address these issues. Because of the efficient inference on mixtures of trees, we can use a very large number of features (≈ 150), but require only a few (≈ 10) to declare a detection. Therefore, we can handle occlusions and multiple aspects, and use the same model to represent all orientations of a face. However, the orientation is not discarded; instead, it is implicitly encoded by the mixture of trees. We can recover the pose by examining the feature arrangement obtained for a face image, and using the types of the features constituting an arrangement, as well as their geometric relationships, to estimate the orientation.

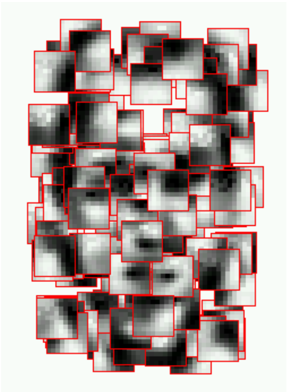

Figure 3: *Cluster centers found by grouping subimages extracted from training face images with the modified K-Means algorithm. The clustering procedure learns both the average grey-level appearance of each feature and its average warped position (according to which each cluster is positioned in the figure).*

3.1. Facial features

Each facial feature is represented as a small image patch; an assembly is a group of features satisfying some constraints imposed by the geometry of a face. For each feature type, the *candidate features* are image patches whose appearance is sufficiently similar to the "canonical" appearance of that feature.

To reduce the time it takes to find candidate features, each image – both training and test – is represented as a collection of small (9×9) image patches centered at interest points (found with the Harris operator, e.g. [9]). Local contrast normalization is applied to counter the variations in brightness and contrast due to different lighting.

Instead of manually specifying what features compose a face, we learn a set of features that are stable, (i.e., present in a large number of face images, at roughly the same place relative to the face), distinct from other features, and distinct from the background. First, we cluster the image patches in training images using the K-Means algorithm which we modified so that it learns not only the appearances but also the *warped positions* of the cluster centers; the *warp* for each training image is computed as an affine transforma-

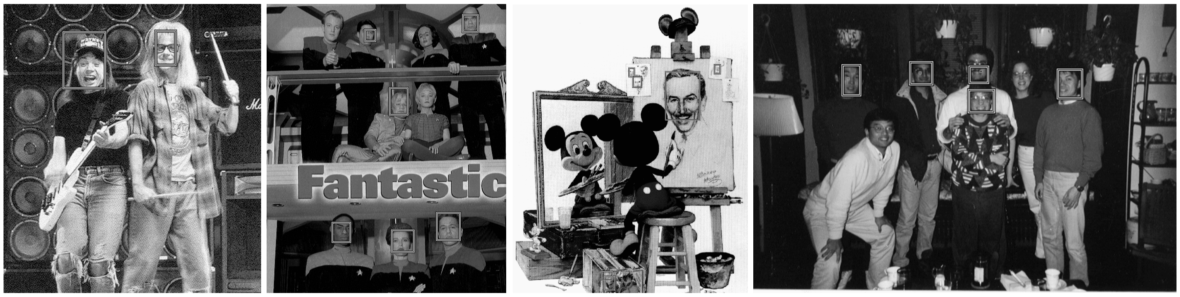

Figure 4: *Examples of faces detected in the test data (with the threshold $\lambda = 1$). The boxes indicate the bounding boxes of the feature assemblies representing the faces, and the dot shows the position of the root node in the mixture of trees, which indicates the “general position” of a face.*

tion that maps 3 “landmark points” on a face image to their canonical positions. The similarity between a patch and a cluster center is computed as the Euclidean distance between their pixel representations, subject to proximity between their warped positions. By clustering image patches, we convert each training image to an *assembly*; these assemblies are now used to learn the face model. Figure 3 shows the cluster centers obtained with our algorithm.

Because pixels within a patch are not independent, we represent each image patch with its projections onto several (e.g. 15) dominant independent components (found by applying PCA to the sub-images of face images). We can capture dependences among these projections conditional on the feature type and still maintain the linear complexity of the model with a tree-structured model. All conditionals in this model are Gaussian, and the tree structure is learned by maximizing mutual information [2].

3.2. Modeling feature arrangements

To model faces with a mixture of trees, we need to learn the pairwise relationships first, and then use entropy minimization to obtain the structure of the generating tree. Conditional probability tables for *feature visibility* are learned by simple counting. The distributions of the *relative positions* have the form $P(X_2 \mid X_1) = P(U_2, V_2 \mid U_1, V_1) = P(U_2 - U_1, V_2 - V_1)$ where $(U_2 - U_1, V_2 - V_1)$ is the displacement between the two features. We use a Gaussian to represent $P(X_2 \mid X_1)$.

To be able to detect faces, we need to learn the model of the background as well as that of a face. We can incorporate the background model into the efficient inference mechanism if it obeys the same independences as those captured in the mixture of trees modeling a face. We choose the simplest such model, in which the features detected in background are independent, and modeled with a Poisson process. The appearance of image patches in non-face images is modeled with a mixture of distributions of the same type as used to model facial features. This allows us to more accurately model the background image patches that look similar to facial features. The probability density of generating an assembly $\{X_k : k \in S\}$ in the background becomes $P_{neg}(\{X_k : k \in S\}) = \prod_{k \in S} \alpha_k P_0(X_k)$ where α_k is the rate of the Poisson process we assume to be generating the candidate features of type k.

$\ln \lambda$	-10	-3	0	3	5	10
Detection	90%	80%	76%	69%	66%	56%
False alarms	1364	279	129	50	22	1

Table 1: *Frontal face detection results. The database contained 117 images, with a total of 511 faces. We show the fraction of faces correctly detected, and the number of non-faces mistakenly detected, for different values of the threshold λ with which the posterior is compared*

To find face-like assemblies of image patches, we compute, for each feature type k and each image patch x_i, the probability $P_k(x_i)$ of seeing this patch in a random view of feature X_k, and the probability $P_0(x_i)$ of seeing the patch in a random view of a non-face. In maximizing the Bayes factor, we associate an extra multiplicative weight $P_k(x_i)/(\alpha_k P_0(x_i))$ with each feature k and patch x_i. In practice, we will make x_i a candidate for feature k only if the ratio $P_k(x_i)/P_0(x_i)$ is sufficiently large – a condition that does not hold for most patch/feature pairs.

4. Detecting frontal faces

We have used mixtures of trees to learn the model of the frontal faces. The training data was kindly provided by Henry Schneiderman. The training background images are chosen at random from the Corel image database.

We tested our face finder on images from the MIT and CMU face databases — 117 photographs with 511 frontal faces. Faces were detected at a range of scales, spaced by a factor of $2^{1/4}$. If two assemblies' bounding boxes overlap by more than a small amount, we remove the one with the smaller posterior. Table 1 shows the performance for our detector for some values of the threshold λ for the Bayes factor. Figure 4 shows some examples of faces we detected in test images. In figure 5, we show an example of our face detector applied to a large group photo (with λ set very low; this does not result in many false detections, since most of those are suppressed by the real faces).

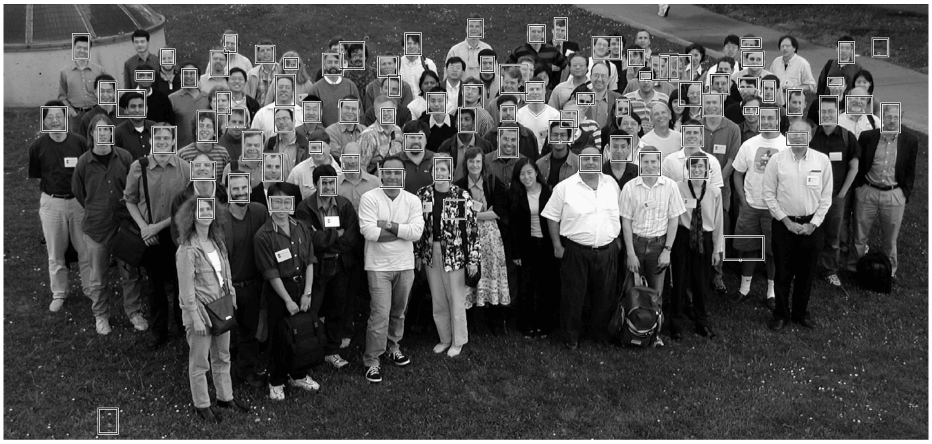

Figure 5: *Faces found in a large group photo. The threshold λ is set very low; most false detections are suppressed by the correctly detected faces. Out of 93 faces in the image, 79 were correctly detected, 14 were missed, and 5 false alarms occurred. Most of the missed faces were not found because they were smaller than the size of the training faces.*

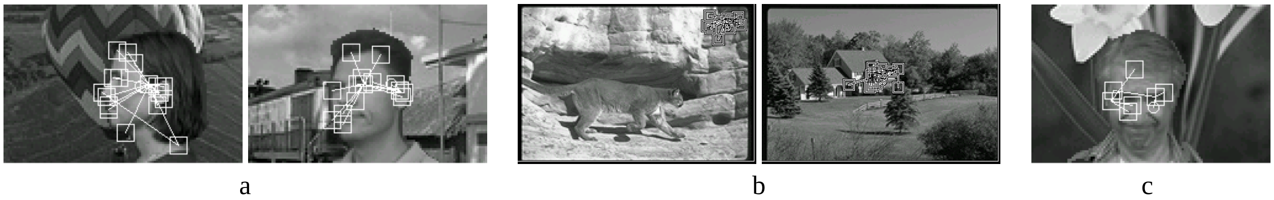

Figure 6: *Examples of the view-invariant face detector applied to faces and backgrounds.* (a) *examples of face assemblies detected correctly. The squares show the features in the assembly, the circle corresponds to the root node of the mixture of trees, and the edges show the edges of the mixture component corresponding to the assembly.* (b) *False detections in background images.* (c) *An example of a missed detection. Even though a feature assembly is found corresponding to a face, its posterior is too low to declare a face detection.*

5. View-invariant face detection

Out-of-plane face rotations suggest that a model that is able to represent aspect would perform well. Our data was kindly provided by M. Weber et al., authors of [13]. It contained faces of 22 subjects, photographed against a uniform background (which we synthetically replaced with random images from the Corel database) at 9 different angles, spaced by 15° and spanning the entire range between the frontal view and the profile; 18 to 36 pictures of each person, for different poses and facial expressions, were included. We randomly chose 14 individuals and placed all of their images into the test set, using the photographs of the remaining 8 subjects for testing.

Each face image was rescaled to be between 40 and 55 pixels in height, and each non-face image was a 128×192 image taken from the Corel database. For each image, we decide whether or not it contains a face by comparing the posterior of the highest Bayes factor feature arrangement with a threshold. The error rate ($(FalsePos + FalseNeg)/2$) ranged between 4% and 8%. Our performance is better than the about 15% error rates reported in [13] for a single detector trained and tested on the entire range of rotations, which shows that a mixture of trees is able to represent the variations of the face appearance as it rotates. In figure 6, we show examples of correctly detected faces, as well as false detections reported in background images.

6. Pose estimation

Representing a face with a large number of features, allowing any features to be absent, and modeling the way in which the feature visibilities and configurations affect one another, allows our face-detection system to be view invariant. However, the orientation is not discarded, but is instead implicitly encoded in the model. By examining the feature arrangement found for a face image, we can estimate an intrinsic coordinate frame for the face. The types of the features constituting a face, as well as their geometric configuration, can be used to derive correspondences between different views of a face, or between an image and a 3D model of the head, and also carry implicit aspect information. For example, having found a feature corresponding to the left eye, we know that the view is not the right profile.

To determine the pose of a face, we learn, for each feature X_k and each view direction θ, the probability $\Pr([X_k] \mid \theta)$ that this feature is present in the view. Our data set contains 9 different view directions, and the feature frequency information is captured in a $9 \times K$ table, where K is the number of available features. The entries of the table are estimated from the training face assemblies.

Given an assembly $A = \{X_k : k \in S\}$, we can compute the probability that it has been generated by a particular view θ: $\Pr(\theta \mid A) \propto P(A \mid \theta)\Pr(\theta) \propto \prod_{k \in S} \Pr([X_k] \mid \theta)$ for a uniform $\Pr(\theta)$. As our estimate of the pose, we use the expected value $\Theta = \sum_\theta \Pr(\theta \mid A)\theta$. If θ^* is the correct view angle, the estimation error is given by $\Theta - \theta^*$, and the RMS error is $\sqrt{\sum_{n=1}^{N} (\Theta_n - \theta_n^*)^2 / N}$, for N test images. In our experiments, the RMS error was 15°, i.e. on the average (and in fact for most test images) the estimated angle is within one angle step of the actual angle. Compare this with the RMS error of 39° that would result from always reporting the average face angle (45°).

7. Conclusions

Mixtures of trees allow us to represent objects as flexible collections of parts, where some part can be missing, and we model the aspect, geometric relationships among the parts, and the individual part appearances. Due to the conditional independences in the model, inference can be performed efficiently, in a bottom-up fashion, where candidate object parts are first detected and then grouped into arrangements using dynamic programming on the mixture of trees. In addition to being able to represent and track people, as we have shown in [4], mixtures of trees can model objects without an intuitive tree decomposition, and we have shown this by applying our model to frontal and view-invariant face detection.

Even though the results we have obtained for frontal face detection are slightly behind the state of the art, our model has the advantage that it can be used to do more than just detection. We can determine not only whether a face is present, but also the configuration of the face, i.e. what facial features are present, and where they are with respect to each other. Our model can use a large number of features (≈ 150), with only a few (≈ 10) needed for detection, and implicitly encodes the aspect; we have shown that we can recover the aspect information by examining the feature arrangements obtained for a face to estimate the out-of-plane rotation of a face. The future work includes using the feature representation of an object that mixtures of trees help us obtain for applications such as recognition of individuals, genders, or facial expressions (by comparing features of the same type in different faces, we do not have to rely on the two faces being in the same pose), matching features between two images of a face, or between an image and a 3D face model, and face tracking (we have already shown [4] that temporal coherence can be incorporated into our model).

Another important advantage of our model is that it is not tailored to a particular type of object: we have shown that it can be used for such diverse objects as human bodies and faces. One of the research directions is to use the model to represent other objects, or entire classes of objects. For example, just as we detect faces in an arbitrary view and then determine the pose, we could use a single mixture of trees to represent many kinds of animal, and use the resulting feature representation of an object in an image to determine what type of animal it is.

Acknowledgements

We thank Michael Jordan for suggesting mixtures of trees for object representation. This research was supported by NSF Graduate Research Fellowship to SI and by the Digital Library grants IIS-9817353 and IIS-9979201.

References

[1] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[2] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[3] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

[4] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees. In *Int. Conf. on Computer Vision*, 2001.

[5] S. Ioffe and D.A. Forsyth. Probabilistic methods for finding people. *Int. J. Computer Vision*, 2001.

[6] M. Meila and M.I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

[7] M. Oren, C. Papageorgiou, P. Sinha, and E. Osuna. Pedestrian detection using wavelet templates. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 193–9, 1997.

[8] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE T. Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[9] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *Int. J. Computer Vision*, 37(2):151–72, 2000.

[10] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 746–51, 2000.

[11] K-K Sung and T. Poggio. Example-based learning for view-based human face detection. *PAMI*, 20(1):39–51, 1998.

[12] R.E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–36, 1977.

[13] M. Weber, W. Einhauser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 20–7, 2000.

[14] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *PAMI*, 19(7):775–9, 1997.