Object Recognition with Pictorial Structures

by

Pedro F. Felzenszwalb

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

Author ....
Department of Electrical Engineering and Computer Science
May, 7, 2001

Certified by .... [illegible] ....
W. Eric L. Grimson
Bernard Gordon Professor of Medical Engineering
Thesis Supervisor

Accepted by .... [illegible] ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Object Recognition with Pictorial Structures

by

Pedro F. Felzenszwalb

Submitted to the Department of Electrical Engineering and Computer Science
on May, 7, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis presents a statistical framework for object recognition. The framework is motivated by the pictorial structure models introduced by Fischler and Elschlager nearly 30 years ago. 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. The problem of detecting an object in an image and the problem of learning an object model using training examples are naturally formulated under a statistical approach. We present efficient algorithms to solve these problems in our framework. We demonstrate our techniques by training models to represent faces and human bodies. The models are then used to locate the corresponding objects in novel images.

Thesis Supervisor: W. Eric L. Grimson
Title: Bernard Gordon Professor of Medical Engineering

Acknowledgments

I would like to thank my advisor, Eric Grimson, for his guidance and support. I would also like to thank Paul Viola for many helpful conversations.
Much of the foundation of this thesis was developed in collaboration with Dan Huttenlocher (thanks dan!).

This research was supported in part by Nippon Telegraph and Telephone.

Contents

Chapter 1

Introduction

The problem of object detection and recognition is central to the field of computer vision. Classical computer vision methods concentrate on objects with fixed or parameterized shapes or with known photometric information (see [17, 23, 30, 19, 25]). This was a good starting point for the field, since it made the recognition problem well defined, and allowed for the development of important mathematical and algorithmic tools. On the other hand, no artificial system can recognize generic objects like a dog, a house or a tree. These objects don't have fixed shape or photometric information.

We believe that many object classes can be characterized solely by their visual appearance, even though the objects in each class have large variations in shape and detailed photometric information. This thesis presents a statistical framework that allows for qualitative descriptions of appearance, making it suitable for many generic recognition problems. Our framework is motivated by the pictorial structure representation introduced in [13]. The problem of detecting an object in an image and the problem of learning an object model using training examples are naturally formulated under a statistical approach. We present efficient algorithms to solve these problems in our framework. We demonstrate our techniques by training models to represent faces and human bodies. The models are then used to locate the corresponding objects in novel images, as shown in Figure 1-1.

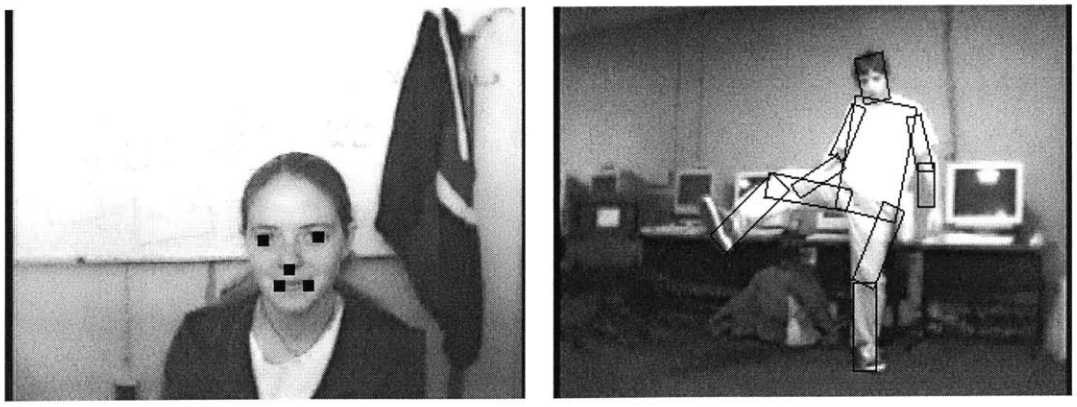

Figure 1-1: Detection results for a face (left) and a human body (right). Each image shows the globally best location for the corresponding object, as computed by our algorithms. The object models were constructed from training examples.

1.1 Pictorial Structures

Pictorial structures were introduced by Fischler and Elschlager [13] nearly 30 years ago. The basic idea is to model an object by a collection of parts arranged in a deformable configuration. We model the appearance of each part is separately, and the deformable configuration is represented by spring-like connections between pairs of parts. The appearance of a part is encoded by a function which measures how much a location in an image looks like the corresponding part. In [13], 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.

The appearance model for each part can be fairly generic. This is because parts are not recognized on their own, but together with the other parts in the object description. This 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 groups them together to form objects. While separate recognition of each part seems attractive from a computational point of view, it forces one to use more complex part models. In the pictorial structure framework, parts can be generic to the point that trying to locate them individually would fail (one would get too many false positives or too many false negatives).

As mentioned, the deformable configuration of parts is represented by connections between them. A connection between two parts indicates relationships between their locations. For example, a connection can enforce precise geometrical constraints, such as a revolute or prismatic joint between two parts. Connections can also represent more generic relationships such as "close to", "to the left of", or even something in between these generic relationships and precise geometrical constraints.

Since both the part models and the relationships between parts can be fairly 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, with a small number of parts connected by flexible revolute joints. In this case it is important that the joints 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 flexible revolute joints should try to enforce that connected parts be aligned at their joint, but allow for small misalignment, penalizing it in the energy function. Moreover, the angle between certain pairs of parts should be arbitrary, while the angle between other pairs should be fairly constrained. Note that it would be impossible to detect generic parts such as "lower-leg" or "upper-arm" on their own. On the other hand, the structure between parts provide sufficient context to detect the human body as a whole.

The pictorial structure framework is general, in the sense that it is independent of the specific scheme used to model the appearance of individual parts, and the exact type of relationships between parts. Articulated objects can be modeled by the appearance of each rigid part and connections that behave like joints. We describe such models in Chapter 5. In [13], faces and terrain maps were modeled by the appearance of local features and spatial relationships between those features. This is the nature of the models presented in Chapter 4. In [22], pictorial structures were used to represent generic scene concepts such as waterfalls, snowy mountains and sunsets. For example, a waterfall was modeled as a bright white region (water) in the middle of darker regions (rocks). There are many other modeling schemes which can be seen as particular implementations of the pictorial structure framework, such as [3] and [9].

1.2 Statistical Formulation

In their original work, Fischler and Elschlager only addressed the problem of finding the best alignment of a pictorial structure model to an image. As mentioned before, 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 present a statistical formulation of the pictorial structure framework. The original matching problem studied by Fischler and Elschlager is equivalent to finding the maximum a posteriori (MAP) estimate of the object location given an observed image in our formulation. The new formulation helps to characterize the different model parameters. In fact, all parameters can be determined empirically using statistical estimation. This way we can construct models automatically, using only a few training examples. The idea is to use the training examples to estimate a model under the maximum likelihood (ML) formalism. This is a big advantage over picking model parameters manually. Learning from examples is an important capability for an intelligent system. Moreover, a user can't usually find the best parameters for a model by trial and error.

Another approach to the object detection problem arises naturally from the statistical formulation (besides MAP estimation). The idea is to sample object locations from their posterior probability distribution. When there is a lot of uncertainty in the object location, sampling is useful to produce multiple hypotheses. Also, sometimes our statistical model only approximates the "true" posterior probability of the object location. Sampling allow us to find many locations for which our posterior is high, and select one of those as the correct one using some other measure. This is similar to the idea behind importance sampling (see [15]). It can also be seen as a mechanism for visual selection (see [2]).

1.3 Efficient Algorithms

Our main motivation is to construct a framework that is rich enough to capture the appearance of many generic objects and for which we can solve the object detection and model learning problems efficiently. We present algorithms to solve these problems for a natural class of pictorial structure models. Our methods require that the set of connections between parts form a tree structure, and that the relationships between connected parts be of a particular (but quite general) form.

Restricting the relationships between parts to a tree structure is natural. 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 the relationships between parts similarly allows a broad range of objects to be modeled.

We present examples illustrating that our algorithms enable efficient search for the *globally* best match of relatively generic objects to an image. Figure 1-1 shows matching results for a face model, and for a model of the human body. Both these models were automatically constructed using training examples.

The asymptotic running time of our matching algorithms is optimal, in the sense that they run as quickly as it takes to match each part separately, without accounting for the relationships between parts. In practice, the algorithms are also fast, finding the globally best match of a pictorial structure to an image in a few seconds.

Chapter 2

General Framework

In this chapter we present the statistical framework for pictorial structures. As described in Section 1.3, our main motivation is to construct a rich class of models for which we can develop efficient algorithms to solve the object detection and model learning problems.

2.1 Statistical Approach

A typical 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 θ 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. If the posterior is low everywhere we might decide that the object is not visible in the image.

- Sampling - this is the problem of sampling from the posterior distribution. In general, the posterior distribution we define is only an approximation of the "true" one. Sampling allows us to find many locations for which our posterior is high, and evaluate them using some other method. In this way, our framework can be used to generate a number of promising hypothesis for the location of the object. Each hypothesis must be verified, but there are only a small number of them.

- Model estimation - this is the problem of finding θ which specifies a good model for a particular object. We would like to build models using some sort of training examples.

The next section describes how we model $p(I|L,\theta)$ and $p(L|\theta)$. Later, we show how to estimate model parameters using training examples, and in Chapter 3 we present efficient algorithms to compute the MAP estimate of the object location and to sample from its posterior distribution.

2.2 Pictorial Structures

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 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

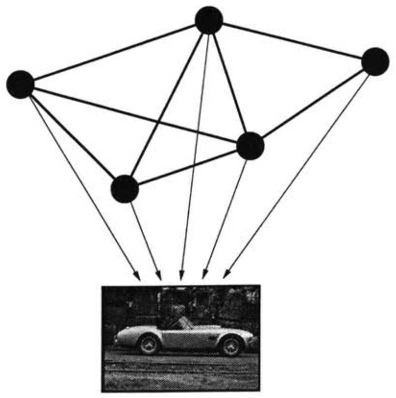
Figure 2-1: Graphical representation of the dependencies between the location of object parts (black nodes) and the image. In the case of a car, each black node would correspond to a part such as a wheel, the body, etc.

more complex parameterizations are also possible. For example, a location can specify the position, angle, 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,

$$p(L|I, \theta) \propto p(I|L, \theta)p(L|\theta), \tag{2.1}$$

where the likelihood, $p(I|L, \theta)$, measures the probability of seeing image I given a particular configuration for the object. Figure 2-1 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 this distribution 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, we can 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 [27]). Similar algorithms are known in the Bayesian Network community as belief propagation and belief revision (see [26]). These algorithms can be implemented to take $O(h^2n)$ time, where n is the number of object parts, and h is a discrete number of possible locations for each part. Unfortunately, this is not good enough. The number of possible locations for each part can be huge, and a quadratic algorithm takes too long. We identify a restriction on the type of dependencies between parts for which we can obtain algorithms that run in $O(hn)$ time. These algorithms are 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 4, a part is modeled as a local image feature, based on image derivatives around a point, while in Section 5 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 as the product of the individual likelihoods,

$$p(I|L, u) \propto \prod_{i=1}^{n} p(I|l_i, u_i). \tag{2.2}$$

This approximation is good if the parts don't overlap, as they would generate different portions of the image. But the approximation can be bad if one part occludes another. The articulated models in Section 5 provide examples where the approximation can be problematic. For those models, the MAP estimate of an object location can be a poor estimate of its position. On the other hand, we show that we can find the true location by sampling from the posterior. We sample to find many locations with high posterior, and select one of those using a different measure.

In our models, the set of connections between parts forms a tree structure. The dependencies between parts are characterized by some parameters $c = \{c_{ij} \mid (v_i, v_j) \in E\}$. For example, one connection might indicate that a given part tends to be at a certain distance to the left of another one. We don't model any preference over the absolute location of object parts, only over their relative configuration. Let $p(l_i) = 1$ for simplicity. Our efficient algorithms require that the joint distribution for the locations of two connected parts be expressed in a specific form. There are a few different possibilities, here we concentrate on the following form. Suppose we have a Normal distribution in a transformed space,

$$p(l_i, l_j|c_{ij}) = \mathcal{N}(T_{ij}(l_i) - T_{ji}(l_j), 0, \Sigma_{ij}), \tag{2.3}$$

where T_{ij}, T_{ji}, and Σ_{ij} are the connection parameters encoded by c_{ij}. The covariance matrix Σ_{ij} should be diagonal, and for simplicity we will assume that T_{ij}, and T_{ji} are invertible. We further require that it be possible to discretize $T_{ji}(l_j)$ in a grid (which in turn specifies a number of discrete locations l_j). 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 Σ_{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 4 we describe simple feature based models where the connections between parts behave like springs. More complex models are described in Section 5, where the connections between parts behave like flexible joints.

The prior distribution over object locations can be defined in terms of the joint distributions for pairs of connected parts,

$$p(L|E, c) = \prod_{(v_i, v_j) \in E} p(l_i, l_j|c_{ij}). \tag{2.4}$$

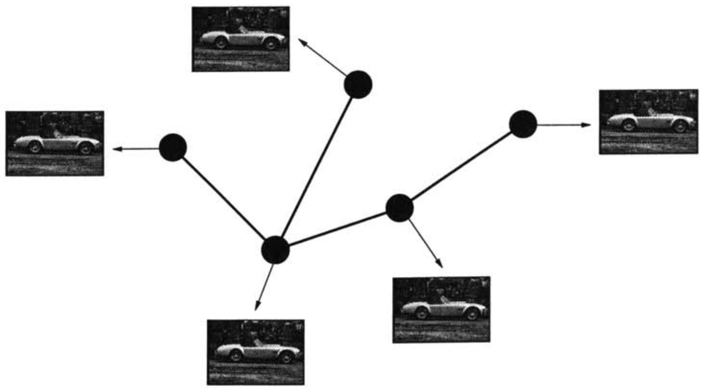

Figure 2-2: Graphical representation of the dependencies between the location of object parts (black nodes) and the image in the restricted models (see text).

Note that this is not a real probability distribution over locations. It actually integrates to infinity! The joint distributions described above have the same problem. What is happening is that we have an uninformative prior over absolute locations (see [4]). We can interpret these functions as distributions over equivalence classes. Each equivalence class corresponds to object configurations which have different absolute locations, but the relative locations between parts are the same.

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 both $p(I|L, \theta)$, 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 $p(L|I, \theta)$, the probability that the object is at some configuration in an image. A graphical representation of our restricted models is shown in Figure 2-2.

2.3 Estimating 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. The problem is 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 θ is, by definition, the value θ^* 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

$$\theta^* = \arg\max_{\theta} \prod_{k=1}^{m} p(I^k|L^k, \theta) \prod_{k=1}^{m} p(L^k|\theta). \tag{2.5}$$

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. Thus, we can independently solve for u^* and the pair E^*, c^*.

2.3.1 Estimating the Appearance Parameters

From equation (2.5) 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 (2.2). 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.

2.3.2 Estimating the Dependencies

From equation (2.5) we get

$$E^*, c^* = \arg\max_{E,c} \prod_{k=1}^{m} p(L^k|E, c). \tag{2.6}$$

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 (2.4) 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 (2.6) and re-ordering the factors we get,

$$E^*, c^* = \arg\max_{E,c} \prod_{(v_i,v_j)\in E} \prod_{k=1}^{m} p(l_i^k, l_j^k|c_{ij}). \tag{2.7}$$

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}^* (since it may vary 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,

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

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 (2.7) we let,

$$E^* = \arg\max_E \prod_{(v_i,v_j)\in E} q(v_i, v_j) = \arg\min_E \sum_{(v_i,v_j)\in E} -\log q(v_i, v_j). \tag{2.8}$$

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 reduces 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 (2.8). 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.

Chapter 3

Matching Algorithms

Now we describe efficient algorithms to match pictorial structure models to images. The first one finds the MAP estimate of the object location given an observed image. That algorithm was first presented in [12]. The second method samples configurations from the posterior distribution. Both algorithms work in a discretized space of locations for each part. They basically run 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, h is a number of discrete locations for $T_{ji}(l_j)$, which usually matches the number of locations for l_j. Sometimes, however, it can be a little larger, as shown in Section 5.

3.1 MAP Estimate

Remember that the MAP estimate of the object location is a configuration with highest probability given an observed image. In some sense, this is our best guess for the object location (see [4] for a theoretical justification for using the MAP estimate). Now we show how to efficiently compute this "best" configuration.

The MAP estimate is given by

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

Equation (2.2) characterizes the likelihood $p(I|L,\theta)$ in terms of an appearance model for each part, and equation (2.4) 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,

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right), \tag{3.1}$$

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 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 (3.1) 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, where h is a number of discrete locations for each part, and n is the number of parts in the model. (see [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, 20]). 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 3.1.1. However, this $O(h^2n)$ algorithm is not practical, because the number of possible locations for each part, h, is usually huge. When searching for the best possible match of a pictorial structure to an image, there is no natural way to limit the space of locations. For example, the number of locations for a part is usually at least as large as the number of pixels in the image, making h on the order of 10^5 possible values.

The restricted form of the joint distribution for the locations of two connected parts in equation (2.3) is,

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

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

$$d_{ij}(l_i, l_j) = (T_{ij}(l_i) - T_{ji}(l_j))^T \, \Sigma'_{ij} \, (T_{ij}(l_i) - T_{ji}(l_j)), \tag{3.2}$$

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

3.1.1 Efficient Minimization

In this section, we show how to use dynamic programming to find the configuration $L^* = (l_1^*, \ldots, l_n^*)$, minimizing equation (3.1) when the graph G is a tree. This is an instance of a known class of dynamic programming techniques and is a generalization of the technique for chains that is used in solving snakes problems (e.g., [1]). The computation involves $(n-1)$ functions, each of which specifies the best location of one part with respect to the possible locations of another part.

Given a tree $G = (V, E)$, 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 $(d_i - 1)$.

First we note that 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 (3.1) is $m_j(l_j) + d_{ij}(l_i, l_j)$. Hence, the quality of the best location of v_j given location l_i of v_i is

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) \right), \tag{3.3}$$

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

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) + \sum_{v_c \in C_j} B_c(l_j) \right). \tag{3.4}$$

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 (3.3) 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 (3.1) 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, in turn, specify an 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 (3.3). 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 (3.4). 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 now be computed by tracing 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 $B_j'(l_i)$. The typical way to compute these functions takes $O(h^2)$ time. This is done by considering 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 equation (3.2). The method will compute each pair $B_j(l_i)$ and $B_j'(l_i)$ in $O(h)$, yielding an $O(hn)$ algorithm overall.

3.1.2 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, 19]). The trivial way to compute this function takes $O(h|B|)$ time, where h is the number of locations in the grid. On the other hand, efficient algorithms exist to compute the distance transform in $O(h)$ time, independent of the number of points in B (see [5, 21]). 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 ∞ otherwise. This suggests a generalization of distance transforms to functions as follows. Let the distance transform of a function f defined over a grid $\mathcal{G}$ be

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

Intuitively, for each grid location x, this function finds a location y that is close to x and for which $f(y)$ is small. Note that difference between the value of $\mathcal{D}_f$ at two locations is bounded by the distance between the locations, regardless of how quickly the function f changes. 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 (3.2), the functions $B_j(l_i)$ that must be computed by the dynamic programming algorithm can be rewritten as generalized distance transforms under the Mahalanobis distance d_{ij},

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

where

$$f(y) = m_j(T_{ji}^{-1}(y)) + \sum_{v_c \in C_j} B_c(T_{ji}^{-1}(y)),$$

and the grid $\mathcal{G}$ specifies a discrete set of 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 changes slowly).

It turns out that some of the efficient algorithms used to compute the classical distance transform can be modified to compute the generalized distance transform under different distances. The method of Karzanov (originally in [21], but see [29] for a better description) can be changed to compute the transform of a function under a Mahalanobis distance with diagonal covariance matrix. The algorithm can also compute $B'_j(l_i)$ as it computes $B_j(l_i)$.

3.2 Sampling from the Posterior

We now turn to the problem of sampling from the posterior distribution of object configurations. When there is a lot of uncertainty in the object location, sampling is useful to produce multiple hypotheses. Sometimes our statistical model only approximates the "true" posterior probability of an object location in an image. In that case, simply computing the MAP estimate might give poor results. By sampling we can find many locations for which our posterior is high, and select one of those as the correct one using some other measure.

The sampling problem can be solved with an algorithm similar to the one used to compute the MAP estimate. 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,

$$S_j(l_i) = \sum_{l_j} \left(p(I|l_j,u_j) p(l_i,l_j|c_{ij}) \prod_{v_c \in C_j} S_c(l_j) \right). \tag{3.5}$$

These recursive functions already give a polynomial algorithm to compute $p(l_r|I,\theta)$. As in the MAP estimation algorithm we can compute them 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 $S_j(l_i)$ in $O(h)$ time for the case where $p(l_i,l_j|c_{ij})$ is in the special form given by equation (2.3). 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,

$$p(l_j|l_i,I,\theta) \propto p(I|l_j,u_j) p(l_i,l_j|c_{ij}) \prod_{v_c \in C_j} S_c(l_j). \tag{3.6}$$

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 $O(h)$ time, yielding a $O(hn)$ 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 (3.6) 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.

3.2.1 Computing the S functions

We want to efficiently compute the function in equation (3.5). 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,

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

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

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

where

$$f(y) = p(I|T_{ji}^{-1}(y), u_j) \prod_{v_c \in C_j} S_c(T_{ji}^{-1}(y)).$$

Just like when computing the generalized distance transform, the convolution is done over a discrete grid which specifies possible values for $T_{ji}(l_j)$ (which in turn specify a set of locations l_j). The Gaussian filter $\mathcal{G}$ is separable since the covariance matrix Σ_{ij} is diagonal. We can compute a good approximation for the convolution in linear time using the techniques from [31].

3.3 Summary

In this chapter, we have presented two different algorithms that can be used to locate pictorial structure models in images. Together with the model learning method from Section 2.3, these algorithms form the base of a complete recognition system. The next two chapters describe two modeling schemes that represent objects in very different ways. The two schemes use the same computational mechanisms, which are exactly the algorithms presented so far.

Chapter 4

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 features and spatial relationships between those features. This type of model has been popular in the context of face detection (see [13, 9, 32]). We first describe how we model the appearance of a feature, and later describe how we model spatial relationships between features. In Section 4.3 we show experiments of face detection.

4.1 Features

The location of a feature 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 features we use the iconic representation developed in [28]. The iconic representation is based on Gaussian derivative filters of different orders, orientations and scales. To describe an image patch centered at some position we collect the response of all filters at that point in a high-dimensional vector. This vector is normalized and called the iconic index at that position. Figure 4-1 shows the nine filters used to build the iconic representation at a fixed scale. In practice, we use three scales, given by $\sigma_1 = 1$,

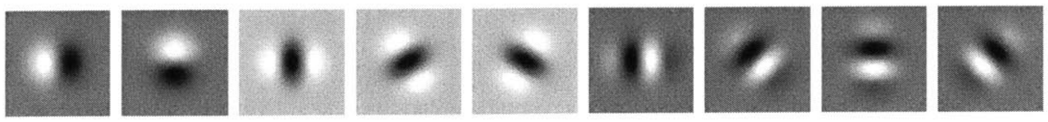

Figure 4-1: Gaussian derivative basis functions used in the iconic representation.

$\sigma_2 = 2$, 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 [28]).

The appearance of a feature is modeled by a distribution over iconic indices. Specifically, we model the distribution of iconic indices at the location of a feature 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,

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

where $\alpha(l_i)$ is the iconic index at location l_i in the image. So each dimension of $\alpha(l_i)$ is the response of a different Gaussian derivative filter at location l_i. 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 can use other methods to represent the appearance of features. In particular, we experimented with the eigenspace techniques from [24]. 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 efficiently by convolving each level of a Gaussian pyramid with small x-y separable filters (see [14]).

4.2 Spatial Distribution

The spatial configuration of features is modeled by a collection of springs connecting pairs of them. Each connection (v_i, v_j) is characterized by the ideal relative location of the two connected parts s_{ij}, and a covariance matrix Σ_{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_j with respect to the location of part v_i as a Gaussian with mean s_{ij} and covariance Σ_{ij},

$$p(l_i, l_j | c_{ij}) = \mathcal{N}(l_j - l_i, s_{ij}, \Sigma_{ij}). \tag{4.1}$$

So, ideally the location of part v_j is the location of part v_i shifted by s_{ij}. Since the models are deformable, the location of v_j can vary (which corresponds to stretching the spring), by paying a cost that depends on the covariance matrix. 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 in the appearance model, the maximum likelihood parameters of this distribution 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 (2.3). 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

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

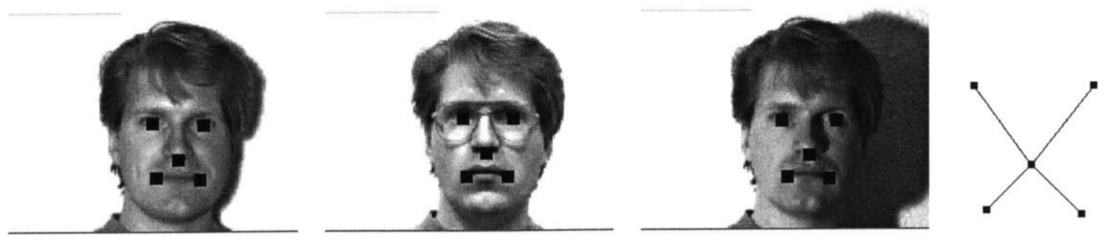

Figure 4-2: Three examples from the first training set and the structure of the learned model.

which allow us to write equation (4.1) in the right form,

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

4.3 Experiments

In this section, we present experiments of using the iconic models we 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 features, corresponding to the eyes, nose, and corners of the mouth. To generate training examples we labeled the location of each feature 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 training examples and the structure of the learned model are shown in Figure 4-2. Remember that we never told the system which features should be connected together. Picking a structure is part of the ML parameter estimation procedure.

We tested the 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 4-3. Both the learning and matching algorithms are extremely fast. Using a desktop computer it took a few seconds to learn the model and about a second to compute the MAP estimate in each image.

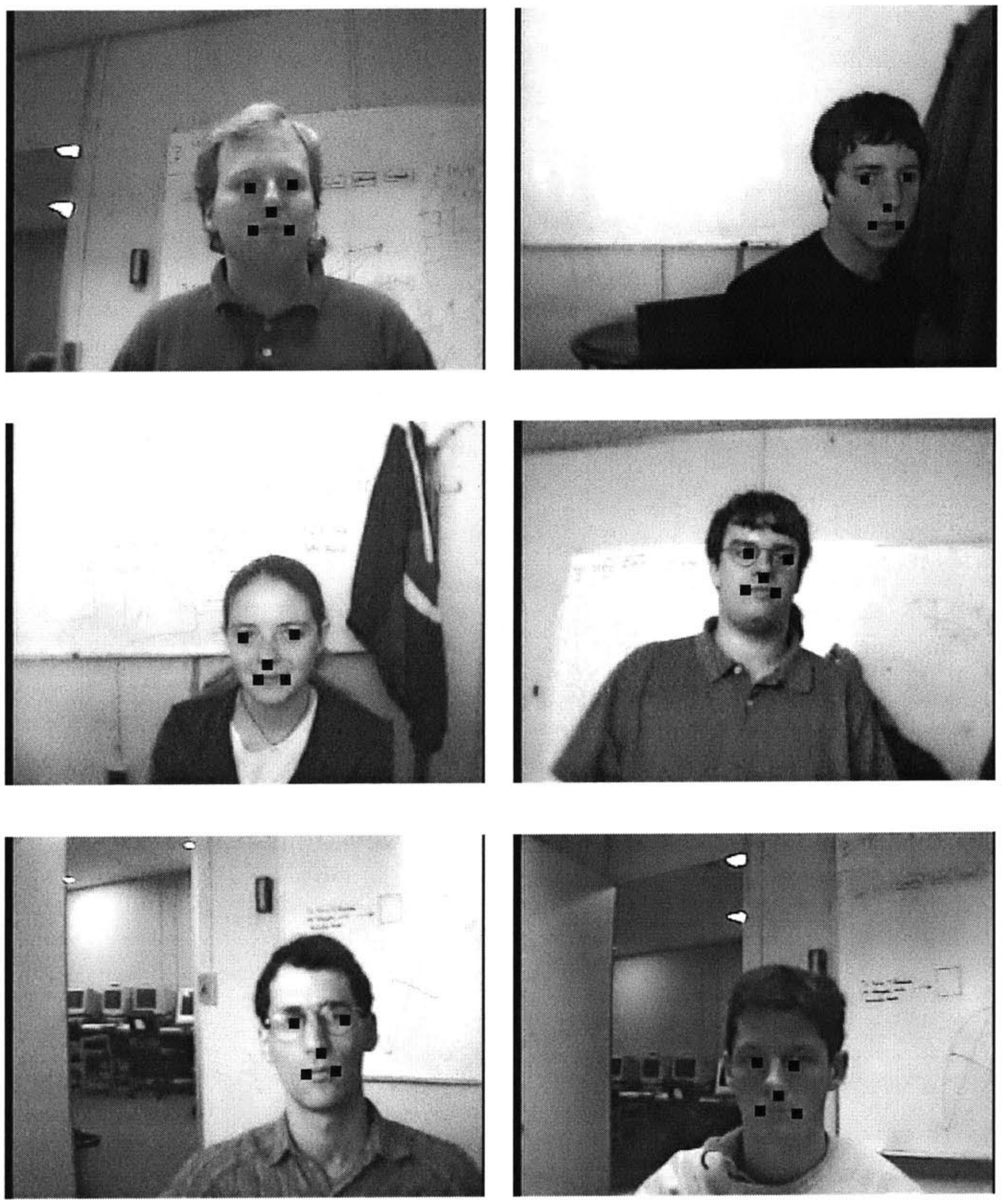

Figure 4-3: Matching results.

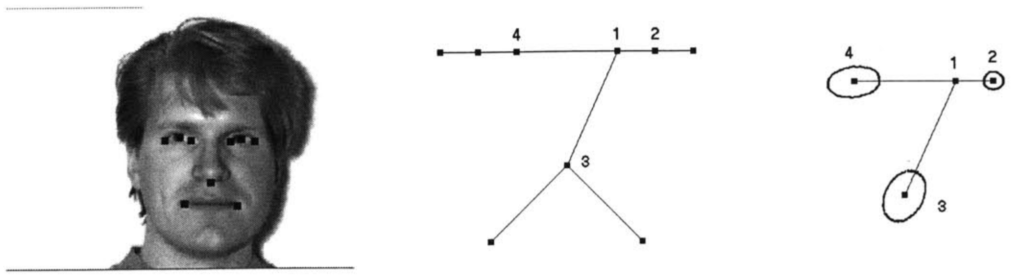

Figure 4-4: One example from the second training set, the structure of the learned model, and a pictorial illustration of the connections to one of the parts, showing the location uncertainty for parts 2, 3, and 4, when part 1 is at a fixed position.

The first experiment demonstrates that we can learn a useful model from training examples. The structure of this model is not particularly interesting. All parts are connected through a central part, and the properties of each connection are similar. So we tried learning a larger model, this one with nine parts. We now have three features 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 4-4 shows one of the training examples and the learned model. Also, in Figure 4-4, there is a detailed illustration of the connections to the left corner of the right eye. The ellipses illustrate 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. The location of the pupil is much more constrained with respect to the location of the eye corner than any other part. Also note that the distributions are not centrally symmetric. We see that the algorithm learned an interesting structure for the model, and automatically determined the constraints between the locations of different pairs of parts.

Chapter 5

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 model from the original input image. Figure 5-1 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 won't assume "perfect" silhouette images. In fact, our method works with very noisy input.

5.1 Parts

For simplicity, assume that the image of an object is generated by a scaled orthographic projection, and that the scale factor of the projection is known. We can easily add an extra parameter in our search space to relax this later.

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 comes from the length of the cylinder and can vary due to foreshortening. In practice, we model the projection of a part as

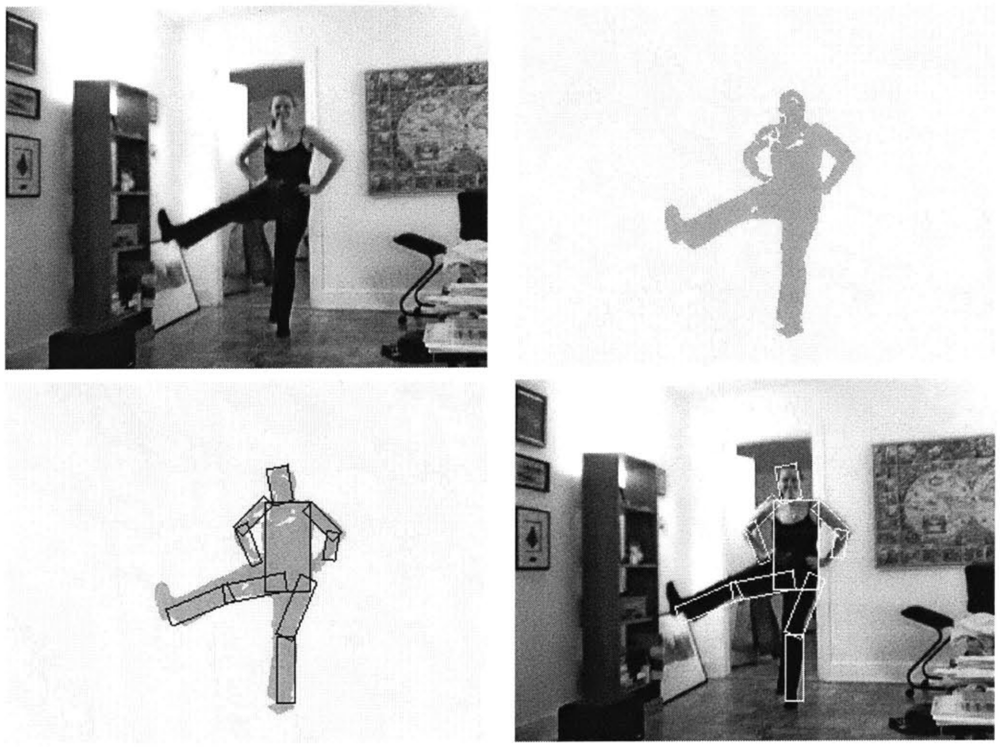

Figure 5-1: Input image, silhouette obtained by background subtraction, and matching result.

a rectangle parameterized by (x, y, s, θ). 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 θ. So we have a four-dimensional pose space for each part.

We model $p(I|l_i, u_i)$ in the following way. First, each pixel in the image is generated independently. Pixels inside the rectangle specified by l_i 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 5-2). 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, when computing $count_1$, we consider a slightly dilated version of the silhouette, and to compute $count_2$ we erode the silhouette. 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 good. 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, the 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

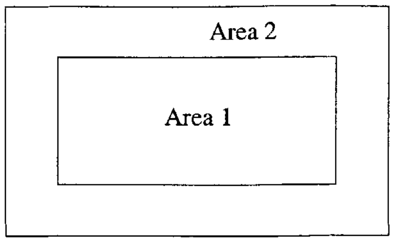

Figure 5-2: A rectangular part. $area_1$ is the area inside the part, and $area_2$ is the border area around it.

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 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. This is similar to the idea behind importance sampling.

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. The problem is that 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 trick, borrowed from the principle of annealing (see [16]). Note that $p'(I|L,u)$ is just the product of the smoothed likelihoods for each part. In all our experiments we used $T = 10$.

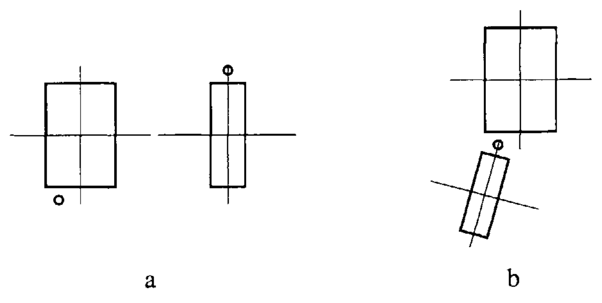

Figure 5-3: Two parts of an articulated object, (a) in their own coordinate system and (b) the ideal configuration of the pair.

5.2 Geometry

For the articulated objects, pairs of parts are connected by flexible joints. A pair of connected parts is illustrated in Figure 5-3. 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 5-3a. In an ideal configuration these points coincide, as illustrated in Figure 5-3b. The ideal relative orientation is given by θ_{ij}, the difference between the orientation of the two parts, and the ideal relative length is given by s_{ij}.

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 values and the observed values,

$$\begin{aligned} p(l_i, l_j | c_{ij}) \;=\; & \mathcal{N}(x'_j - x'_i, 0, \sigma_x^2) \\ & \mathcal{N}(y'_j - y'_i, 0, \sigma_y^2) \\ & \mathcal{N}(s_j - s_i, s_{ij}, \sigma_s^2) \\ & \mathcal{M}(\theta_j - \theta_i, \theta_{ij}, k), \end{aligned} \tag{5.1}$$

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

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

The distribution over angles, $\mathcal{M}$, is the von Mises distribution (see [18]),

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

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 between the relative sizes of the two parts and the ideal relative size. The last term measures the difference between the relative angle of the two parts and the ideal relative angle. Usually σ_x and σ_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,

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

Finding the maximum likelihood estimate of (s_{ij}, σ_s^2) is easy since we just have a Gaussian distribution over the size differences. s_{ij} is just the mean size difference over the examples and σ_s^2 is the sample variance. Similarly, there are known methods to find the ML parameters (θ_{ij}, k) of a von Mises distribution (see [18]). The ML estimate of the joint location in each part is 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 (σ_x^2, σ_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 (2.3). 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 α and β. 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

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

which allow us to write equation (5.1) in the right form,

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

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

5.3 Experiments

In this section, we present experiments of using the articulated models just described 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 shown in Figure 5-4. 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).

We tested the model by matching it to novel images. As described in Section 5.1, the MAP estimate can be a bad guess for the object location. Therefore we sample configurations from the posterior distribution and rate each sample using a separate measure. For each sample we computed a Chamfer distance between the shape of

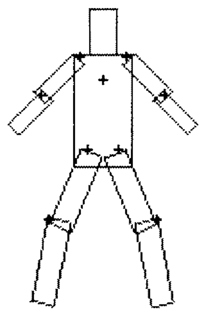
Figure 5-4: Model of human body.

the object under that 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 5-5. First, a silhouette is obtained from the original image using background subtraction. We use the silhouette as input to the sampling algorithm and obtain a number of different pose hypothesis. The best pose is then selected using the Chamfer measure.

More matching results are shown in Figure 5-6. 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 example. 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, θ) parameters.

Figure 5-7 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 perfect. The most common source of error comes from ambiguities in the silhouette. Figure 5-8 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 samples with equally good Chamfer distance.

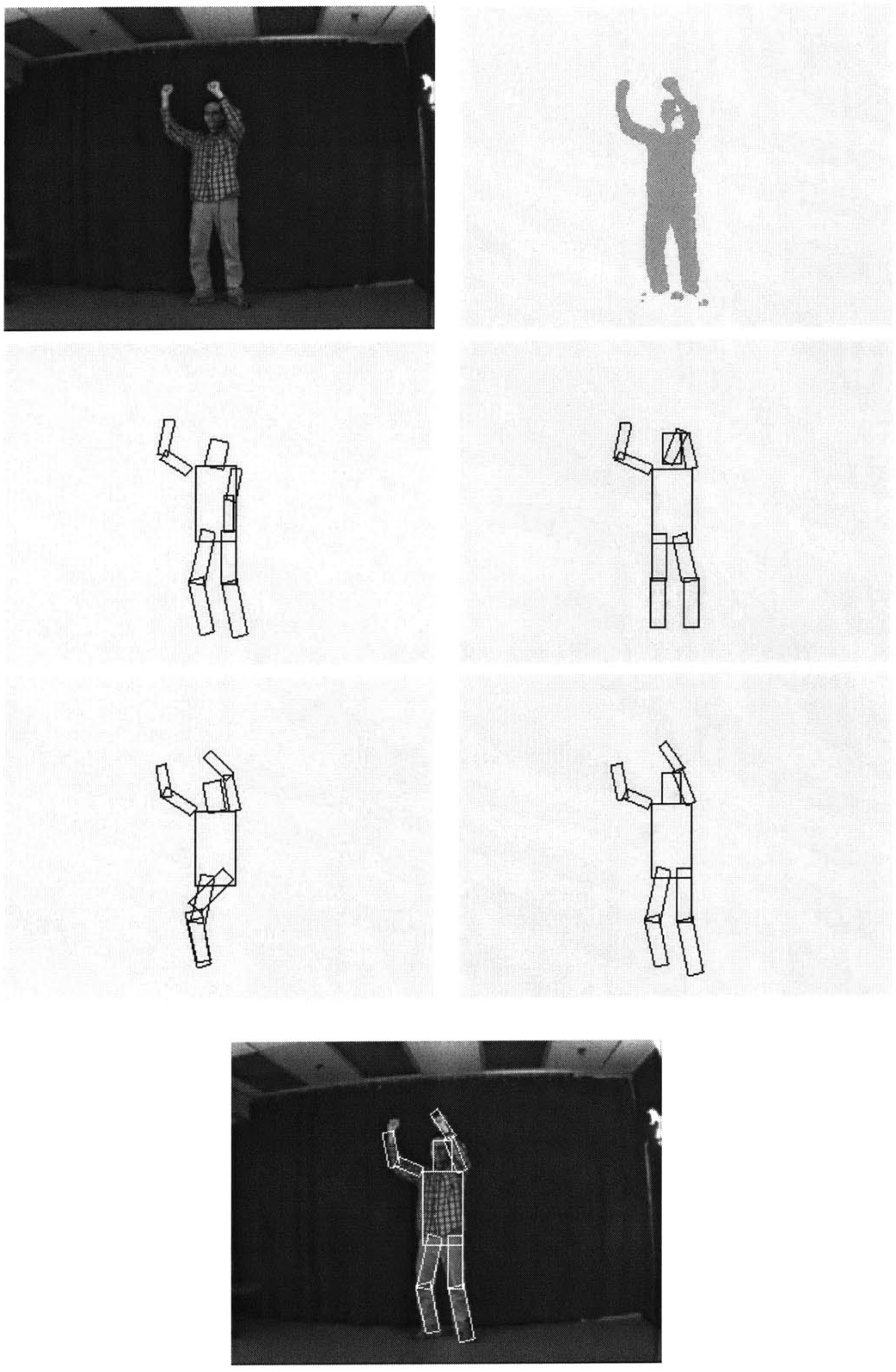

Figure 5-5: Input image, silhouette, random samples, and best result selected using the Chamfer distance.

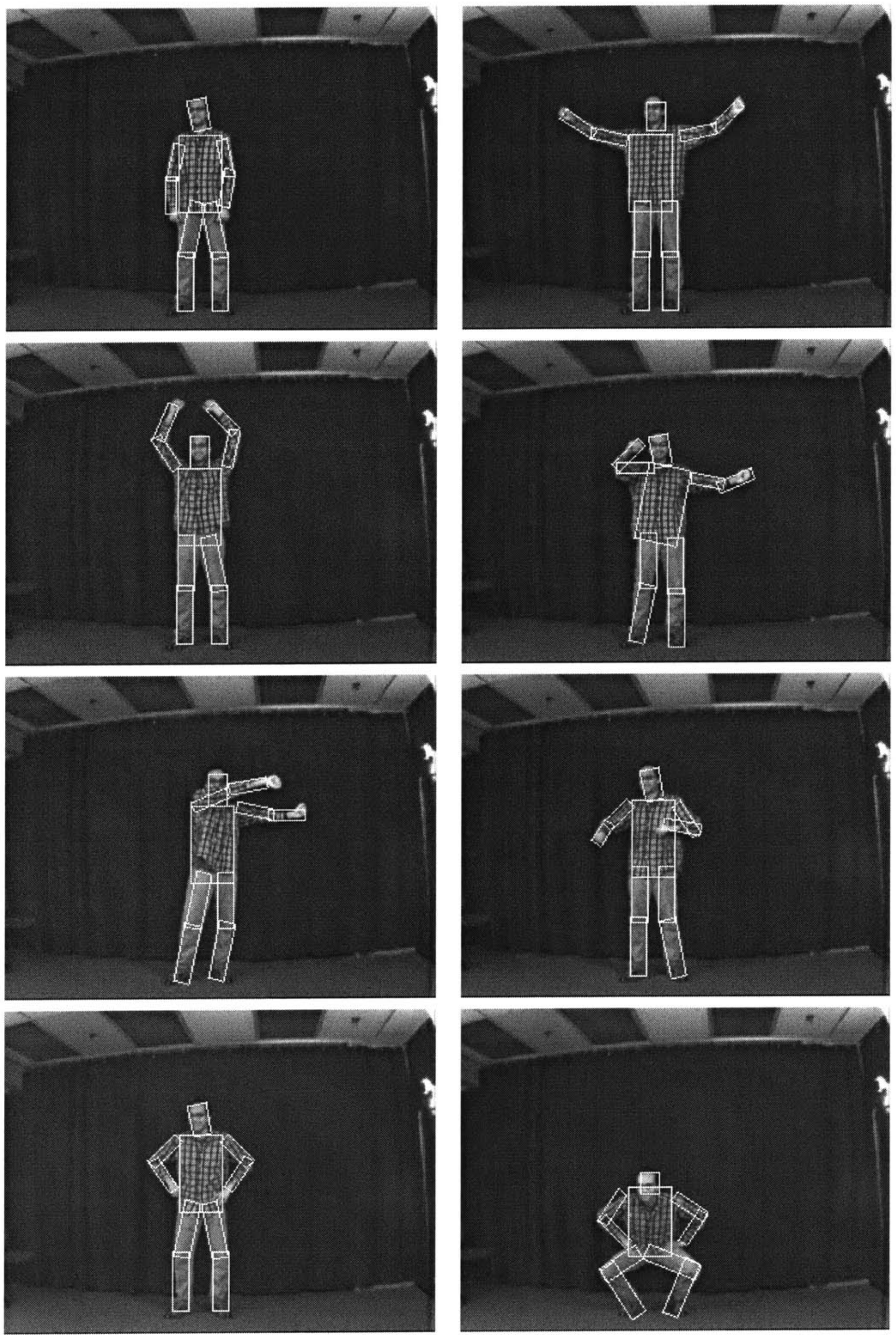

Figure 5-6: Matching results (sampling 100 times).

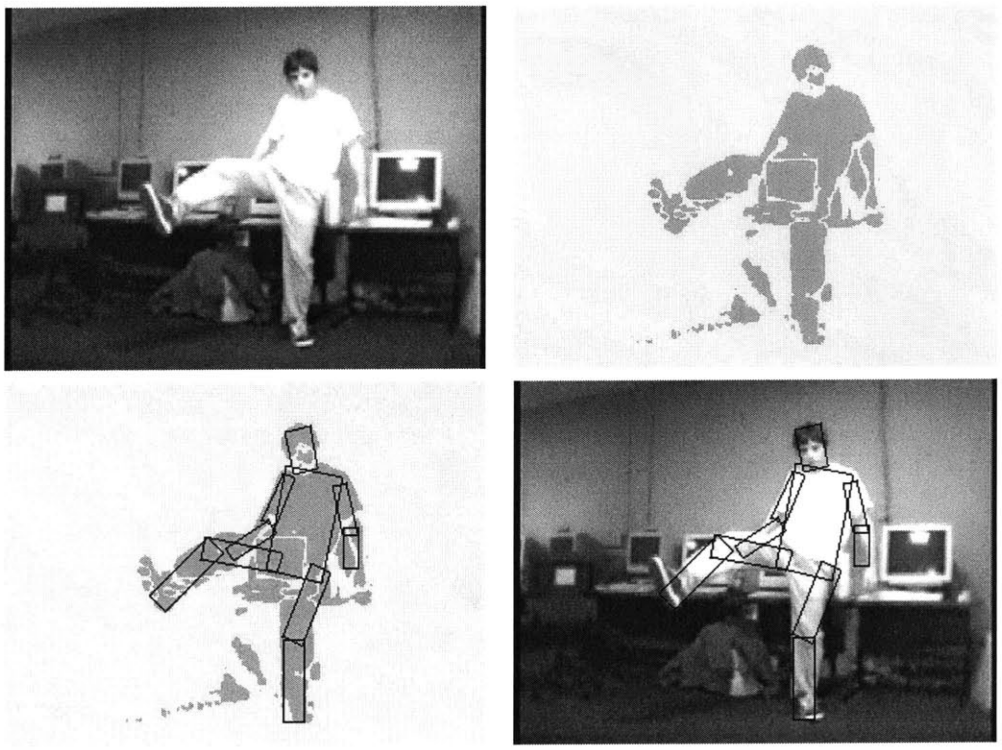

Figure 5-7: Even with noisy silhouettes we get good results.

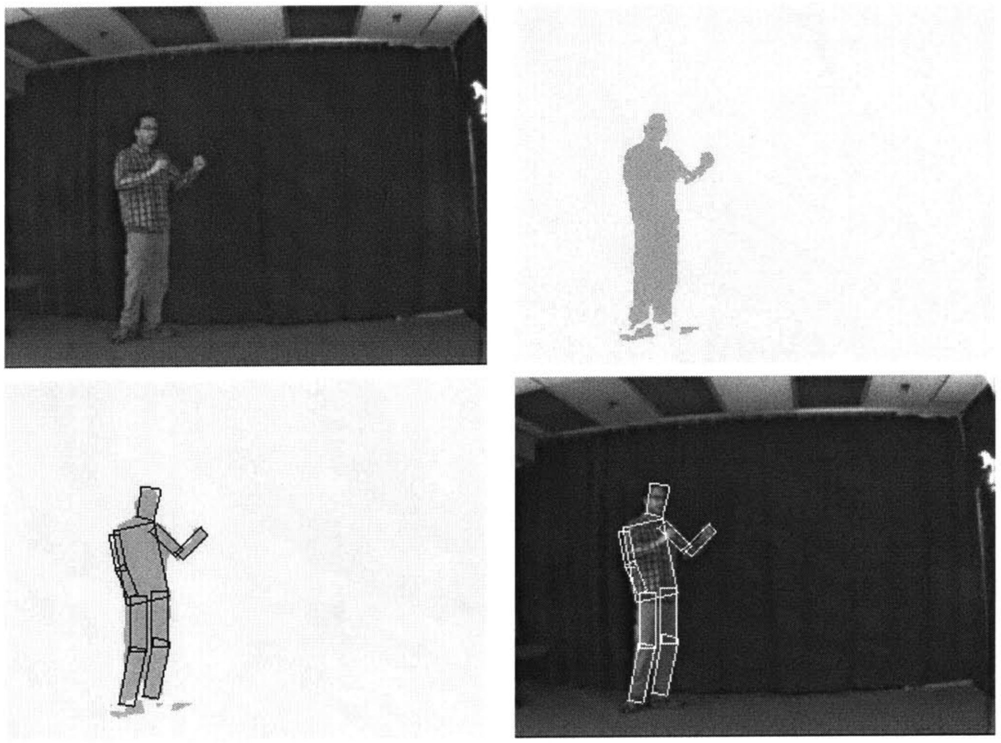

Figure 5-8: In this case, the silhouette doesn't provide enough information to estimate the position of one arm.

Chapter 6

Summary

This thesis described a statistical framework to represent the visual appearance of objects. With a statistical approach we can define the object detection and model learning problems in a principled way. Our major contribution is a rich class of models for which we can solve these problems efficiently. The models are based on the pictorial structure representation developed in [13], which allows for qualitative descriptions of appearance and is suitable for generic recognition problems.

One of the difficulties in representing generic objects is the large variation in shape and photometric information in each object class. Using a representation by parts, we can model the appearance variation in each part separately. We also explicitly model the geometric configuration of the parts, independent of their individual appearances. We demonstrated that our methods can be used to learn models for generic objects, such as faces and human bodies. Using these models we can detect the corresponding objects and estimate their pose.

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. We presented two concrete modeling schemes, but there are many other possibilities. By using a general framework we provided a set of computational mechanisms that can be shared among many different modeling schemes.

6.1 Extensions

1. We can deal with occluded parts by making $p(I|l_i, u_i)$ robust. 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.

2. We can detect multiple instances of an object using the MAP estimation algorithm. The algorithm can 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. So we could just pick all locations for the root that yield a high posterior. We could also look at the configuration we get for each possible location of the root and classify them using a separate method. This would select h configurations to be tested, out of the possible h^n. Another option is to sample multiple times from the posterior.

3. If we have an image sequence, we can detect an object in the first frame and use that location as prior information for the detection in the next frame. All our algorithms can be modified to take into account prior information over absolute locations. This would yield a tracking system.

Bibliography

[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] Y. Amit and A. Kong. Graphical templates for model registration. *PAMI*, 18(3):225–236, March 1996.

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

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

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

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

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

[17] W.E.L. Grimson. *Object Recognition by Computer, The Role of Geometric Constraints.* MIT Press, 1990. With contributions from Tomas Lozano-Perez and Daniel P. Huttenlocher.

[18] 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.

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

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

[21] 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.

[22] P. Lipson, E. Grimson, and P. Sinha. Configuration based scene classification and image indexing. In *CVPR97*, pages 1007–1013, 1997.

[23] D.G. Lowe. Fitting parameterized three-dimensional models to images. *PAMI*, 13(5):441–450, May 1991.

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

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

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

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

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

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

[30] S. Ullman and R. Basri. Recognition by linear combinations of models. *PAMI*, 13(10):992–1005, October 1991.

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

[32] 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.