# Pictorial Structures for Object Recognition

Pedro F. Felzenszwalb
Artificial Intelligence Lab, Massachusetts Institute of Technology
pff@ai.mit.edu

Daniel P. Huttenlocher
Computer Science Department, Cornell University
dph@cs.cornell.edu

## Abstract

In this paper we present a statistical framework for modeling the appearance of objects. Our work is motivated by the pictorial structure models introduced by Fischler and Elschlager. The basic idea is to model an object by a collection of parts arranged in a deformable configuration. The appearance of each part is modeled separately, and the deformable configuration is represented by spring-like connections between pairs of parts. These models allow for qualitative descriptions of visual appearance, and are suitable for generic recognition problems. We use these models to address the problem of detecting an object in an image as well as the problem of learning an object model from training examples, and present efficient algorithms for both these problems. We demonstrate the techniques by learning models that represent faces and human bodies and using the resulting models to locate the corresponding objects in novel images.

## 1 Introduction

Research in object recognition is increasingly concerned with the ability to recognize generic classes of objects rather than just specific instances of objects. In this paper, we consider both the problem of detecting objects using generic part-based models and that of learning such models from example images. Our work is motivated by the pictorial structure representation introduced by Fischler and Elschlager [15] thirty

$(a)$ $(b)$

Figure 1: Detection results for a face (a); and a human body (b). Each image shows the globally best location for the corresponding object, as computed by our algorithms. The object models were learned from training examples.

years ago. The basic idea is to describe an object by a collection of parts arranged in a deformable configuration. Each part models local visual properties of the object, and the deformable configuration is characterized by spring-like connections between certain pairs of parts.

A statistical approach provides a natural formulation for the problem of detecting objects in images as well as for the problem of learning object models from training examples. We present such a statistical framework for pictorial structure models together with efficient algorithms for solving both the learning and detection problems. These algorithms apply to a specific but large class of pictorial structures. We demonstrate our techniques by learning models of two different object classes: faces and human bodies. We then use the resulting models to detect instances of these objects in novel images, as illustrated in Figure 1. The matches found by our techniques are based on a global search over all possible configurations of the parts in an image. This search is done efficiently by exploiting properties of the models. In contrast, most of the existing non-rigid matching techniques use local search methods and require that an initial configuration be provided close to the final answer. In many situations one wants to be able to detect objects wherever they are in the image, in such cases local search methods are not sufficient.

The pictorial structure framework we use is general, in the sense that it is independent of the specific scheme used to model the appearance of individual parts and of the type of connections between parts. For the face model in Figure 1a the appearance of each part is characterized by the response of Gaussian derivative filters

of different orders, orientations and scales. Each part is simply allowed to translate in the image plane, yielding a pose space with $2n$ dimensions for an object with $n$ parts. The connections between parts enforce that their relative locations be consistent with a typical face. For the person model in Figure 1b, the individual parts look like rectangles in the image. Each rectangular part is allowed to translate, rotate and undergo foreshortening, yielding a pose space with $4n$ dimensions for an object with $n$ parts. The connections between parts behave like simple revolute joints.

Despite the differences in both the part models and the type of connections between parts the same framework applies for the face and person models, and the same learning and matching algorithms can be used. In particular, the pose spaces are very different for the two kinds of models, but the same search techniques provide efficient means of finding the best global match of the models to an image.

## 1.1 Pictorial Structures

In [15] the problem of matching a pictorial structure to an image is defined in terms of an energy function to be minimized. The quality of a particular configuration for the parts depends both on how well each part matches the image data at its location, and how well the configuration agrees with the deformable model. This approach is different than most methods that use part-based representations. In those methods, parts are recognized individually in an initial phase, and a second phase uses the relations between parts to detect a configuration that matches the object model. In contrast, the pictorial structure approach does not involve making any initial decisions about the locations of individual parts. An overall decision is made based on the whole object model.

With pictorial structures we are able to use fairly generic models of appearance for the individual parts. The connections between parts provide the necessary context to detect them without obtaining a large number of false positive matches. A connection indicates spatial relationships between two parts. For example, a connection can enforce precise geometrical constraints, such as a revolute or prismatic joint. Connections can also represent more generic relationships such as "close to", "to the left of", or something in between these generic relationships and precise geometrical constraints.

Since both the part models and the relationships between parts can be generic, pictorial structures provide a powerful framework for recognition problems. For example, suppose we want to model the appearance of the human body. It makes sense to represent the body as an articulated object, with joints connecting different body parts. With pictorial structures we can use a fairly coarse model, consisting of a

small number of parts connected by flexible revolute joints. The structural relations between parts provide sufficient context to detect the human body as a whole even where it would be difficult to detect generic parts such as "lower-leg" or "upper-arm" on their own. With such coarse models, the connections between parts don't behave exactly like rigid joints, since a small number of parts can only approximate the geometrical structure of the human body. The joint models we use try to ensure that connected parts be aligned at their joint, while still allowing for small mis-positioning.

## 1.2 Statistical Formulation

In their original work, Fischler and Elschlager only addressed the problem of finding the best match of a pictorial structure model to an image. As mentioned above, they characterized this problem by defining an energy function to be minimized. While the energy function intuitively makes sense, it has many free parameters. For each different object, one has to construct a model, which includes picking an appearance model for each part, the characteristics of the connections between parts, and weighting parameters for the energy function.

We are interested not only in matching models to images but also in learning models from training examples. To that end we present a statistical formulation of the pictorial structure framework. In this formulation, the matching problem introduced by Fischler and Elschlager is equivalent to finding the maximum a posteriori (MAP) estimate of the object location given an observed image. The statistical formulation helps to characterize the different parameters of a model. In fact, *all* model parameters can be learned from a few training examples using maximum likelihood (ML) estimation. This is of practical as well as theoretical interest, since a user generally cannot find the best parameters for a deformable model by trial and error.

The statistical framework also provides a natural way of addressing another important detection problem, of finding all good matches of a model to an image rather than finding just the best match. The idea is to consider primarily good matches without considering many bad ones. We can achieve this by sampling object locations from their posterior probability distribution. Sampling allows us to find many locations for which our posterior is high, and select one or more of those as correct using an independent method. This procedure lets us use somewhat inaccurate models for generating hypothesis and can be seen as a mechanism for visual selection (see [2]). It is also similar to the idea behind importance sampling (see [17]).

## 1.3 Efficient Algorithms

Our goal is not only to construct a framework that is rich enough to capture the appearance of many generic objects, but also to be able to efficiently solve the object detection and model learning problems. We present such algorithms for detecting and learning a natural class of pictorial structure models. In particular, our methods require that the set of connections between the parts of an object form a tree structure, and that the spatial relationships between connected pairs of parts be expressed in a special form.

Restricting the connections between parts to a tree structure is natural for many classes of objects. For example, the connections between parts of many animate objects form a tree corresponding to the skeletal structure. Many other kinds of objects can be represented using a tree structure such as a star-graph, where there is one central part to which all the other parts are connected. The restriction that we impose on the form of connections between pairs of parts allows for a broad range of relationships to be represented. In particular, we require that there be mappings from the pose space of each part to a space where the Euclidean distance between transformed poses measure the extent to which two parts deviate from their ideal relative configuration.

The asymptotic running time of our matching algorithms is optimal, in the sense that the methods run as fast as matching each part to the image separately, without accounting for the connections between them. This means that we can take into account the context provided by the connections for "free". In practice, the algorithms are quite fast, finding the globally best match of a pictorial structure to an image in a few seconds.

## 1.4 Related Work

For nearly 40 years, research in object recognition has been dominated by approaches that separate processing into distinct stages of feature extraction and matching. In the first stage, discrete primitives, or "features" are extracted from an image. In the second stage, stored models are matched against the features that were extracted from the image. For instance, in the pioneering work of Roberts [33] childrens blocks were recognized by first extracting edges and corners from images and then matchnig these features to polyhedral models of the blocks. The model-based recognition paradigm of the 1980's similarly followed this approach. Model-based recognition methods focus largely on the problem of efficiently searching for correspondences between features that have been extracted from an image, and features of a stored model. Examples

of this paradigm include interpretation tree search [19, 3], the alignment method [22], RANSAC [14] and geometric hashing [26].

The problem of extracting features or parts of objects from images can itself be seen as another form of recognition problem. This is particularly clear with approaches such as part-based recognition (e.g., [12] and [32]), where the primitives are sub-parts of objects – for example the shade on a lamp. Not only is the feature or part extraction itself a recognition task, in some ways it is actually more difficult than the subsequent problem of recognizing the full object. Sub-parts or features are in general less distinctive than an overall object, making it more difficult to determine whether or not such parts are present in the image.

Limitations of the simple features used by most model-based recognition techniques led to a quite different class of recognition methods, developed in the 1990's, which operate directly on images rather than first extracting discrete features or parts. These include both appearance-based methods (e.g., [35] and [28]) and template-based methods such as Hausdorff matching [21]. Such approaches treat images as the artifacts to be recognized, rather than having more abstract models based on features or other primitives. A single example or multiple training images are generally used to form a "template" that is used as a model. This model is then compared to new images to determine whether or not the target is present, generally by explicitly searching possible transformations of the template with respect to the image.

The matching of pictorial structures, introduced by Fischler and Elschlager is an alternative approach to recognition that in many ways combines template matching with the combinatorial model matching approach. A major drawback, however, is that the problem of finding the best match of a pictorial structure model to an image is NP hard in general. In the past local minimization techniques that need a starting solution near the correct answer have been used. In this paper we introduce algorithms that can be used to match a large class of pictorial structure models to images efficiently.

In [9] a formulation similar to the one presented here was used to model pictorial structures consisting of a constellation of features. In their work instead of having connections between pairs of parts, all parts are constrained with respect to a central coordinate system by a Gaussian distribution. This makes it impossible to represent objects with more than one articulation point. Moreover they only provide heuristic algorithms that don't necessarily find the optimal match of the model to an image.

Techniques to find people in images using models similar to the ones we present in Section 6 were described in [23]. However, their methods are more in line with the classical part-based recognition systems that first detect a number of hypothesis for

the individual parts, and later find groups of parts which match the object model. As the problem of detecting generic parts such as "lower leg" or "upper-arm" in cluttered images is very hard, the examples in [23] are limited to simple images.

# 2 General Framework

A standard way to approach object detection from a statistical perspective is to model two different distributions. One distribution corresponds to the imaging process, and measures the likelihood of seeing a particular image given that an object is at some location. The other distribution measures the prior probability that an object would be at a particular location.

Let $\theta$ be a set of parameters that define an object model. The likelihood of seeing image $I$ given that the object is at location $L$ is given by $p(I|L,\theta)$. The prior probability of the object being at location $L$ is given by $p(L|\theta)$. Using Bayes' rule we can compute $p(L|I,\theta)$, the probability that the object is at location $L$, given an observed image $I$ (this will be called the posterior distribution from now on). A number of interesting problems can be characterized in terms of these probability distributions:

- MAP estimation - this is the problem of finding the location $L$ with highest posterior probability. In some sense, the MAP estimate is our best guess for the location of the object.
- Sampling - this is the problem of sampling from the posterior distribution. Sampling provides a natural way to find many good matches of a model to an image, rather than just the best one.
- Model estimation - this is the problem of finding $\theta$ which specifies a good model for a particular object. Using maximum likelihood estimation we can learn the model parameters from training examples.

We provide efficient algorithms to solve these problems for a large class of models. Together the algorithms form the base of a complete object detection system that learns from examples.

## 2.1 Pictorial Structure Models

In the pictorial structure framework, an object is represented by a collection of parts, or features, with connections between certain pairs of parts. A natural way to express

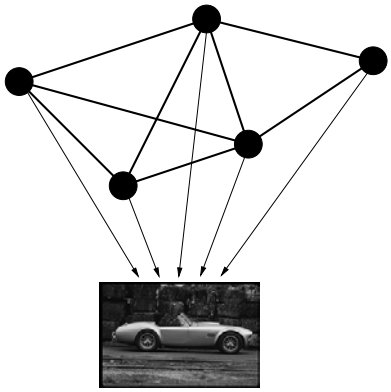

Figure 2: Graphical representation of the dependencies between the location of object parts (black nodes) and the image. In the case of a car, each black node would correspond to a part such as a wheel, the body, etc.

such a model is in terms of an undirected graph $G = (V, E)$, where the vertices $V = \{v_1, \ldots, v_n\}$ correspond to the parts, and there is an edge $(v_i, v_j) \in E$ for each pair of connected parts $v_i$ and $v_j$.

An instance of the object is given by a configuration $L = (l_1, \ldots, l_n)$, where $l_i$ is a random variable specifying the location of part $v_i$. Sometimes we refer to $L$ simply as the object location, but "configuration" emphasizes the part-based representation. The location of a part, $l_i$, can simply be the position of the part in the image, but more complex parameterizations are also possible. For example, a location can specify the position, orientation, and scale parameters for two dimensional parts. Each connection $(v_i, v_j) \in E$ indicates that the locations $l_i$ for $v_i$ and $l_j$ for $v_j$ are dependent. To be precise, the prior distribution over object configurations, $p(L|\theta)$, is a Markov Random Field, with structure specified by the graph $G$.

Using Bayes' rule, the posterior distribution over object configurations given an observed image can be characterized by the prior model and a likelihood function,

$$p(L|I, \theta) \propto p(I|L, \theta)p(L|\theta),$$

where the likelihood, $p(I|L, \theta)$, measures the probability of seeing image $I$ given a particular configuration for the object. Figure 2 shows a graphical representation of this statistical model. The random variable corresponding to the location of each object part is represented by a black node. Thick edges correspond to dependencies coming from the prior model, and the thin directed edges correspond to the dependency of the image with respect to the object configuration.

This posterior distribution is too complex to deal with in its most general form. In fact, finding the MAP estimate or sampling from such general distributions is an NP-

hard problem. Our framework is based on restricting the form of the prior model and the likelihood function so that the posterior distribution is more tractable. First of all, the graphical representation of the posterior should have no loops. In that case, it is possible to find the MAP estimate and sample from the distribution in polynomial time. This is done using a generalization of the Viterbi and Forward-Backward algorithms (see [30]). Similar algorithms are known in the Bayesian Network community as belief propagation and belief revision (see [29]).

Such polynomial time algorithms run in $O(h^2 n)$ time, where $n$ is the number of object parts, and $h$ is a discrete number of possible locations for each part. Unfortunately this is too slow for general detection problems because the number of possible locations for a single part is usually quite large – in the hundreds of thousands or millions. We have identified a natural restriction on the type of dependencies between parts for which we obtain algorithms that run essentially in $O(hn)$ time. These algorithms are also quite fast in practice.

We assume that there is an appearance model for each part, and that the appearances are characterized by some parameters $u = \{u_i \mid v_i \in V\}$. The exact method used to model the appearance of parts is not important. In Section 5, a part is modeled as a local image feature, based on image derivatives around a point, while in Section 6 parts are modeled as fairly large shapes. In practice, the appearance modeling scheme just needs to provide a distribution $p(I|l_i, u_i)$ (up to a normalizing constant), which measures the likelihood of seeing a particular image, given that a part with appearance parameters $u_i$ is at location $l_i$. Note that this distribution doesn't have to be a precise generative model, an approximate measure is enough in practice. We model the likelihood of seeing an image given that the object is at some configuration by the product of the individual likelihoods,

$$p(I|L, u) \propto \prod_{i=1}^{n} p(I|l_i, u_i). \tag{1}$$

This approximation is good if the parts don't overlap, as in this case they generate different portions of the image. But the approximation can be bad if one part occludes another. For the iconic models described in Section 5 the prior distribution over configurations enforces that the parts never overlap (the probability of a configuration with overlap is very small). On the other hand, for the articulated models described in Section 6 there is much less constraint on the locations of parts, and parts can easily overlap. For those models, the MAP estimate of an object location can be a poor estimate of its position. Even when this is the case, however, the correct match still has high posterior probability, it is just not the maximum. We show that we can find the correct location by obtaining multiple samples from the posterior and then

selecting one of the random samples using an independent method.

In our models, the set of connections between parts forms a tree structure. In general, when the set of dependencies between random variables forms a tree, their joint distribution can be expressed as,

$$p(L) = \frac{\prod_{(v_i,v_j)\in E} p(l_i, l_j)}{\prod_{v_i \in V} p(l_i)^{\deg v_i - 1}},$$

where $\deg v_i$ is the degree of vertex $v_i$ in the dependency graph. We don't model any preference over the absolute location of object parts, only over their relative configuration. This means $p(l_i)$ is a constant, and we let the constant be one for simplicity. The dependencies between parts are characterized by some parameters $c = \{c_{ij} \mid (v_i, v_j) \in E\}$. For example, such a connection might indicate that a given part tends to be at a certain distance to the left of another one. Our efficient algorithms require that the joint distribution for the locations of two connected parts be expressed in a special form. We use a Normal distribution for the distances between part locations in a transformed space,

$$p(l_i, l_j | c_{ij}) \propto \mathcal{N}(T_{ij}(l_i) - T_{ji}(l_j), 0, \Sigma_{ij}), \tag{2}$$

where $T_{ij}$, $T_{ji}$, and $\Sigma_{ij}$ are the connection parameters encoded by $c_{ij}$. The covariance matrix $\Sigma_{ij}$ should be diagonal, and for simplicity we will assume that $T_{ij}$, and $T_{ji}$ are one-to-one. We further require that it be possible to represent the set of possible transformed locations as positions in a grid. The functions $T_{ij}$ and $T_{ji}$ together capture the ideal relative locations for parts $v_i$ and $v_j$. The distance between the transformed locations, weighted by $\Sigma_{ij}$, measures the deformation of a "spring" connecting $v_i$ and $v_j$. This special form for the joint distribution of two parts arises naturally from our algorithmic techniques. Moreover, it allows for a broad class of interesting models. In Section 5 we describe simple feature based models where the connections between parts behave like springs. More complex models are described in Section 6, where the connections between parts behave like flexible joints.

Since we let $p(l_i) = 1$, the prior distribution over object locations is defined by the joint distributions for pairs of connected parts,

$$p(L|E, c) = \prod_{(v_i,v_j)\in E} p(l_i, l_j | c_{ij}). \tag{3}$$

Note that both the $p(l_i, l_j|c_{ij})$ and $p(L|E, c)$ are improper priors (they integrate to infinity). This is a consequence of using an uninformative prior over absolute locations for each part (see [4]).

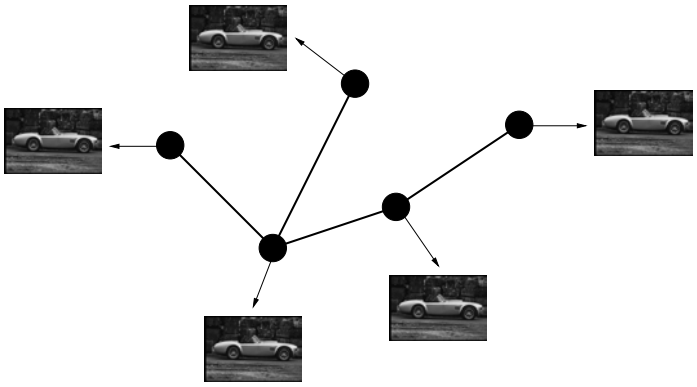

Figure 3: Graphical representation of the dependencies between the location of object parts (black nodes) and the image in the restricted models (see text).

So our models depend on parameters $\theta = (u, E, c)$, where $u = \{u_1, \ldots, u_n\}$ are the appearance parameters for each part, $E$ indicates which parts are connected, and $c = \{c_{ij} \mid (v_i, v_j) \in E\}$ are the connection parameters. We have defined the form of $p(I|L, \theta)$, the likelihood of seeing an image given that the object is at a some configuration, and $p(L|\theta)$, the prior probability that the object would assume a particular configuration. This is sufficient to characterize the posterior $p(L|I, \theta)$, the probability that the object is at some configuration in an image. A graphical representation of our restricted models is shown in Figure 3.

# 3 Learning Model Parameters

Suppose we are given a set of example images $I^1, \ldots, I^m$ and corresponding object configurations $L^1, \ldots, L^m$ for each image. We want to use the training examples to obtain estimates for the model parameters $\theta = (u, E, c)$, where $u = \{u_1, \ldots, u_n\}$ are the appearance parameters for each part, $E$ is the set of connections between parts, and $c = \{c_{ij} \mid (v_i, v_j) \in E\}$ are the connection parameters. The maximum likelihood (ML) estimate of $\theta$ is, by definition, the value $\theta^*$ that maximizes

$$p(I^1, \ldots, I^m, L^1, \ldots, L^m|\theta) = \prod_{k=1}^{m} p(I^k, L^k|\theta),$$

where the right hand side is obtained by assuming that each example was generated independently. Since $p(I, L|\theta) = p(I|L, \theta)p(L|\theta)$, the ML estimate is

$$\theta^* = \arg\max_{\theta} \prod_{k=1}^{m} p(I^k|L^k, \theta) \prod_{k=1}^{m} p(L^k|\theta). \tag{4}$$

The first term in this equation depends only on the appearance of the parts, while the second term depends only on the set of connections and connection parameters. Below

we show that one can independently solve for the appearance models of the individual parts and the structural model given by the connections and their parameters. As a consequence, any kind of part models can be used in our framework as long as there is a maximum likelihood estimation procedure for learning the model parameters from examples. We use quite simple part models in this paper because our focus is on developing a general framework and providing efficient algorithms that can be used with many different modeling schemes.

## 3.1 Estimating the Appearance Parameters

From equation (4) we get

$$u^* = \arg\max_u \prod_{k=1}^{m} p(I^k|L^k, u).$$

The likelihood of seeing image $I^k$, given the configuration $L^k$ for the object is given by equation (1). Thus,

$$u^* = \arg\max_u \prod_{k=1}^{m}\prod_{i=1}^{n} p(I^k|l_i^k, u_i) = \arg\max_u \prod_{i=1}^{n}\prod_{k=1}^{m} p(I^k|l_i^k, u_i).$$

Looking at the right hand side we see that to find $u^*$ we can independently solve for the $u_i^*$,

$$u_i^* = \arg\max_{u_i} \prod_{k=1}^{m} p(I^k|l_i^k, u_i).$$

This is exactly the ML estimate of the appearance parameters for part $v_i$, given independent examples $(I^1, l_i^1), \ldots, (I^m, l_i^m)$. Solving for $u_i^*$ depends on picking a specific modeling scheme for the parts, and we return to this in Sections 5 and 6.

## 3.2 Estimating the Dependencies

From equation (4) we get

$$E^*, c^* = \arg\max_{E,c} \prod_{k=1}^{m} p(L^k|E, c). \tag{5}$$

We need to pick a set of edges that form a tree and the properties for each edge. This can be done in a similar way to the Chow and Liu algorithm in [10], which estimates a tree distribution for discrete random variables. Equation (3) defines the prior probability of the object assuming configuration $L^k$ as,

$$p(L^k|E, c) = \prod_{(v_i,v_j)\in E} p(l_i^k, l_j^k|c_{ij}).$$

Plugging this into equation (5) and re-ordering the factors we get,

$$E^*, c^* = \arg\max_{E,c} \prod_{(v_i,v_j)\in E} \prod_{k=1}^{m} p(l_i^k, l_j^k | c_{ij}). \tag{6}$$

We can estimate the parameters for each possible connection independently, even before we know which connections will actually be in $E$ as

$$c_{ij}^* = \arg\max_{c_{ij}} \prod_{k=1}^{m} p(l_i^k, l_j^k | c_{ij}).$$

This is the ML estimate for the joint distribution of $l_i$ and $l_j$, given independent examples $(l_i^1, l_j^1), \ldots, (l_i^m, l_j^m)$. Solving for $c_{ij}^*$ depends on picking a specific representation for the joint distributions. Independent of the exact form of $p(l_i, l_j | c_{ij})$, and how to compute $c_{ij}^*$ (which we consider later, as it varies with different modeling schemes), we can characterize the "quality" of a connection between two parts as the probability of the examples under the ML estimate for their joint distribution,

$$q(v_i, v_j) = \prod_{k=1}^{m} p(l_i^k, l_j^k | c_{ij}^*).$$

Intuitively, the quality of a connection between two parts measures the extent to which their locations are related. These quantities can be used to estimate the connection set $E^*$ as follows. We know that $E^*$ should form a tree, and according to equation (6) we let,

$$E^* = \arg\max_{E} \prod_{(v_i,v_j)\in E} q(v_i, v_j) = \arg\min_{E} \sum_{(v_i,v_j)\in E} -\log q(v_i, v_j). \tag{7}$$

The right hand side is obtained by taking the negative logarithm of the function being maximized (and thus finding the argument minimizing the value, instead of maximizing it). Solving this equation is equivalent to the problem of computing the minimum spanning tree (MST) of a graph. We build a complete graph on the vertices $V$, and associate a weight $-\log q(v_i, v_j)$ with each edge $(v_i, v_j)$. By definition, the MST of this graph is the tree with minimum total weight, which is exactly the set of edges $E^*$ defined by equation (7). The MST problem is well known (see [11]) and can be solved efficiently. Kruskal's algorithm can be used to compute the MST in $O(n^2 \log n)$ time, since we have a complete graph with $n$ nodes.

# 4 Matching Algorithms

In this section we describe efficient algorithms to match our pictorial structure models to images. The first algorithm finds the MAP estimate of the object location given

an observed image. The second algorithm samples configurations from the posterior distribution. Recall that the MAP estimate of the object location is a configuration with highest posterior probability given an observed image. In some sense, this is our best guess for the object location. On the other hand, sampling from the posterior distribution is useful to produce multiple hypotheses. In [13] we presented a version of the MAP estimation algorithm that uses a different restriction on the form of connections between parts. That form did not allow for efficient sampling from the posterior distribution.

Both algorithms work in a discretized space of locations for each part. They run essentially in $O(hn)$ time, where $h$ is a number of discrete locations for each part and $n$ is the number of parts. To be precise, the running time of the algorithms is $O(h'n)$, where $h'$ is the number of discrete positions in the space of transformed poses for a part. Sometimes $h'$ can be slightly bigger than $h$, as seen in Section 6. To understand how the algorithms work it is useful to consider the configuration space for an object. There are $h^n$ possible configurations for the object. It is not necessary to explicitly consider such a huge state space because the structure of dependencies between parts forms a tree. This restriction allows us to use dynamic programming to solve the original problems by solving a sequence of subproblems. Each subproblem depends only on the location of two connected parts. There are $h^2$ possible configurations for two parts, but our algorithms use a second level of dynamic programming to solve each subproblem in time that is linear in $h'$.

## 4.1 MAP Estimate

The MAP estimate of the object location is a configuration with highest probability given an observed image. Such a configuration is defined by

$$L^* = \arg\max_L p(L|I,\theta) = \arg\max_L p(I|L,\theta)p(L|\theta).$$

Equation (1) characterizes the likelihood $p(I|L,\theta)$ in terms of an appearance model for each part, and equation (3) characterizes the prior $p(L|\theta)$ in terms of the connections between parts. Thus, in our framework we have,

$$L^* = \arg\max_L \left( \prod_{i=1}^{n} p(I|l_i,u_i) \prod_{(v_i,v_j)\in E} p(l_i,l_j|c_{ij}) \right).$$

By taking the negative logarithm of this equation, a typical energy minimization problem arises,

$$L^* = \arg\min_L \left( \sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i,l_j) \right), \tag{8}$$

where $m_i(l_i) = -\log p(I|l_i, u_i)$ is a match cost, which measures how well part $v_i$ matches the image data at location $l_i$, and $d_{ij}(l_i, l_j) = -\log p(l_i, l_j|c_{ij})$ is a deformation cost, which measures how well the relative locations for $v_i$ and $v_j$ agree with the prior model. The form of this minimization is quite general, and it appears in a number of problems in computer vision, such as MAP estimation of Markov Random Fields for low-level vision (such as image restoration and stereo) and optimization of dynamic contour models (snakes). While the form of the minimization is shared with these other problems, the structure of the graph and space of possible configurations differ substantially. This changes the computational nature of the problem.

Solving equation (8) for arbitrary graphs and arbitrary functions $m_i$, $d_{ij}$ is an NP-hard problem (see [7]). However, when the graph $G = (V, E)$ has a restricted form, the problem can be solved more efficiently. For instance, with first-order snakes the graph is simply a chain, which enables a dynamic programming solution that takes $O(h^2 n)$ time as described in [1]. Moreover, with snakes the minimization is done over a small number of locations for each vertex (e.g., the current location plus the 8 neighbors on the image grid). This minimization is then iterated until the change in energy is small. The key to an efficient solution is that the number of locations, $h$, be small, as the dynamic programming solution is quadratic in this value. Another source of efficient algorithms has been in restricting $d_{ij}$ to a particular form. This approach has been particularly fruitful in some recent work on MRFs for low-level vision ([8, 7, 24]). In our algorithm, we use constraints on both the structure of the graph and the form of $d_{ij}$.

By restricting the graphs to trees, a similar kind of dynamic programming can be applied as is done for chains, making the minimization problem polynomial rather than exponential time. The precise technique is described in Section 4.1.1. However, this $O(h^2 n)$ algorithm is not practical, because the number of possible locations for each part, $h$, is usually huge.

The restricted form of the joint distribution for the locations of two connected parts in equation (2) is,

$$p(l_i, l_j|c_{ij}) \propto \mathcal{N}(T_{ij}(l_i) - T_{ji}(l_j), 0, \Sigma_{ij}).$$

So $d_{ij}(l_i, l_j)$ is a Mahalanobis distance between transformed locations,

$$d_{ij}(l_i, l_j) = (T_{ij}(l_i) - T_{ji}(l_j))^T \; \Sigma'_{ij} \; (T_{ij}(l_i) - T_{ji}(l_j)), \tag{9}$$

where $\Sigma'_{ij} = \Sigma_{ij}/2$, and we ignored an additive constant since it doesn't change the solution of the minimization problem. This form for $d_{ij}$ yields a minimization algorithm which runs in $O(h'n)$ rather than $O(h^2 n)$ time. This makes it quite practical

to find the *globally optimal match* of a pictorial structure to an image, up to the discretization of the possible locations.

### 4.1.1 Efficient Minimization

In this section, we discuss how to efficiently find the configuration $L^* = (l_1^*, \ldots, l_n^*)$, minimizing equation (8) when the graph $G$ is a tree. Given $G = (V, E)$, let $v_r \in V$ be an arbitrarily chosen root vertex. From this root, each vertex $v_i \in V$ has a depth $d_i$ which is the number of edges between it and $v_r$ (and the depth of $v_r$ is 0). The children, $C_i$, of vertex $v_i$ are those neighboring vertices, if any, of depth $(d_i+1)$. Every vertex $v_i$ other than the root has a unique parent, which is the neighboring vertex of depth $(d_i - 1)$.

For any vertex $v_j$ with no children (i.e., any leaf of the tree), the best location $l_j^*$ of that vertex can be computed as a function of the location of just its parent, $v_i$. The only edge incident on $v_j$ is $(v_i, v_j)$, thus the only contribution of $l_j$ to the energy in (8) is $m_j(l_j) + d_{ij}(l_i, l_j)$. The quality of the best location of $v_j$ given location $l_i$ of $v_i$ is

$$B_j(l_i) = \min_{l_j} \left( m_j(l_j) + d_{ij}(l_i, l_j) \right), \tag{10}$$

and the best location of $v_j$ as a function of $l_i$ can be obtained by replacing the min in the equation above with arg min.

For any vertex $v_j$ other than the root, assume that the function $B_c(l_j)$ is known for each child $v_c \in C_j$. That is, the quality of the best location of each child is known with respect to the location of $v_j$. Then the quality of the best location of $v_j$ given the location of its parent $v_i$ is

$$B_j(l_i) = \min_{l_j} \left( m_j(l_j) + d_{ij}(l_i, l_j) + \sum_{v_c \in C_j} B_c(l_j) \right). \tag{11}$$

Again, the best location of $v_j$ as a function of $l_i$ can be obtained by replacing the min in the equation above with arg min. This equation subsumes (10) because for a leaf node the sum over its children is simply empty. Finally, for the root $v_r$, if $B_c(l_r)$ is known for each child $v_c \in C_r$ then the best location of the root is

$$l_r^* = \arg\min_{l_r} \left( m_r(l_r) + \sum_{v_c \in C_r} B_c(l_j) \right).$$

That is, the minimization in (8) can be expressed recursively in terms of the $(n - 1)$ functions $B_j(l_i)$ for each vertex $v_j \in V$ (other than the root). These recursive equations suggest a simple algorithm. Let $d$ be the maximum depth node in the tree.

For each node $v_j$ with depth $d$, compute $B_j(l_i)$, where $v_i$ is the parent of $v_j$. These are all leaf nodes, so clearly $B_j(l_i)$ can be computed as in (10). Next, for each node $v_j$ with depth $(d-1)$ compute $B_j(l_i)$, where again $v_i$ is the parent of $v_j$. Clearly, $B_c(l_j)$ has been computed for every child $v_c$ of $v_j$, because the children have depth $d$. Thus $B_j(l_i)$ can be computed as in (11). Continue in this manner, decreasing the depth until reaching the root at depth zero. Besides computing each $B_j$ we also compute $B'_j$, which indicates the best location of $v_j$ as a function of its parent location (obtained by replacing the min in $B_j$ with $\arg\min$). At this point, we compute the optimal location $l_r^*$ of the root. The optimal location $L^*$ of all the parts can be computed by tracing back from the root to each leaf. We know the optimal location of $v_j$ given the location of its parent, and the optimal location of each parent is now known starting from the root.

The overall running time of this algorithm is $O(Hn)$, where $H$ is the time required to compute each $B_j(l_i)$ and $B'_j(l_i)$. In the general case this takes $O(h^2)$ time as it is necessary to consider every location of a child node for each possible location of the parent. In the next section, we show how to compute each $B_j(l_i)$ and $B'_j(l_i)$ more efficiently when $d_{ij}$ is restricted to be in the form of equation (9).

### 4.1.2 Generalized Distance Transforms

Traditional distance transforms are defined for sets of points on a grid. Suppose we have a grid $\mathcal{G}$. Given a point set $B \subseteq \mathcal{G}$, the distance transform of $B$ specifies for each location in the grid, the distance to the closest point in the set,

$$\mathcal{D}_B(x) = \min_{y \in B} d(x, y).$$

In particular, $\mathcal{D}_B$ is zero at any point in $B$, and is small at nearby locations. The distance transform is commonly used for matching edge based models (see [6, 21]). The trivial way to compute this function takes $O(k|B|)$ time, where $k$ is the number of locations in the grid. On the other hand, efficient algorithms exist to compute the distance transform in $O(k)$ time, independent of the number of points in $B$ (see [5, 25]). These algorithms have small constants and are very fast in practice. In order to compute the distance transform, it is commonly expressed as

$$\mathcal{D}_B(x) = \min_{y \in \mathcal{G}} \left( d(x, y) + 1_B(y) \right),$$

where $1_B(y)$ is an indicator function for membership in the set $B$, that has value 0 when $y \in B$ and $\infty$ otherwise. This suggests a generalization of distance transforms where the indicator function is replaced with some arbitrary function over the grid $\mathcal{G}$,

$$\mathcal{D}_f(x) = \min_{y \in \mathcal{G}} \left( d(x, y) + f(y) \right).$$

Intuitively, for each grid location $x$, the transform finds a location $y$ that is close to $x$ and for which $f(y)$ is small. Note that difference between the values of $\mathcal{D}_f$ at two locations is bounded by the distance between the locations, regardless of how quickly the function $f$ changes (the indicator function of the classical distance transform is a limiting case, being either 0 or $\infty$). In particular, if there is a location where $f(x)$ has a small value, $\mathcal{D}_f$ will have small value at $x$ and nearby locations.

Given the restricted form of $d_{ij}$ in equation (9), the functions $B_j(l_i)$ that must be computed by the dynamic programming algorithm can be rewritten as generalized distance transforms using the Mahalanobis distance defined by $\Sigma'_{ij}$ as the distance in the grid,

$$B_j(l_i) = \mathcal{D}_f(T_{ij}(l_i)),$$

where

$$f(y) = \begin{cases} m_j(T_{ji}^{-1}(y)) + \sum_{v_c \in C_j} B_c(T_{ji}^{-1}(y)) & \text{if } y \in \text{range}(T_{ji}) \\ \infty & \text{otherwise} \end{cases}$$

and the grid $\mathcal{G}$ specifies a discrete set of possible values for $T_{ji}(l_j)$ that are considered during the minimization (this in turn specifies a discrete set of locations $l_j$). There is an approximation being made, since the set of discrete values for $T_{ji}(l_j)$ (the locations in the grid) might not match the set of discrete values for $T_{ij}(l_i)$ (where we need the value of $\mathcal{D}_f$). We can simply define the value of the distance transform at a non-grid position to be the value of the closest grid point. The error introduced by this approximation is small (as the transform by definition changes slowly).

The same algorithms that efficiently compute the classical distance transform can be used to compute the generalized distance transform under different distances, by replacing the indicator function $1_B(x)$ with an arbitrary function $f(x)$. In particular we use the method of Karzanov (originally in [25], but see [34] for a better description) to compute the transform of a function under a Mahalanobis distance with diagonal covariance matrix. This algorithm can also compute $B'_j(l_i)$, the best location of $v_j$ as a function of its parent location, as it computes the cost function $B_j(l_i)$.

## 4.2 Sampling from the Posterior

We now turn to the problem of sampling from the posterior distribution of object configurations. The sampling problem can be solved with an algorithm almost identical to the one used to compute the MAP estimate. The relationship between the two cases is analogous to the relationship between the Viterbi and the Baum-Welch algorithms for Hidden Markov Models (HMM's). Basically the sampling algorithm works directly with the probability distributions instead of their negative logarithms,

and the maximizations in the recursive equations are replaced by summations.

The posterior distribution is

$$p(L|I,\theta) \propto p(I|L,\theta)p(L|\theta) \propto \left( \prod_{i=1}^{n} p(I|l_i,u_i) \prod_{(v_i,v_j)\in E} p(l_i,l_j|c_{ij}) \right).$$

Like before, let $v_r \in V$ be an arbitrarily chosen root vertex, and the children of $v_i$ be $C_i$. The algorithm works by first computing $p(l_r|I,\theta)$. We then sample a location for the root from that distribution. Next we sample a location for each child, $v_c$, of the root from $p(l_c|l_r,I,\theta)$. We can continue in this manner until we have sampled a location for each part. The marginal distribution for the root location is,

$$p(l_r|I,\theta) \propto \sum_{l_1} \cdots \sum_{l_{r-1}} \sum_{l_{r+1}} \cdots \sum_{l_n} \left( \prod_{i=1}^{n} p(I|l_i,u_i) \prod_{(v_i,v_j)\in E} p(l_i,l_j|c_{ij}) \right).$$

Computing the distribution in this form would take exponential time. But since the set of dependencies between parts form a tree, we can rewrite the distribution as,

$$p(l_r|I,\theta) \propto p(I|l_r,u_r) \prod_{v_c\in C_r} S_c(l_r).$$

The functions $S_j(l_i)$ are similar to the $B_j(l_i)$ we used for the MAP estimation algorithm,

$$S_j(l_i) \propto \sum_{l_j} \left( p(I|l_j,u_j)p(l_i,l_j|c_{ij}) \prod_{v_c\in C_j} S_c(l_j) \right). \tag{12}$$

These recursive functions already give a polynomial algorithm to compute $p(l_r|I,\theta)$. As in the MAP estimation algorithm we can compute the $S$ functions starting from the leaf vertices. The trivial way to compute each $S_j(l_i)$ takes $O(h^2)$ time. For each location of $l_i$ we evaluate the function by explicitly summing over all possible locations of $l_j$. We will show how to compute each $S_j(l_i)$ more efficiently for the case where $p(l_i,l_j|c_{ij})$ is in the special form given by equation (2). But first let's see what we need to do after we sample a location for the root from its marginal distribution. If we have a location for the parent $v_i$ of $v_j$ we can write,

$$p(l_j|l_i,I,\theta) \propto p(I|l_j,u_j)p(l_i,l_j|c_{ij}) \prod_{v_c\in C_j} S_c(l_j). \tag{13}$$

If we have already computed the $S$ functions we can compute this distribution in $O(h)$ time. So once we have sampled a location for the root, we can sample a location for each of its children. Next we sample a location for the nodes at the third level of the tree, and so on until we sample a location for every part. In the next section we show how to compute the $S$ functions in time linear in $h'$, yielding an $O(h'n)$ algorithm

for sampling a configuration from the posterior distribution. Note that if we want to sample multiple times we only need to compute the $S$ functions once. And when the location of a parent node is fixed, we only need to compute the distribution in (13) for locations of the children where $p(l_i, l_j|c_{ij})$ is not too small. So sampling multiple times isn't much more costly than sampling once.

### 4.2.1 Computing the $S$ functions

We want to efficiently compute the function in equation (12). We will do this by writing the function as a Gaussian convolution in the transformed space (given by $T_{ij}$ and $T_{ji}$). Using the special form of $p(l_i, l_j|c_{ij})$ we can write,

$$S_j(l_i) \propto \sum_{l_j} \left( \mathcal{N}(T_{ij}(l_i) - T_{ji}(l_j), 0, \Sigma_{ij})\ p(I|l_j, u_j) \prod_{v_c \in C_j} S_c(l_j) \right).$$

This can be seen as a Gaussian convolution in the transformed space:

$$S_j(l_i) \propto (G \otimes f)\ (T_{ij}(l_i)),$$

where $G$ is a Gaussian filter with covariance $\Sigma_{ij}$, $\otimes$ is the convolution operator, and

$$f(y) = \begin{cases} p(I|T_{ji}^{-1}(y), u_j) \prod_{v_c \in C_j} S_c(T_{ji}^{-1}(y)) & \text{if } y \in \text{range}(T_{ji}) \\ 0 & \text{otherwise} \end{cases}$$

Just like when computing the generalized distance transform, the convolution is done over a discrete grid which specifies possible values for $T_{ji}(l_j)$. The Gaussian filter $G$ is separable since the covariance matrix $\Sigma_{ij}$ is diagonal. We can compute a good approximation for the convolution in time linear in the set of grid locations using the techniques from [36].

# 5 Iconic Models

The framework presented so far is general in the sense that it doesn't fully specify how objects are represented. A particular modeling scheme must define the pose space for the object parts, the form of the appearance model for each part, and the type of connections between parts. Here we present models that represent objects by the appearance of local image patches and spatial relationships between those patches. This type of model has been popular in the context of face detection (see [15, 9, 37]). We first describe how we model the appearance of a part, and later describe how we model spatial relationships between parts. Learning an iconic model

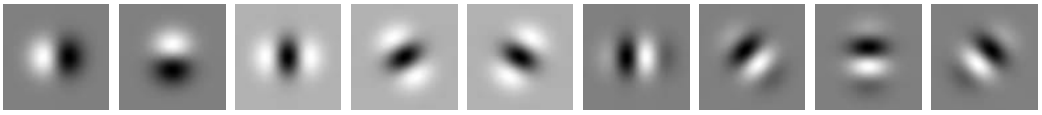

Figure 4: Gaussian derivative basis functions used in the iconic representation.

involves picking labeled landmarks on a number of instances of the target object. From these training examples both the appearance models for each part and the spatial relationships between parts are automatically estimated, using the procedure described in Section 3. In Section 5.3 we show some experiments with face detection.

## 5.1 Parts

In this class of models the location of a part is specified by its $(x, y)$ position in the image, so we have a two-dimensional pose space for each part. To model the appearance of each individual part we use the iconic representation introduced in [31]. The iconic representation is based on the response of Gaussian derivative filters of different orders, orientations and scales. An image patch centered at some position is represented by a high-dimensional vector that collects all the responses of a set of filters at that point. This vector is normalized and called the iconic index at that position. Figure 4 shows the nine filters used to build the iconic representation at a fixed scale. In practice, we use three scales, given by $\sigma_1 = 1$, $\sigma_2 = 2$, and $\sigma_3 = 4$, the standard deviations of the Gaussian filters. So we get a 27 dimensional vector. The iconic index is fairly insensitive to changes in lighting conditions. For example, it is invariant to gain and bias. We get invariance to bias as a consequence of using image derivative filters, and the normalization gives us the invariance to gain. Iconic indices are also relatively insensitive to small changes in scale and other image deformations. They can also be made invariant to image rotation (see [31]), although we use an orientation-sensitive representation here.

The appearance of a part is modeled by a distribution over iconic indices. Specifically, we model the distribution of iconic indices at the location of a part as a Gaussian with diagonal covariance matrix. Using a diagonal covariance matrix makes it possible to estimate the distribution parameters with a small number of examples. If many examples are available, a full Gaussian distribution or even more complex distributions such as a mixture of Gaussians, or a non-parametric estimate could be used. Under the Gaussian model, the appearance parameters for each part are $u_i = (\mu_i, \Sigma_i)$, a mean vector and a covariance matrix. We have,

$$p(I|l_i, u_i) = \mathcal{N}(\alpha(l_i), \mu_i, \Sigma_i),$$

where $\alpha(l_i)$ is the iconic index at location $l_i$ in the image. If we have some training examples, we can easily estimate the maximum likelihood parameters of this distribution as the sample mean and covariance.

Note that we could use other methods to represent the appearance of image patches. In particular, we experimented with the eigenspace techniques from [27]. With a small number of training examples the eigenspace methods are no better than the iconic representation, and the iconic representation can be computed more efficiently. In fact, the iconic representation can be computed very fast by convolving each level of a Gaussian pyramid with small x-y separable filters (see [16]).

## 5.2 Spatial Distribution

The spatial configuration of the parts is modeled by a collection of springs connecting pairs of parts. Each connection $(v_i, v_j)$ is characterized by the ideal relative location of the two connected parts $s_{ij}$, and a covariance matrix $\Sigma_{ij}$ which in some sense corresponds to the stiffness of the spring connecting the two parts. So the connection parameters are $c_{ij} = (s_{ij}, \Sigma_{ij})$. We model the distribution of the relative location of part $v_i$ with respect to the location of part $v_j$ as a Gaussian with mean $s_{ij}$ and covariance $\Sigma_{ij}$,

$$p(l_i, l_j | c_{ij}) = \mathcal{N}(l_i - l_j, s_{ij}, \Sigma_{ij}). \tag{14}$$

So, ideally the location of part $v_i$ is the location of part $v_j$ shifted by $s_{ij}$. Since the models are deformable, the location of $v_i$ can vary by paying a cost that depends on the covariance matrix. This corresponds to stretching the spring. Because we have a full covariance matrix, stretching in different directions can have different costs. For example, two parts can be highly constrained to be at the same vertical position, while their relative horizontal position may be uncertain. As with the appearance models for the individual parts, the maximum likelihood parameters of these spatial distributions for pairs of parts can easily be estimated using training examples.

In practice, we need to write the joint distribution of $l_i$ and $l_j$ in the specific form required by our algorithms. It must be a Gaussian distribution with zero mean and diagonal covariance in a transformed space, as described by equation (2). To do this, we first compute the singular value decomposition of the covariance matrix $\Sigma_{ij} = U_{ij} D_{ij} U_{ij}^T$. Now let

$$T_{ij}(l_i) = U_{ij}^T (l_i - s_{ij}), \text{ and } T_{ji}(l_j) = U_{ij}^T (l_j),$$

which allow us to write equation (14) in the correct form,

$$p(l_i, l_j | c_{ij}) = \mathcal{N}(T_{ij}(l_i) - T_{ji}(l_j), 0, D_{ij}).$$

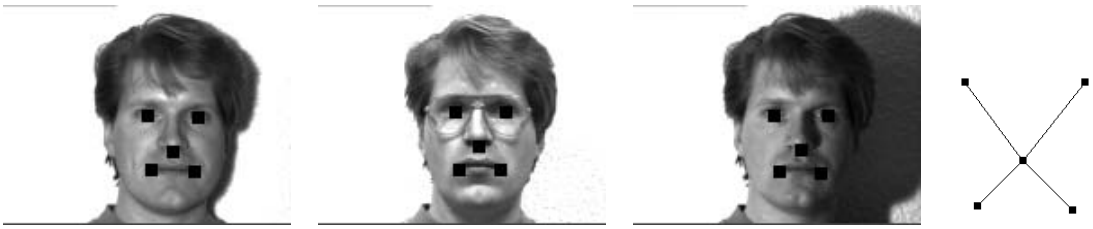

Figure 5: Three examples from the first training set and the structure of the learned model.

## 5.3 Experiments

Now we present some experiments of using the iconic models just described to detect faces. The basic idea is to use ML estimation to train a model of frontal faces, and MAP estimation to detect faces in novel images. Our first model has five parts, corresponding to the eyes, nose, and corners of the mouth. To generate training examples we labeled the location of each part in twenty different images (from the Yale face database). More training examples were automatically generated by scaling and rotating each training image by a small amount. This makes our model handle some variation in orientation and scale. Some of the training examples and the structure of the learned model are shown in Figure 5. Remember that we never told the system which pairs of parts should be connected together. Determining the structure is part of the ML parameter estimation procedure.

We tested the resulting model by matching it to novel images using MAP estimation. Note that *all* model parameters are automatically estimated under the maximum likelihood formalism. Thus, there are no "knobs" to tune in the matching algorithm. Some matching results are shown in Figure 6. Both the learning and matching algorithms are extremely fast. Using a desktop computer it took a few seconds to learn the model and less than a second to compute the MAP estimate in each image. These experiments demonstrate that we can learn a useful model from training examples. However the structure of this model is not particularly interesting as all the parts are connected through a central part, and the properties of all the connections are similar.

We also tried learning a larger model, this one with nine parts. We now have three parts for each eye, one for the left corner, one for the right corner and one for the pupil. This is a useful model to detect gaze direction. Figure 7 shows one of the training examples and the learned model. Also, in Figure 7, there is a detailed illustration of the connections to the left corner of the right eye. The ellipses illustrate

Figure 6: Matching results.

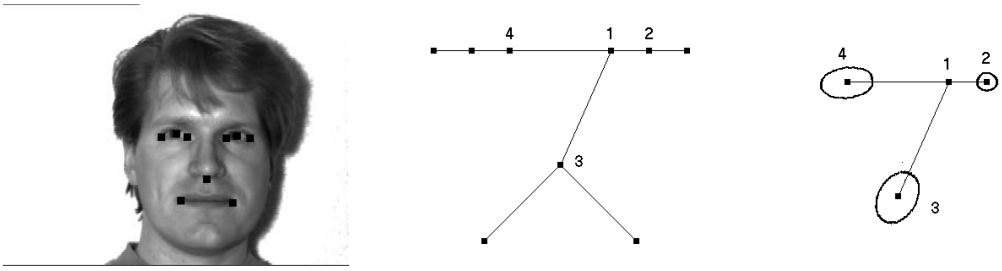


Figure 7: One example from the second training set, the structure of the learned model, and a pictorial illustration of the connections to one of the parts, showing the location uncertainty for parts 2, 3, and 4, when part 1 is at a fixed position.

the location uncertainty for the other parts, when this part is at some fixed location. They are level sets of the probability distribution for the location of parts 2, 3, and 4, given that part 1 is fixed. Note that the location of the pupil (part 2) is much more constrained with respect to the location of the eye corner than any other part, as would be expected intuitively. Also note that the distributions are not centrally symmetric, as they reflect the typical variation in the relative locations of parts. We see that the algorithm both learned an interesting structure for the model, and automatically determined the constraints between the locations of different pairs of parts.

# 6 Articulated Models

Now we present a scheme to model articulated objects. Our main motivation is to construct a system that can estimate the pose of human bodies. We concentrate on detecting objects in silhouette images. These images can be generated by subtracting a background image from the input image. Figure 8 shows an example input and matching result. Silhouette images characterize well the problem of pose estimation for an articulated object. We want to find an object configuration that covers the foreground pixels and leaves the background pixels uncovered. Note that we don't assume "perfect" silhouette images. In fact, our method works with very noisy input. In order to detect articulated bodies in such silhouette images we use the sampling techniques instead of computing the MAP estimate for the object location. This is important because the models for articulated bodies are not accurate generative models. This is explained in more detail below.

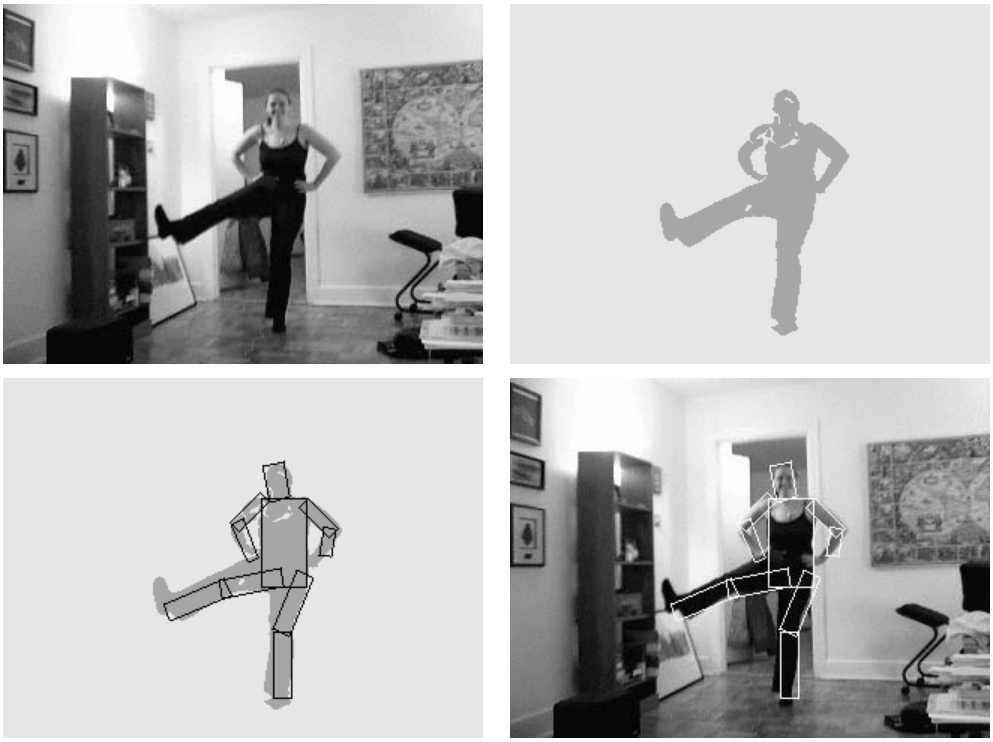

Figure 8: Input image, silhouette obtained by background subtraction, and matching result.

## 6.1 Parts

For simplicity, we assume that the image of an object is generated by a scaled orthographic projection, so that parallel features in the model remain parallel in the image. For images of human forms this is generally a reasonable assumption. We further assume that the scale factor of the projection is known. We can easily add an extra parameter to our search space in order to relax this latter assumption.

Suppose that objects are composed of a number of rigid parts, connected by flexible joints. If a rigid part is more or less cylindrical, its projection can be approximated by a rectangle. The width of the rectangle comes from the diameter of the cylinder and is fixed, while the length of the rectangle depends on the length of the cylinder but can vary due to foreshortening. We model the projection of a part as a rectangle parameterized by $(x, y, s, \theta)$. The center of the rectangle is given in image coordinates $(x, y)$, the length of the rectangle is defined by the amount of foreshortening $s \in [0, 1]$, and the orientation is given by $\theta$. So we have a four-dimensional pose space for each part.

We model the likelihood of observing an image given a particular location for a

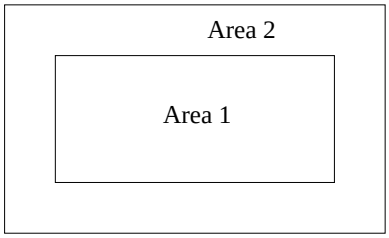


Figure 9: A rectangular part. $area_1$ is the area inside the part, and $area_2$ is the border area around it.

part in the following way. First, each pixel in the image is generated independently. Pixels inside a part are foreground pixels with probability $q_1$. Intuitively, $q_1$ should be close to one, expressing the idea that parts occlude the background. We also model a border area around each part (see Figure 9). In this area, pixels belong to the foreground with probability $q_2$. In practice, when we estimate $q_2$ from data we see that pixels around a part tend to be background. We assume that pixels outside both areas are equally likely to be background or foreground pixels. Thus,

$$p(I|l_i, u_i) = q_1^{count_1} \ (1 - q_1)^{(area_1 - count_1)} \ q_2^{count_2} \ (1 - q_2)^{(area_2 - count_2)} \ 0.5^{(t - area_1 - area_2)},$$

where $count_1$ is the number of foreground pixels inside the rectangle, and $area_1$ is the area of the rectangle. $count_2$ and $area_2$ are similar measures corresponding to the border area, and $t$ is the total number of pixels in the image. So the appearance parameters are $u_i = (q_1, q_2)$, and it is straightforward to estimate these parameters from training examples.

To make the probability measure robust we consider a slightly dilated version of the silhouette when computing $count_1$, and to compute $count_2$ we erode the silhouette (in practice we dilated and eroded the silhouette by two pixels). Computing the likelihood for every possible location of a part can be done efficiently by convolving the image with uniform filters. Each convolution counts the number of pixels inside a rectangle (specified by the filter) at every possible translation.

Intuitively, our model of $p(I|l_i, u_i)$ is reasonable for a single part. The likelihood favors large parts, as they explain a larger area of the image. But remember that we model $p(I|L, u)$ as a product of the individual likelihoods for each part. For a configuration with overlapping parts, this measure "overcounts" evidence. Suppose we have an object with two parts. The likelihood of an image is the same if the two parts are arranged to explain different areas of the image, or if the two parts are on top of each other and explain the same area twice. Therefore, with this measure the MAP estimate of an object configuration can be a bad guess for its true position. This is not because the posterior probability of the true configuration is low, but because there

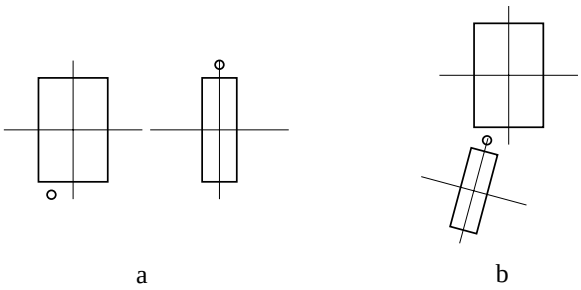


Figure 10: Two parts of an articulated object, (a) in their own coordinate system and (b) the ideal configuration of the pair.

are configurations which have high posterior and are wrong. In our experiments, we obtain a number of configurations which have high posterior probability by sampling from that distribution. We then select one of the samples by computing a quality measure that doesn't overcount evidence.

There is one more thing we have to take into account for sampling to work. When $p(I|L,u)$ overcounts evidence, it tends to create high peaks. This in turn creates high peaks in the posterior. When a distribution has a very strong peak, sampling from the distribution will almost always obtain the location of the peak. To ensure that we get a number of different hypothesis from sampling we use a smoothed version of $p(I|L,u)$, defined as

$$p'(I|L,u) \propto p(I|L,u)^{1/T} \propto \prod_{i=1}^{n} p(I|l_i,u_i)^{1/T},$$

where $T$ controls the degree of smoothing. This is a standard technique, borrowed from the principle of annealing (see [18]). Note that $p'(I|L,u)$ is just the product of the smoothed likelihoods for each part. In all our experiments we used $T = 10$.

## 6.2 Geometry

For the articulated objects, pairs of parts are connected by flexible joints. A pair of connected parts is illustrated in Figure 10. The location of the joint is specified by two points $(x_{ij}, y_{ij})$ and $(x_{ji}, y_{ji})$, one in the coordinate frame of each part, as indicated by circles in Figure 10a. In an ideal configuration these points coincide, as illustrated in Figure 10b. The ideal relative orientation is given by $\theta_{ij}$, the difference between the orientation of the two parts.

Suppose $l_i = (x_i, y_i, s_i, \theta_i)$ and $l_j = (x_j, y_j, s_j, \theta_j)$ are the locations of two connected parts. The joint probability for the two locations is based on the deviation between

their ideal relative values and the observed ones,

$$\begin{aligned} p(l_i, l_j | c_{ij}) &= \mathcal{N}(x'_i - x'_j, 0, \sigma_x^2) \\ &\quad \mathcal{N}(y'_i - y'_j, 0, \sigma_y^2) \\ &\quad \mathcal{N}(s_i - s_j, 0, \sigma_s^2) \\ &\quad \mathcal{M}(\theta_i - \theta_j, \theta_{ij}, k), \end{aligned} \tag{15}$$

where $(x'_i, y'_i)$ and $(x'_j, y'_j)$ are the positions of the joints in image coordinates. Let $R_\theta$ be the matrix that performs a rotation of $\theta$ radians about the origin. Then,

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + s_i R_{\theta_i} \begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix}, \text{ and } \begin{bmatrix} x'_j \\ y'_j \end{bmatrix} = \begin{bmatrix} x_j \\ y_j \end{bmatrix} + s_j R_{\theta_j} \begin{bmatrix} x_{ji} \\ y_{ji} \end{bmatrix}.$$

The distribution over angles, $\mathcal{M}$, is the von Mises distribution (see [20]),

$$\mathcal{M}(\theta, \mu, k) \propto e^{k \cos(\theta - \mu)}.$$

The first two terms in the joint distribution measure the horizontal and vertical distances between the observed joint positions in the image. The third term measures the difference in foreshortening between the two parts. The last term measures the difference between the relative angle of the two parts and the ideal relative angle. Usually $\sigma_x$ and $\sigma_y$ will be small so parts tend to be aligned at their joint. And if $k$ is small the angle between the two parts is fairly unconstrained, modeling a revolute joint.

The connection parameters under this model are,

$$c_{ij} = (x_{ij}, y_{ij}, x_{ji}, y_{ji}, \sigma_x^2, \sigma_y^2, \sigma_s^2, \theta_{ij}, k).$$

Finding the maximum likelihood estimate of $\sigma_s^2$ is easy since we just have a Gaussian distribution over $s_i - s_j$. Similarly, there are known methods to find the ML parameters $(\theta_{ij}, k)$ of a von Mises distribution (see [20]). The ML estimate of the joint location in each part are the values $(x_{ij}, y_{ij}, x_{ji}, y_{ji})$ which minimize the sum of square distances between $(x'_i, y'_i)$ and $(x'_j, y'_j)$ over the examples. We can compute this as a linear least squares problem. The variances $(\sigma_x^2, \sigma_y^2)$ are just the sample variances.

We need to write the joint distribution of $l_i$ and $l_j$ in the specific form required by our algorithms. It must be a Gaussian distribution with zero mean and diagonal covariance in a transformed space, as described by equation (2). First note that a von Mises distribution over angular parameters can be specified in terms of a Gaussian over the unit vector representation of the angles. Let $\vec{\alpha}$ and $\vec{\beta}$ be the unit vectors corresponding to two angles $\alpha$ and $\beta$. That is, $\vec{\alpha} = [cos(\alpha), sin(\alpha)]^T$, and similarly for $\vec{\beta}$. Then,

$$cos(\alpha - \beta) = \vec{\alpha} \cdot \vec{\beta} = -\frac{\|\vec{\alpha} - \vec{\beta}\|^2 - 2}{2}.$$

Now let

$$T_{ij}(l_i) = (x'_i, y'_i, s_i, \cos(\theta_i + \theta_{ij}), \sin(\theta_i + \theta_{ij})),$$
$$T_{ji}(l_j) = (x'_j, y'_j, s_j, \cos(\theta_j), \sin(\theta_j)),$$
$$\Sigma_{ij} = diag(1/\sigma_x^2, 1/\sigma_y^2, 1/\sigma_s^2, k, k),$$

which allow us to write equation (15) in the right form,

$$p(l_i, l_j | c_{ij}) \propto \mathcal{N}(T_{ji}(l_j) - T_{ij}(l_i), 0, \Sigma_{ij}).$$

For these models, the number of discrete locations in the transformed space is a little larger than the number of locations for each part. This is because we represent the orientation of a part as a unit vector which lives in a two-dimensional grid. In practice, we use 32 possible angles for each part, and represent them as points in a $11 \times 11$ grid.

## 6.3 Experiments

We use an articulated model to represent the human body. Our model has ten parts, corresponding to the torso, head, two parts per arm and two parts per leg. To generate training examples we labeled the location of each part in ten different images (without too much precision). The learned model is illustrated in Figure 11. The crosses indicate joints between parts. We never told the system which parts should be connected together, this is automatically learned during the ML parameter estimation. Note that the correct structure was learned, and the joint locations agree with the human body anatomy (the joint in the middle of the torso connects to the head). The configuration of parts shown in Figure 11 was obtained by fixing the position of the torso and placing all other parts in their optimal location with respect to each other.

We tested the model by matching it to novel images. As described in Section 6.1, the MAP estimate can be a bad guess for the object location, because our model does not explicitly account for overlap between parts. Therefore we sample configurations from the posterior distribution to obtain multiple hypothesis and rate each sample using a separate measure. For each sample we computed the Chamfer distance between the shape of the object under the hypothesized configuration and the silhouette obtained from the input image. The Chamfer distance is a robust measure of binary correlation (see [6]). The matching process is illustrated in Figure 12. First, a silhouette is obtained from the original image using background subtraction. We use this silhouette as input to the sampling algorithm to obtain a number of different pose hypotheses for the object. The best pose is then selected using the Chamfer measure.

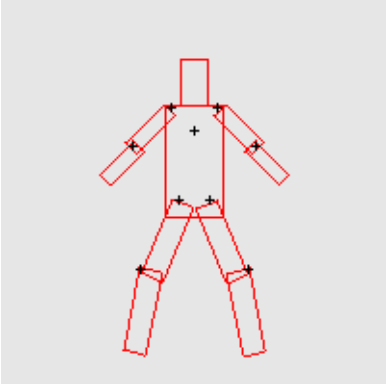

Figure 11: Model of human body.

More matching results are shown in Figure 13. For each image, we sampled two-hundred object configurations from the posterior distribution and picked the best one under the Chamfer distance. Using a desktop computer it took about one minute to process each image. The space of possible locations for each part was discretized into a $70 \times 70 \times 10 \times 32$ grid, corresponding to $(x, y, s, \theta)$ parameters. There are over 1.5 million locations for each part, making any algorithm that considers locations for pairs of parts at a time impractical.

Figure 14 shows that our method works well with noisy input. There is no way to detect body parts individually on inputs like that. But the dependencies between parts provide sufficient context to detect the human body as a whole. Of course, sometimes the estimated pose is not correct. The most common source of error comes from ambiguities in the silhouette. Figure 15 shows an example where the silhouette doesn't provide enough information to estimate the position of one arm. Even in that case we get a fairly good estimate. We can detect when ambiguities happen because we obtain many different poses with equally good Chamfer score. Thus we know that there are different configurations that are equally good interpretations of the image.

# 7 Summary

This paper presents a statistical framework for representing the visual appearance of objects composed of rigid parts arranged in a deformable configuration. The models are based on the pictorial structure representation developed in [15], which allows for qualitative descriptions of appearance and is suitable for generic recognition problems. The statistical approach provides a principled way of defining both the object detection and model learning problems, and yields efficient methods for solving both of these problems. We have illustrated how models can be learned from a small num-

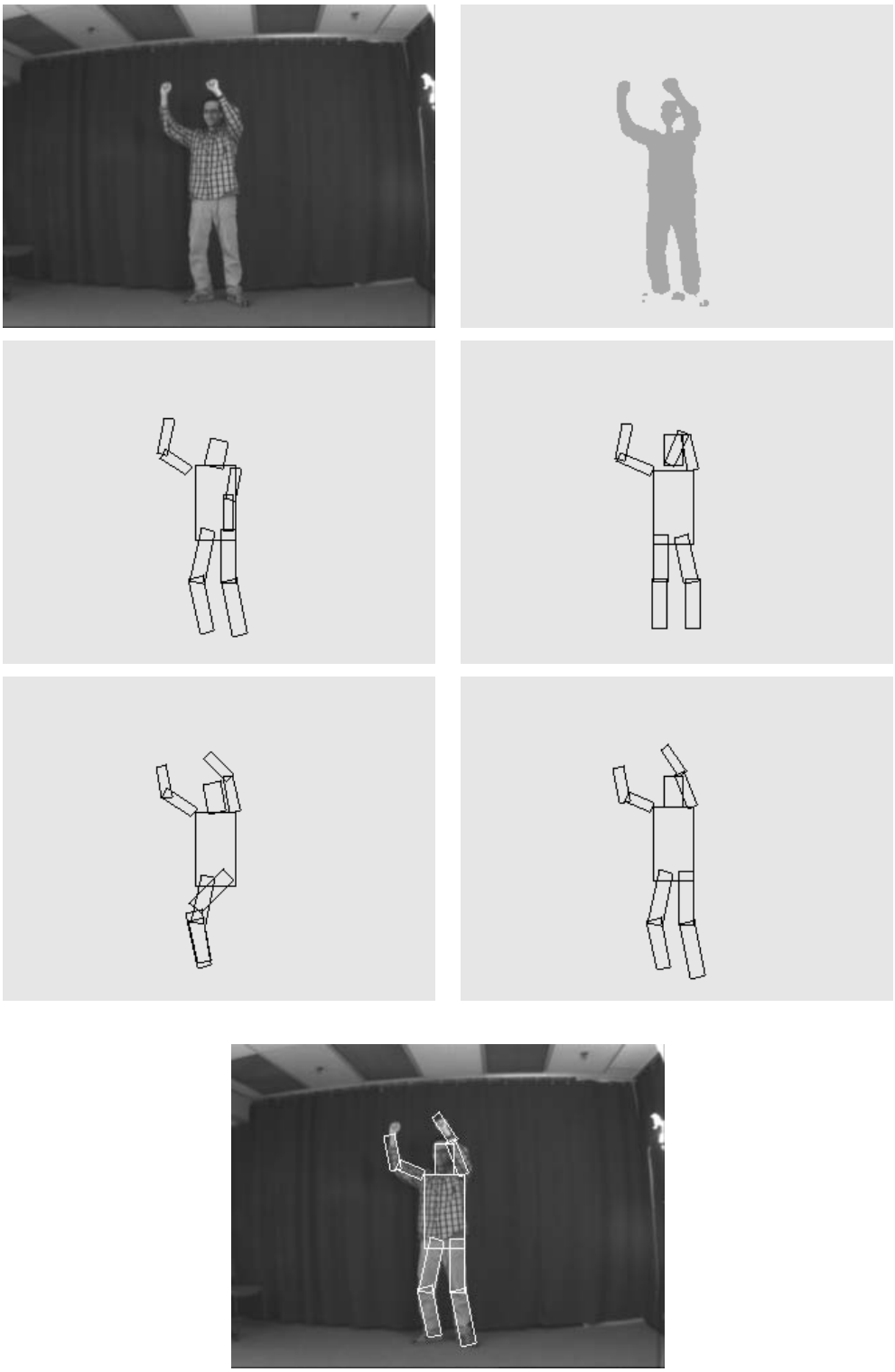

Figure 12: Input image, silhouette, random samples, and best result selected using the Chamfer distance.

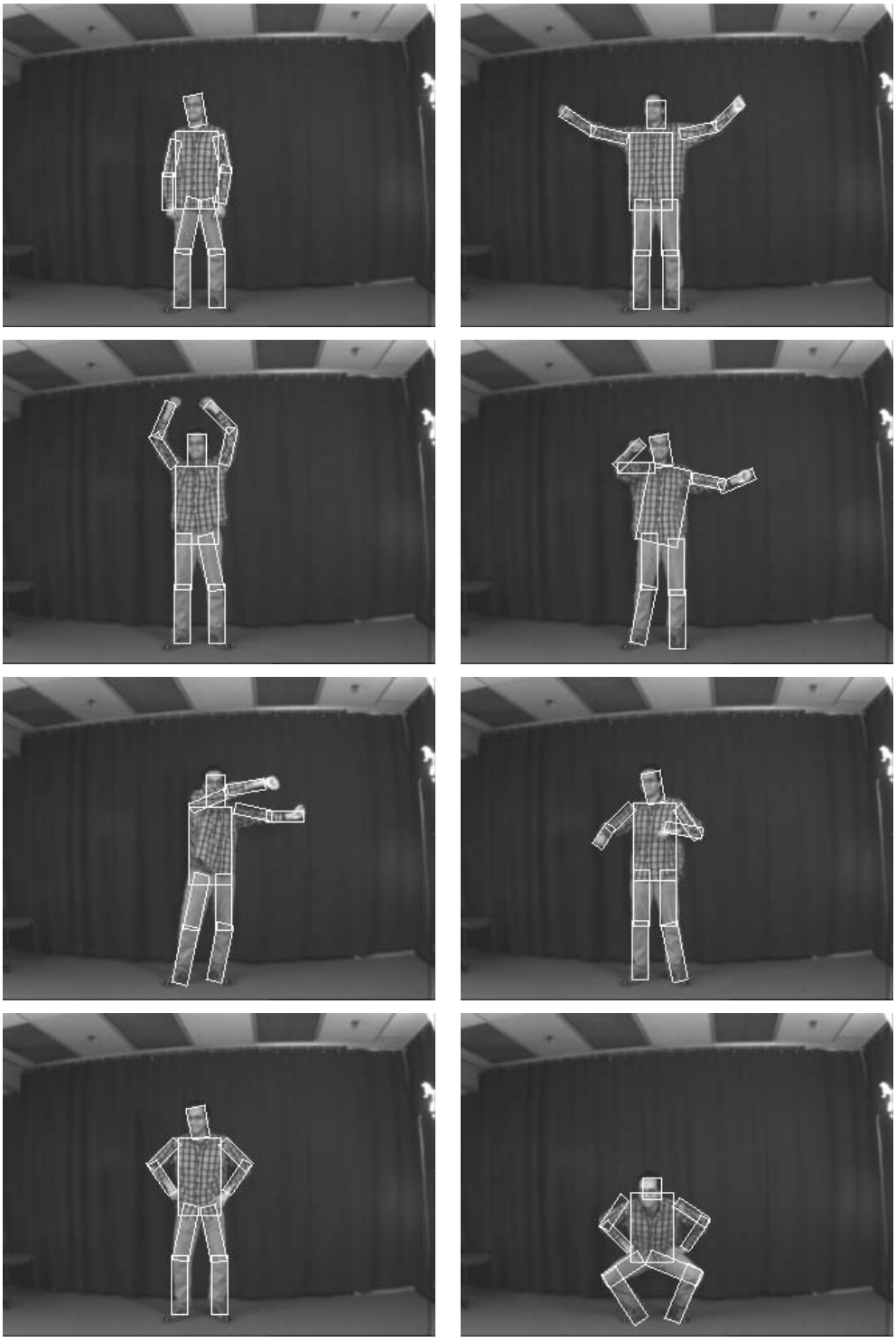

Figure 13: Matching results (sampling 100 times).

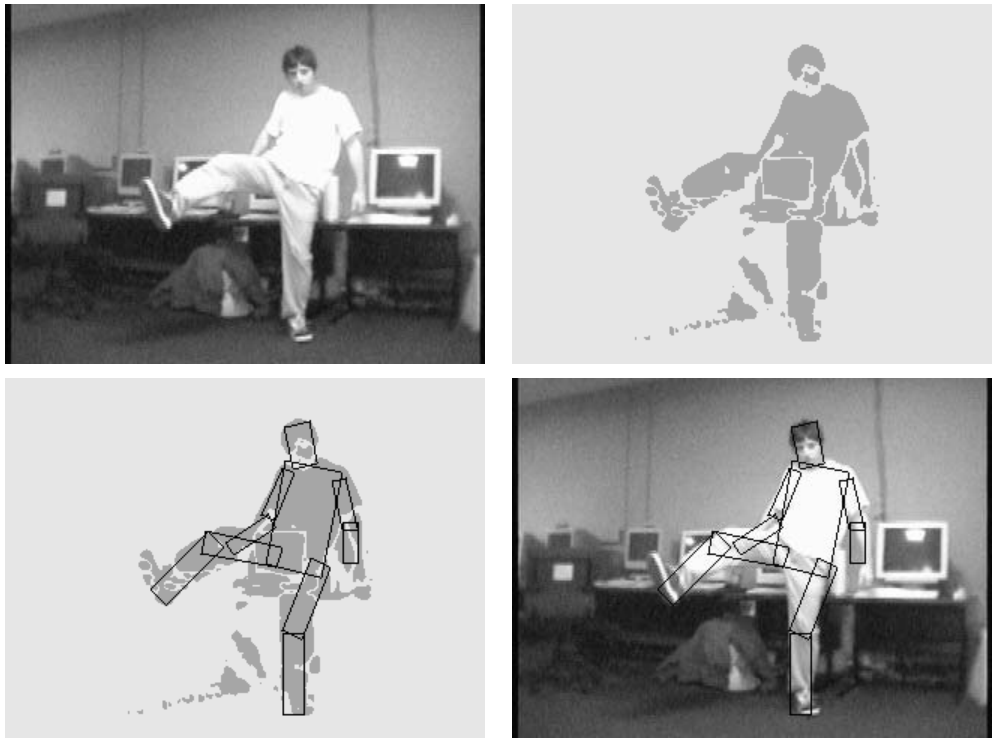

Figure 14: Even with noisy silhouettes we get good results.

ber of training images, and how the resulting models can be used to find instances of objects in new images.

There are three main contributions in this paper, which set it apart from other work on pictorial structures and flexible templates for detecting objects in images. First, we introduce efficient algorithms for finding the best *global* match of a model to an image. In contrast, prior work uses local search techniques that must be somehow initialized near the right answer. Second, we introducte the use of statistical sampling techniques to identify multiple *good* matches of a model to an image. In contrast, prior work focuses on finding the best match. Third, our use of a statistical formulation provides a natural way of learning pictorial structure models from example images. Most of the prior work uses manually constructed models, which are difficult to create and to validate. We believe that the ability to learn such models is particularly important as good models of flexible objects are generally too complex to be constructed manually.

One of the difficulties in representing generic objects is the large variation in shape and photometric information in each object class. Using a part based representation we can model the appearance variation in each part separately. We also explicitly

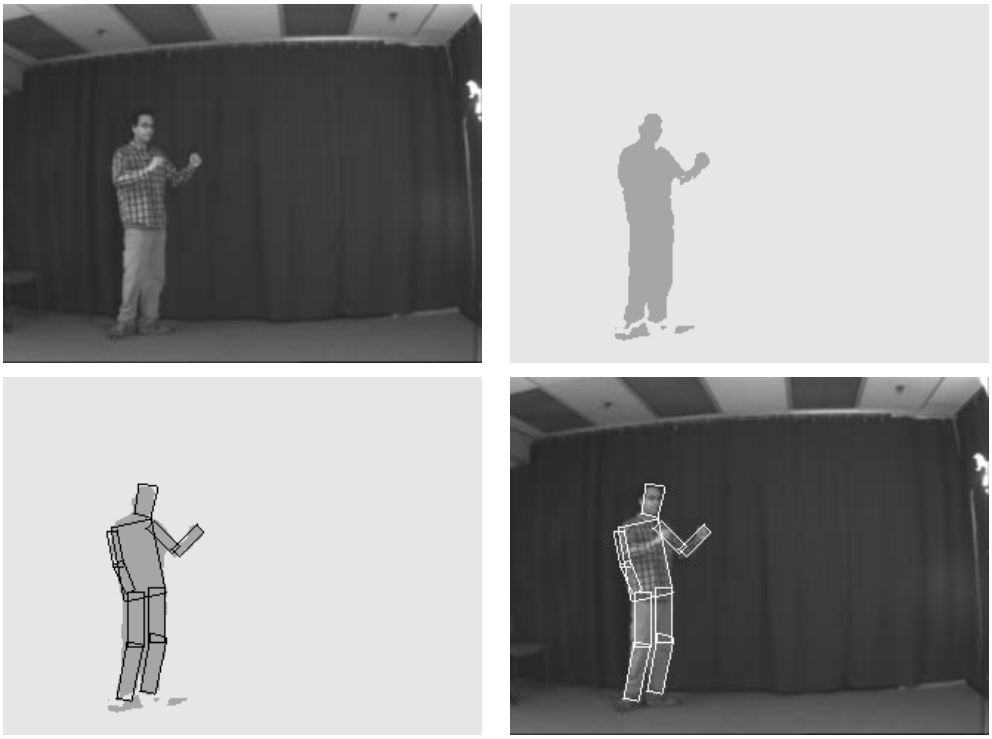

Figure 15: In this case, the silhouette doesn't provide enough information to estimate the position of one arm.

model the geometric configuration of the parts, which is treated as independent of their individual appearances. Our framework is general, in the sense that it is independent of the specific method used to represent the appearance of parts, and the type of the geometric relationships between the parts. It allows for a variety of kinds of part models and geometric relations between parts. By using a general framework we have provided a set of computational mechanisms that can be used for many different modeling schemes. In this paper we presented two quite different modeling schemes, one was used to model faces and the other to model articulated bodies.

Given a particular kind of part model and geometric constraints, the parameters of the models are learned from training examples. We demonstrated the learning techniques using both simple face models and a simple model of the human body. With these learned models we were then able to detect the corresponding objects and estimate their pose in novel images. The learning and the detection techniques are both computationally efficient – in a theoretical (asymptotic) sense and in practice. Learning the models from training examples takes just a couple of minutes on a standard desktop workstation. Detection, searching over the entire space of possible

configurations, takes less than a second for the face models and less than a minute for the body model.

### 7.1 Extensions

There are a number of possible extensions to our work:

- Occluded parts can be handled by making $p(I|l_i, u_i)$ a robust measure. Basically the likelihood should never be too small, even when there is no evidence for the part at some location. The context provided by the unoccluded parts can be rich enough to constrain the location of occluded parts.

- There are other ways of detecting multiple instances of an object that we did not consider here. For instance, the MAP estimation algorithm can be used to output the configuration with maximum posterior probability conditioned on each location for the root part. This doesn't take any more time than computing the MAP estimate itself. We could select all locations for the root that yield a high posterior.

- Given an image sequence, our method can be used to detect an object in the first frame and then use that location as prior information for the detection in the next frame. The detection algorithms can be modified to take into account prior information over absolute locations. This would yield a system for tracking flexible objects.

## References


[1] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *PAMI*, 12(9):855–867, September 1990.

[2] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11(7):1691–1715, October 1999.

[3] N.J. Ayache and O.D. Faugeras. Hyper: A new approach for the recognition and positioning of two-dimensional objects. *PAMI*, 8(1):44–54, January 1986.

[4] J.O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, 1985.

[5] G. Borgefors. Distance transformations in digital images. *CVGIP*, 34(3):344–371, June 1986.

[6] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *PAMI*, 10(6):849–865, November 1988.

[7] Y. Boykov, O. Veksler, and R. Zabih. Energy minimization with discontinuities. *Under Review*, 1998.

[8] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *CVPR98*, pages 648–655, 1998.

[9] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *ECCV98*, pages II:628–641, 1998.

[10] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Information Theory*, 14(3):462–467, May 1968.

[11] T.H. Cormen, C.E. Leiserson, and Rivest R.L. *Introduction to algorithms*. MIT Press and McGraw-Hill, 1996.

[12] S.J. Dickinson, I. Biederman, A.P. Pentland, J.O. Eklundh, R. Bergevin, and R.C. Munck-Fairwood. The use of geons for generic 3-d object recognition. In *IJCAI93*, pages 1693–1699, 1993.

[13] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient matching of pictorial structures. In *CVPR00*, pages II:66–73, 2000.

[14] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, June 1981.

[15] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *TC*, 22(1):67–92, January 1973.

[16] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *PAMI*, 13(9):891–906, September 1991.

[17] A.E. Gelfand and A.F.M. Smith. Sampling-based approaches to calculating marginal densities. *J. Royal Stat. Association*, 85:398–409, 1990.

[18] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 6(6):721–741, November 1984.

[19] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *PAMI*, 9(4):469–482, July 1987.

[20] E.J. Gumbel, J.A. Greenwood, and D. Durand. The circular normal distribution: Theory and tables. *J. American Stat. Association*, 48:131–152, March 1953.

[21] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *PAMI*, 15(9):850–863, September 1993.

[22] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *IJCV*, 5(2):195–212, November 1990.

[23] S. Ioffe and D.A. Forsyth. Probabilistic methods for finding people. *IJCV*, 43(1):45–68, June 2001.

[24] H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *CVPR98*, pages 125–131, 1998.

[25] A.V. Karzanov. Quick algorithm for determining the distances from the points of the given subset of an integer lattice to the points of its complement. *Cybernetics and System Analysis*, pages 177–181, April-May 1992. Translation from the Russian by Julia Komissarchik.

[26] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Affine invariant model-based object recognition. *RA*, 6:578–589, 1990.

[27] B. Moghaddam and A.P. Pentland. Probabilistic visual learning for object representation. *PAMI*, 19(7):696–710, July 1997.

[28] H. Murase and S.K. Nayar. Visual learning and recognition of 3-d objects from appearance. *IJCV*, 14(1):5–24, January 1995.

[29] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[30] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[31] R.P.N. Rao and D.H. Ballard. An active vision architecture based on iconic representations. *AI*, 78(1-2):461–505, October 1995.

[32] E. Rivlin, S.J. Dickinson, and A. Rosenfeld. Recognition by functional parts. *CVIU*, 62(2):164–176, September 1995.

[33] L.G. Roberts. Machine perception of 3-d solids. In *Optical and Electro-optical Information Processing*, pages 159–197, 1965.

[34] W. Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*. Springer-Verlag, 1996. LNCS 1173.

[35] M. Turk and A.P. Pentland. Eigenfaces for recognition. *CogNeuro*, 3(1):71–96, 1991.

[36] W.M. Wells, III. Efficient synthesis of gaussian filters by cascaded uniform filters. *PAMI*, 8(2):234–239, March 1986.

[37] L. Wiskott, J. Fellous, N. Kruger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *PAMI*, 19(7):775–669, July 1997.