International Journal of Computer Vision 43(1), 45–68, 2001

Probabilistic Methods for Finding People

S. IOFFE AND D.A. FORSYTH
Computer Science Division, University of California at Berkeley, Berkeley, CA 94720
ioffe@cs.berkeley.edu
daf@cs.berkeley.edu

Received July 2, 2000; Revised March 12, 2001; Accepted March 12, 2001

Abstract. Finding people in pictures presents a particularly difficult object recognition problem. We show how to find people by finding candidate body segments, and then constructing assemblies of segments that are consistent with the constraints on the appearance of a person that result from kinematic properties. Since a reasonable model of a person requires at least nine segments, it is not possible to inspect every group, due to the huge combinatorial complexity.

We propose two approaches to this problem. In one, the search can be pruned by using projected versions of a classifier that accepts groups corresponding to people. We describe an efficient projection algorithm for one popular classifier, and demonstrate that our approach can be used to determine whether images of real scenes contain people.

The second approach employs a probabilistic framework, so that we can draw samples of assemblies, with probabilities proportional to their likelihood, which allows to draw human-like assemblies more often than the non-person ones. The main performance problem is in segmentation of images, but the overall results of both approaches on real images of people are encouraging.

Keywords: object recognition, human detection, probabilistic inference, grouping correspondence search

1. Introduction

Finding people in images is a difficult task, due to the high variability in the appearance of people. This variability may be due to the configuration of a person (e.g., standing vs. sitting vs. jogging), the pose (e.g. frontal vs. lateral view), clothing, and variations in illumination. There are two usual strategies for object recognition:

- Search over model parameters (kinematic variables, camera parameters, etc.) using a comparison between a predicted view of the object and the image. This problem is often stated as optimization of an objective function, which measures the similarity between the predicted and the actual views. This is usually called the *top-down* approach.
- Assemble image features into increasingly large groups, using the current group as a rough hypothesis about the object identity to select the next grouping activity. This is usually called the *bottom-up* approach.

1.1. Why Proceed Bottom-Up?

Current activities in vision emphasize top-down recognition and tracking. There are three standard approaches to finding people described in the literature. Firstly, the problem can be attacked by template matching (e.g. (Oren et al., 1997), where upright pedestrians with arms hanging at their side are detected by a template matcher; (Niyogi and Adelson, 1995; Liu and Picard, 1996; Cutler and Davis, 2000), where walking is detected by the simple periodic structure that it generates in a motion sequence; (Haritaoglu et al., 2000; Wren et al., 1997), which rely on background subtraction—that is, a template that describes

"non-people"). Matching templates to people (rather than to the background) is inappropriate if people are going to appear in multiple configurations, because the number of templates required is too high.

This motivates the second approach, which is to find people by finding faces (e.g. (Poggio and Sung, 1995; Rowley et al., 1996a, 1996b; Rowley et al., 1998a, 1998b; Sung and Poggio, 1998)). The approach is most successful when frontal faces are visible.

The third approach is to use the classical technique of search over correspondence (this is an important early formulation of object recognition; the techniques we describe have roots in (Faugeras and Hebert, 1986; Grimson and Lozano-Pérez, 1987; Thompson and Mundy, 1987; Huttenlocher and Ullman, 1987)). In this approach, we search over correspondence between image configurations and object features. There are a variety of examples in the literature (for a variety of types of object; see, for example, (Huang et al., 1997; Ullman, 1996)). Perona and collaborators find faces by searching for correspondences between eyes, nose and mouth and image data, using a search controlled by probabilistic considerations (Leung et al., 1995; Burl et al., 1995). Unclad people are found by (Forsyth et al., 1996; Forsyth and Fleck, 1999), using a correspondence search between image segments and body segments, tested against human kinematic constraints.

It is difficult to evaluate the correspondences between image regions and model parts and reliably choose the best ones. A crucial difficulty in both finding and tracking people is that extended, straight, coherent image regions—which could be body segments—can be relatively common. This means in turn that any objective function is going to have a local extremum where a hypothesized body segment lies over that region. The result is a tendency for trackers to drift or finding methods to become confused. The problem is simplified for the trackers since the configuration in a frame can be used to start the search for the next frame, but local extrema still present a significant problem and cause body parts to be lost due to occlusions, to other objects nearby that look like body parts, or to especially rapid motions. In addition, most trackers need to be started by hand, by specifying the configuration in the initial frame.

It is natural to try and simplify matters with a continuation method: take a series of simplified versions of the evaluation function, search the simplest, and use the result as a start point for a search on a less simple version, ending at an extremum of the original evaluation function. The annealed particle filter of (Deutscher et al., 2000) uses this strategy, but apparently cannot deal with much clutter because it creates too many difficult peaks. Furthermore, using this strategy to find and track people requires a detailed search of a high dimensional domain (the number of people being tracked times the number of parameters in the person model plus camera parameters). This implies that a method is needed that is able to explore large search spaces and thus provide an efficient alternative to blank search.

Bottom-up methods offer the promise of significantly reduced search, but have become unpopular because it appears to be very difficult to realize this promise. A typical bottom-up method would (1) detect a variety of features and then (2) group these features incrementally into assemblies, using a grouping procedure that takes into account the features in a group before adding features. An example of this process—which dates back at least to (Binford, 1971)—would involve finding objects by: (1) finding edges; (2a) pairing edge fragments that appear to lie locally on a generalized cylinder; (2b) collecting pairs that together appear to lie on a generalized cylinder; (2c) collecting pairs of straight homogeneous generalized cylinders with roughly constant cross-section which lie nearby and (2d) asserting that all such pairs could be arms. There are many instances of this line of reasoning *which does not require the use of any particular primitive* (finding curved objects in range data (Agin, 1972; Nevatia and Binford, 1977); finding people in images (Forsyth et al., 1996); finding lamps and mugs in images (Dickinson et al., 1992; Ulupinar and Nevatia, 1988; Zerroug and Nevatia, 1999) amongst others). Note that the grouping process is maintaining an increasingly more precise hypothesis of the object's identity, which is used to direct grouping activities. The success of the method depends on being able to supply a series of grouping activities that: (1) can cope with bad features; (2) have little ambiguity at each stage about what should be done—because this results in search; and (3) can robustly recognize many different types of object.

Many researchers have modeled a person as a kinematic chain, and recognized people as collections of generalized cylinders, subject to constraints given by the kinematics of human joints. This approach has been successful in tracking (e.g., (Gavrila and Davis, 1996; O'Rourke and Badler, 1980)). Generally, the tracker is initialized by marking the subject's configuration in the first frame, and the configuration is updated from frame to frame. In (Hogg, 1983) and (Rohr, 1993), the

frame-to-frame configuration update is accomplished by a search of the parameter space to minimize a model-to-image matching cost. Often, the update involves non-linear optimization (e.g., using gradient descent) to find the new configuration, where the old one is used to start the search (Bregler and malik, 1998; Rehg and Kanade, 1994).

The dichotomy between the top-down and bottom-up approaches is not precise. For example, in the people finding method of (Felzenszwalb and Huttenlocher, 2000), the *global extremum* of the objective function is found by first computing a response to the body-part filter at each orientation, position and scale (using convolution with a bank of filters), and then extracting the optimal group of body parts using dynamic programming. To be able to find the global extremum, however, they restrict the class of object models: the appearance model for the body parts is constrained to have a particular size and color, so that convolution could be used for body-part detection; the kinematic model must have the form of a tree so that the optimal configuration can be found efficiently. Furthermore, their framework lacks a discriminative component, and the system cannot determine whether a given image contains a person, but can only guess the person's configuration if one is known to be present.

In our work, we avoid such constraints and still are able to efficiently find people by pruning the search—that is, ignoring entire regions of the search space which have been determined not to contain the solution.

1.2. Outline

We assume that an image of a human can be decomposed into a set of distinctive segments, so that there is a segment of each type in each image (so that in each image a correspondence from model segments to image segments can be established). While this representation is restrictive, since body parts may often be absent due to either occlusion or their unusual appearance, we show that it can be used to detect and count people.

A simple segment detector is used to find image regions that could correspond to body parts. However, the appearance of limbs is not nearly as distinctive as that of the whole body (especially in the absence of cues such as the skin color), and thus many spurious body parts are found along with the actual ones (Fig. 1(a)). In addition, the correspondence between image regions and the body parts is hard to establish since many body parts are indistinguishable from one another (e.g., the two upper arms). However, the human body is subject to rather strong kinematic constraints. For example, if two segments have been matched to a left upper arm and a left lower arm, then the lengths of these segments and their relative positions are constrained so as to correspond to a possible elbow configuration.

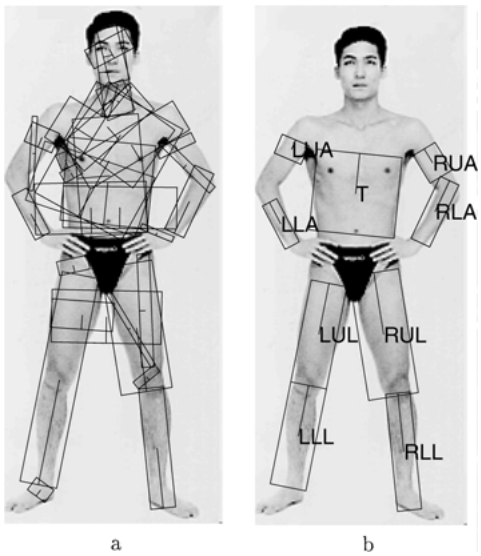

Figure 1. (a) All segments extracted for an image. (b) A labeled segment configuration corresponding to a person, where `T` = torso, `LUA` = left upper arm, etc. The head is not marked because we are not looking for it with our method.

The goal of this paper is to demonstrate that we can efficiently group the candidate segments found in an image (*image segments*) into *assemblies*—human-looking groups of image segments, with each element marked with a label specifying to which body part that segment corresponds (see Fig. 1(b)). Alternatively, we can think of a matching of the model to the image, whereby each model segment is coupled with an image segment.

The main issue with grouping is that the brute force approach of testing each combination of segments doesn't work because of the huge number of such assemblies (e.g., for 100 segments in the image, we have about 10^{18} assemblies with 9 segments (torso plus two segments per limb)). However, we can make the search much more manageable if we use the fact that, for most segment groups, it is impossible to add other segments found in the image in such a way that the resulting assembly looks like a person. For example, if it has been detected that two segments cannot represent the upper

and lower left arm, as in Fig. 6(a), then no assembly containing them will correspond to a person, and would not need to be considered.

We will show:

- How to represent body parts (Section 2) and learn models of relationships between them (Sections 3.3.2 and 4.3.2).
- How to prune the search. In Sections 3.3.3 and 4.3.4, we will demonstrate how we can, for a partial model-to-image matching (an assembly with some segments missing), determine that, no matter what segments are added to an assembly, it could not look like a person. In that case the current branch of the interpretation tree could be ignored.
- How to structure the search to make it efficient. In Sections 3.3.4 and 4.3.3, we give strategies for incrementally adding image segments to an assembly so that the pruning mechanism could be effectively exploited.
- That these methods can find and count people in images (Section 5).

1.3. Representation

As in much of the previous work mentioned above, we model the person as a collection of cylinders, and an image of a person as a collection of bar-shaped *segments*. We ignore a person's head. Bars can be detected using image edges, segmentation techniques such as normalized cuts (Shi and Malik, 1997), or motion cues if the image is a part of a video sequence. We picked the simplest type of segment detector, which looks for pairs of parallel edges. Such a detector is described in Section 2 and will, much of the time, detect all the body parts of a person, but will also produce many spurious segments for non-person image regions. This is acceptable, since kinematic constraints will help discriminate human segments from spurious ones.

By adopting the simplest segment-detection mechanism, we are able to concentrate on the search process. Even though our segment detector can deal with only one type of body part (bars, and not, for example, the head) and produces many spurious segments, kinematic constraints are a powerful cue as to which segments are the actual body parts, and we show that our search and pruning strategies use these constraints effectively. However, many of the failures of our method are due to missing or inaccurately detected segments, which suggests that our inference method is effective, but the overall system would benefit from an improved segment-finder. We discuss such improvements in Section 6.

The disadvantage of using our segment detector is that the range of images we can use is limited: our subjects may not wear baggy clothes. In this paper, we restrict ourselves to images of models wearing swimsuits or no clothes. However, the grouping process is independent of how the body segments are represented. Therefore, the restrictions imposed by the way we model the segments can be overcome. Section 6 discusses detecting more than one type of segments (e.g., limbs and the face), and handling clothes.

In this paper, we show two ways of efficiently assembling potential body parts into human assemblies. These two methods use different models of a person, which leads to different search and pruning strategies.

Classification: In Section 3 we show how to efficiently learn a *top-level classifier* that discriminates human assemblies of segments from non-human ones, and how to efficiently extract all assemblies in the image that look like people by adding segments to assemblies *incrementally*. The efficiency is achieved by pruning small assemblies, using *projected classifiers*. We describe how to derive projected classifiers from the top-level one. The pruning of subtrees of the interpretation tree is model-driven, which means that we backtrack the search if an assembly has been found such that, no matter what segments are added to it, the result will not look like a person.

Inference: In Section 4, we demonstrate a probabilistic method that uses a *soft classifier*, which associates a measure of likelihood of a person with each assembly, rather than a binary "person/non-person" decision. We describe a method for detecting people by sampling from the likelihood, and this method is made efficient by sampling sub-assemblies of increasing size, using *importance sampling*. The pruning is accomplished by first computing upper bounds on the likelihoods of sub-assemblies (using dynamic programming), and using these bounds to define the intermediate distributions, from which the sub-assemblies are sampled. Such pruning is data-driven: an assembly is no longer considered if the computed bounds indicate that no *image segments* can be added to make a human assembly.

2. Finding Segments

It is usual to model the appearance of body parts in the image as bars (i.e., projections of straight cylinders),

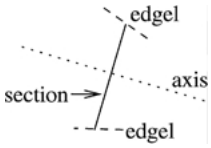

Figure 2. A symmetry: the two edgels (dashed lines) are symmetrical about the symmetry axis (dotted). We represent symmetries by their sections (solid line), which are line segments that connect the midpoints of the two edgels.

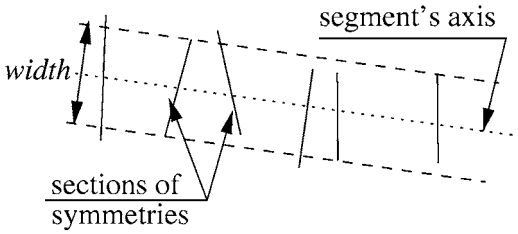

Figure 3. The segment finder groups symmetries into segments. The EM algorithm finds the optimal positions of the segments' axes and the widths, and also estimates the posterior probabilities that a given symmetry is assigned to a particular segment or to noise.

e.g. (Bregler and Malik, 1998; Gavrila and Davis, 1996; Hogg, 1983; O'Rourke Badler, 1980; Rohr, 1993). This seems to be a reasonable and natural representation, which, as we show in this paper, delivers promising results. We use pictures of people wearing swimsuits or no clothes, which allows us to detect bars by grouping parallel edges. We discuss the problem of body-part detection for clothed people in Section 6.

To find rectangular approximations to candidate body segments, we extract edges and identify sets of edge elements that form, approximately, pairs of parallel line segments (which become the length-wise sides of the rectangular segments). This grouping process is hierarchical, whereby we first identify "symmetries"—pairs of symmetrical edgels—and then group them (Brady and Asada, 1984; Brooks, 1981).

Two edges constitute a symmetry if, within some margin of error, they are reflections of each other about some symmetry axis. The verification is done by considering the *section*—the line connecting the midpoints of the two edgels—and finding the axis perpendicular to the section and passing through its midpoint. We then declare the edgel pair a symmetry if the angles the edgels form with the axis are sufficiently similar and small (Fig. 2).

Each edgel can be a part of zero, one or more symmetries. We will now group symmetries into segments. A set of symmetries constitutes a segment if the lengths of their sections are roughly the same (this corresponds to the segment's *width*), the midpoints of their sections are roughly on the same line (the segment's *axis*), to which the sections are perpendicular. There should also not be large gaps between symmetries of the same segment (Fig. 3).

We group symmetries by fixing the number of segments and searching for the best segment parameters (their axes and widths) and assignments of symmetries to segments (each symmetry being assigned to one of the segments or to noise). This can be formulated as an optimization problem (segment parameters) in the presence of missing data (labels representing segment assignment for each symmetry). The natural method for solving such a problem is the Expectation-Maximization algorithm (Dempster et al., 1977). This algorithm produces the optimal segment parameters and posterior probabilities for the assignment labels.

Each segment is represented with a *symmetry axis* and a *width*. Each symmetry has a label showing which segment (or noise) it belongs to. A symmetry fits a segment best when the midpoint of the symmetry lies on the segment's symmetry axis, the endpoints lie half a segment width away from the axis, and the symmetry is perpendicular to the axis. This yields the conditional likelihood for a symmetry given a segment as a four-dimensional Gaussian (two numbers for each endpoint), and an EM algorithm can now fit a fixed number of segments to the symmetries. After that, we determine where each segment begins and ends by finding the range of symmetries for which this segment has the largest posterior. If there is a large gap between these symmetries (that is, symmetries from different image regions are attributed to the same segment), then the segment is broken into two or more pieces. The Fig. 4 shows example images produced by the segment finder.

Note that the segments we have found do not have an orientation (for instance, if we hypothesize that one of them is the torso, we don't know which end corresponds to the shoulders and which to the hips). Another problem is that if a limb is straight in an image, then only a single segment is obtained rather than the upper and lower halves. We deal with this by replacing each segment with its two oriented versions, and also splitting each segment in half length-wise and adding both halves to the segment set (Fig. 5). We also add a constraint, when grouping segments, that if one of the segment halves was labeled as a lower limb, then the other half of the same segment has to be the corresponding upper limb.

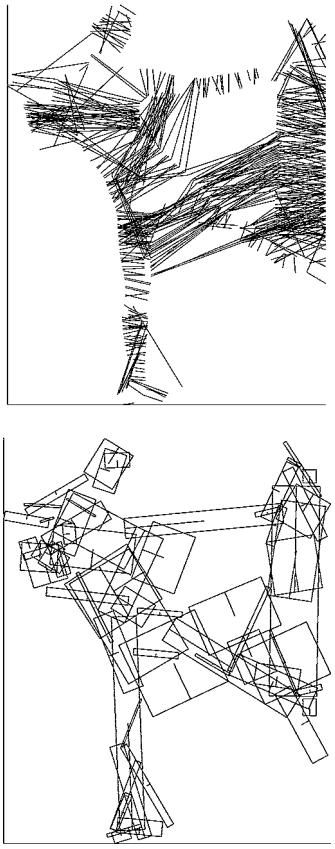

Figure 4. An example run of the segment finder. (top) A set of symmetries obtained for an image. Each symmetry is represented by its section. We show every 4th symmetry to avoid clutter. (bottom) The EM algorithm fits a fixed number of rectangular segments to the symmetries. These are the candidate body segments which become the input to our assembly-builder (grouper).

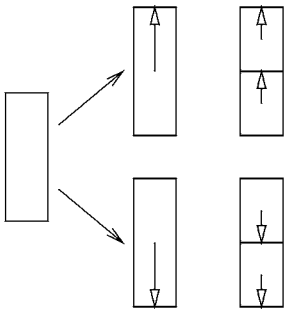

Figure 5. Each segment found using EM is replaced with its two oriented versions, each of which is then split in half. This procedure allows us to deal with the situation when a single segment is found for a straight limb. Also, it allows us to specify, for example, which end of a lower-arm segment is the wrist and which is the elbow. The arrows within the segments indicate their orientation.

It is not necessary to get the segments exactly right, as the kinematic information used in grouping them is powerful enough to handle inaccuracies.

3. Finding People Using Classification

After the segment detector has identified the image regions that could, possibly, correspond to human body parts, we need to assemble these *image segments* into groups that look like people. The brute-force approach of classifying every segment group doesn't work because of the huge number of such assemblies. Instead, we build such assemblies incrementally, by sequentially considering groups of increasing size, and growing a group by trying to add another image segment to it.

The advantage of incremental search is that it could be made quite efficient if we can detect early that a group of segments could *not* be a part of a person, no matter what other segments are added. For example, if two segments can under no circumstances represent the upper and lower left arm, as in Fig. 6(a), then no assembly containing them will correspond to a person.

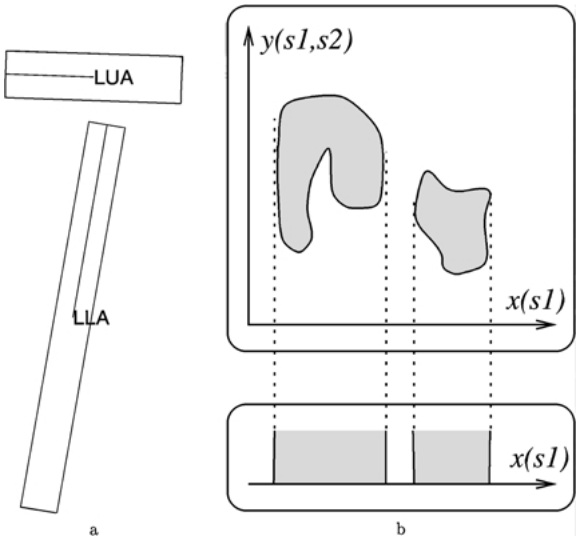

Figure 6. (a) Two segments that cannot correspond to the left upper and lower arm. Any configuration where they do can be rejected using a projected classifier regardless of the other segments that might appear in the configuration. (b) Projecting a classifier $C\{(l_1, s_1), (l_2, s_2\}$. We are considering two features, one of them, $x(s_1)$, depending only on the segment labeled as l_1, and the other, $y(s_1, s_2)$, depending on both segments. (top) The shaded area is the volume classified as positive, for the feature set $\{x(s_1), y(s_1, s_2)\}$. (bottom) Finding the projection C_{l_1} amounts to projecting off the features that cannot be computed from s_1 only, i.e., $y(s_1, s_2)$.

This means that we can prune all model-to-image correspondences containing this one. This observation dates back to (Grimson and Lozano-Pérez, 1987), who use it to prune an *interpretation tree* of correspondences. We show how to derive this pruning mechanism in a principled way from a single, learned classifier.

We learn the *top-level classifier* as the classifier that takes an assembly of 9 segments labeled as the 9 different body parts, and determines whether the assembly corresponds to a person. From that classifier we construct a family of tests that determine, for a group of 8 or fewer segment-label pairs, whether it may be possible for this group to be augmented to a full 9-segment assembly that is classified as a person by the top-level classifier. If it has been determined that no such augmentation is possible for an assembly, then it, and all the assemblies containing it, can be rejected. Each test used for the early rejection is obtained from the top-level classifier by *projection*. By projecting a classifier, we mean obtaining a new classifier that uses a subset of features used by the original one, so that the volume classified as "positive" by the projected classifier is the projection of the positive volume of the original classifier onto a subspace of the feature space. An example of a projected classifier is given in Fig. 6(b).

Projected classifiers allow us to search for human assemblies efficiently, by incrementally considering assemblies of increasing size. At each stage, an assembly is discarded if it is classified as "non-human" by the corresponding project classifier—that is, if no segments can be added to the assembly to obtain a 9-segment human assembly. We grow the assemblies that are not discarded by trying to pair them with each remaining image segment. The search becomes efficient if many assemblies are discarded at an early stage, so that, for a subset of labels, only a small fraction of the set of all the assemblies corresponding to that subset need to be considered.

By introducing projected classifiers we do not make our system more prone to overfitting, since each projected classifier is not learned independently, but rather is *deterministically derived from the top-level classifier*. It follows from the definition of the projected classifier that using them to discard smaller segment groups does not affect the final result: the set of human assemblies found in the image does not change. The gain from the use of projected classifiers is only in *efficiency*: they allow us to find every human assembly in the image without considering all the possible 9-segment groups.

In this section, we describe how to learn a *classifier* that identifies human assemblies, how to organize the search so that assemblies are built *incrementally*, and how to *project* the classifier so that the *projected classifiers* could be used to determine that a segment group could not be a part of a human assembly and thus prune the search.

3.1. Building Segment Configurations

Suppose that the set of candidate segments has been found for an image. We will define an *assembly* as a set

$$A = \{(l_1, s_1), (l_2, s_2), \ldots, (l_k, s_k)\}$$

of pairs where each segment s_i is labeled with the *label* l_i. The segments $\{s_i\}$ are a subset of candidate image segments, and each label l_i specifies what body part is matched to the the segment s_i and can thus take on one of the 9 distinct values ($l_i \in \{\mathtt{T}, \mathtt{LUA}, \mathtt{RLL}, \ldots\}$), corresponding to the torso, left upper arm, right lower leg, etc. All the segments within an assembly are distinct, as are the labels.

An assembly is *complete* if it contains exactly m distinct segments, one for each label. A complete assembly thus represents a full 9-segment configuration (Fig. 1(a) and (b)); we could test whether or not it looks like a person.

Assume we have a classifier C that for any complete assembly A outputs $C(A) > 0$ if A corresponds to a person-like configuration, and $C(A) < 0$ otherwise. Finding all the possible body configurations in an image is equivalent to finding all the complete assemblies A for which $C(A) > 0$. This cannot be done with brute-force search through the entire set because of its size. However, the search can be pruned. It is often possible to determine, for an (incomplete) assembly A, that no bigger assembly A' containing A is classified as a person. For instance, if two segments cannot represent the upper and lower left arm, as in Fig. 6(a), then we do not consider any complete assemblies where they are labeled as such. We prune the search by introducing *projected classifiers*.

3.2. Projected Classifiers

Projected classifiers make the search for body configurations efficient by pruning branches of the interpretation tree (a search tree whose nodes correspond

to matching a particular model segment with an image segment) using the properties of smaller sub-assemblies. Given a classifier C which is a function of a set of features whose values depend on segments with labels $1 \ldots 9$, the *projected classifier* $C_{l_1 \ldots l_k}$ is a function of all those features that depend only on the segments with labels $l_1 \ldots l_k$ and is used to separate sub-assemblies that could possibly be extended to a human configuration from those that could not. In particular, if a complete assembly A can be formed by adding some label-segment pairs to A' so that $C(A) > 0$, then $C_{l_1 \ldots l_k}(A') > 0$ (see Fig. 6(b)). The converse need not be true: the feature values required to bring a projected point inside the positive volume of C may not be realized with any assembly of the current set of segments $1, \ldots, N$.

Notice that, even though many projected classifiers are used (each corresponding to a different subset of labels), this does not increase the possibility of overfitting. The reason is that the projected classifiers do not affect the final classification, but merely make the inference more efficient.

For a projected classifier $C_{l_1 \ldots l_k}$ to be useful, the following two conditions must hold:

- The decision $C_{l_1 \ldots l_k}(A)$ must be easy to compute
- $C_{l_1 \ldots l_k}$ must be effective in rejecting assemblies at an early stage.

These are strong requirements which are not satisfied by most good classifiers. For example, separating hyperplanes (Vapnik, 1996) often provide good discrimination, but the projected classifier is generally useless as it classifies everything as positive. The exception is a hyperplane that is parallel to the direction of projection, in which case it projects to another separating hyperplane. We take advantage of this fact, by using a committee of axis-aligned hyperplanes, described in Section 3.3.2, as our classifier.

3.3. Implementation

Using projected classifiers, we are able to efficiently find all the human-like (i.e. those classified as positive by the top-level classifier) assemblies of image segments. Because the projected classifier rejects a sub-assembly only if it can be determined that no segments can be added to it to make up a human-like assembly, using projected classifiers to prune the interpretation tree never changes the outcome of the search; it does, however, make the search much more efficient, since most medium-size (e.g., >3 segments) sub-assemblies can be rejected early.

In Section 3.3.2 we describe a classifier that both yields good separation of people and non-people and projects well, using the methods of Section 3.3.3. Then, in Section 3.3.4, we show how to use such a classifier (or any other one that satisfies the above two properties).

3.3.1. Features. To classify assemblies, we need to represent each assembly as a point in a feature space, each feature corresponding to a measurement obtained from the assembly. Because the large degree of independence among the kinematics of different human joints, we make each feature depend on 1 segment (i.e. aspect ratios), 2 (angles, length ratios and relative positions) or 3 segments (e.g. the ratio of a distance between the upper legs to the length of the torso). Because we want to be able to detect the person regardless of orientation, position within the image or size, the features are invariant to translation, uniform scaling or rotation of the segment set.

3.3.2. Classifiers that Project. In our problem, each *segment* from the set $\{1 \ldots N\}$ is a bar in some position and orientation. Given a complete assembly $A = \{(l_1, s_1), \ldots, (l_9, s_9)\}$, we want to have $C(A) > 0$ iff the segment arrangement produced by A looks like a person.

We expect the features that correspond to human configurations to lie within small fractions of their possible value ranges. This suggests using an axis-aligned bounding box, with bounds learned from a collection of positive assemblies, for a good first separation, and then bootstrapping it with a weighted committee of *weak* classifiers each of which splits the feature space on a single feature value. The committee is learned by boosting, which is an algorithm to convert a weak classifier to a strong one by sequentially training a number of weak classifiers and determining the contributions each of them makes to the weighted vote. Our classifier projects particularly well, using a simple algorithm described in Section 3.3.3.

We start by computing the bounding box, in the feature space, of the set of human assemblies, and compute projected classifiers (which are also bounding boxes, in feature subspaces). Then, a collection of images without people is searched for human configurations using the method of Section 3.3.4, and the resulting

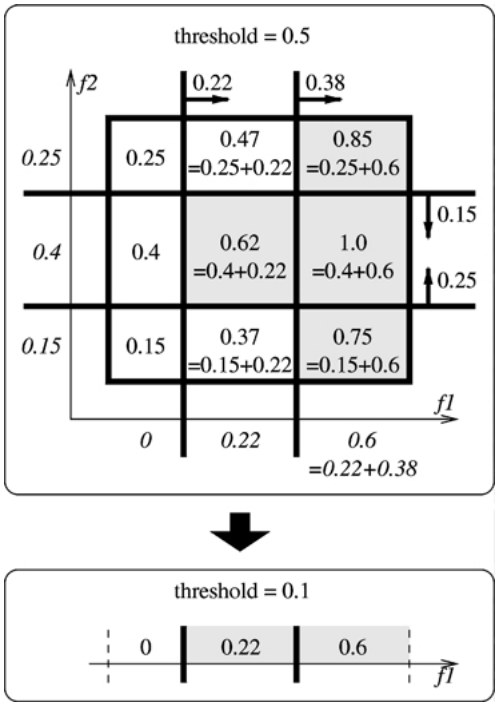

Figure 7. (top) A combination of a bounding box (the thick rectangle) and a boosted classifier, for two features f_1 and f_2. Each plane in the boosted classifier is a thick line with the positive half-space indicated by an arrow; the associated weight β is shown next to the arrow. The shaded area is the positive volume of the classifier, which are the points F where $\sum_k w_k(f_k) > 1/2$, $k = 1, 2$. The weights $w_1(\cdot)$ and $w_2(\cdot)$ are shown (in italics) along the f_1- and f_2-axes, respectively, and the total weight $w_1(f_1) + w_2(f_2)$ is shown for each region of the bounding box. (bottom). The projected classifier, given by $w_1(f_1) > 1/2 - \delta = 0.1$ where $\delta = \max_{f_2} w_2(f_2) = \max\{0.25, 0.4, 0.15\} = 0.4$.

assemblies are now used as negative data. A boosting algorithm (AdaBoost (Freund and Schapire, 1996)) is now applied to these negative assemblies and the original positive ones to learn a weighted committee of weak classifiers. The final classifier accepts an assembly if both the bounding box and the learned committee do so too. An example of a classifier is shown in Fig. 7(top).

Let $F = (f_1 \ldots f_K)$ be the values of all features computed for some complete assembly. The boosted classifier is learned as a weighted committee of *weak classifiers*, each of which splits the feature space on a single feature. The tth classifier ($t = 1 \ldots T$) compares the value of the k_tth feature with some threshold p_t and classifies an assembly as a person if

$$d_t\left(f_{k_t} - p_t\right) > 0,$$

where $d_t \in \{1, -1\}$ determines which half-space is classified as positive.

The output of AdaBoost is a set of such weak classifiers (at each iteration of the algorithm, the feature f_{k_t} to split on, the threshold p_t and the orientation d_t of the separating plane are chosen so as to minimize the classification error, which is computed as the sum of the weights of the misclassified data points, and the weights are updated at each iteration, assigning more weight to the points which have previously been misclassified). Additionally, AdaBoost will associate a weight β_t with each weak classifier; the resulting weighted committee will classify an assembly as positive iff

$$\sum_{d_t(f_{k_t} - p_t) > 0} \beta_t > \sum_{d_t(f_{k_t} - p_t) < 0} \beta_t,$$

that is, if the total weight of the weak classifiers that classify the configuration as a person is greater than the weight of those classifying it as a non-person. By normalizing the weights so that $\sum_t \beta_t = 1$, we can rewrite the decision rule as

$$\sum_{d_t(f_{k_t} - p_t) > 0} \beta_t > 1/2.$$

The set $\{k_t\}$ may have repeating indices (that is, a feature may be split by several planes, which may have different p, d and β values), and does not need to span the entire set $1 \ldots K$. By grouping together the weights corresponding to planes splitting on the same feature, we rewrite the classifier as

$$\sum_{k=1}^{K} w_k(f_k) > 1/2,$$

where

$$w_k(f_k) = \sum_{k_t = k,\ d_t(f_k - p_t) > 0} \beta_t$$

is the weight associated with the particular value of feature f_k. This weight is a piece-wise constant function of f_k; its points of discontinuity are given by $\{p_t \mid k_t = k\}$.

3.3.3. Projecting a Boosted Classifier. Given a classifier constructed as above, we need to construct classifiers that depend on some identified subset of the features. The *projected classifier* $C_{l_1 \ldots l_k}$ is a function of all those features that depend only on the segments with labels $l_1 \ldots l_k$ and is used to separate subassemblies that could possibly be extended to a human

configuration from those that could not. In particular, if a complete assembly A can be formed by adding some label-segment pairs to A' so that $C(A) > 0$, then $C_{l_1 \ldots l_k}(A') > 0$. The geometry of our classifiers—whose positive regions consist of unions of axis-aligned bounding boxes—makes projection easy.

To obtain a projected classifier, one or more features—those that involve the segments left out by projection—must be projected away. For example, the projected classifier for assemblies consisting of a `torso` segment alone is obtained by keeping only the features that can be computed from the `torso` segment.

For convenience of notation, we show how to project away the feature f_K; to project away several features, this process can be applied in a sequence to all of them. The idea that makes projection easy is that, since each weak classifier splits on a feature, projection will remove those of them that split on f_K, and will change the weights of the remaining classifiers in the committee.

The projection of the classifier should classify a point F in the $(K-1)$-dimensional feature subspace as positive iff

$$\max_{F'} \sum_{k=1}^{K} w_k(f'_k) > 1/2$$

where F' is a point in the K-dimensional feature space that projects to F but can have any value for f_K. We can rewrite this expression as

$$\sum_{k=1}^{K-1} w_k(f_k) + \max_{f'_K} w_K(f'_K) > 1/2.$$

The value of

$$\delta = \max_{f'_K} w_K(f'_K)$$

is readily available and independent of f_K. We can see that, with the feature projected away, we obtain

$$\sum_{k=1}^{K-1} w_k(f_k) > 1/2 - \delta,$$

which has the same form as the original classifier, except that the threshold is no longer 1/2. Any number of features can be projected away in a sequence in this fashion. An example of the projected classifier is shown in Fig. 7(bottom).

3.3.4. Building Assemblies Incrementally. Assume we have a classifier C that accepts assemblies corresponding to people and that we can construct projected classifiers as we need them. We will now show how to use them to construct assemblies, using a *pyramid of classifiers*.

A pyramid of classifiers (Fig. 8), determined by the classifier C and a permutation of labels $(l_1 \ldots l_9)$ consists of nodes $N_{l_i \ldots l_j}$ corresponding to each of the projected classifiers $C_{l_i \ldots l_j}$, $i \leq j$. Each of the bottom-level nodes N_{l_i} receives the set of all segments in the image as the input. The top node $N_{l_1 \ldots l_9}$ outputs the set of all complete assemblies $A = \{(l_i, s_i) \ldots (l_9, s_9)\}$ such that $C(A) > 0$, i.e. the set of all assemblies in the image classified as people. Further, each node $N_{l_i \ldots l_j}$ outputs the set of all sub-assemblies $A = \{(l_i, s_i) \ldots (l_j, s_j)\}$ such that $C_{l_i \ldots l_j}(A) > 0$.

The nodes N_{l_i} at the bottom level work by selecting all segments s_i in the image for which $C_{l_i}\{l_i, s_i)\} > 0$ (the only single-segment feature we use is the ratio of segment's width to its length, so we simply select the segments with appropriate aspect ratios). Each of the remaining nodes has two parts:

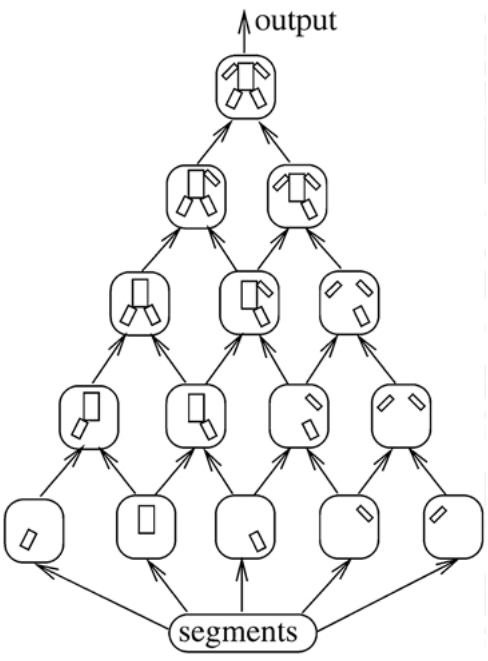

Figure 8. A pyramid of classifiers. Each node outputs sub-assemblies accepted by the corresponding projected classifier. Each node except those in the bottom row works by forming assemblies from the outputs of its two children, and filtering the result using the corresponding projected classifier. The top node outputs the set of all complete assemblies that correspond to body configurations.

- The *merging* stage of node $N_{l_i \ldots l_j}$ merges the outputs of its children by computing the set of all assemblies $\{(l_i, s_i) \ldots (l_j, s_j)\}$ such that

$$C_{l_i \ldots l_{j-1}}\{(l_i, s_i) \ldots (l_{j-1}, s_{j-1}\} > 0$$

and

$$C_{l_{i-1} \ldots l_j}\{(l_{i+1}, s_{i+1}) \ldots (l_j, s_j)\} > 0.$$

- The *filtering* stage then selects, from the resulting set of assemblies, those for which

$$C_{l_i \ldots l_j}(\cdot) > 0,$$

and the resulting set is the output of $N_{l_i \ldots l_j}$.

Because $C\{(l_1, s_1) \ldots (l_9, s_9)\} > 0$ implies $C_{l_i \ldots l_j}\{(l_i, s_i) \ldots (l_j, s_j)\} > 0$ for any i and j, it is clear that the output of the pyramid is, in fact, the set of all complete A for which $C(A) > 0$ (note that $C_{l_1 \ldots l_9} = C$, as a assembly is defined only by the set of segments and their labels but not their order).

The only constraint on the order in which the outputs of nodes are computed is that children nodes have to be applied before parents. In our implementation, we use nodes $N_{l_i \ldots l_j}$ where j changes from 1 to m, and, for each j, i changes from j down to 1. This is equivalent to computing sets of assemblies of the form $\{(l_1, s_1) \ldots (l_j, s_j)\}$ in order, where getting $(j+1)$-segment assemblies from j-segment ones is itself an incremental process, whereby we check labels against l_{j+1} in the order $l_j, l_{j-1}, \ldots, l_1$. In practice, we choose the latter order on the fly for each increment step using a greedy algorithm, to minimize the size of assembly sets that are constructed (note that in this case the classifiers may no longer form a pyramid). The order $(l_1 \ldots l_9)$ in which labels are added to an assembly needs to be fixed. We determine this order off-line with a greedy algorithm by running a large segment set through the assembly builder and choosing the next label to add so as to minimize the number of assemblies that result.

3.3.5. Efficient Classification. The type of classifier C we are using allows for an efficient building of assemblies, in that the features do not need to be recomputed when we add a label-segment pair and move from $C_{l_1 \ldots l_i}$ to $C_{l_1 \ldots l_{i+1}}$.

We achieve this efficiency by carrying, along with an assembly $A = \{(l_1, s_1) \ldots (l_i, s_i)\}$, the sum

$$\sigma(A) = \sum_{k \in \mathcal{K}(l_1 \ldots l_i)} w_k(f_k)$$

where $\mathcal{K}(l_1 \ldots l_i)$ is the set of all features computable from the segments labeled as $l_1, \ldots, l_i$, and $\{f_k\}$—the values of these features.

When we add another segment to get $A' = \{(l_1, s_1) \ldots (l_{i+1}, s_{i+1}\}$, we can compute

$$\sigma(A') = \sigma(A) + \sum_{k \in \mathcal{K}(l_1 \ldots l_{i+1}) \setminus \mathcal{K}(l_1 \ldots l_i)} w_k(f'_k).$$

In other words, when we add a label l_{i+1}, we need to compute only those features that require the segment s_{i+1} for their computation.

4. Finding People by Sampling

The approach of Section 3 has the disadvantage of employing a binary classifier, which tries to discriminate people from non-people. The problems with such a classifier include:

- Discrimination is made difficult by humans that appear in unusual configurations or poses, have occluded or low-contrast body parts, as well as by spurious limb-like segments that do not correspond to people but may, by chance, occur in groups that resemble people.
- The classifier we described cannot determine how *likely* an assembly is to be a person, and thus makes it difficult to revise the classification if new information is added. For example, if two human-like assemblies overlap each other, only one of them could represent the true configuration of a person, but the classifier will not be able to choose which assembly it is, since both of them are classified as people. This makes it difficult to count people, since a person often produces several human-like assemblies, and a subset of all assemblies must be chosen to determine how many people there are.

We address these problems by associating a *likelihood* with each assembly, which is proportional to the probability of seeing an assembly in a random view of a person and is high for the more human-like assemblies and low for the non-human ones. Then, if we know that there is exactly one person in the image, we

may choose the maximum of the likelihood. To count people, we would count sufficiently separated peaks (modes). If the image is a frame in a video sequence, we may choose an assembly different from the likelihood mode if it agrees with the assemblies in the nearby frames—this could be used for robust tracking.

We represent the likelihood with a set of samples of assemblies, with probabilities of drawing an assembly proportional to its likelihood. As a result, the more an assembly looks like a person the more often it will be chosen, but assemblies that look less like people are not completely discarded, which will allow for incorporation of future evidence (for example, in the counting procedure of Section 5.3).

To draw samples from the likelihood, we cannot use the brute-force method of computing the likelihood for each assembly, because of the huge number of possible assemblies. Instead, we propose to sample the likelihood by drawing samples from a sequence of approximations, using resampling at each stage (cf. (Blake and Isard, 1998; Deutscher et al., 2000; Kanazawa et al., 1995; Neal, 1998)). In particular, we incrementally build segment groups of *increasing size*, by sampling them from appropriate *intermediate distributions*.

The intermediate distributions are derived from the likelihood so that, for a partial (<9-segment) assembly, the value of the corresponding intermediate distribution is high if it is likely that the assembly can be augmented to produce a human assembly, and low otherwise. This allows us to devote more attention to assemblies with a "higher potential." The intermediate distributions make sampling from the likelihood possible, much like the projected classifiers of Section 3 help find assemblies accepted by the top-level classifiers, and the process of deriving them from the likelihood is quite similar to projection (see Fig. 9).

The samples from an intermediate distribution are augmented by adding a segment with a new label and *resampling* so as to get samples from the new intermediate distribution. Using *importance sampling*, we are able to make sure that the incremental process of iterated resampling yields 9-segment assemblies that are, approximately, samples from the likelihood.

In this section, we explain why the likelihood makes sense, and show how to learn it from data. We show how to use *importance sampling* to incrementally sample assemblies of increasing size from the appropriate *intermediate distributions*, so that at the end samples from the likelihood are obtained. The intermediate distributions are not learned independently but are *directly derived from the likelihood*, thus avoiding overfitting. We show how to count people in images, using the samples we obtain.

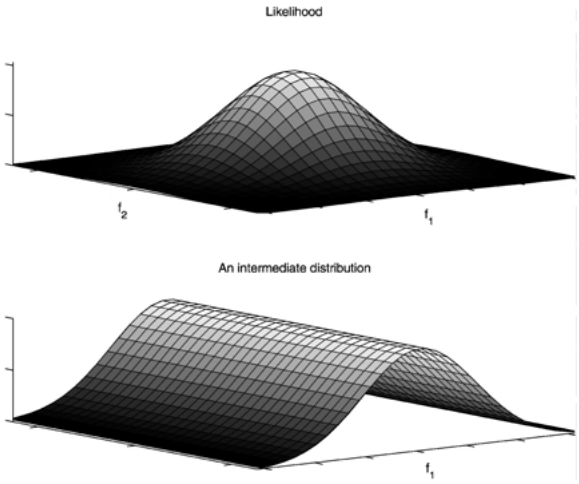

Figure 9. The process of deriving intermediate distributions from the likelihood is quite similar to projection (cf. Fig. 6(b)). (top) The original likelihood depends on two features of an assembly: f_1 and f_2. (bottom). An intermediate distribution for assemblies for which f_2 cannot be computed (because these smaller assemblies do not contain all of the segments on which f_2 depends).

4.1. Sampling from Likelihood

We propose the probability of drawing an assembly as a sample to be proportional to its *likelihood*. The likelihood is defined as the probability density, in an appropriate feature space, of the set of features corresponding to a 9-segment assembly, *given* that this assembly represents a person.

This density can be learned from data. Thus, the likelihood will favor assemblies that are, in some sense, similar to those in the training set. The model we use to learn the density is discussed in Section 4.3.2, but any other reasonable model could be used instead without changes to the general framework (a reasonable model is one that assigns a higher distribution value to an assembly that is more like a person).

The likelihood is an attractive distribution to sample from, since it is higher for the assemblies that are more people-like (according to the training set). It has to be noted that, although we refer to the likelihood as a *distribution*, it is not really because the likelihoods of all assemblies in the image may not sum up to 1. The actual distribution we sample from is only proportional

to the likelihood; the constant of proportionality is hard to compute, but, fortunately, the sampling methods we use (importance sampling) do not need the distributions to be normalized.

Another justification for the use of likelihood is as follows. Suppose that we know that the segment set of an image contains exactly one 9-segment human assembly, and we need the posterior on which assembly it is. Assuming a uniform prior, we see that the posterior is

$$\Pr[A = \text{person} \mid \text{segments},\ 1\,\text{person}] \propto L(A) \times P(\text{segments} \mid A = \text{person}, 1\,\text{person})$$

($\propto$ means proportional to). The first term of the right-hand side is the likelihood—how likely a random person is to look as the assembly A. The second term is the probability density of seeing the image segments, given that exactly one person, given by assembly A, is present. Assuming that segments are independent of one another, this is just the probability of seeing the segments *not* in A. This probability doesn't depend on the configuration of A. We assume that the distribution on non-person segments is uniform; let α be the value of the corresponding probability density. Then,

$$P(\text{segments} \mid A = \text{person}, 1\,\text{person}) \propto P(\text{segments not in } A) = \alpha^{\#\text{ segments not in } A},$$

and the last term is the same for any 9-segment assembly A. Thus,

$$\Pr[A = \text{person} \mid \text{segments}, 1\,\text{person}] \propto L(A),$$

and sampling from likelihood is the right thing to do.

4.2. Resampling

There are too many nine-segment assemblies to compute the likelihood for each. However, as in Section 3, we can build assemblies incrementally, and exclude smaller segment groups from further consideration if it can be determined that they cannot be a part of a person. For example, having generated a set of samples of the form $\{(\mathtt{T}, s_{\mathtt{T}})\}$ of single-segment assemblies each consisting of a `torso` segment, and samples of the form $\{(\mathtt{LUA}, s_{\mathtt{LUA}})\}$ of assemblies each of which contains only a `left upper arm`, we can form all combinations of the form $\{(\mathtt{T}, s_{\mathtt{T}}), (\mathtt{LUA}, s_{\mathtt{LUA}})\}$ and then resample those, so that the resulting samples of two-segment assemblies $\{(\mathtt{T}, s_{\mathtt{T}}), (\mathtt{LUA}, s_{\mathtt{LUA}})\}$ come from the appropriate marginal likelihood or other appropriate intermediate distribution. (Presumably, among the resulting samples, the groups similar to those found in people will occur more frequently.) We can proceed by similarly sampling 3-, 4-, ..., 9-segment sub-assemblies, in such a way that the resulting set of 9-segment assemblies is sampled from $L(\cdot)$.

At each stage, we use *importance sampling*, which is a method for drawing samples from (possibly intractable) distributions (as used in (Blake and Isard, 1998)). In particular, to draw a sample from $g(x)$, we first draw a large number of independent samples $\{s_1, \ldots, s_n\}$ from a proposal distribution $f(x)$, and then set $s = s_i$ with probability proportional to $w_i = \frac{g(x)}{f(x)}$. As $n \to \infty$, the distribution for the sample s will approach $g(x)$. In our case, the proposal distribution $f(\cdot)$ corresponds to the *intermediate distribution* L_k on k-segment assemblies, while the target distribution $g(\cdot)$ corresponds to the intermediate distribution L_{k+1} on $(k+1)$-segment assemblies. The intermediate distributions should be chosen in such a way that we are more likely to propose, for example, a two-segment assembly $\{(\mathtt{RUA}, s_{\mathtt{RUA}}), (\mathtt{RLA}, s_{\mathtt{RLA}})\}$ if the two segments individually are more likely to be `upper right arm` and `lower right arm` of a person; we discuss this choice in Section 4.3.4.

4.3. Implementation

Our system starts by finding segments as described in Section 2. From the segments, we use a learned *likelihood model* to form *assemblies* by sampling. To generate assemblies, we use incremental sampling, whereby segment groups of increasing size are drawn from appropriate *intermediate distributions*, each of which roughly corresponds to a marginal likelihood but takes all image segments into account and is computed using dynamic programming. Finally, the set of assemblies is replaced with a smaller set of *representatives*, which are used to count people in the image.

By incrementally sampling segment groups of increasing size, we guide the sampler to larger assemblies that are more human-like. Asymptotically (as the number of samples increases), the resulting samples of 9-segment assemblies will be drawn from the likelihood $L(A)$. Therefore, as in the method of Section 3, the intermediate distributions do not affect the final result; they do, however, help us obtain the samples from $L(A)$ more efficiently.

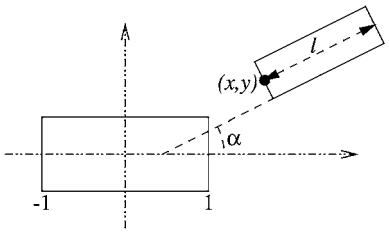

Figure 10. The parameterization of a joint. With one of the two segments, a frame of reference is associated, so that the coordinate axes are parallel to the sides of that rectangle, the origin is at its center, and all lengths are relative to the length of the segment (we scale the frame of reference so that the abscissae of the ends of the reference segment are 1 and -1). The 4 features that parameterize the joint are the coordinates (x, y) of the midpoint of the adjacent side of the second segment, the length l of the second segment (relative to that of the first), and the angle α between the segments.

4.3.1. Features. An assembly of 9 rectangular segments, modulo rotation, translation and uniform scaling, has 41 degrees of freedom. A natural way to parameterize such an assembly is by a set of 9 aspect ratios, and 4 features at each of the 8 joints that encode the relative lengths, angles and displacements between adjacent segments (Fig. 10). Thus, each feature in our model depends on either one or two segments, and the two-segment features can be computed either from the two halves of the same limb (such as `right upper arm` and `right lower arm`), or from an upper limb and the torso.

4.3.2. Learning Likelihoods for People. We assume that the configurations at the joints are independent of one another. For example, the arms and legs can move almost independently of each other, and there is little correlation among the the configurations of the elbows and knees. We make the assumption for 3D configurations of people and extend it to their 2D projections onto the image plane. Another way of putting it in 2D is to say that, if we have several human assemblies, normalized so that, for example, the lengths and positions of their torsos are fixed, then by pasting together left arm of one of these assemblies, right leg of another, torso of the third, and so on, we will get another human assembly.

The main errors will be due to interactions between kinematic constraints on the hips and shoulders, and viewing pose. For instance, if we consider both frontal and lateral views of people, then the configuration of the torso and arms imposes constraints on the orientation of the person (frontal to lateral), which in turn constrains the configuration of the torso and legs.

While it is convenient to parameterize the body by independent joint configurations, we have found that such a model is not sufficient. Often, it results in assemblies being found in the image such that several model segments are matched with the same image segment (e.g., a segment may be labeled as both the lower arm and upper leg). This causes the assemblies found for people to incorrectly represent their configuration, and also results in spurious non-human assemblies being classified as people (since, with coinciding segments, fewer than 9 segments are needed to build an assembly). We solve the problem by adding a constraint that all the segments in an assembly be distinct, and define the indicator function $I_{dist}(A)$ to be 1 if this constraint is satisfied for assembly A, and 0 otherwise. Then, we can write the likelihood from which we sample as

$$L(A) \propto I_{dist}(A) \times \prod_{k=1}^{41} \ell_k(f_k), \tag{1}$$

where f_k is the value of the kth feature, and $\ell_k(f_k)$ is the corresponding one-dimensional marginal likelihood. This representation avoids the problems with learning high-dimensional distributions: each $\ell_k(\cdot)$ can be learned independently, from a relatively small data set. In our experiments, we chose for $\ell_k(\cdot)$ to be a histogram for the values f_k (Fig. 11).

Notice that while, without the $I_{dist}(A)$ term, our model could be described as a tree and dealt with using efficient inference methods available for tree-structured graphical models (such as dynamic programming (Felzenszwalb and Huttenlocher, 2000), it is no longer a tree. For example, the choices of segments corresponding to the left arm and the right arm are no longer conditionally independent given the torso. We will show, however, that dynamic programming could still be used to drive the search—namely, to compute the intermediate distributions from which subassemblies are drawn.

4.3.3. Building Assemblies Incrementally by Resampling. We fix a permutation $(l_1, \ldots, l_9)$ of labels $\{$`T`, `LUA`, $\ldots\}$, and generate a sequence

$$(S_1, \ldots, S_9)$$

of multisets (sets with possibly repeating elements) of samples, where each S_k contains N (not necessarily

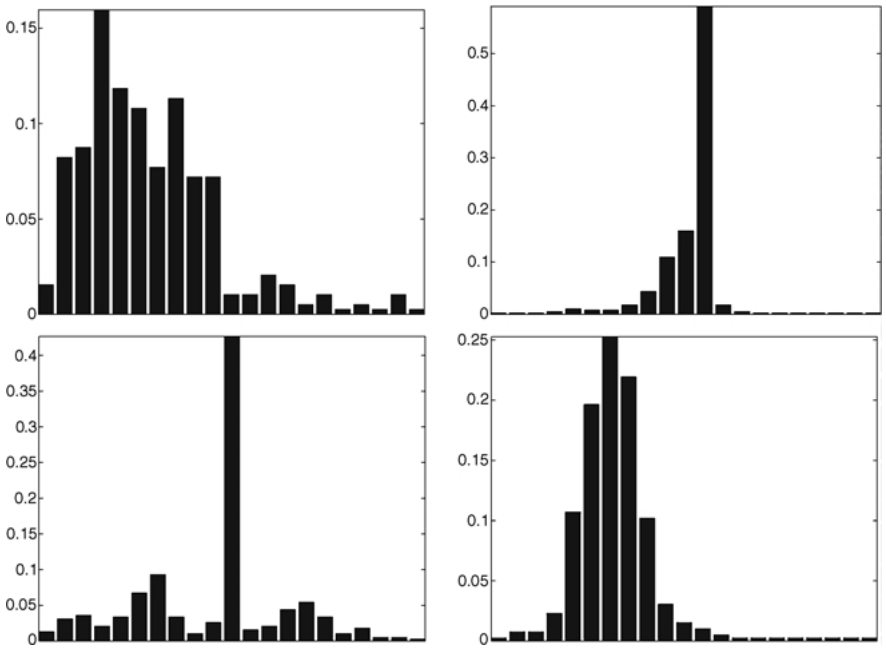

Figure 11. Examples of histograms that are used to model one-dimensional distributions ℓ_i of features. Note the sharp spikes on some of the histograms. They are due to the training data in which, for many limbs, a single segment was found, and thus the upper and lower halves had to be obtained by splitting this segment. This has biased our system towards such limb configurations.

distinct) assemblies of the form

$$\{(l_1, s_{l_1}), \ldots, (l_k, s_{l_k})\}$$

of k segments labeled as $l_1, \ldots, l_k$ (Fig. 12). For example, in our implementation, $(l_1 \ldots l_9) = (\mathtt{T}, \mathtt{LUA}, \mathtt{LLA}, \ldots)$, and so S_1 will contain the samples $\{(\mathtt{T}, s_{\mathtt{T}})\}$ of `torso` segments, while S_3 will contain samples

$$\{(\mathtt{T}, s_{\mathtt{T}}), (\mathtt{LUA}, s_{\mathtt{LUA}}), (\mathtt{LLA}, s_{\mathtt{LLA}})\}$$

of triples corresponding to the `torso`, the `left upper arm` and the `left lower arm`. The samples in S_k are drawn from appropriately chosen *intermediate distributions* $L_k(\cdot)$, discussed in Section 4.3.4.

We generate the set of samples S_{k+1} from S_k using importance sampling. First, we form the set of sub-assemblies

$$\{(l_1, s_{l_1}), \ldots, (l_k, s_{l_k}), (l_{k+1}, s_{l_{k+1}})\}$$

for all groups

$$\{(l_1, s_{l_1}), \ldots, (l_k, s_{l_k})\} \in S_k$$

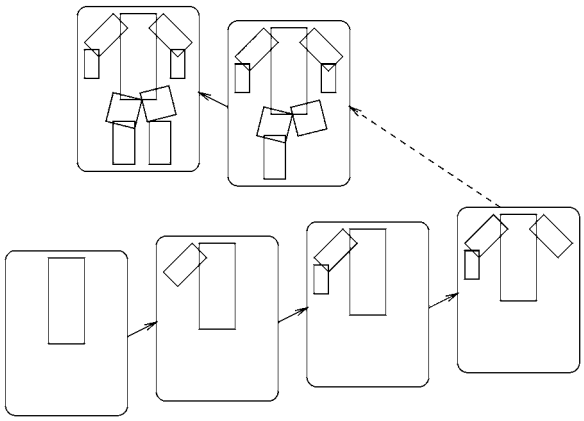

Figure 12. We sample assemblies incrementally, by generating sets of samples of 1-, 2-,. . . , 9-segment assemblies, so that the latter are drawn from the likelihood $L(\cdot)$. The set S_k of k-segment assemblies is drawn from the intermediate distributions L_k, with $L_9 = L$. To generate the set S_{k+1} from S_k, we use importance resampling, with resampling weights equal to $L_{k+1}(\cdot)/L_k(\cdot)$. Compare this to building assemblies incrementally in the classification approach (cf. Fig. 8).

(which are assumed to be samples from L_k) and all choices of $s_{l_{k+1}}$. This way, we obtain samples of $(k+1)$-segment assemblies but which are drawn from L_k rather than L_{k+1}. We now *resample* this set of samples, by

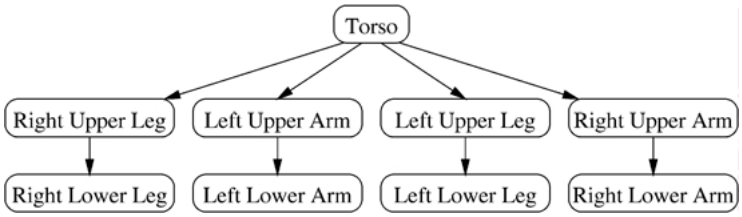

Figure 13. A tree-structured graphical model that could be used for human configurations if we lifted the constraint that all segments in an assembly be distinct. However, even for non-tree-shaped models, graphical models like this one can be used to drive the search, by allowing us to efficiently compute intermediate distributions for sub-assemblies that take into account all the image segments.

independently drawing N samples, with the probability of drawing $\{(l_1, s_{l_1}), \ldots, (l_{k+1}, s_{l_{k+1}})\}$ proportional to

$$w\{(l_1, s_{l_1}), \ldots, (l_{k+1}, s_{l_{k+1}})\} = \frac{L_{k+1}(\cdot)}{L_k(\cdot)}.$$

4.3.4. Intermediate Distributions. In the first approximation, we could sample S_k by taking the corresponding intermediate distribution L_k to be the *marginal likelihood*

$$L_{l_1 \ldots l_k}(A) \propto I_{dist}(A) \times \prod_i \ell_i(f_i),$$

where the product is over all the features computable from segments labeled as $l_1, \ldots, l_k$, and $I_{dist}(A) = 1$ iff all of those segments are distinct, and $= 0$ otherwise. We write s_l for the segment of the sub-assembly whose label is l. For our feature set and the choice of $(l_1 \ldots l_9)$, each of the marginal likelihoods $L_{l_1 \ldots l_k}(s_{l_1}, \ldots, s_{l_k})$ models the probability that the sub-assembly $(s_{l_1}, \ldots, s_{l_k})$ is seen in a random view of a human.

A disadvantage of using marginal likelihoods as intermediate distributions is that the marginal likelihood of a sub-assembly does not depend on segments around it. For example, the value of such an intermediate distribution on a segment pair that looks like an arm will not depend on whether the segments around the pair look like the rest of a person or not. It would be desirable to be able to efficiently compute distributions that would take the sub-assembly's context into account and provide a better guide as to whether the sub-assembly is a part of a person.

If we can do this and find intermediate distributions that approximate

$$\max\{L(A) \mid A \text{ contains the given subassembly}\},$$

then incremental sampling will sample complete assemblies from a distribution that better approximates $L(A)$. We believe this to be so because, when importance sampling is used to sample from an unnormalized distribution $g(\cdot)$ via the intermediate distribution $f(\cdot)$, the target distribution is best approximated when the two distribution are proportional to each other.

When incrementally building assemblies, we add segments in the following order: the torso; the 4 upper limbs; the 4 lower limbs. We will define, as our intermediate distributions $L_k(\cdot)$ on k-segment assemblies, the maximum

$$\max_{A' \supset A} L(A')$$

over all the complete assemblies that contain the sub-assembly. When computing the maximum, we *lift the constraint that the same segment not be assigned several labels* (so that now a segment can be, for example, left and right lower arm at the same time). By doing this, we ensure that all features computed for an assembly are computed from either single segments or pairs of adjacent segments, and thus the body model is tree-structured (Fig. 13). Indeed, all the numeric features described in Section 4.3.2 have this property; to compute intermediate distributions, we remove the binary feature for which this property does not hold: the feature that is uniform if all the segments in an assembly are distinct, and is 0 otherwise.

The removal of the distinct-segments constraint is essential to making maximization efficient, since it reduces the graphical model representing a person to a tree (with the torso as the root) for which efficient inference methods, such as dynamic programming, are available. On the other hand, this implies that the assembly A' thus found may have $I_{dist}(A') = 0$, and so we still need resampling steps to make sure that we sample from $L(A)$ rather than its relaxation.

The highest likelihood assembly, containing a given sub-assembly, is found by a simple Dynamic Programming algorithm, whereby we find the best lower half of a limb for each upper half, and then find the best limb of each type for each torso.

As an example, suppose that all of the segments in an assembly, except the lower left arm, are fixed, and we are to choose the lower left arm that maximizes the likelihood of the resulting assembly. It is easy to see that, in our model, the lower left arm can be found by considering all the pairs of a lower left arm (which can be any segment) and the upper left arm (which is fixed), and choosing the one with the highest marginal likelihood $L_{\mathtt{LUA,LLA}}$. This is true because of the fact that all features that involve the left lower arm may involve either no other segments or the left upper arm only.

Now, let us suppose that we have fixed a `torso` and, possibly, some limbs, and we want to add the left arm that would maximize the likelihood of the result. First, for each choice $s_{\mathtt{LUA}}$ of the left upper arm, we will find the best left *lower* arm $\mathtt{best}_{\mathtt{LLA}}(s_{\mathtt{LUA}})$ by maximizing

$$\mathtt{best}_{\mathtt{LLA}}(s_{\mathtt{LUA}}) = \arg\max_{s_{\mathtt{LLA}}} L_{\mathtt{LUA,LLA}}\{(\mathtt{LUA}, s_{\mathtt{LUA}}), (\mathtt{LLA}, s_{\mathtt{LLA}})\}.$$

Since no feature involves the left arm and any other limb, we can choose the best left arm for a given `torso` by considering all possible choices of the left *upper* arm, each of which defines the whole arm as shown above. For a torso segment s_T, we find the best left upper arm $best_{\mathtt{LUA}}(s_{\mathtt{T}})$ as follows:

$$\mathtt{best}_{\mathtt{LUA}}(s_{\mathtt{T}}) = \arg\max_{s_{\mathtt{LUA}}} L_{\mathtt{T,LUA,LLA}}\big\{(T, s_{\mathtt{T}}), (\mathtt{LUA}, s_{\mathtt{LUA}}), (\mathtt{LLA}, \mathtt{best}_{\mathtt{LLA}}(s_{\mathtt{LUA}}))\big\}.$$

Thus, for a torso segment $s_{\mathtt{T}}$, the best left arm will be

$$\{(\mathtt{LUA}, \mathtt{best}_{\mathtt{LUA}}(s_{\mathtt{T}})), (\mathtt{LLA}, \mathtt{best}_{\mathtt{LLA}}(\mathtt{best}_{\mathtt{LUA}}(s_{\mathtt{T}})))\}.$$

Now we have an algorithm to compute

$$\max_{A' \supset A} L(A')$$

for any assembly A which contains a torso segment $s_{\mathtt{T}}$, some whole limbs and some upper limbs—which, due to the order in which we add segments, is the only type of assemblies we deal with. For each of the 4 limbs, we do the following.

- If the whole limb is missing, add the best upper segment

  $$s_{\mathtt{upper}} = \mathtt{best}_{\mathtt{upper}}(s_{\mathtt{T}})$$

  and the corresponding lower segment

  $$\begin{aligned} s_{\mathtt{lower}} &= \mathtt{best}_{\mathtt{lower}}(s_{\mathtt{upper}}) \\ &= \mathtt{best}_{\mathtt{lower}}(\mathtt{best}_{\mathtt{upper}}(s_{\mathtt{T}})). \end{aligned}$$

- If just the lower half is missing, add the segment

  $$s_{\mathtt{lower}} = \mathtt{best}_{\mathtt{lower}}(s_{\mathtt{upper}})$$

  that corresponds to the upper half $s_{\mathtt{upper}}$.

This algorithm is efficient: because only pairs of segments are considered, it runs in $O(n^2)$ time for n segments (and faster in practice, if we try to pair up only those segments that are close to each other). Although the upper bounds provided by this algorithm are very effective for directing the sampler to relevant image regions, they may not be tight. For example, in the resulting assemblies the legs may coincide. Therefore, the above algorithm does not replace sampling, but merely provides a good proposal mechanism.

Notice that a similar proposal mechanism could be used for any likelihood function $L(A)$. To make use of efficient tree inference algorithms, we would simply have to specify an approximation $L_{tree}(A) \approx L(A)$ such that the model L_{tree} could be represented as a tree-shaped graphical model, computed efficiently with dynamic programming, and used to compute intermediate distributions.

5. Experiments

In this section, we describe the tests we used to verify our methods. In Section 5.1, we describe the performance of the system described in Section 3, which learns a top-level classifier, and identifies all the image assemblies that are classified as people. To make the search efficient, assemblies are built incrementally, and non-human assemblies are rejected early by using projected classifiers, derived from the top-level one.

The results are encouraging but not quite satisfactory, in part because the system of Section 3 cannot count people. This suggests that it would be beneficial to have a measure of human-likeness associated with each assembly, and to retain some uncertainty

by recognizing not only the "best" assembly but the sufficiently "good" ones as well. We do so using the system of Section 4, which samples assemblies from the likelihood. The sampling is made possible by employing a series of approximations to the likelihood, from which assemblies of increasing size are sampled. In Section 5.2, we show that such a system is able to find and count people.

5.1. Finding People Using a Classifier

We report results for a system that automatically identifies potential body segments, and then applies the assembly process described above. (Since, at the time of the experiments in this section, the EM-based segment detector was not available, we used more ad hoc techniques (Forsyth and Fleck, 1997) to group symmetries into segments.) Images for which assemblies that are kinematically consistent with a person are reported as having people in them. The segment finder may find either 1 or 2 segments for each limb, depending on whether it is bent or straight; because the pruning is so effective, we can allow segments to be broken into two equal halves lengthwise (like the left leg in Fig. 1(b)), both of which are tested.

5.1.1. Training. The training set included 79 *negative* images without people, selected randomly from the COREL database, and 274 *positive* images each with a single person on uniform background. The grey-scale positive images have been scanned from books of human models (Shuppan, 1993–1996), and all segments in the those images were reported. The negative images were originally in color, but were converted to grey-scale after removing regions that did not corresponded to human skin in colour and texture (censored regions in Fig. 17 are shown in white). Without such censoring, too many spurious segments were produced, which generated spurious person-like assemblies. Negative images, both for the training and for the test set, were chosen so that all had at least 30% of their pixels similar to human skin in colour and texture. This gives a more realistic test of the system performance by excluding regions that are obviously not human, and reduces the number of segments in the negative images to the same order of magnitude as those in the positive images.

The models are all wearing either swim suits or no clothes, otherwise segment finding fails; it is an open problem to segment people wearing loose clothing (see Section 6). There is a wide variation in the poses of the training examples, although all body segments are visible. The sets of segments corresponding to people were then hand-labeled. Of the 274 images with people, segments for each body part were found in 193 images. The remaining 81 resulted in incomplete configurations, which could still be used for computing the bounding box used to obtain a first separation. Since we assume that if a configuration looks like a person then its mirror image would too, we double the number of body configurations by flipping each one about a vertical axis. The bounding box is then computed from the resulting 548 points in the feature space, without looking at the images without people.

The boosted classifier was trained to separate two classes: the $193 \times 2 = 386$ points corresponding to body configurations, and 60727 points that did not correspond to people but lay in the bounding box, obtained by using the bounding box classifier to incrementally build assemblies for the images with no people. We added 1178 synthetic positive configurations obtained by randomly selecting each limb and the torso from one of the 386 real images of body configurations (which were rotated and scaled so the torso positions were the same in all of them) to give an effect of joining limbs and torsos from different images rather like children's flip-books. Remarkably, the boosted classifier classified each of the real data points correctly but misclassified 976 out of the 1178 synthetic configurations as negative; the synthetic examples were unexpectedly more similar to the negative examples than the real positive examples were.

5.1.2. Results. The test dataset was separate from the training set and included 120 images with a person on a uniform background, and varying numbers of negative images, reported in Table 1. We report results for two

Table 1. Number of images of people (positive) and without people (negative) processed by the classifiers with 367 and 567 features, followed by the false negative (images with a person where no body configuration was found) and false positive (images with no people where a person was detected) rates. The majority of the false negative cases were due to the segment finder failing to extract all the relevant segments, while the assembly builder cannot handle missing segments.

Feats	Pos	Neg	False −	False +
367	120	28	37%	4%
567	120	86	49%	10%

classifiers, one using 567 features and the other using a subset of 367 of those features. Figure 1 also shows the false positive and false negative rates achieved for each of the two classifiers. By marking 51% of positive images and only 10% of negative images, the classifier using 567 features compares extremely favorably with that of (Forsyth et al., 1996), which marked 54% of positive images and 38% of negative images using hand-tuned tests to form groups of four segments. In 55 of the 59 images where there was a false negative, a segment corresponding to a body part was missed by the segment finder, meaning that the overall system performance significantly understates the classifier performance (these experiments were done with an earlier version of the segment finder, which was more prone to missing segments; the EM-based version, described in Section 2, is the one used in our second approach of Section 4). There are few signs of overfitting, probably because the features are highly redundant. Using the larger set of features makes assembly faster (by a factor of about five), because more configurations are rejected earlier.

5.2. *Finding People by Sampling*

To learn the likelihood model $L(\cdot)$, we used a set, of 193 training images, scanned from (Shuppan, 1993, 1996). Each contained a photograph of a single person, standing against a uniform background. All the views were frontal and all limbs were visible, although the configurations varied. The models wore swimsuits or no clothes, since clothes make it hard to propose body segments. The symmetries produced for each image were used to determine sets of segments, although the segment finder was not the EM-based one used on the test data. We hand-labeled the segments by marking those corresponding to the 9 body segments. In fact, the training images were the part of a larger collection that resulted in complete assemblies (no segment finder misses). Since the likelihood should not favor an assembly over its mirror image, we expanded the training set by adding the mirror image of each assembly, thus resulting in 386 configurations. The likelihood $L(\cdot)$ was defined as in Eq. (1), where $\ell_i(\cdot)$ were the histograms (with 20 bins) for each of the 41 geometric features for the training set.

5.2.1. Test Data. The test data included 145 *negative images* with no people, and 228, 72, and 65 images with 1, 2, and 3 people, respectively. The negative images came from the COREL database, while those with people were obtained by combining single-person images from the same collection as, but distinct from, the training data.

The sets of symmetries were produced for each test image. The parts of the negative images differing significantly in color from people's skin (no more than 1/2 of each image) were blanked out before finding symmetries; no such preprocessing was done for images with people. The EM-based segment finder was applied to each set of symmetries by fitting 50 mixture components to each negative image, 20 and 40 (on separate runs) to the 1-person images, and 40 and 60 to both 2- and 3-person images. The actual number of segments produced varied, due to splitting of segments with gaps. The resulting collections of segments were then used for testing.

To be able to find both straight (1 segment) and bent (2 segments) limbs, we added both halves (lengthwise) of each segment to the segment sets. The halves of a segment, however, could appear only either together in the same limb, or as the `torso`.

5.2.2. Directing the Sampler. Our sampler is working in a discrete space of labels and image segments. It can be difficult to focus the activity of such samplers on components with large probability. For example, if there are two people in the image, and one results in a large group of segments and the other in a small group (due to mischief in the segment finder), the sampler may repeatedly draw samples from the large group corresponding to the one person, and never get to the other. A natural strategy is to break the domain into a set of equivalence classes, sample the classes, and then sample within the classes drawn by that sampler.

We define equivalent assemblies to be those that label the same segment as a `torso`. This is a good choice, because different people in an image will tend to have their torsos in different places. We represent the class by the assembly that has the highest likelihood. This means that we have a tight upper bound for the likelihoods within the equivalence class, which means that classes that are omitted when we sample classes tend to be those which contain elements of relatively low likelihood. For an exact algorithm we would need elements within classes to have similar likelihoods; our results suggest that this is not particularly important.

In practice, we find the representative assembly for each torso segment by drawing samples of assemblies

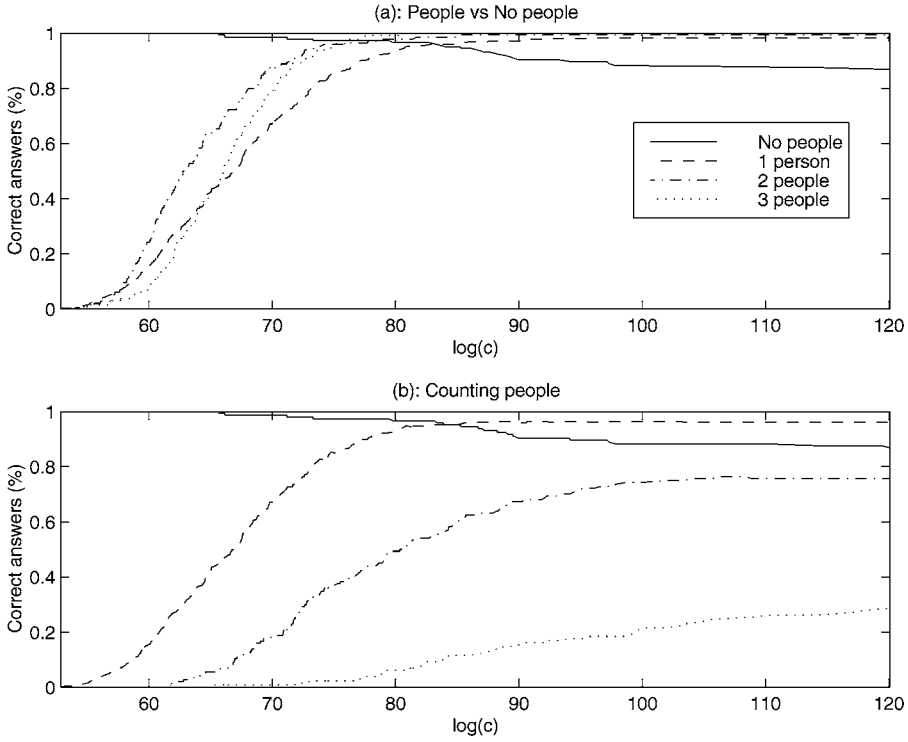

Figure 14. Percentage of correct decisions for Person vs No person classification (a) and Counting (b), as a function of the parameter c. Each figure shows the percentages separately for images with 0, 1, 2, and 3 people. We believe that the decrease in the count accuracy as the number of people goes up is due to the segment finder, which fails to extract all the relevant segments.

with that torso, and then choosing the one with the highest likelihood. We can reduce the number of representatives that need to be computed if we know a lower bound on the likelihood of a person-like assembly. In that case, we do not need to sample for torso segments for which the *upper bound* $\max_{A' \supset A} L(A')$, computed in Section 4.3.4, is too low.

5.2.3. People vs No People. We used sampling and representative selection to count people, as in Section 5.3.2. For each image, we found the MAP subset $\{A_i \mid i \in G\}$ of representatives classified as people, and classify the image as containing a person if $|G| \geq 1$, and no people if $G = \emptyset$. Figure 14(a) shows how the success of this classification depends on the value of c, from Eq. (2).

5.3. Counting People

Our sampling algorithm makes it possible to count people in images. For efficiency, we reduce the set of samples drawn from the likelihood by selecting a small set of *representative assemblies* in the image, so that for each sample assembly there is a representative such that the torsos of the assembly and the representative overlap (Section 5.3.1). The representatives are then used for counting (Section 5.3.2).

5.3.1. Finding Representative Assemblies. We assume that distinct people have distinct torsos, accepting that occlusion of one torso by another will lead to a miscount. We break the set of all assemblies in the image into (not necessarily disjoint) *blocks*—sets of assemblies such that any two assemblies from the same block have overlapping `torsos`. Then, the *representative* is chosen from each block as the assembly with the highest likelihood, over all assemblies available from the block. Because we have assumed that any people in the image are spaced apart, we can use representatives to count people—by replacing the set of assemblies with that of representatives, we do not diminish the count. Indeed, any assembly that is not a representative must be overlapped by a higher-likelihood representative, and so if there was a human assembly in some region of the

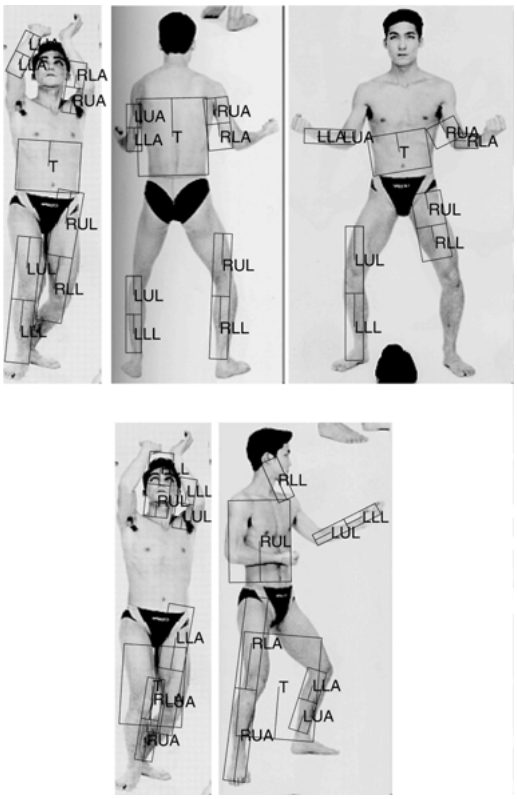

Figure 15. Examples of representative assemblies found for images of people. Each representative assembly is the highest-likelihood sample from a set of samples with overlapping torsos. We use representatives to count people and argue that using representatives does not change the count. Often (top) representatives can also be used to infer configurations of people, although (bottom) that is not always the case.

image, there will be a representative there as well. In fact, the configuration of a person can often be inferred from that of representatives (Fig. 15).

We can efficiently find representatives, since we can use the upper bounds on the likelihoods, computed in Section 5.2.2. In particular, if the algorithm of Section 5.2.2 produced a valid assembly (no coinciding segments) for some `torso` segment, then sampling need not be performed for that `torso` (since this assembly has a higher likelihood than any other we can obtain by sampling). If, however, the assembly obtained for the upper bound is not a valid one, we have to sample assemblies with the given `torso`, but retain only the one with the highest likelihood (since all of the assemblies share the `torso`). Furthermore, we need not sample for a given `torso` segment if there is already an overlapping assembly, whose likelihood is greater than the upper bound for the given `torso`.

5.3.2. Estimating the Number of People. Once the representative set has been computed for an image, we want to obtain the estimate on the number of people in the image. We assume that assemblies corresponding to people do not overlap and have independent configurations. Let the set of representatives be $\{A_1, \ldots, A_m\}$, and let us consider any set $G \subseteq \{1 \ldots m\}$, such that no assemblies from $\{A_i \mid i \in G\}$ overlap. We will look at the *posterior probability* pr[each of A_i represents a person|image data] that the representatives $\{A_i \mid i \in G\}$ are people while $\{A_j \mid j \notin G\}$ are not. To count people, we choose the set G for which the posterior is largest, and the size $|G|$ will give the MAP estimate of the number of people in the image. We could also represent this posterior as a set of samples to give some insight into the reliability of a particular count.

We assume that each assembly has the *a priori* probability β of being a person, independently of the others. Then, the prior for G is

$$\pi(G) = \beta^{|G|}(1-\beta)^{m-|G|},$$

and the posterior is proportional to

$$\Pr[A_1, \ldots, A_m \mid G]\pi(G).$$

Since the human assemblies do not overlap, the posterior $\Pr[A_1, \ldots, A_m \mid G] = 0$ if some of $\{A_i \mid i \in G\}$ overlap. Otherwise, we have

$$\Pr[A_1, \ldots, A_m|G] = \prod_{i \in G} L(A_i) \prod_{i \notin G} L_{\text{non}}(A_i),$$

where we still use

$$L(A) = \Pr[\text{person in random configuration looks like } A],$$

and define

$$L_{\text{non}}(A) = \Pr[A \mid \text{random view not containing a person}].$$

Finally, we assume $L_{\text{non}}(\cdot)$ to be uniform. We get that, for non-overlapping $\{A_i \mid i \in G\}$, the posterior is proportional to

$$L_{\text{non}}^{m-|G|} \prod_{i \in G} L(A_i)\beta^{|G|}(1-\beta)^{m-|G|} \propto c^{|G|} \prod_{i \in G} L(A_i), \quad (2)$$

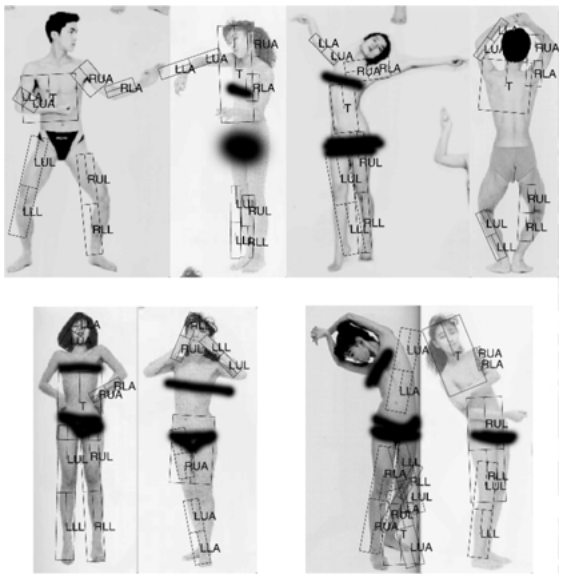

Figure 16. Examples showing representatives for images with two people; these representatives give quite a good guide to the person's configuration (top row); the bottom row shows some cases where the configurations are not represented correctly. Images have been airbrushed so they can be shown salve pudore.

where the constant $c = \frac{\beta}{(1-\beta)L_{\text{non}}}$ is to be estimated so as to yield best classification.

5.3.3. Counting Results. We used the size $|G|$ of the MAP set G as the estimate of the number of people. Figure 14(b) shows, for images with $k = 0 \ldots 3$ people, the fraction of segment sets that yielded the correct estimate $|G| = k$.

The 3-person images did not yield as good results as those with fewer people. We believe this to be due to the fact that with more people in the image the EM-based segment finder is more likely to miss a body segment. Indeed, when looking for the MAP estimate of the number of people in the image, we select the largest non-overlapping set of assemblies with likelihoods greater than $1/c$, where c is the constant used in Eq. (2).

If all relevant segments were found regardless of the number of people in the image, then, if a person was found in each of 3 pictures, 3 people would be found in a collage of those 3 images. This, however, is not the case: as shown in Figure 14(b), the accuracy of the estimated count for images with 3 people is much worse than that for images with 1 or 2 people. Upon inspection, we have found that in 3-person images, more often than in the images with fewer people, some of the body parts were not found by the segment finder (perhaps due to the EM algorithm used for detection, which searches for a fixed number of segments and is prone to getting stuck in local extrema). This leads us to suspect that a large proportion of false negatives are due to missing segments. A better segment finder that doesn't search for a fixed number of segments may be the solution (see Section 6).

For many cases, the representatives give quite a good indication of the configuration of the people present (Fig. 16), but there is no guarantee that the highest-likelihood assembly used as a representative does in fact correspond to the true configuration of the person. We have shown, however, that representatives yield good counting results, because a high-likelihood representative is usually found for a person, even if its configuration is incorrect.

6. Discussion

Finding people in images is a very difficult problem, and we do not know of any system capable of finding clothed people in arbitrary configurations in static images (except (Felzenszwalb and Huttenlocher, 2000), discussed in Section 1, who assume a very constrained appearance model for body parts and do not perform discrimination). Most tracking work relies on motion to identify the foreground, and requires the user to hand-label the configuration of a person in the first frame.

This paper is an attempt to address the problem of finding people in a bottom-up fashion, whereby we first extract possible body parts, then use kinematic constraints to group them into human assemblies. We have shown that, even for unreliable body-part detectors, the kinematic constraints on relationships between the parts significantly increase our ability to discriminate people from non-people. A better segment finder would improve the performance even further by both increasing the detection rate and eliminating some false detections such as that shown in Fig. 17.

The main contribution of this work is the framework that allows candidate body parts to be grouped efficiently. We do so by *pruning* the search: assemblies of increasing size are built incrementally, and segment groups that are unlikely to be a part of a human configuration are rejected at an early stage.

As we demonstrated in Section 5, the performance of the system is promising. As we argued in Sections 5.1.2 and 5.3.3, many of the false negatives are due to the

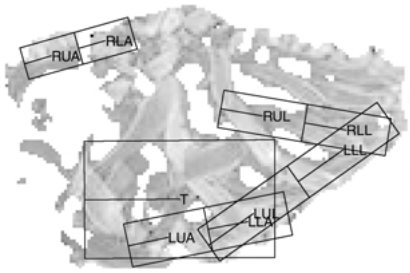

Figure 17. A negative image for which a human assembly was found. The assembly indeed looks like a configuration of a person. A better segment finder would not produce these segments and thus a person would not be detected. The white regions in the image are the pixels that have been masked out because they could not belong to a person due to their color.

segment finder which failed to identify some of the body parts of the people present in the image. This suggests that, while kinematic constraints on relationships among body parts are effective in discriminating people from non-people and can be exploited efficiently using pruning, the overall performance would benefit from a better body segment model. In particular, we need to be able to:

- Instead of finding people as assemblies of *body parts*, look for groups of *features*, which may or may not correspond to body parts, but must be stable and discriminative.
- Deal with a large number of features of different types (e.g., bars found in Section 2, cylinders found using shading patterns (Haddon and Forsyth, 1997), textured regions corresponding to clothed limbs (Shi and Malik, 1997), edges, regions found by template matching such as faces (Rowley et al., 1998a; Sung and Poggio, 1998), etc.).
- Build assemblies of these features, not requiring all of the features to be present in an assembly, but rather determining automatically when an assembly can be classified as a person or a non-person without adding any more segments.
- Structure the search for assemblies in an opportunistic manner, automatically determining which features would be best to add (in contrast to the current system which adds body parts to an assembly in a fixed order).

Addressing these concerns is a topic of current research. We propose to do so by learning a mixture of trees (Meila and Jordan, 2000), whose components correspond to different subsets of features. By making the mixture components share structure, and because of the conditional independences captured in the tree structure, we can efficiently select the optimal assembly, or sample assemblies from the likelihood.

Acknowledgments

This research was supported by NSF Graduate Research Fellowship to SI and by the Digital Library grant IIS-9817353.

References

Agin, G.J. 1972. Representation and description of curved objects. Ph.D. Thesis, Stanford University, Stanford, CA.

Binford, T.O. 1971. Visual perception by computer. In *Proc. IEEE Conference on Systems and Control*, Miami, FL.

Blake, A. and Isard, M. 1998. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer Verlag, London.

Brady, J.M. and Asada, H. 1984. Smoothed local symmetries and their implementation. *International Journal of Robotics Research*, 3(3):36–61, New York.

Bregler, C. and Malik, J. 1998. Tracking people with twists and exponential maps. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 8–15, Santa Barbara, CA.

Brooks, R.A. 1981. Symbolic reasoning among 3-D models and 2-D images. Ph.D. Thesis, Stanford University, Computer Science Dept. Stanford, CA.

Burl, M.C., Leung, T.K., and Perona. P. 1995. Face localisation via shape statistics. In *Int. Workshop on Automatic Face and Gesture Recognition*.

Cutler, R. and Davis, L.S. 2000. Robust real-time periodic motion detection, analysis and applications. *IEEE T. Pattern Analysis and Machine Intelligence*, 22(8):781–796.

Dempster, A.P., Laird, N.M., and Rubin, D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* (39), pp. 185–197.

Deutscher, J., Blake, A., and Reid, I. 2000. Articulated body motion capture by annealed particle filtering. In *IEEE Conf. on Computer Vision and Pattern Recognition*.

Dickinson, S. Pentland, A.P., and Rosenfeld. A. 2000. 3D shape recovery using distributed aspect matching. *IEEE Trans. Patt. Anal. Mach. Intell.*, 14(2):174–198.

Faugeras, O.D. and Hebert, M. 1986. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52.

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

Forsyth, D.A. and Fleck, M.M. 1997. Body plans. In *IEEE Conf. on Computer Vision and Pattern Recognition*.

Forsyth, D.A. and Fleck, M.M. 1999. Automatic detection of human nudes. *Int. J. Computer Vision*, 32(1):63–77.

Forsyth, D.A., Fleck, M.M., and Bregler, C. 1996. Finding naked people. In *European Conference on Computer Vision*.

Freund, Y. and Schapire, R.E. 1996. Experiments with a new boosting algorithm. In *Machine Learning*—13.

Gavrila, D.M. and Davis, L.S. 1996. 3d model-based tracking of humans in action: A multi-view approach. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 73–80.

Grimson, W.E.L. and Lozano-Pérez, T. 1987. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Patt. Anal. Mach. Intell.*, 9(4):469–482.

Haddon, J. and Forsyth, D.A. 1997. Shading primitives. In *Int. Conf. on Computer Vision*.

Haritaoglu, I., Harwood, D., and Davis, L.S. 2000. W4: Real-time surveillance of people and their activities. *IEEE T. Pattern Analysis and Machine Intelligence*, 22(8):809–830.

Hogg, D. 1983. Model based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5–20.

Huang, C-Y., Camps, O.T., and Kanungo, T. 1997. Object recognition using appearance-based parts and relations. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 877–883.

Huttenlocher, D.P. and Ullman, S. 1987. Object recognition using alignment. In *Proc. Int. Conf. Comp. Vision*, London, U.K. pp. 102–111.

Kanazawa, K., Koller, D., and Russell, S. 1995. Stochastic simulation algorithms for dynamic probabilistic networks. In *Uncertainty in Artificial Intelligence. Proceedings of the Eleventh Conference*.

Leung, T.K., Burl, M.C., and Perona, P. 1995. Finding faces in cluttered scenes using random labelled graph matching. In *Int. Conf. on Computer Vision*.

Liu, F. and Picard, R.W. 1996. Detecting and segmenting periodic motion. Media lab vision and modelling tr-400, MIT, Cambridge, MA.

Meila, M. and Jordan, M. 2000. Learning with mixtures of trees. *submitted Journal of Machine Learning Research*.

Neal, R.M. 1998. Annealed importance sampling. Technical Report no. 9805, University of Toronto.

Nevatia, R. and Binford, T.O. 1977. Description and recognition of complex curved objects. *Artificial Intelligence*, 8(1):77–98.

Niyogi, S.A. and Adelson, E.H. 1995. Analyzing and recognizing walking figures in xyt. Media lab vision and modelling tr-223, MIT, Cambridge, MA.

Oren, M., Papageorgiou, C., Sinha, P., and Osuna, E. 1997. Pedestrian detection using wavelet templates. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 193–199.

O'Rourke, J. and Badler, N. 1980. Model-based image analysis of human motion using constraint propagation. *IEEE T. Pattern Analysis and Machine Intelligence*, 2(6):522–546.

Poggio, T. and Sung, K.-K. 1995. Finding human faces with a gaussian mixture distribution-based face model. In *Asian Conf. on Computer Vision*, pp. 435–440.

Rehg, J. and Kanade, T. 1994. Visual tracking of high dof articulated structures: An application to human hand tracking. In *European Conference on Computer Vision*, pp. 35–46.

Rohr, K. 1993. Incremental recognition of pedestrians from image sequences. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 9–13.

Rowley, H.A., Baluja, S., and Kanade, T. 1996a. Human face detection in visual scenes. In Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E. (Eds.). *Advances in Neural Information Processing*, 8:875–881, MIT Press: Cambridge, MA, USA.

Rowley, H.A., Baluja, S., and Kanade, T. 1996b. Neural network-based face detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 203–208.

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

Rowley, H.A., Baluja, S., and Kanade, T. 1998b. Rotation invariant neural network-based face detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 38–44.

Shi, J. and Malik, J. 1997. Normalised cuts and image segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 731–737.

Shuppan, E. *Pose File*, 1993–1996. Vol. 1–7. Books Nippan. A collection of photographs of human models, annotated in Japanese, Japan.

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

Thompson, D.W. and Mundy, J.L. 1987. Three-dimensional model matching from an unconstrained viewpoint. In *IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp. 208–220.

Ullman, S. 1996. *High-level Vision: Object Recognition and Visual Cognition*. MIT Press: Cambridge, MA, USA.

Ulupinar, F. and Nevatia, R. 1988. Using symmetries for analysis of shape from contour. In *Proc. Int. Conf. Comp. Vision*, Tampa, FL, pp. 414–426.

Vapnik, V.N. 1996. *The Nature of Statistical Learning Theory*. Springer Verlag.

Wren, C.R., Azarbayejani, A., Darrell, T., and Pentland, A.P. 1997. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785.

Zerroug, M. and Nevatia, R. 1999. Part-based 3d descriptions of complex objects from a single image. *PAMI*, 21(9):835–848.