

# Efficient Matching of Pictorial Structures \*

Pedro F. Felzenszwalb  
Artificial Intelligence Laboratory  
MIT  
Cambridge, MA 02139  
pff@ai.mit.edu

Daniel P. Huttenlocher  
Computer Science Department  
Cornell University  
Ithaca, NY 14853  
dph@cs.cornell.edu

## Abstract

*A pictorial structure is a collection of parts arranged in a deformable configuration. Each part is represented using a simple appearance model and the deformable configuration is represented by spring-like connections between pairs of parts. While pictorial structures were introduced a number of years ago, they have not been broadly applied to matching and recognition problems. This has been due in part to the computational difficulty of matching pictorial structures to images. In this paper we present an efficient algorithm for finding the best global match of a pictorial structure to an image. The running time of the algorithm is optimal and it takes only a few seconds to match a model with five to ten parts. With this improved algorithm, pictorial structures provide a practical and powerful framework for qualitative descriptions of objects and scenes, and are suitable for many generic image recognition problems. We illustrate the approach using simple models of a person and a car.*

## 1. Introduction

In this paper we consider the problem of matching pictorial structures to images, as introduced by Fischler and Elschlager [8] nearly 30 years ago. A pictorial structure is a collection of parts arranged in a deformable configuration. The deformable configuration is represented by spring-like connections between the parts. The matching problem is that of finding the best placement of the parts in an image, where the quality of a placement depends both on how well each part matches the image and on how well the placements agree with the deformable configuration.

The main contribution of our work is the development of an efficient matching algorithm for a natural class of pictorial structures. A secondary contribution

is providing a Bayesian interpretation of the problem, in terms of MAP estimation. The running time of our algorithm is optimal, in the sense that it runs as quickly as simply matching each part separately, without accounting for the relationships between parts. In practice the algorithm is also fast, finding the globally best match of a pictorial structure to an image in just a few seconds.

Pictorial structures provide a powerful framework for qualitative descriptions of objects and scenes, making them suitable for many generic image recognition problems. In [8] and in [7], pictorial structures were used to form generic models of a human face. Simple generic appearance models were used for parts such as the eyes, mouth, etc., and the connections between parts ensured that the geometric arrangement of the parts was face-like. In [16], pictorial structures were used to model generic scene concepts such as a waterfall, a snowy mountain, or a sunset. For example, a waterfall was modeled as a bright white region (water) in the middle of darker regions (rocks). The method that we present in this paper can be used to efficiently solve a broad range of generic recognition problems, including those investigated in [8, 7, 16], as well as the recognition of articulated objects.

Following [8], a deformable configuration is represented by pairwise relationships between the locations of the parts. Certain pairs of parts are connected by virtual “springs” that pull one part to be in a given relative location with respect to the other. These “springs” can enforce different kinds of relationships between the locations of the parts. For example, in the person model in Section 6.1, the body parts of a person are connected together by flexible revolute joints. This allows the limbs to move while remaining connected. In the car model in Section 6.2, the wheels of a car are connected to the body using a flexible prismatic joint. This allows the wheels to move along the bottom of the car but not in any other direction.

The matching problem as posed in [8] involves min-

---

\*Work supported in part by DARPA contract DAAL01-97-K-0104

imizing a certain energy function which takes into account both the “spring” forces between parts and the match quality for each part. Minimizing this energy is a hard problem in general. We develop a new solution to this minimization problem for a natural class of pictorial structures. This method requires that the set of relationships between parts form a tree structure, and that the relationships between each pair of parts be of a particular form. The restriction to a tree structure allows us to use standard dynamic programming techniques, and the restriction in the form of the pairwise relationship between parts allows us to use a novel generalization of distance transforms. Combining the two techniques we obtain an algorithm that runs as fast as simply matching each part separately, without considering the relationships between the parts.

Restricting the relationships between parts to a tree structure is natural because the connections between parts of many animate objects form a tree corresponding to the skeletal structure. Many other kinds of objects can be represented as a star-graph, where there is one central part that all the other parts are connected to. For instance, in modeling a car the wheels and windows can all be positioned relative to the body. The restriction that we impose on the form of the pairwise relationships between parts similarly allows a broad range of objects to be modeled, such as those with flexible revolute joints as we use in modeling a person and flexible prismatic joints as we use in modeling a car.

We present examples illustrating that the algorithm enables efficient search of an image for the *globally* best match of relatively generic objects, such as a “dark-haired fair-skinned person wearing a blue shirt and black pants” or a “red car”.

## 2. Related Work

As noted in the introduction, we use the same framework as Fischler and Elschlager [8], posing the matching problem as energy minimization. They also proposed a dynamic programming approach as we do here. There are two key differences in our work. First, we focus specifically on the case of tree-structured objects, because they are an interesting class that can be efficiently recognized. Second, we develop a method that is linear rather than quadratic in the number of possible placements for each part. This difference is important because the quadratic time method is simply not of practical use for most cases.

In the introduction we discussed the work of [16] and [7], both of whom use pictorial structures. In [16], only coarse pictorial structures are used and no algorithm is presented. The work in [7] uses a similar energy

function to the one presented here. In their work instead of having connections between pairs of parts, all parts are constrained with respect to a central coordinate system, this makes it impossible to represent objects with more than one articulation point. Moreover, they only give heuristic algorithms that don’t necessarily find the global optimal solution. Our approach can find the globally optimal match for this kind of model by locating each part relative to a virtual part that acts as the central coordinate system.

There are many approaches to recognizing models represented by parts and connections or constraints between the parts, such as [11],[2],[12],[6] and [15]. While these techniques address a similar problem to the matching of pictorial structures, one key difference is that they all make hard (yes or no) decisions as to whether a given location in the image can contain a given part. These methods then seek a set of such detected parts that are arranged in a valid configuration. Moreover, all such valid configurations are treated as being equally good. The use of binary decisions is not well suited to simple qualitative parts models such as the ones used in this paper and in other applications of matching pictorial structures [8, 16, 7]. With simple generic part models, it is better to determine how well each part matches at each location, rather than only finding acceptable locations for the parts. The work in [17] is similar to ours, in that it represents articulated objects using a tree of dependencies between parts. Again, however, all valid configurations of the parts are treated as equally good. Furthermore, the algorithm presented in [17] is for local minimization only, as opposed to a global match that searches over all possible placements of the model as is done here.

## 3. The Recognition Framework

We are concerned with representing pictorial structures using the scheme first proposed in [8]. A pictorial structure is represented as a collection of parts, with connections between certain pairs. A natural way to express such a model is in terms of a graph  $G = (V, E)$  where the vertices  $V = \{v_1, \dots, v_n\}$  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 a part in an image is specified by a location  $l$ . For the examples in this paper,  $l$  specifies position, rotation, and scale parameters for simple two dimensional parts. For each part  $v_i$ , a match cost function  $m_i(I, l)$  measures how well the part matches the image  $I$  when placed at location  $l$ . The examples in this paper use fairly simple template matching for this cost function. Other possibilities would be to use more

complex appearance models (e.g., [18]) or edge based techniques (e.g., [13]).

The connections between parts indicate relationships between their locations. For each connection  $(v_i, v_j)$  there is a deformation cost function  $d_{ij}(l_i, l_j)$  measuring how well the locations  $l_i$  of  $v_i$  and  $l_j$  of  $v_j$  agree with the object model. For instance, in the person model in Section 6.1 the connections enforce that the body parts of a person be arranged in a human-like configuration.

A configuration  $L = (l_1, \dots, l_n)$  specifies a location for each of the parts  $v_i \in V$  with respect to the image. The goal is to find the best such configuration, as measured by the match cost for the individual parts and by the pairwise cost for the connected pairs of parts. Following [8], we express this best match as,

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

The form of this minimization problem is quite general, and it appears in a number of problems in computer vision, such as MAP estimation of Markov Random Fields (MRFs) and optimization of dynamic contour models (snakes). While the form of the minimization problem 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.

In its most general form the minimization in (1) takes exponential time,  $O(m^n)$  where  $m$  is the number of discrete values for each  $l_i$  and  $n$  is the number of vertices in the graph. 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  $O(m^2n)$  time ([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,  $m$ , be small, as the dynamic programming solution is quadratic in  $m$ . 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 ([5, 14]). In our algorithm, we use constraints on both the structure of the graph and the form of  $d_{ij}$ .

For the matching problem that we consider here, the graph structure can in principle be arbitrary. However, as discussed in the introduction, many objects

can naturally be represented using tree structures. By restricting the graphs to trees, a similar kind of dynamic programming can be applied to trees as is done for chains, making the minimization problem polynomial rather than exponential time. The precise technique is described in Section 4. However, this  $O(m^2n)$  algorithm is not practical, because the number of possible locations for each part,  $m$ , is generally quite large. When searching for the best possible match of a pictorial structure to an image, there is no natural way to limit this location space. Even if the location parameters are coarsely sampled – say with 50 values each for  $x$ -translation,  $y$ -translation, rotation and scale –  $m$  is already over six million. Thus a straightforward application of dynamic programming on trees is not practical.

We investigate a restriction of the pairwise cost function,  $d_{ij}$ , that yields a minimization algorithm which runs in  $O(mn)$  rather than  $O(m^2n)$  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. Intuitively the pairwise cost function measures the degree to which the model is deformed, increasing as connected parts are moved away from their ideal relative locations. We restrict  $d_{ij}$  to the following form,

$$d_{ij}(l_i, l_j) = \|T_{ij}(l_i) - T_{ji}(l_j)\| \quad (2)$$

where  $\|\cdot\|$  is some norm (recall that a norm obeys the properties of identity, symmetry and triangle inequality), and  $T_{ij}, T_{ji}$  are invertible functions. We further require that it be possible to discretize  $T_{ji}(l_j)$  in a grid. Intuitively, the deformation cost is restricted to be a distance between reparameterized location vectors. The functions  $T_{ij}$  and  $T_{ji}$  together capture the ideal relative configuration of the parts  $v_i$  and  $v_j$ . That is, if  $l_i$  and  $l_j$  specify ideal locations of  $v_i$  and  $v_j$  with respect to one another, then  $T_{ij}(l_i) = T_{ji}(l_j)$ . The norm then measures the degree of deviation from this ideal relative configuration. In Section 6.1 we show how  $d_{ij}$  can behave like a flexible revolute joint between two parts and in Section 6.2 we show how it can behave like a flexible prismatic joint.

## 4. Efficient Minimization

In this section we show how to use dynamic programming to find the configuration  $L^* = (l_1^*, \dots, l_n^*)$ , minimizing equation (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  $(v_i, v_j)$  and thus the only contribution of  $l_j$  to the energy in (1) is  $d_{ij}(l_i, l_j) + m_j(I, l_j)$ . Hence the quality of the best location of  $v_j$  given location  $l_i$  of  $v_i$  is

$$B_j(l_i) = \min_{l_j} (d_{ij}(l_i, l_j) + m_j(I, l_j)) \quad (3)$$

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

$$B_j(l_i) = \min_{l_j} \left( d_{ij}(l_i, l_j) + m_j(I, l_j) + \sum_{v_c \in C_j} B_c(l_j) \right) \quad (4)$$

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  $\arg \min$ . Note that this equation subsumes (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

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

That is, the minimization in (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). 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 are of depth  $d$ . Thus  $B_j(l_i)$  can be computed as in (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. That is, 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  $O(nM)$ , where  $M$  is the time required to compute each  $B_j(l_i)$  and  $B'_j(l_i)$ . We now show how to compute  $B_j(l_i)$  and  $B'_j(l_i)$  in  $O(m)$ , yielding an  $O(mn)$  algorithm overall.

## 5. Generalized Distance Transforms

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

$$\mathcal{D}_B(z) = \min_{w \in B} \|z - w\|$$

In order to compute the distance transform, it is commonly expressed as

$$\mathcal{D}_B(z) = \min_{w \in \mathcal{G}} (\|z - w\| + 1_B(w)) \quad (5)$$

where  $1_B(w)$  is an indicator function for membership in the set  $B$ , that has the value 0 when  $w \in B$  and  $\infty$  otherwise. The algorithm presented in [3, 4] to compute (5) efficiently, still works if we replace the indicator function by an arbitrary function on the grid. 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

$$\mathcal{D}_f(z) = \min_{w \in \mathcal{G}} (\|z - w\| + f(w)) \quad (6)$$

Intuitively, for each grid location  $z$ , this function finds a location  $w$  that is close to  $z$  and for which  $f(w)$  is small. Note that differences between the value of  $\mathcal{D}_f$  at two locations are bounded by the distance between the locations, regardless of how quickly the function  $f$  changes. In particular, if there is a location where  $f(z)$

has a small value,  $\mathcal{D}_f$  will have small value at  $z$  and nearby locations.

Given the restricted form of  $d_{ij}$  in equation (2), equation (4) can be rewritten as a generalized distance transform,

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

where

$$f(w) = m_j(I, T_{ji}^{-1}(w)) + \sum B_c(T_{ji}^{-1}(w))$$

and the grid  $\mathcal{G}$  specifies a discrete set of  $T_{ji}(l_j)$  that are considered during the minimization (this in turn specifies a discrete set of locations  $l_j$ ).

The algorithm in [3, 4] for computing (5) runs in  $O(mD)$  time for  $m$  locations on a  $D$ -dimensional grid. As just noted, this same algorithm works for (6). In general the dimension  $D$  is a small fixed number. In this paper the grid is the four-dimensional space of,  $x$ -translation,  $y$ -translation, rotation  $\theta$ , and scale  $s$ . In order to compute (6) under the  $L_1$  norm in this four-dimensional parameter space, an array  $D[x, y, \theta, s]$  is initialized to the values of the function  $f(w)$ . The first pass over this array goes from the minimum  $[x, y, \theta, s]$  location (in order of increasing  $x$ , then  $y$ , then  $\theta$ , then  $s$ ) computing

$$\begin{aligned} D[x, y, \theta, s] = \min(&D[x, y, \theta, s], \\ &D[x - 1, y, \theta, s] + k_x, \\ &D[x, y - 1, \theta, s] + k_y, \\ &D[x, y, \theta - 1, s] + k_\theta, \\ &D[x, y, \theta, s - 1] + k_s) \end{aligned}$$

This computation is done “in place”, changing values in the array as it goes. Note that the value  $x - 1$  indicates the neighbor in the  $x$ -dimension of the array, and so forth. The constants  $k_x, k_y, k_\theta$  and  $k_s$  account for different scales of the axes. The second pass over the array goes from the maximum  $[x, y, \theta, s]$  location (in order of decreasing  $x$ , then  $y$ , then  $\theta$ , then  $s$ ) computing

$$\begin{aligned} D[x, y, \theta, s] = \min(&D[x, y, \theta, s], \\ &D[x + 1, y, \theta, s] + k_x, \\ &D[x, y + 1, \theta, s] + k_y, \\ &D[x, y, \theta + 1, s] + k_\theta, \\ &D[x, y, \theta, s + 1] + k_s) \end{aligned}$$

This algorithm does not consider the fact that  $\theta$  is periodic. Special handling of the boundary cases and two additional passes can be performed to handle this periodicity. Computing the generalized distance transform under any  $L_p$  norm can be approximated using similar procedures. The resulting approximation has a fixed percentage error (see [3, 4]). This solves the problem

of computing  $B_j(l_i)$  in  $O(m)$  time. The algorithm can be easily modified to keep track of the location that gets propagated to each position in the grid, which allow us to compute  $B'_j$  as we compute  $B_j$ .

## 6. Object Models

The experiments reported in this paper use a simple model of a person and of a car, shown in Figures 4 and 5 respectively. The parts of these models are just rectangles with fixed aspect ratio, an average color and a color variance. However, nothing about the matching algorithm requires such simple part models. The parts can be anything for which a match cost can be computed efficiently at each possible location in the image (e.g., more complicated appearance models, color or edge-based templates, etc).

The color models for the parts use the opponent color space defined in [20]. The average color of each part was estimated from one example and the covariance matrix was chosen by hand (currently we use a diagonal matrix instead of a full covariance matrix). The location of each part in the image is defined by a 4-tuple,  $(x, y, \theta, s)$  specifying the position of the center of mass, the orientation and the height of the rectangle. The match costs are computed using a convolution kernel composed of a “match” rectangle embedded in a larger “no match” rectangle, as illustrated in Figure 1. To compute a match cost for a part at any given location, we first generate a new image that measures how much each pixel in the input image matches the color of the part, using a truncated quadratic error function. That is, generate  $I'$  by:

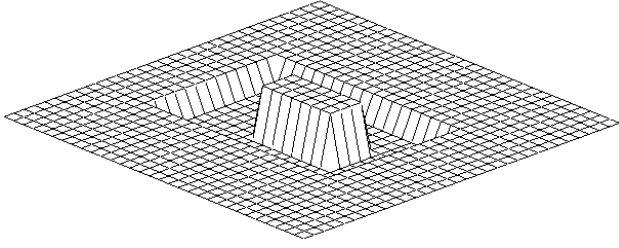
$$I'(x, y) = \min\left(\frac{1}{2}(I(x, y) - \mu)^T \Sigma^{-1}(I(x, y) - \mu), o\right)$$

where  $I(x, y)$  is the color of pixel  $(x, y)$  in the input image,  $\mu$  and  $\Sigma$  are the average color and color variance of a part, and  $o$  is an upper bound on the error. Truncating the error function allows for partial occlusion of parts. We then convolve  $I'$  with the “match”/“no match” kernel at each possible orientation and scale to generate a match cost for the part at every location.

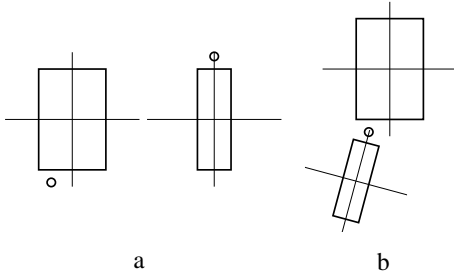
The connections between parts behave differently for the person and car models. In the person model, body parts are connected by flexible revolute joints while in the car model, the wheels are connected to the body by flexible prismatic joints. These connections are described in detail below.

### 6.1. Person Model

For the person model, the deformation costs model flexible revolute joints between two connected parts. A



**Figure 1: Convolution kernel used to compute match cost for a rectangular part.**



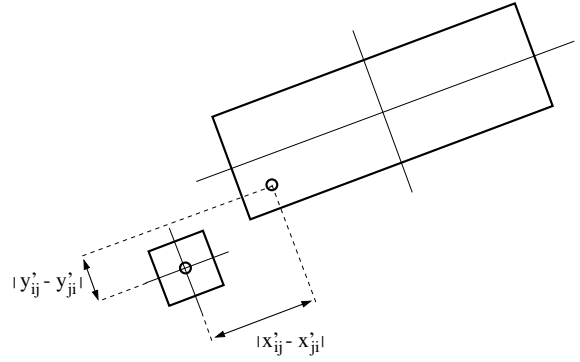
**Figure 2: Two parts of the person model, (a) in their own coordinate system and (b) the ideal configuration of the pair.**

pair of connected parts is illustrated in Figure 2. 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 2a. In an ideal configuration these points coincide, as illustrated in Figure 2b. The ideal relative orientation is given by  $\theta_{ij}$ , the angle between the main axes (in the “height” direction). The ideal relative size is given by  $s_{ij}$ , the ratio of the two heights.

Given the observed locations  $l_i = (\theta_i, s_i, x_i, y_i)$  and  $l_j = (\theta_j, s_j, x_j, y_j)$  of two parts, the deformation cost measures the deviation between the ideal values and these observed values. Each joint specifies weights  $w_{ij}^\theta, w_{ij}^s, w_{ij}^x, w_{ij}^y$  for the cost associated with deviations in each of the relative orientation, size and joint alignment. Thus we define the pairwise deformation cost for the person model to be,

$$\begin{aligned} d_{ij}(l_i, l_j) &= w_{ij}^\theta |\theta_j - \theta_i| - \theta_{ij}| \\ &+ w_{ij}^s |(\log s_j - \log s_i) - \log s_{ij}| \\ &+ w_{ij}^x |x'_{ij} - x'_{ji}| \\ &+ w_{ij}^y |y'_{ij} - y'_{ji}| \end{aligned}$$

where the first term is the difference between the ideal relative angle and the observed relative angle, the second term is the difference between the ideal relative size



**Figure 3: Two parts of the car model. The distance between the joints is measured along the horizontal and vertical directions with respect to their orientation.**

and the observed relative size (using log for size ratios), and the third and fourth terms are the horizontal and vertical distances between the observed joint positions in the image  $(x'_{ij}, y'_{ij})$  for  $v_i$  and  $(x'_{ji}, y'_{ji})$  for  $v_j$ .

In our experiments we let  $w_{ij}^s, w_{ij}^x, w_{ij}^y$  be large values and  $w_{ij}^\theta$  be a small value. Thus the deformation cost stays relatively small as the parts rotate about the joint and is large if the parts are not aligned at the joint or have different relative sizes.

The deformation cost must be expressed in the form of equation (2), as a distance between  $T_{ij}(l_i)$  and  $T_{ji}(l_j)$ . Given the location  $l_i = (\theta_i, s_i, x_i, y_i)$  we define  $T_{ij}(l_i) = (\theta'_i, s'_i, x'_i, y'_i)$  where

$$\begin{aligned} \theta'_i &= w_{ij}^\theta (\theta_i - \theta_{ij}/2) \\ s'_i &= w_{ij}^s (\log s_i - \log s_{ij}/2) \\ (x'_i, y'_i)^T &= W_{ij}((x_i, y_i)^T + s_i R_{\theta_i}(x_i, y_i)^T) \end{aligned}$$

$W_{ij}$  is a diagonal weight matrix with entries  $w_{ij}^x$  and  $w_{ij}^y$ , and  $R_{\theta_i}$  is a matrix that performs a rotation of  $\theta_i$  radians about the origin. The new position coordinates  $x'_i, y'_i$  indicate the position of the joint in the image (scaled according to the weights). Now we note that the  $L_1$  distance between  $T_{ij}(l_i)$  and  $T_{ji}(l_j)$  is equal to the deformation cost  $d_{ij}(l_i, l_j)$  just given above, yielding an expression as in (2).

## 6.2. Car Model

For the car model, the deformation costs model flexible prismatic joints between the car wheels and the car body. The wheels have an ideal position with respect to the car body and can move along the bottom of the car with a cost proportional to the distance from their ideal position.

Like in the person model, the location of a joint is specified by two points  $(x_{ij}, y_{ij})$  and  $(x_{ji}, y_{ji})$ , one in

the coordinate frame of each part. In an ideal configuration these points coincide. The ideal relative size is given by  $s_{ij}$ , the ratio of the two heights. In the joints used in the car model, the parts always have the same orientation. Each joint specifies weights  $w_{ij}^s$ ,  $w_{ij}^x$ ,  $w_{ij}^y$  for the cost associated with deviations in the relative size and joint alignment. We define the pairwise deformation cost for the car model to be,

$$\begin{aligned} d_{ij}(l_i, l_j) &= \infty |\theta_j - \theta_i| \\ &+ w_{ij}^s |\log s_j - \log s_i| \\ &+ w_{ij}^x |x'_{ij} - x'_{ji}| \\ &+ w_{ij}^y |y'_{ij} - y'_{ji}| \end{aligned}$$

where the first term insures that the orientation of  $v_i$  and  $v_j$  is the same, the second term is the difference between the ideal relative size and the observed relative size (using log for size ratios), and the third and fourth terms are the horizontal and vertical distances between the joint positions in the image, measured with respect to the orientation of the parts (see Figure 3).

In our experiments we let  $w_{ij}^y = \infty$  so the wheels of the car can only slide horizontally with respect to the orientation of the body. We let  $w_{ij}^s$  be a large value, making a strong connection between sizes of the parts, and  $w_{ij}^x$  be a smaller value, allowing the wheels to slide along the bottom of the car body without increasing the deformation cost too much.

Given the location  $l_i = (\theta_i, s_i, x_i, y_i)$  we define  $T_{ij}(l_i) = (\theta'_i, s'_i, x'_i, y'_i)$  where

$$\begin{aligned} \theta'_i &= \infty \theta_i \\ s'_i &= w_{ij}^s (\log s_i - \log s_{ij}/2) \\ (x'_i, y'_i)^T &= W_{ij} (R_{-\theta_i}(x_i, y_i)^T + s_i(x_{ij}, y_{ij})^T) \end{aligned}$$

$W_{ij}$  is a diagonal weight matrix with entries  $w_{ij}^x$  and  $w_{ij}^y$ , and  $R_{-\theta_i}$  is a matrix that performs a rotation of  $-\theta_i$  radians about the origin. The new position coordinates  $x'_i, y'_i$  indicate the position of the joint in the image with respect to a coordinate frame that is rotated to be in the same orientation as the part. Now we note that the  $L_1$  distance between  $T_{ij}(l_i)$  and  $T_{ji}(l_j)$  is equal to the deformation cost  $d_{ij}(l_i, l_j)$  just given above, yielding an expression as in (2).

## 7. Implementation and Experiments

We discretized the space of possible part locations into 50 buckets for each of the  $x$  and  $y$  positions, 10 buckets for size and 20 buckets for orientation. Using these parameters our system finds the best configuration of a model with five to ten parts in a  $320 \times 240$  pixel image in about 10 seconds on a 450 MHz Pentium-III.

Figure 4 shows recognition results using the person model. Note that we can recognize the person under a broad range of lighting condition and varied poses, with a simple generic model. This illustrates the ability to recognize articulated objects. Figure 5 shows recognition results using the car model. Note that we can recognize fairly different kinds of cars under a wide variety of poses with a single model. This illustrates the ability of our method to recognize generic objects such as a “red car”.

## 8. Bayesian Formulation

In this section we note that the best match as expressed by equation (1) can be reformulated as the maximum a posteriori (MAP) estimate of the model configuration.

The MAP estimate is the configuration with maximum probability given the input image:

$$L^* = \arg \max_L \Pr(L|I)$$

Bayes rule then implies

$$L^* = \arg \max_L \Pr(I|L)\Pr(L) \quad (7)$$

The prior,  $\Pr(L)$ , captures information about the object that is known before observing the image. This prior information is given by the “spring” connections between parts. The larger the deformation costs in a configuration, the less probable this configuration should be. We can capture this notion using:

$$\Pr(L) = \frac{1}{K} e^{-\sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j)} \quad (8)$$

where  $d_{ij}$  is the deformation cost associated with edge  $(v_i, v_j)$ , and  $K$  is a normalization constant. Note that this is the typical Gibbs distribution used in Markov Random Field estimation problems.

The likelihood function,  $\Pr(I|L)$ , measures the probability of observing image  $I$  given a particular configuration of the model. Intuitively, the likelihood should be high when the appearance of the parts agree with the image data at the positions they are placed, and low otherwise. Even though the parts of a model may overlap in the image, we approximate the likelihood by assuming it is proportional to the product of match qualities for individual parts,

$$\Pr(I|L) \propto \prod_{i=1}^n g_i(I, l_i) \quad (9)$$

The function  $g_i(I, l_i)$  measures the match quality for part  $v_i$  at location  $l_i$  in image  $I$ . This function depends

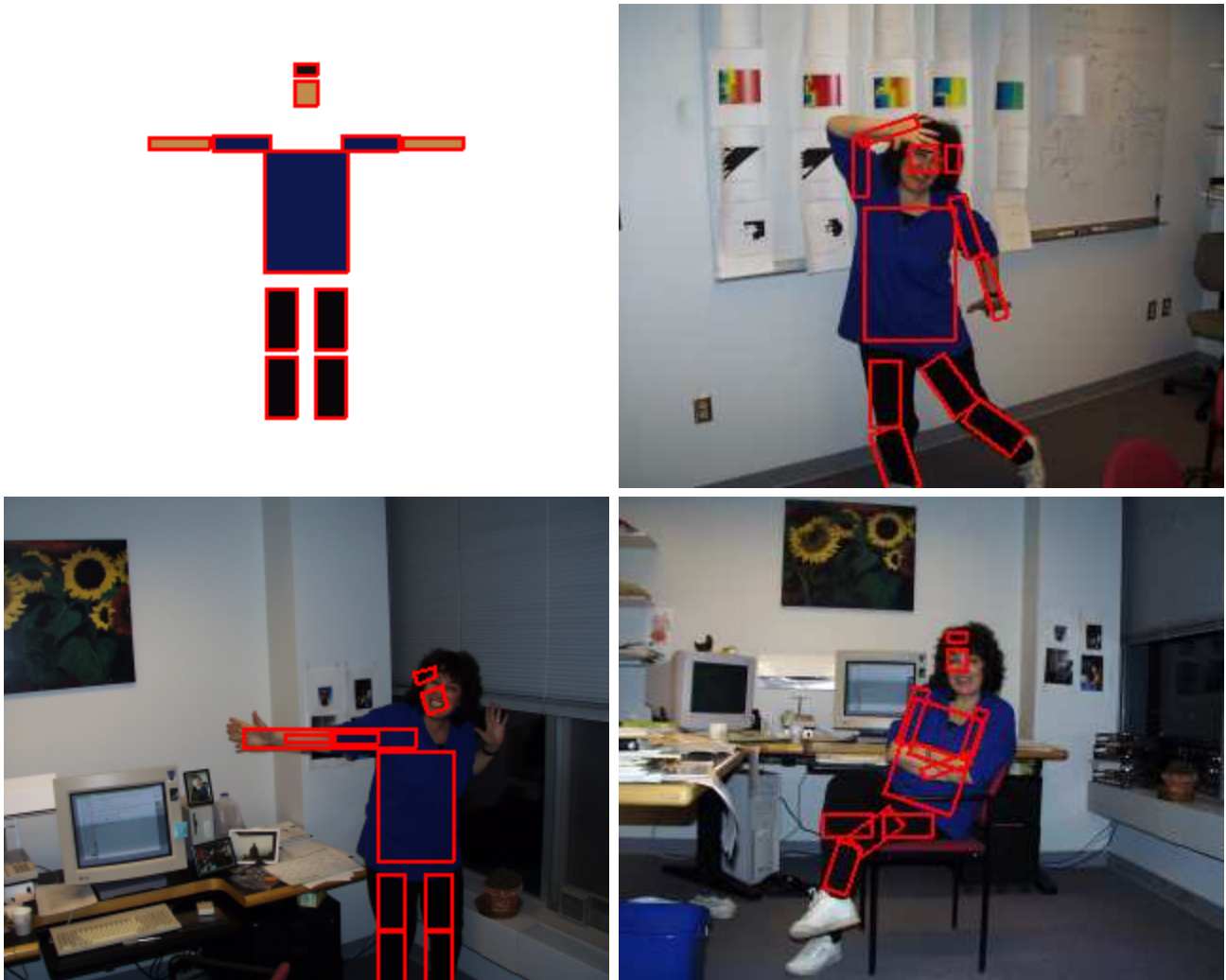


Figure 4: Recognition results using the person model. The top left image shows the model, which has revolute joints, in its default configuration. The remaining three images show examples of the globally best matching configuration.

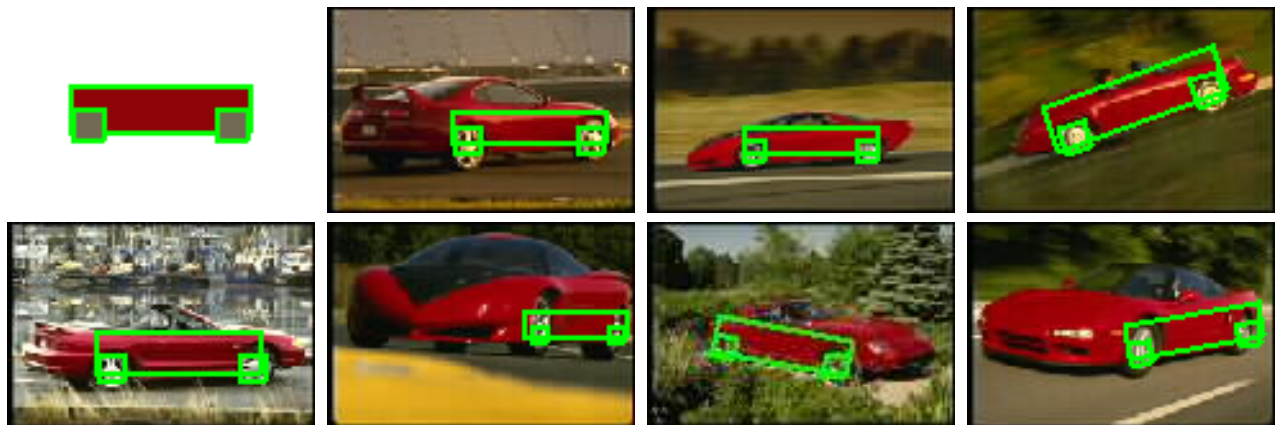


Figure 5: Recognition results using the car model. The top left image shows the model, which has prismatic joints, in its default configuration. The remaining seven images show the globally best matching configuration.

on the particular modeling scheme used for individual parts.

The standard approach to finding the MAP estimate in (7) is to minimize the energy function obtained by taking the negative logarithm. Substituting (8) and (9) into (7) we obtain,

$$L^* = \arg \min_L \left( \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j) - \sum_{v_i \in V} \ln g_i(I, l_i) \right)$$

which is exactly the same as (1) if we let the match costs  $m_i$  be the negative logarithm of the match qualities  $g_i$ .

## 9. Future Work

In this paper the quality of an object configuration is based purely on the relative positions of individual parts. It is simple to incorporate prior information about the absolute location of each part in the framework. This allows an integrated approach to model-based tracking and recognition. The recognition on the first image of a sequence can be done using uniform prior on absolute locations. On subsequent images information about the previous match can be used to impose a non-uniform prior on absolute locations.

We also have new results that allow the deformation cost  $d_{ij}$  to be the squared distance between transformed locations. This better models the idea that flexible joints can be “pulled” by small amounts without great cost, while large deformations are very expensive. The algorithm by Karzanov described in [19] can be changed to efficiently compute a modified generalized distance transform, with the norm replaced by its square.

## References

- [1] A. Amini, T. Weymouth, R. Jain. Using Dynamic Programming for Solving Variational Problems in Vision. *PAMI*, Vol. 12, No. 9, September 1990, Pages 855-867.
- [2] A. Beinglass, H. Wolfson. Articulated Object Recognition, or: How to Generalize the Generalized Hough Transform. *CVPR*, 1991, Pages 461-466.
- [3] G. Borgefors. Distance Transformations in Arbitrary Dimensions. *CVGIP*, Vol. 27, No. 3, September 1984, Pages 321-345.
- [4] G. Borgefors. Distance Transformations in Digital Images. *CVGIP*, Vol. 34, No. 3, June 1986, Pages 344-371.
- [5] Y. Boykov, O. Veksler, R. Zabih. Markov Random Fields with Efficient Approximations. *CVPR*, 1998, Pages 648-655.
- [6] M.C. Burl, T.K. Leung, P. Perona. Recognition of Planar Object Classes. *CVPR*, 1996, Pages 223-230.
- [7] M.C. Burl, M. Weber, P. Perona. A Probabilistic Approach to Object Recognition Using Local Photometry and Global Geometry. *ECCV*, Vol. 2, 1998, Pages 628-641.
- [8] M.A. Fischler and R.A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Trans. on Computers*, Vol. 22, No. 1, January 1973, Pages 67-92.
- [9] D.A. Forsyth and M.M. Fleck. Body Plans. *CVPR*, 1997, Pages 678-683.
- [10] S. Geman, D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *PAMI*, Vol. 6, No. 6, November 1984, Pages 721-741.
- [11] W.E.L. Grimson. On the Recognition of Parameterized 2D Objects. *IJCV*, Vol. 2, No. 4, April 1989, Pages 353-372.
- [12] Y. Hel-Or, M. Werman. Constraint-Fusion for Localization and Interpretation of Constrained Objects. *CVPR*, 1994, Pages 39-45.
- [13] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge. Comparing Images Using the Hausdorff Distance. *PAMI*, Vol. 15, No. 9, September 1993, Pages 850-863.
- [14] H. Ishikawa and D. Geiger. Segmentation by Grouping Junctions. *CVPR*, 1998, Pages 125-131.
- [15] J. Krumm. Object Detection with Vector Quantized Binary Features. *CVPR*, 1997, Pages 179-185.
- [16] P. Lipson, E. Grimson, P. Sinha. Configuration Based Scene Classification and Image Indexing. *CVPR*, 1997, Pages 1007-1013.
- [17] D.G. Lowe. Fitting Parameterized Three-Dimensional Models to Images. *PAMI*, Vol. 13, No. 5, May 1991, Pages 441-450.
- [18] H. Murase and S.K. Nayar. Visual Learning And Recognition Of 3-D Objects From Appearance. *IJCV*, Vol. 14, No. 1, January 1995, Pages 5-24.
- [19] W. Rucklidge. Efficient Visual Recognition Using the Hausdorff Distance. *LNCS 1173*, Springer-Verlag, 1996.
- [20] M.J. Swain and D.H. Ballard. Color Indexing. *IJCV*, Vol. 7, No. 1, November 1991, Pages 11-32.