To appear in Early Visual Learning, edited by Tomaso Poggio and Shree Nayar.
January 1995

1 Learning Object Recognition Models from Images

Arthur R. Pope and *David G. Lowe*
University of British Columbia

Abstract

To recognize an object in an image one must have an internal model of how that object may appear. We describe a method for learning such models from training images. An object is modeled by a probability distribution describing the range of possible variation in the object's appearance. This distribution is organized on two levels. Large variations are handled by partitioning the training images into clusters that correspond to distinctly different views of the object. Within each cluster, smaller variations are represented by distributions that characterize the presence, position, and measurements of various discrete features of appearance. The learning process combines an incremental conceptual clustering algorithm for forming the clusters with a generalization algorithm for consolidating each cluster's training images into a single description. Recognition employs information about feature positions, numeric measurements, and relations in order to constrain and speed the search. Preliminary experiments have been conducted with a system that implements some aspects of the method; the system can learn to recognize a single characteristic view of an object in the presence of occlusion and clutter.

1 Introduction

To recognize an object in an image one must have some expectation of how the object may appear. That expectation is based on an internal model of the object's form or appearance. We are investigating how a system might acquire such models directly from intensity images, and then use those models to recognize the objects in other images.

The following scenario illustrates how this recognition learning system would operate. One presents to the system a series of example images that depict a particular object from various viewpoints. From those examples the system develops a model of the object's appearance. This training is repeated for each of the objects the system is to recognize. When given a test image, the system can then identify and rank apparent instances of the known objects in the image.

Few object recognition systems have been designed to acquire their models directly from intensity images. Instead, most systems are simply given models in the form of manually-produced shape descriptions. During recognition, the object's shape description must be combined with a model of the image formation process in order to determine whether the object might have a particular appearance. Successful recognition depends on having good models not only of the objects, but also of the scene illumination, surface reflectance, optical projection, and the sensor. Because modeling image formation has proven difficult, most object recognition systems that follow this approach have been restricted to object models that are relatively simple and coarse.

A system that can learn its models directly from images, on the other hand, may enjoy important advantages:

- The learning system will avoid the problem of having to estimate the actual appearance of an object from some idealized model of its shape. Instead, all of the properties of the object's appearance needed for recognition will be learned directly by observation. Eliminating inaccuracies in appearance estimation should allow recognition to be accomplished more robustly.
- The learning system will acquire new models more conveniently. A new model will be defined not by measuring and encoding its shape, as with traditional object recognition systems, but by merely displaying the object to the system in various poses.
- The learning system could endow a robot with the ability to learn objects as they are encountered for later recognition. This ability would be important in a dynamic, unknown environment.

The difficulty of the recognition learning problem is largely due to the fact that an object's appearance has a large range of variation. It varies with changes in camera position and lighting and, if the object is flexible, with changes in shape. Further variation is due to noise and to differences among individual instances of the same type of object. Accommodating this variation is a central problem in the design of a recognition learning system. The scheme used to describe image content must be stable so that small changes in an appearance induce only small changes in its description. The scheme used to model an object's appearance must describe just what variations are possible. The learning procedure must generalize enough to overcome insignificant variation, but not so much as to confuse dissimilar objects. And the procedure used to identify modeled objects in images must tolerate the likely range of mismatch between model and image.

In investigating the recognition learning problem we have concentrated on one particular version of it: learning to recognize 3-D objects in 2-D intensity images. Objects are recognized solely by the intensity edges they exhibit (although nothing about the approach precludes an extension to other properties, such as color and texture). As for the objects themselves, they may be entirely rigid, possess a small number of articulations, or be somewhat flexible. The system is able to learn to recognize a class of similar objects, accommodating the variation among objects just as it accommodates the variation among images of a single object.

Our method models an object by a probability distribution that describes the range of possible variation in the object's appearance. This probability distribution is organized on two levels. Large variations are handled by partitioning the training images into clusters that correspond to distinctly different views or configurations of the object. Within each cluster, smaller variations are represented by probability distributions characterizing various appearance features. For each feature we represent the probability of detecting that feature in various positions with respect to the overall object, and the probability of the feature having various values of feature-specific, numeric measurements. A rich, partially-redundant, and extensible repertoire of features is used to describe appearance.

The learning method combines an incremental conceptual clustering algorithm for forming the clusters with a generalization algorithm for consolidating each cluster's training images into a single description. Recognition, which involves matching the features of a training image with those of a cluster's description, can employ information about feature positions, numeric measurements, and relations in order to constrain and speed the search for a match. Preliminary experiments have been conducted with a system that implements some aspects of the method; that system can learn to recognize a single view of an object among other occluding and distracting objects.

We have just described the problem being considered, its significance, the source of its difficulty, and the outline of a solution method. The next section begins a detailed description of the method by discussing the representations used for images and models. The process of finding a match between a model and an image is guided by a match quality measure, which is the subject of section 3. This measure supports both the matching procedure described in section 4, and the procedure for learning models described in section 5. Section 6 presents experimental results from a system implemented to test the approach. Section 7 discusses relevant work by others on this and similar problems, and section 8 summarizes the chapter's main ideas. Sections flagged by † contain technical details that can be safely skipped on a first reading. More information may be found in other recent publications [18, 19, 20].

2 Representation Schemes

2.1 Image representation

We represent an image in terms of discrete properties called *features*. Each feature has a particular type, a location within the image, and a vector of numeric *attributes* that further characterize it. A feature may, for example, be a segment of intensity edge, a particular arrangement of such segments, or a region of uniform texture or color. Low-level features may be found as responses to feature detectors, such as edge or corner detectors; other, higher-level features may be found by grouping or abstracting the low-level ones. Numerous features of various types describe a typical image.

What attributes a feature has depends on its type. A junction of two circular arcs, for example, may have one attribute for the junction's angle, and another for the ratio of the two arcs' radii. Attributes are expressed so that they are invariant with respect to translation, rotation, and scaling of the feature within the image (using, for example, scale-normalized measures [23]).

The repertoire of feature types must be sufficient to provide a rich description of any relevant image. A degree of redundancy is desirable, for it helps to ensure the completeness of the representation and it contributes stability. Good features are those that can be detected reliably, and are relatively invariant with respect to modest changes in viewpoint or lighting. For efficiency in recognition, it is useful to have some highly-selective features that usually occur only in the presence of certain objects. In recognizing manufactured objects, for example, features denoting various geometric arrangements of intensity edges may serve this role well. Some more commonplace features, such as simple line and curve segments, should supplement the highly-selective ones so that the overall repertoire can still describe a wide variety of objects, at least at a basic level. Of course, distinctions among objects can only be made if the repertoire includes features that express those distinctions.

Apart from these requirements, the recognition learning method is not particular about what features are used or what their attributes are. As any feature is bound to be unreliable in certain situations, the method attempts to compensate for feature shortcomings by learning how reliable various features are for recognizing each object.

The collection of features found in an image is represented by an *image graph.* Graph nodes represent features; directed arcs represent grouping and abstraction relations among them. Formally, an image graph G is denoted by a tuple $\langle F, R\rangle$, where F is a set of image features and R is a relation over elements of F. An image feature $f_k \in F$ is a tuple $\langle t_k, \mathbf{a}_k, \mathbf{b}_k, \mathbf{C}_k\rangle$, where t_k is the feature's type, $\mathbf{a}_k$ is a vector of attributes describing the feature, $\mathbf{b}_k$ is its measured position, and $\mathbf{C}_k$ is a covariance matrix describing the uncertainty in that position. The domain of a feature's attribute vector depends on the feature's type. Section 2.3, below, describes how positions such as $\mathbf{b}_k$ are represented. Finally, an element of R is a tuple $\langle k, l_1, \ldots, l_m\rangle$, indicating that image feature k was found by grouping or abstracting image features l_1 through l_m.

2.2 Model representation

A model is organized on two levels in order to fully and accurately describe the range of possible variation in its object's appearance. Significant variations in appearance are handled by subdividing the model into a set of characteristic views, each independent of the others. Smaller variations are handled within each characteristic view by allowing the view to represent a probability distribution over a range of similar image graphs.

Because this explicitly represents how an object appears from various, discrete viewpoints, it is called a *viewer-centered*, or *multi-view*, representation. To learn a viewer-centered model, it is not necessary to recover the object's 3-D structure. Another common approach is the *object-centered* representation, which instead explicitly represents the 3-D geometry of the object. To learn an object-centered model, however, one has to recover the 3-D location of each feature. Recovery is difficult because the viewpoint of each training image is unknown, there is uncertainty in the measurement of each feature's image location, and there may be errors in matching features among images.

In our method, each characteristic view describes a range of possible appearances by defining a joint probability distribution over image graphs. Because the space of image graphs is enormous, however, it is not practical to represent or learn this distribution in its most general form. Instead, the joint distribution is approximated by treating its component features as though they were independent. This approximation allows the joint distribution to be decomposed into a product of marginal distributions, thus greatly simplifying the representation, matching, and learning of models.

One consequence of the simplification is that statistical dependence (association or covariance) among model features cannot be accurately represented within a single characteristic view. An extreme example of such dependence is an object with two subsets of features such that only one subset appears in any one image. Because of the simplification, this object with its strongly covariant features would be poorly represented by a single characteristic view. However, where one characteristic view cannot capture an important statistical dependence, multiple views can. In this example, two characteristic views, each containing one of the two subsets of features, could represent perfectly the statistical dependence among features.

By using a large enough set of characteristic views we can model any object as accurately as we might wish. For the sake of efficiency, however, we would prefer to use relatively few views and let each represent a moderate range of possible appearances. One challenge for our model learning method is to strike an appropriate balance between the number of characteristic views used and the accuracy of those views over their respective ranges. This issue will be revisited when we discuss the model learning procedure in section 5.

A single characteristic view is described by a *model graph*. Like an image graph, a model graph has nodes that represent features and arcs that represent composition and abstraction relations among features. Each node records the information needed to estimate three probabilities:

- *The probability of observing this feature in an image depicting the characteristic view of the object.* The node records the number of times the feature has been identified in training images. A count is also kept of the training images used to learn the overall model graph. The probability is estimated from these two numbers as described in section 3.1.

- *Given that this feature is observed, the probability of it having particular attribute values.* This is characterized by a probability distribution over vectors of attribute values. Little can be assumed about the form of this distribution because it may depend on many factors: the type of feature, how its attributes are measured, possible deformations of the object, and various sources of measurement uncertainty. Thus we use a non-parametric density estimator that makes relatively few assumptions. To support this estimator, which is described in section 3.3, the model graph node records sample attribute vectors acquired from training images.

- *Given that this feature is observed, the probability of it having a particular position.* This is characterized by a probability distribution over feature positions. We approximate this distribution as Gaussian to allow use of an efficient matching procedure based on least-squares estimation. The parameters of the distribution are estimated from sample feature positions acquired from training images.

Formally, a model graph $\bar{G}$ is denoted by a tuple $\langle \bar{F}, \bar{R}, \bar{m} \rangle$, where $\bar{F}$ is a set of model features, $\bar{R}$ is a relation over elements of $\bar{F}$, and $\bar{m}$ is the number of training images used to produce $\bar{G}$. A model feature $\bar{f}_j \in \bar{F}$ is a tuple $\langle \bar{t}_j, \bar{m}_j, \bar{A}_j, \bar{B}_j \rangle$, where $\bar{t}_j$ is the feature's type, $\bar{m}_j$ is the number of training images in which the feature was observed, and $\bar{A}_j$ and $\bar{B}_j$ are sequences of attribute vectors and positions drawn from those training images. Finally, an element of $\bar{R}$ is a tuple $\langle j, l_1, \ldots, l_m \rangle$, indicating that model feature j is a grouping or abstraction of model features l_1 through l_m.

2.3 Coordinate systems

A feature's position is expressed in terms of a 2-D, Cartesian coordinate system by a location, orientation, and scale. Image features are located in an *image coordinate system* identified with pixel rows and columns. Model features are located in a *model coordinate system* shared by all features within a model graph. The absolute positions of these coordinate systems are not important as they are used only to measure features' relative positions.

Two different schemes are used to describe a feature's position in either coordinate system:

$xy\theta s$ The feature's location is specified by $[x\ y]$, its orientation by θ, and its scale by s.

$xyuv$ The feature's location is specified by $[x\ y]$, and its orientation and scale are represented by the orientation and length of the 2-D vector $[u\ v]$.

We will prefer the $xy\theta s$ scheme for measuring feature positions, and the $xyuv$ scheme for estimating viewpoint in the course of matching a model with an image. They are related by $\theta = tan^{-1}(v/u)$ and $s = \sqrt{u^2 + v^2}$. Where it is not otherwise clear we will indicate schemes using the superscripts ${}^{xy\theta s}$ and ^{xyuv}.

Part of the task of matching a model with an image is to determine a *viewpoint transformation* that brings the image and model features into close correspondence. In our case, this viewpoint transformation is a 2-D similarity transformation. The $xyuv$ scheme allows such a transformation to be expressed as a linear operation with the advantage that it can then be estimated from a set of feature pairings by solving a system of linear equations.[1]

We take the viewpoint transformation, T, to be from image to model coordinates, and use it to transform the position of an image feature before comparing it with that of a model feature. The result of applying T to the position $\mathbf{b}_k$ is denoted $T(\mathbf{b}_k)$.

A transformation consisting of a rotation by θ_t, a scaling by s_t, and a translation by $[x_t\, y_t]$, in that order, can be expressed in two ways as a matrix operation. We will have occasion to use both. In one case, a matrix $\mathbf{A}_k$ represents the position $\mathbf{b}_k = [x_k\, y_k\, u_k\, v_k]$ being transformed:

$$\mathbf{b}'_k = \begin{bmatrix} x'_k \\ y'_k \\ u'_k \\ v'_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_k & -y_k \\ 0 & 1 & y_k & x_k \\ 0 & 0 & u_k & -v_k \\ 0 & 0 & v_k & u_k \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \end{bmatrix} = \mathbf{A}_k \mathbf{b}_t. \qquad (1)$$

In the other case, a matrix $\mathbf{A}_t$ represents the rotation and scaling components of the transformation:

$$\mathbf{b}'_k = \begin{bmatrix} x'_k \\ y'_k \\ u'_k \\ v'_k \end{bmatrix} = \begin{bmatrix} u_t & -v_t & 0 & 0 \\ v_t & u_t & 0 & 0 \\ 0 & 0 & u_t & -v_t \\ 0 & 0 & v_t & u_t \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ u_k \\ v_k \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \\ 0 \\ 0 \end{bmatrix} = \mathbf{A}_t \mathbf{b}_k + \mathbf{x}_t. \qquad (2)$$

These linear formulations allow a transformation to be estimated easily from a set of feature pairings. Given a model feature at $\mathbf{b}_j$ and an image feature at $\mathbf{b}_k$, the transformation aligning the two features can be obtained as the solution to the system of linear equations $\mathbf{b}_j = T(\mathbf{b}_k)$. With additional feature pairings, the problem of estimating the transformation becomes over-constrained; then the solution that is optimal in the least-squares sense can be found by least-squares estimation, as described in section 4.2.

3 Match Quality Measure

Recognition requires finding a consistent set of pairings between some model features and some image features, plus a viewpoint transformation that brings the

[1]Ayache and Faugeras [1], among others, have also used this formulation to express the transformation as a linear operation.

paired features into close correspondence. Together, the pairings and transformation are called a *match*. The pairings will often be incomplete, with some image features not explained by the model (perhaps there are other objects in the scene) and some model features not found in the image (perhaps due to shadows or occlusion). Nevertheless, the desired match should be a good one that jointly maximizes both the number of features paired and the resemblance between paired features. We use a *match quality measure* to evaluate these qualities.

The match quality measure considers what features are paired, how significant those features are, how similar their attribute values are, and how well their positions correspond. Each factor is evaluated according to past matching experience as recorded by the model. The factors are combined using Bayesian theory to estimate the probability that a particular match represents a true occurrence of the object in the image.

A set of pairings is represented by the tuple $E = \langle e_1, e_2, \ldots \rangle$, where $e_j = k$ if model feature j is paired with image feature k, and $e_j = -$ if it is not paired. The hypothesis that the object is present in the image is denoted by H. Match quality is associated with the probability that this hypothesis is correct given a set of pairings and a viewpoint transformation. Bayes theorem allows us to write this probability as:

$$\mathrm{P}(H \mid E, T) = \frac{\mathrm{P}(E \mid T, H)\, \mathrm{P}(T \mid H)}{\mathrm{P}(E \wedge T)} \mathrm{P}(H). \tag{3}$$

There is no practical way to represent the high-dimensional, joint probability distributions $\mathrm{P}(E \mid T, H)$ and $\mathrm{P}(E \wedge T)$ in their most general form. Instead, we approximate them using the feature independence simplification discussed previously in section 2.2. This reduces equation 3 to a product of marginal probability distributions.

$$\mathrm{P}(H \mid E, T) \approx \prod_j \frac{\mathrm{P}(e_j \mid T, H)}{\mathrm{P}(e_j)} \frac{\mathrm{P}(T \mid H)}{\mathrm{P}(T)} \mathrm{P}(H). \tag{4}$$

The approximation is a perfect one when two independence properties hold:

(a) $\{e_i\}$ is collectively independent given knowledge of T and H, and

(b) $\{e_i, T\}$ is collectively independent in the absence of any knowledge of H.

In practice we can expect these properties to hold at least somewhat. Given that an object is present at a particular pose, features detected at widely separate locations on the object will be independently affected by occlusion and noise; these features satisfy property (a). And in a random scene cluttered with unknown objects, even nearby features may be largely independent because they could come from any of numerous objects; these features satisfy property (b).

On the other hand, the independence properties fail to the extent that there is redundancy among features. For example, a feature representing a perceptually-significant grouping is not independent of the features it groups; in this case, equation 4 may overstate the significance of pairing these features because it separately counts both the individual features and the feature that groups them. With redundancy uniformly present among all model graphs, however, the resulting bias should have little effect on the outcome of any particular matching problem. Having adopted the hypothesis that this is so, we use equation 4 as the basis for our match quality measure.

The measure is defined using log-probabilities to simplify calculations. Moreover, it is assumed that all positions of a modeled object within an image are equally likely, and thus $\mathrm{P}(T \mid H) = \mathrm{P}(T)$. With these simplifications the match quality measure becomes

$$g(E,T) = \log \mathrm{P}(H) + \sum_j \log \mathrm{P}(e_j \mid T, H) - \sum_j \log \mathrm{P}(e_j). \tag{5}$$

$\mathrm{P}(H)$ is the prior probability that the object, as modeled, is present in an image; it can be estimated from the proportion of training images that matched the model and were used to create it. Estimates of the conditional and prior probabilities of individual feature pairings, $\mathrm{P}(e_j \mid T, H)$ and $\mathrm{P}(e_j)$, will be described in the next two sections. We will use the following notation for specific random events within the universe of matching outcomes:

$\tilde{e}_j = k$	model feature j is paired with image feature k
$\tilde{e}_j = -$	model feature j is paired with nothing
$\tilde{\mathbf{a}}_j = \mathbf{a}$	model feature j is paired with a feature whose attributes are $\mathbf{a}$
$\tilde{\mathbf{b}}_j = \mathbf{b}$	model feature j is paired with a feature at position $\mathbf{b}$

3.1 Conditional probability of a feature pairing

There are two cases to consider in estimating $\mathrm{P}(e_j \mid T, H)$, the conditional probability of a pairing involving model feature j.

1. When j is not paired, this probability is estimated by considering how often j failed to be paired with an image feature during training. We use a Bayesian estimator with a uniform prior, and the $\bar{m}$ and $\bar{m}_j$ statistics recorded by the model:

$$\mathrm{P}(\tilde{e}_j = - \mid T, H) = 1 - \mathrm{P}(\tilde{e}_j \neq - \mid T, H) \approx 1 - \frac{\bar{m}_j + 1}{\bar{m} + 2}. \tag{6}$$

2. When j is paired with image feature k, this probability is estimated by considering how often j was paired with image features during training, and how the attributes and position of k compare with those of the training features:

$$\begin{aligned} \mathrm{P}(\tilde{e}_j = k \mid T, H) &= \mathrm{P}(\tilde{e}_j \neq - \wedge \tilde{\mathbf{a}}_j = \mathbf{a}_k \wedge \tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid T, H) \\ &\approx \mathrm{P}(\tilde{e}_j \neq - \mid T, H)\, \mathrm{P}(\tilde{\mathbf{a}}_j = \mathbf{a}_k \mid \tilde{e}_j \neq -, H) \\ &\qquad \mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \tilde{e}_j \neq -, T, H). \end{aligned} \tag{7}$$

$\mathrm{P}(\tilde{e}_j \neq - \mid \ldots)$ is estimated as shown in equation 6. $\mathrm{P}(\tilde{\mathbf{a}}_j = \mathbf{a}_k \mid \ldots)$ is estimated using the series of attribute vectors $\bar{A}_j$ recorded with model feature j, and a non-parametric density estimator described in section 3.3. $\mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \ldots)$, the probability that model feature j is paired with an image feature at position $\mathbf{b}_k$ under viewpoint transformation T, is estimated as described in section 3.4, below.[2]

3.2 Prior probability of a feature pairing

Estimates of the prior probabilities, $\mathrm{P}(e_j)$, are based on measurements of a large collection of images typical of those in which the object will be sought. This *milieu collection* is used to estimate "background" probability distributions that characterize features found independently of whether any particular object is present. In other words, these distributions describe what can be expected in the absence of any knowledge of H or T. By an analysis similar to that underlying estimates of the conditional probabilities, we obtain estimates for two cases of e_j.

1. The probability of j remaining unpaired regardless of H and T is

$$\mathrm{P}(\tilde{e}_j = -) = 1 - \mathrm{P}(\tilde{e}_j \neq -).$$

The latter term is estimated from the frequency with which features of j's type, $\tilde{t}_j$, occur in the milieu collection.

2. The probability of j being paired with k regardless of H and T is

$$\begin{aligned}\mathrm{P}(\tilde{e}_j = k) &= \mathrm{P}(\tilde{e}_j \neq - \wedge \tilde{\mathbf{a}}_j = \mathbf{a}_k \wedge \tilde{\mathbf{b}}_j = T(\mathbf{b}_k)) \\ &\approx \mathrm{P}(\tilde{e}_j \neq -)\, \mathrm{P}(\tilde{\mathbf{a}}_j = \mathbf{a}_k \mid \tilde{e}_j \neq -) \\ &\qquad \mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \tilde{e}_j \neq -). \end{aligned} \tag{8}$$

$\mathrm{P}(\tilde{\mathbf{a}}_j = \mathbf{a}_k \mid \ldots)$ is estimated using samples of attribute vectors drawn from the milieu collection, and the density estimator described in section 3.3. $\mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k \mid \ldots)$ is a constant estimated by assuming a uniform distribution of features throughout a bounded region of model coordinate space.

3.3 Probability distribution over feature attributes †

One component of the match quality measure is the probability that a feature may have a particular attribute vector. To help us estimate this probability, we have samples of attribute vectors that have been acquired by observing the feature in training images. The estimation problem is therefore of the following form: given

[2]For simplicity, our notation does not distinguish between probability mass and probability density. $\mathrm{P}(\tilde{e}_j)$ is a mass because $\tilde{e}_j$ assumes discrete values, whereas $\mathrm{P}(\tilde{\mathbf{a}}_j)$ and $\mathrm{P}(\tilde{\mathbf{b}}_j)$ are densities because $\tilde{\mathbf{a}}_j$ and $\tilde{\mathbf{b}}_j$ are continuous. But since equation 4 divides each conditional probability mass by a prior probability mass, and each conditional probability density by a prior probability density, here we can safely neglect the distinction.

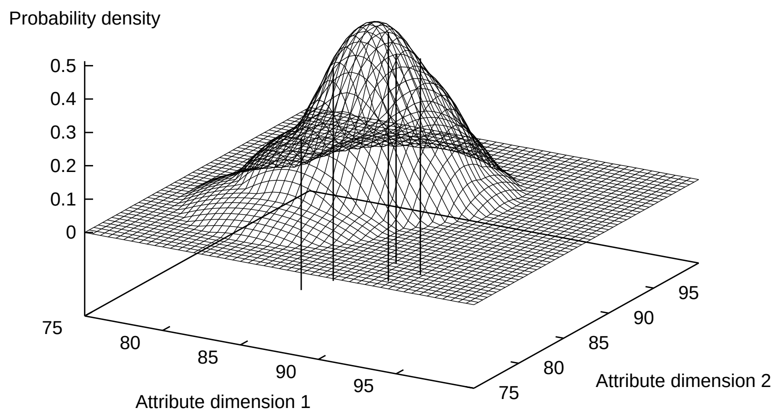

Figure 1: *Example of a locally-adaptive probability density estimate for attribute vectors. The spikes denote the samples from which the density estimate was computed.*

a sequence of d-dimensional observation vectors $\{\mathbf{x}_i : 1 \leq i \leq n\}$ drawn at random from an unknown distribution, estimate the probability that another vector drawn from that same distribution would have the value $\mathbf{x}$.

This could be solved by assuming that the distribution has some parameterized form (e.g., normal), and then estimating its parameters from the observations $\mathbf{x}_i$. However, the attribute vector distributions could be complex as they depend not only on sensor noise and measurement errors, but also on systematic variations in object shape, lighting, and pose. Hence we use a non-parametric estimation method [28]. In its simplest, form, this method estimates probability density by summing contributions from a series of overlapping kernels. The density at $\mathbf{x}$ is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_i K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \tag{9}$$

where h is a constant smoothing factor, and K is a kernel function. We use the Epanechnikov kernel because it has finite support and can be computed quickly. Its definition is

$$K(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1 - \mathbf{x}^{\mathrm{T}}\mathbf{x}) & \text{if } \mathbf{x}^{\mathrm{T}}\mathbf{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where c_d is the volume of a d-dimensional sphere of unit radius. The smoothing factor h appearing in equation 9 strikes a balance between the smoothness of the estimated distribution and its fidelity to the observations $\mathbf{x}_i$.

We can adjust h using a locally-adaptive method: with $\hat{f}$ as a first density estimator, we create a second estimator, $\hat{f}_a$, whose smoothing factor varies according

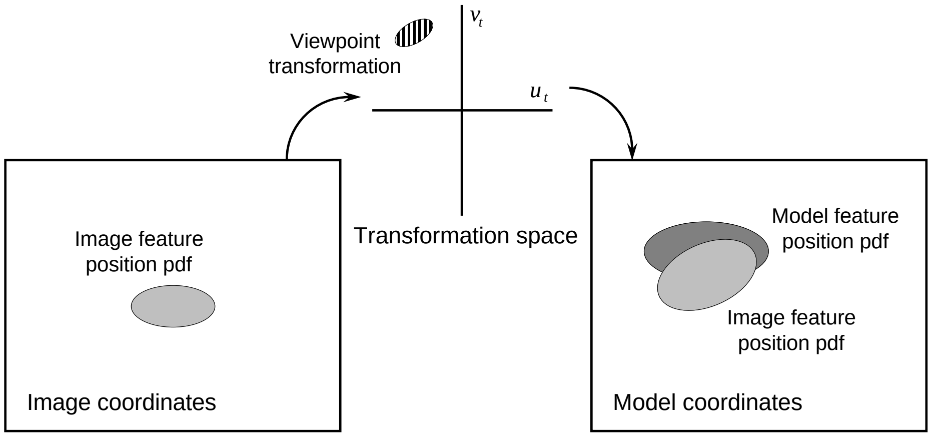

Figure 2: *An image feature's position is transformed from image coordinates (left) to model coordinates (right) according to an estimate of the viewpoint transformation (center). A model feature's position is estimated in model coordinates (right). Uncertainty in the positions and the transformation are characterized by Gaussian distributions. Overlap of the two distributions in model coordinates corresponds to the probability that the two features match given the viewpoint transformation and their respective positions.*

to the first estimator's density estimate:

$$\hat{f}_a(\mathbf{x}) = \frac{1}{nh^d} \sum_i \lambda_i^{-d} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h\lambda_i}\right), \tag{11}$$

$$\text{where } \lambda_i = \left(\frac{\hat{f}(\mathbf{x}_i)}{\nu}\right)^{-\frac{1}{2}} \text{ and } \nu = \left(\prod_i \hat{f}(\mathbf{x}_i)\right)^{\frac{1}{n}}.$$

The various λ_i incorporate the first density estimates at the points $\mathbf{x}_i$, and ν is a normalizing factor. This adaptive estimator smoothes more in low-density regions than in high-density ones. Thus a sparse outlier is thoroughly smoothed while a central peak is accurately represented in the estimate (see figure 1).

3.4 Probability distribution over feature positions †

Another component of the match quality measure is the probability that a model feature is paired with an image feature given the positions of the two features and a viewpoint transformation that somewhat aligns them. This position- and transformation-dependent portion of the match quality measure is represented by $\mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \tilde{e}_j \neq -, T, H)$ in equation 7. To estimate it, we use the viewpoint transformation to map the image feature's position into model coordinates, where we compare it with the model position (see figure 2). The positions and transformation are characterized by Gaussian probability density functions (pdfs), allowing the comparison to take into account the uncertainty in each.

Figure 3: *The Gaussian distribution of an image feature's position in $xy\theta s$ coordinates (left) is approximated by a Gaussian distribution in $xyuv$ coordinates (right), with the parameters of the approximating distribution determined as shown.*

Image feature k's position is conveniently characterized by a Gaussian pdf in $xy\theta s$ image coordinates. Its mean is the feature's position, $\mathbf{b}_k^{xy\theta s}$, as measured in the image. However, because our system's feature detectors and grouping processes do not supply uncertainty estimates for individual features, we define the covariance matrix for this pdf using system parameters:

$$\mathbf{C}_k^{xy\theta s} = \begin{bmatrix} \sigma_l^2 & 0 & 0 & 0 \\ 0 & \sigma_l^2 & 0 & 0 \\ 0 & 0 & (\frac{\sigma_\theta}{s_k})^2 & 0 \\ 0 & 0 & 0 & \sigma_s^2 \end{bmatrix}. \tag{12}$$

The parameters σ_l, σ_θ, and σ_s are our estimates of the standard deviations in measurements of location, orientation, and scale. The orientation variance includes a factor based on the feature scale, s_k, because the orientation of a large feature can usually be measured more accurately than that of a small one.

This Gaussian pdf is then re-expressed in $xyuv$ image coordinates so that the viewpoint transformation can be applied as a linear operation. Unfortunately, a pdf that is Gaussian in $xy\theta s$ coordinates is not necessarily Gaussian in $xyuv$ coordinates. Nevertheless, in this case a good approximating Gaussian can be obtained in $xyuv$ coordinates because the θ and s variances are small. The approximation places the $xyuv$ mean at the same position as the $xy\theta s$ mean, and aligns the Gaussian envelope radially, away from the $[u\,v]$ origin (see figure 3). Its mean and covariance matrix are

$$\mathbf{b}_k^{xyuv} = [x_k \;\; y_k \;\; s_k \cos\theta_k \;\; s_k \sin\theta_k] \tag{13}$$

$$\text{and } \mathbf{C}_k^{xyuv} = \mathbf{R} \begin{bmatrix} \sigma_l^2 & 0 & 0 & 0 \\ 0 & \sigma_l^2 & 0 & 0 \\ 0 & 0 & \sigma_s^2 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 \end{bmatrix} \mathbf{R}^{\mathrm{T}}, \tag{14}$$

$$\text{where } \mathbf{R} \;=\; \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\theta_k & -\sin\theta_k \\ 0 & 0 & \sin\theta_k & \cos\theta_k \end{bmatrix}.$$

The viewpoint transformation is characterized by a Gaussian pdf over $[x_t y_t u_t v_t]$ vectors with mean $\mathbf{t}$ and covariance $\mathbf{C}_t$. (In the course of matching a model with an image, $\mathbf{t}$ and $\mathbf{C}_t$ are estimated as described in section 4.2, below.) We transform the image feature's position from *xyuv* image coordinates to *xyuv* model coordinates using the viewpoint transformation. If we would disregard the uncertainty in the transformation estimate, we would obtain a Gaussian pdf in model coordinates with mean $\mathbf{A}_k\mathbf{t}$ and covariance $\mathbf{A}_t\mathbf{C}_k\mathbf{A}_t^{\mathrm{T}}$. On the other hand, disregarding the uncertainty in the image feature position produces a Gaussian pdf in model coordinates with mean $\mathbf{A}_k\mathbf{t}$ and covariance $\mathbf{A}_k\mathbf{C}_t\mathbf{A}_k^{\mathrm{T}}$. With Gaussian uncertainty in both the image feature position and the transformation, however, the pdf in model coordinates cannot be characterized as Gaussian. At best we can approximate it as Gaussian, which we do with a mean and covariance given in *xyuv* coordinates by

$$\mathbf{b}_{kt} \;=\; \mathbf{A}_k\mathbf{t} \tag{15}$$

$$\text{and } \mathbf{C}_{kt} \;\approx\; \mathbf{A}_t\mathbf{C}_k\mathbf{A}_t^{\mathrm{T}} + \mathbf{A}_k\mathbf{C}_t\mathbf{A}_k^{\mathrm{T}}. \tag{16}$$

The position of model feature j is also characterized by a Gaussian pdf in *xyuv* model coordinates. Its mean $\mathbf{b}_j$ and covariance $\mathbf{C}_j$ are estimated from the series of position vectors $\bar{B}_j$ recorded by the model.[3]

We can now estimate the probability that j is paired with k according to their position pdfs in *xyuv* model coordinates. The estimate is obtained by integrating over all positions $\mathbf{r}$ the probability that both the image feature is at $\mathbf{r}$ and the model feature matches something at $\mathbf{r}$:

$$\mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \tilde{e}_j \neq -, T, H) \approx \int_{\mathbf{r}} \mathrm{P}(\tilde{\mathbf{r}}_j = \mathbf{r})\, \mathrm{P}(\tilde{\mathbf{r}}_{kt} = \mathbf{r})\, \mathrm{d}\mathbf{r}. \tag{17}$$

Here $\mathbf{r}$ ranges over *xyuv* model coordinates while $\tilde{\mathbf{r}}_j$ and $\tilde{\mathbf{r}}_{kt}$ are random variables drawn from the Gaussian distributions $\mathrm{N}(\mathbf{b}_j, \mathbf{C}_j)$ and $\mathrm{N}(\mathbf{b}_{kt}, \mathbf{C}_{kt})$. It would be costly to evaluate this integral by sampling at various $\mathbf{r}$. Fortunately, however, the integral can be rewritten as a Gaussian in $\mathbf{b}_j - \mathbf{b}_{kt}$, as can be seen from the fact that it is essentially a convolution of two Gaussians. Thus it is equivalent to

$$\mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \tilde{e}_j \neq -, T, H) \approx G(\mathbf{b}_j - \mathbf{b}_{kt}, \mathbf{C}_j + \mathbf{C}_{kt}) \tag{18}$$

where $G(\mathbf{x}, \mathbf{C})$ is a Gaussian with zero mean and covariance $\mathbf{C}$. In this form, the desired probability estimate is easily computed.

[3]Two practical considerations enter into the estimation of $\mathbf{C}_j$. First, when $\bar{B}_j$ contains too few samples for a reliable estimate of $\mathbf{C}_j$, the estimate that $\bar{B}_j$ yields is blended with another determined by system parameters. Second, minimum variances are imposed on $\mathbf{C}_j$ in case some dimension of $\bar{B}_j$ has zero variance.

4 Matching Procedure

Both recognition and learning require that we find a match between a model graph and an image graph—one that maximizes the match quality measure defined in section 3. It does not seem possible to find an optimal match through anything less than exhaustive search. In practice, however, good matches can be found quickly by *iterative alignment* [1, 13, 15]. This process hypothesizes some initial pairings between model and image features, uses those pairings to estimate the viewpoint transformation, uses the transformation estimate to evaluate and choose additional pairings, refines the transformation estimate using the additional pairings, and so on until as many features a possible have been matched.

In our version of the iterative alignment method, we explicitly represent the uncertainty in the position of each feature and the resulting uncertainty in the transformation estimate. Thus features that are well-localized contribute most to the transformation estimate, and those whose positions vary most are sought over the largest image neighborhoods. This version of iterative alignment is called *probabilistic alignment* to emphasize its basis in probability theory. It uses feature uncertainty information that has been acquired from training images and recorded in the model.

4.1 Probabilistic alignment

To choose the initial pairings, possible pairings of higher-level features are rated according to the contribution each would make to the match quality measure. The pairing $\langle j, k\rangle$ receives the rating

$$g_j(k) = \max_T \log \mathrm{P}(\tilde{e}_j = k \mid T, H) - \log \mathrm{P}(\tilde{e}_j = k). \tag{19}$$

This rating favors pairings in which the model feature has a high likelihood of matching, the two features have similar attribute values, and the resulting transformation estimate's variance would be small. Moreover, because the component of $\mathrm{P}(\tilde{e}_j = k \mid T, H)$ that depends on T is Gaussian, its maximum over T can be computed readily.

A search is begun from each of the several highest-ranked pairings. It starts by estimating a viewpoint transformation from the initial pairing, and proceeds by repeatedly identifying additional consistent pairings, adopting the best pairings, and using those to update the transformation estimate. (A method of computing the viewpoint transformation is described in section 4.2, below.) During this search, possible pairings are rated according to the contribution each would make to the match quality measure. Provided it is consistent with pairings adopted so far, the pairing $\langle j, k\rangle$ receives the rating

$$g_j(k; T) = \log \mathrm{P}(\tilde{e}_j = k \mid T, H) - \log \mathrm{P}(\tilde{e}_j = k) \tag{20}$$

This rating considers the same criteria as the initial ratings (equation 19), while also favoring pairings whose feature positions correspond closely according to the transformation estimate.

For efficiency, a priority queue is used to manage pairing choices during a search. Each pairing is placed on the queue as it is rated so that, once all pairings have been evaluated, the queue contains a few dozen of the best pairings. Queued pairings that conflict are considered ambiguous; they are downrated so that they will be postponed in favor of less ambiguous pairings. Finally, the highest-ranked pairings are adopted and used to update the transformation estimate. Backtracking is performed when ambiguity forces a choice among conflicting pairings, and a search branch is terminated when no additional pairings can be identified to improve the match quality measure.

From several starting hypotheses and the various search branches that result from backtracking, we obtain a number of consistent matches. As they are found only the best is retained, and its match quality measure provides a threshold for pruning subsequent search branches.

Note that the match quality measure provides an estimate of the (logarithm of the) probability that the match represents a true instance of the object in the image. One way to judge recognition, then, is to require that this probability exceeds some specified threshold. Setting the threshold to the ratio of costs of Type II and Type I decision errors produces recognition decisions that minimize the expected error cost.

4.2 Estimating the viewpoint transformation †

The matching procedure requires that we estimate a viewpoint transformation from one or more feature pairings, with the desired estimate being that which maximizes the match quality measure for the given pairings. Fortunately, this is a linear, least-squares estimation problem for which good algorithms exist.

The estimation problem is formulated as follows. Each pairing $\langle j, k \rangle$ of model and image features is related by the transformation $\mathbf{t}$ and a residual error $\tilde{\mathbf{e}}$:

$$\mathbf{A}_k \, \mathbf{t} = \mathbf{b}_j + \tilde{\mathbf{e}}. \tag{21}$$

Here, $\mathbf{A}_k$ is the matrix representation of image feature k's mean position (see equation 1), $\mathbf{t}$ is the transformation estimate vector $[x_t y_t u_t v_t]$, and $\mathbf{b}_j$ is the vector representation of model feature j's mean position. The residual $\tilde{\mathbf{e}}$ is assumed to have a Gaussian distribution whose covariance $\mathbf{C}_j$ can be estimated from the series of position vectors, $\bar{B}_j$, recorded by the model. We can rewrite this relation so that the residual has unit variance by multiplying both sides by the upper triangular square root of $\mathbf{C}_j$ (a process called *whitening*).

$$\mathbf{U}_j^{-1} \, \mathbf{A}_k \, \mathbf{t} = \mathbf{U}_j^{-1} \, \mathbf{b}_j + \tilde{\mathbf{e}}', \text{ where } \mathbf{C}_j = \mathbf{U}_j \mathbf{U}_j^{\mathrm{T}} \text{ and } \tilde{\mathbf{e}}' \sim \mathrm{N}(0, I). \tag{22}$$

A series of feature pairings gives us a series of such relations. From them, a linear, least-squares estimator determines both the transformation $\mathbf{t}$ that minimizes the sum of the residual errors, and its covariance $\mathbf{C}_t$.

During a match search, feature pairings are adopted sequentially. We need to refine the transformation estimate with each new pairing or group of pairings adopted so that an improved estimate can then be used to identify additional pairings. Thus a recursive estimator is used.

The square root information filter (SRIF) [3] is a recursive estimator well suited for this problem. Compared to the Kalman filter it is numerically more stable and faster for batched measurements; it also has the nice property of computing the total residual error as a side effect. As its name implies, the SRIF works by updating the square root of the information matrix, which is the inverse of the estimate's covariance matrix. The initial square root $\mathbf{R}_1$ and state vector $\mathbf{z}_1$ are obtained from the first pairing $\langle j,k\rangle$ of model and image features:

$$\mathbf{R}_1 = \mathbf{U}_j^{-1}\,\mathbf{A}_k \;\text{ and }\; \mathbf{z}_1 = \mathbf{U}_j^{-1}\,\mathbf{b}_j. \tag{23}$$

Then, with each subsequent pairing $\langle j,k\rangle$, the estimate is updated by triangularizing a matrix composed of the previous estimate and data from the new pairing:

$$\begin{bmatrix} \mathbf{R}_{i-1} & \mathbf{z}_{i-1} \\ \mathbf{U}_j^{-1}\mathbf{A}_k & \mathbf{U}_j^{-1}\mathbf{b}_j \end{bmatrix} \overset{\triangle}{\rightarrow} \begin{bmatrix} \mathbf{R}_i & \mathbf{z}_i \\ 0 & \mathbf{e}_i \end{bmatrix}. \tag{24}$$

When estimates of the viewpoint transformation and its covariance are needed, they can be obtained by

$$\mathbf{t}_i = \mathbf{R}_i^{-1}\mathbf{z}_i \;\text{ and }\; \mathbf{C}_{t_i} = \mathbf{R}_i^{-1}\mathbf{R}_i^{-\mathrm{T}}. \tag{25}$$

This requires only back substitution since $\mathbf{R}_i$ is triangular. The SRIF also makes the total residual error available as $\mathbf{e}_i\mathbf{e}_i^{\mathrm{T}}$, which conveniently corresponds to the $\log \mathrm{P}(\tilde{\mathbf{b}}_j = T(\mathbf{b}_k) \mid \tilde{e}_j \neq -, T, H)$ component of our match quality measure. Thus, following each update of the transformation estimate, the match quality measure for the new transformation can be computed easily; there is no need to re-evaluate equation 18 for the new transformation and each previous feature pairing.

5 Learning Procedure

The learning procedure assembles one or more model graphs from a series of training images showing various views of an object. Two tasks are required:

clustering	The learning procedure must divide the training images into clusters, each destined to form one characteristic view.
generalizing	For each cluster, it must construct a model graph summarizing the members of that cluster. The model graph represents a generalization of the cluster's contents.

Since clustering decisions ought to consider how well the resulting clusters can be generalized, the two tasks are closely related. Each will be discussed separately, however, in the following two sections.

We use $\mathcal{X}$ to denote the series of training images for one object. During learning, the object's model $\mathcal{M}$ consists of a series of clusters $\mathcal{X}_i \subseteq \mathcal{X}$, each with an associated model graph $\bar{G}_i$. Once learning is complete, only the model graphs must be retained to support recognition.

5.1 Clustering training images into characteristic views[4]

An incremental conceptual clustering algorithm is used to create clusters among the training images. Clustering is incremental in that, as each training image is acquired, it is assigned to an existing cluster or used to form a new one. Like other conceptual clustering algorithms (e.g., COBWEB [11]), the algorithm uses a global measure of overall clustering quality to guide clustering decisions. This measure is chosen to promote and balance two somewhat-conflicting qualities. On one hand, it favors clusterings that result in simple, concise, and efficient models, while on the other hand, it favors clusterings whose resulting model graphs accurately characterize (or match) the training images.

Maximum a posterior (MAP) estimation provides a nice framework for combining these two qualities. It suggests that the learning procedure choose a model $\mathcal{M}$ that maximizes the posterior probability $\mathrm{P}(\mathcal{M} \mid \mathcal{X})$. By Bayes's theorem, this is equivalent to maximizing the product $\mathrm{P}(\mathcal{M})\mathrm{P}(\mathcal{X} \mid \mathcal{M})$. The prior distribution $\mathrm{P}(\mathcal{M})$ can be designed to favor simple models, while the conditional distribution $\mathrm{P}(\mathcal{X} \mid \mathcal{M})$ can be designed to favor models that characterize the training images accurately.

- *Prior distribution.* We apply the minimum description length (MDL) principle [22] to define a prior distribution favoring simple models. Briefly, the MDL principle provides a method of constructing a prior probability distribution over a family of statistical models by relating the probability of each to the length of its description as written in some minimal-length encoding scheme. To encode a model, we concisely enumerate its model graphs, nodes, arcs, attribute vectors, and position vectors, using a fixed number of bits for each component. With $L(\mathcal{M})$ denoting the length of $\mathcal{M}$'s encoding, the prior probability of $\mathcal{M}$ is given by $\log \mathrm{P}(\mathcal{M}) = -L(\mathcal{M})$.

- *Conditional distribution.* We use the match quality measure to define a conditional distribution favoring accurate models. Recall that the measure is based on an estimate of the probability that the match represents a true occurrence of the modeled object in the image. For this match probability to be high, the model must accurately depict how the object appears in the image. Thus, to rate the accuracy of a model, we combine match probability estimates for each of the model's training images:

$$\mathrm{P}(\mathcal{X} \mid \mathcal{M}) = \prod_i \prod_{X \in \mathcal{X}_i} \max_{\langle E,T \rangle} \mathrm{P}(H \mid E, T; X, \bar{G}_i), \tag{26}$$

  where $\mathrm{P}(H \mid E, T; X, \bar{G}_i)$ is defined by equation 4. The maximum over matches $\langle E, T \rangle$ is found by the matching procedure described in section 4.

[4]At the time of writing, the clustering method described here had not yet been validated experimentally.

As each training image is acquired it is assigned to an existing cluster or used to form a new one. Choices among clustering alternatives are made to maximize the resulting $P(\mathcal{M} \mid \mathcal{X})$. When evaluating an alternative, each cluster's subset of training images $\mathcal{X}_i$ is first generalized to form a model graph $\mathcal{M}_i$ as described below.

5.2 Generalizing training images to form a model graph

Within each cluster, training images are merged to form a single model graph that represents a generalization of those images. An initial model graph is formed from the first training image's graph. That model graph is then matched with each subsequent training image's graph and revised after each match according to the match result. A model feature j that matches an image feature k receives an additional attribute vector $\mathbf{a}_k$ and position $\mathbf{b}_k$ for its series $\bar{A}_j$ and $\bar{B}_j$. Some unmatched image features are used to extend the model graph, while model features that remain largely unmatched are eventually pruned. After several training images have been processed in this way the model graph nears an equilibrium, containing the most consistent features with representative populations of sample attribute vectors and positions for each.

6 Experimental Results

A system that learns a single, characteristic view has been implemented using facilities of the Vista computer vision environment [21]; implementation of the clustering procedure needed to learn multiple views is in progress. The system recognizes 3-D objects in 2-D intensity images, employing a repertoire of features designed to describe the appearance of manufactured objects. Straight and circular segments of intensity edges are the lowest-level features. These are augmented by features representing various perceptually-significant groupings, including junctions, pairs and triples of junctions, pairs of parallel segments, chains of such pairs, and convex regions. Features that are rotationally symmetric, such as straight lines, are simply represented by multiple graph nodes, one per orientation.

Experiments with this system have produced encouraging results. For example, figure 4 shows a model of a characteristic view of a stool learned from nine training images acquired over a 20-degree range of viewpoint. Figure 5 shows the result of matching that model with a cluttered test image. The match search begins with a pairing of junctions (shown with a bold $\times$ in figure 5) that is rated highly by equation 19 primarily due to the image feature's attribute values. Matching proceeds with a pairing of parallel arcs (also shown in bold) that is favored in part due to the model feature's low positional uncertainty (apparent in figure 4(d)).

We are studying the models produced to gain further insight. As evident from the model depiction in figure 4 and from the histogram in figure 6, the stool model records significant differences in the positional uncertainty of various

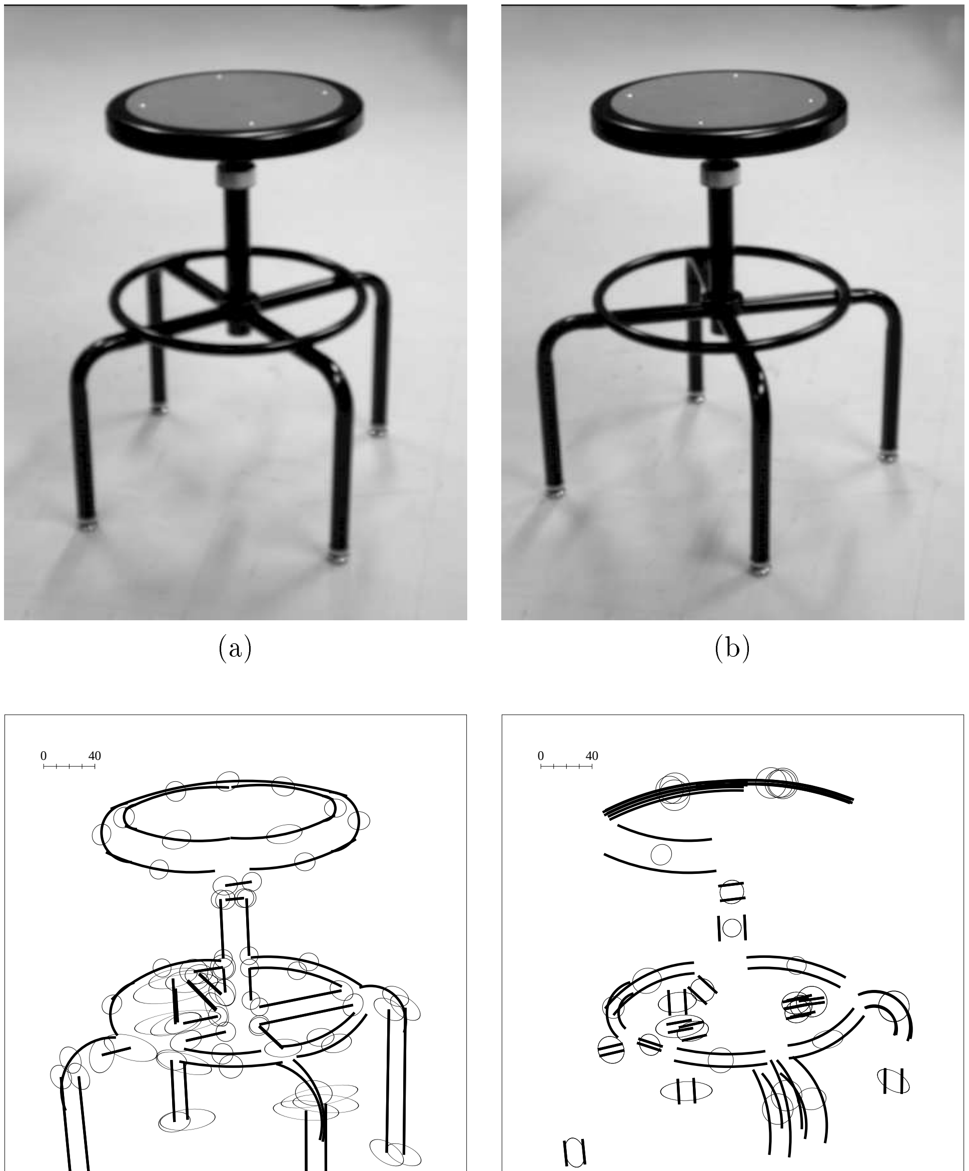

Figure 4: *Nine training images spanning 20 degrees of viewing angle, from (a) to (b), yield a single characteristic view model. Among model features, those denoting straight and circular segments of intensity edge are shown in (c); those denoting pairs of parallel segments are shown in (d). Ellipses depict two standard deviations of feature location uncertainty.*

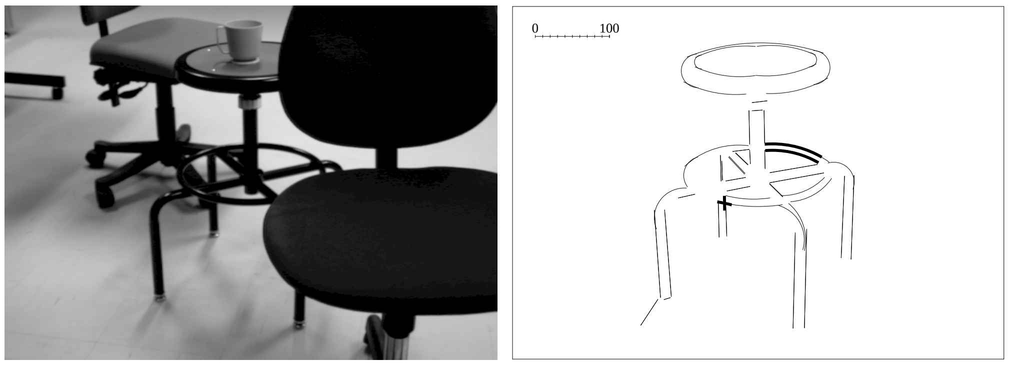

Figure 5: *A cluttered test image (a) in which the partially-occluded stool is recognized (b). Model features representing segments of intensity edge are shown projected into the image according to the final viewpoint transformation estimate. See the text for further explanation.*

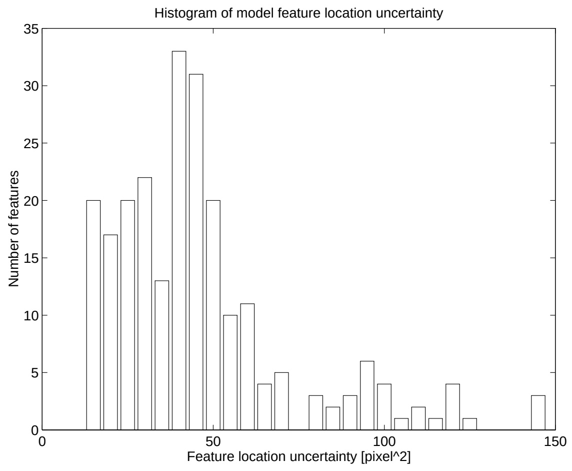

Figure 6: *Features of the stool model vary widely in positional uncertainty, as shown by this histogram of feature location uncertainty. Here, location uncertainty is measured as the area of a one-standard-deviation ellipse about the model feature's expected location.*

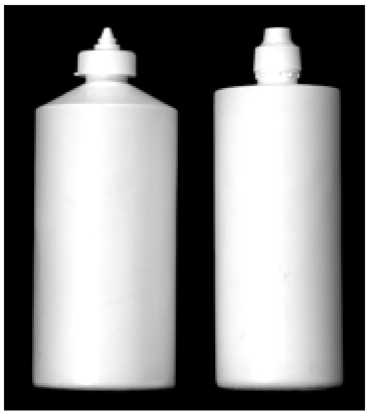

Figure 7: *Bottles A and B, used to test generality and specificity of models.*

Image	Model	Features Paired	Match Quality
Bottle A	Bottle A	43 / 109	60.7
Bottle A	Bottle B	20 / 56	6.7
Bottle B	Bottle A	27 / 109	-2.8
Bottle B	Bottle B	27 / 56	20.7

Table 1: *Results of matching each subimage of figure 7 with each bottle model.* Features Paired *is the proportion of model features paired.* Match Quality *is the value of the match quality measure, $g(E,T)$. Adapted from [19].*

features. Some differences are due to shifts in the relative positions of features with changing viewpoint—the seat and post remain fixed, for example, while the legs shift in various directions. Others are due to inherent differences in the accuracy of localizing various types of features—for example, a right-angle junction might be better localized than an oblique or acute one. Differences would be even greater for a flexible object.

Additional experiments have sought to determine whether the method can generalize across objects of similar appearance while still differentiating them on the basis of small distinctions. In one experiment, models were created for bottles A and B, shown in figure 7, using six training images of each. Each model was then matched with two subimages from figure 7: one containing the identical object, the other containing its counterpart. Table 1 summarizes the results. Each model matches its identical object best, meaning that the two objects are successfully distinguished; however each model also matches its counterpart to a lesser degree, meaning that each model successfully generalizes to match other objects of similar appearance.

In this case the specificity of the two models is due, in part, to differences in attribute value distributions. For example, each model includes a feature for its bottle's lower left corner and one attribute of that feature records the ratio of the corner's two sides. Since the two bottles have different height-to-width ratios, this feature is among those that help to differentiate the bottles. Figure 8 shows how the pdfs estimated for this attribute differ between the two models.

7 Related Research on Learning to Recognize Objects

This section surveys other efforts to build systems that learn to recognize objects. The survey is organized according to the role that learning plays in these systems. A final section summarizes efforts to establish theoretical bounds on the learnability of object recognition.

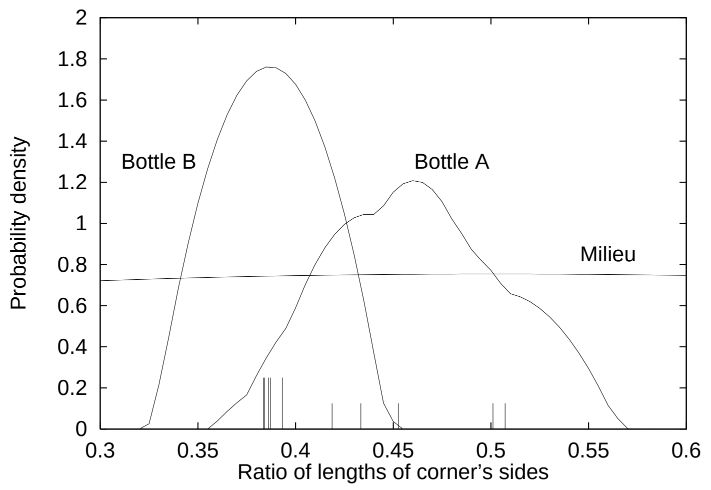

Figure 8: *Comparison of three attribute value distributions. Those labeled* Bottle A *and* Bottle B *are from model features corresponding to the lower left corners of bottles A and B. Spikes represent the two populations of sample values from which these distributions are estimated. The distribution labeled* Milieu *averages all corners from numerous images. From [19].*

7.1 Learning appearance primitives

An object recognition system may learn new types of shape or appearance primitives to use in describing objects. Typically this is done by clustering existing primitives or groups of them, and then associating new primitives with the clusters that have been found. New primitives thus represent particular configurations or abstractions of existing ones. The new configurations may improve the representation's descriptive power, and the new abstractions may allow more appropriate generalizations.

Segen [24] has demonstrated this approach with a system that learns a representation for 2-D contours. The system's lowest-level primitives are distinguished points, such as curvature extrema, found on contours in training images. Nearby points are paired, each pair is characterized by a vector of measurements, the measurement vectors are clustered, and a new primitive is invented for each cluster of significant size. Consequently, each new primitive describes a commonly observed configuration of two distinguished points. The induction process is repeated with these new primitives to generate higher-level primitives describing groups of four, eight, and more distinguished points. Weng et al.'s Cresceptron system [31] is analogous in that it induces a hierarchy of primitives within a pre-programmed framework. Since these primitives are essentially templates, invariance to translation, rotation, and scaling in the image must be provided by prior segmentation or by an attentional mechanism. We would expect both these method to be sensitive to clutter in the training images and to parameters of the clustering algorithm.

Delanoy, Verly, and Dudgeon [7] and Fichera et al. [10] have described object recognition systems that induce fuzzy predicates from training images. Their systems represent models as logical formulae, and, therefore, the systems need appropriate predicates. These are invented by clustering measurements of low-level primitives that have been recovered from training images. Turk and Pentland [29] and Murase and Nayar [17] have induced features using principle component analysis. They compute the several most significant eigenvectors, or principle components, of the set of training images. Since these few eigenvectors span much of the subspace containing the training images, they can be used to concisely describe those images and others like them. However, as this is a global representation, it may have difficulty with matching cluttered images.

7.2 Learning an appearance classifier

The object recognition task can be characterized, in part, as a classification problem: instances represented as vectors, sets, or structures must be classified into categories corresponding to various objects. A wealth of techniques has been developed for classifying, and for inducing classifiers from training examples. For the purposes of object recognition, the important considerations distinguishing these techniques include the expressiveness and complexity of the input representation (e.g., vectors are easier to classify than structures), the generality of the categories learned, the ability to cope with noisy features, the number of training examples needed, and the sensitivity to the order in which examples are presented.

Jain and Hoffman [14] describe a system that learns rules for classifying objects in range images. The instances classified by their system are sets of shape primitives with associated measurements. The classifier applies a series of rules, each contributing evidence for or against various classifications. Each rule applies to a particular type of shape primitive and a particular range of measurements for that primitive. These rules are learned from training images by extracting primitives from the images, clustering them according to their measurements, and associating rules with the clusters that derive primarily from a single object. Because this system does not learn constraints governing the relative positions of the shape primitives, it appears to have a very limited ability to distinguish among objects that have different arrangements of similar features.

Neural networks, including radial basis function networks, have been used as trainable classifiers for object recognition (e.g., [4]). In this role, the network approximates a function that maps a vector of feature measurements to an object identifier, or to a vector of graded yes/no responses, one per object. For this approach to succeed the function must be smooth; furthermore, as with any classifier, the object categories must be shaped appropriately in feature space (e.g., Gaussian radial basis functions are best suited for representing hyperellipsoids). Nearest neighbor classifiers, which make fewer assumptions, are also commonly used. Rare are comparative evaluations of how various classifier/feature space combinations perform on object recognition problems (such as [5]). The most difficult aspect of this approach seems to be deriving appropriate feature vectors from cluttered images.

7.3 Learning a structural model

A structural model explicitly represents both shape primitives and their spatial relationships. Its structure is thus analogous to that of the modeled object, and it is often represented as a graph or, equivalently, as a series of predicates. In general, a structural model is learned from training images by first obtaining structural descriptions from each image, and then inducing a generalization covering those descriptions. Connell and Brady [6] have described a system that learns structural models for recognizing 2-D objects in intensity images. The system incorporates many interesting ideas. They use graphs to represent the part/whole and adjacency relations among object regions described by smoothed local symmetries (ribbon shapes). An attribute of a region, such as its elongation or curvature, is encoded symbolically by the presence or absence of additional graph nodes according to a Gray code. A structural learning procedure forms a model graph from multiple example graphs, most commonly by deleting any nodes not shared by all graphs (the well-known dropping rule for generalization). Similarity between two graphs is measured by a purely syntactic measure: simply by counting the nodes they share. Consequently, this system accords equal importance to all features, and it uses a somewhat arbitrary metric for comparing attribute values.

The approaches described below involve more powerful models that represent probability distributions over graphs. A Markov or independence condition is assumed so that the high-order, joint probability distribution can be approximated by a product of low-order distributions, one per node or arc. The probability of a particular graph instance is then defined according to a partial match between that instance and the model. In these respects, the approaches resemble our own for learning characteristic views. Our approach differs in that it uses feature positions as well as attributes and relations to support an efficient matching procedure based on iterative alignment.

Wong and You [32] represent a model as a random graph in which nodes represent shape primitives, arcs and hyperarcs represent relations among them, and both have attribute values characterized by discrete probability distributions. An attributed graph (i.e., a random graph's outcome) is treated as just a special case of random graph. An entropy measure is defined on random graphs, and the distance between two random graphs is defined as the increment in entropy that would result from merging the two (the minimal increment over all possible mergings). A random graph is synthesized from examples by repeatedly merging the two nearest graphs. This learning method seems to have been demonstrated only with 2-D recognition problems and clean, synthetic images. However, McArthur [16] has extended the random graph formalism to allow continuous attributes, and he has used it to learn 3-D, object-centered models from images with known viewpoints. In comparison, our formalism incorporates continuous attributes more easily using likelihoods rather than entropies, and our learning method does not require knowledge of viewpoint or feature correspondences.

In Segen's system [24] for recognizing 2-D shapes, a graph node represents a relation among shape primitives, which are oriented point features (see page 23). But instead of specifying a particular relation, a node in the model graph specifies a probability distribution over possible relations. For example, relations involving two features may include one that holds when the features are oriented away from each other, and another that holds when they are oriented in the same direction; a model node may specify that the first relation holds with probability 0.5, and the second, with probability 0.3. When a graph instance is matched with the model, the instance's probability can be computed by assessing the probability of each instance node according to the distribution of its corresponding model node. Recognition involves finding the most probable match. A model is learned incrementally from instances by matching it with each one successively; following each match the probabilities recorded in matching model nodes are adjusted and any unmatched instance nodes are added to the model. Whereas Segen's system reduces all measurements to global categories found by clustering, our method retains numeric measurements as attribute and position distributions that are learned individually for each model feature. Consequently, we expect better performance in discrimination and generalization.

7.4 Learning a set of characteristic views

In learning a model that is to be represented as a set of characteristic views, part of the task is to choose those views. One can cluster the training images (a form of unsupervised learning) to create one characteristic view from each cluster. While several researchers have done this with images rendered from CAD models, thus avoiding the feature correspondence problem, Gros [12] and Seibert and Waxman [25] have clustered real images. Gros measures the similarity of an image pair as the proportion of matching shape primitives, whereas Seibert and Waxman use a vector clustering algorithm with fixed-length vectors encoding global appearance. Our method, in comparison, uses a clustering measure based on objective performance goals (accuracy and efficiency), and an appearance representation less affected by occlusion.

7.5 Learning a recognition strategy

Draper [8] has considered how a system equipped with a variety of special-purpose representations and algorithms might learn strategies for employing those techniques to recognize specific objects. A typical recognition task would be to locate a tree by fitting a parabola to the top of its crown. For this task, an appropriate strategy is to segment the image, extract regions that are colored and textured like foliage, group these into larger regions, smooth region boundaries, and fit parabolas to the boundaries. A human supplies training examples by pointing out the desired parabolas in a series of images. The system then evaluates various strategy choices (e.g., whether to smooth region boundaries) and parameter choices (e.g., the degree of smoothing) for cost and effectiveness. From these evaluations it constructs a strategy, including alternatives, for performing the task on new images. By learning what order to try alternative recognition methods, Draper's system differs from those that just select a single set of features useful for recognition. Difficulty remains in limiting the search among strategies and parameters to achieve acceptable performance.

7.6 Theoretical learnability

There have been some efforts to identify theoretical limits on what can be learned from images. These efforts are based on the analogy that learning to recognize an object from images is like learning a concept from examples. Valiant [30] has provided a useful definition for characterizing the class of concepts that can be learned by a particular algorithm: Informally, a class is probably approximately correct (PAC) learnable by an algorithm if, with high probability, the algorithm learns a concept that correctly classifies a high proportion of examples using polynomially bounded resources (and, consequently, number of training examples); the bound is a polynomial of both the accuracy and some natural parameter of the concept class (e.g., vector length for concepts defined on a vector space). Significantly, the algorithm has no prior knowledge about the distribution of examples.

Shvayster [27] has shown that some classes of concepts defined on binary images are not PAC-learnable from positive examples by any algorithm. For example, suppose a template is said to match an image when every black pixel in the template corresponds to a black pixel in the image (though not necessarily vice versa). Then the concept consisting of all instances not matching some unknown template is not PAC-learnable. Shvayster speculates that some nonlearnable concepts may become learnable, however, if some prior knowledge about the distribution of examples is available.

Edelman [9] has argued that Shvayster's negative result is not applicable to object recognition because it uses an instance representation, the binary image, that is inappropriate. If instead instances are represented by vectors of point feature locations, Edelman shows, then recognition of an object can be learned from a polynomial number of positive examples. He concludes that model learning may be practical, provided an appropriate representation is chosen.

8 Summary

We have presented a method for recognizing objects using models acquired from training images. Appearance in an image is represented by an attributed graph of discrete features and their relations, with a typical object described by many features. Since one object can vary greatly in appearance when viewed under different conditions, a model is represented by a probability distribution over such graphs. The range of this distribution is divided among characteristic views, allowing a simplified representation for each view as a model graph of independent features.

A model feature is described by probability distributions for probabilities of detection, various internal attribute values, and various image positions. All three distributions are estimated from samples supplied by training images.

A match quality measure provides a principled means of evaluating a match between a model and an image. It combines probabilities that are estimated using the distributions recorded by the model. The measure leads naturally to an efficient matching procedure called probabilistic alignment. In searching for a solution, the procedure can employ constraints arising both from the topology of the model graph and from the probability distributions describing individual features.

The model learning procedure has two components. A conceptual clustering component determines clusters of training images that correspond to characteristic views by maximizing a global measure of cluster quality. That measure combines a simplicity criterion based on the minimum description length principle with a fit criterion based on the match quality measure. A generalizing component merges the images within each cluster to form a model graph representing a generalization of that cluster. It uses the matching procedure to determine correspondences among the cluster's images.

An important aspect of the recognition learning problem this work has not addressed is how a database of acquired model graphs should be organized and accessed. Possibilities include organizing the model graphs hierarchically [2, 26], or using selected high-level features and their attributes to index the collection of model graphs. This issue is presently beyond the scope of our own work, however.

References

[1] N. Ayache and O. D. Faugeras. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, PAMI-8(1):44–54, Jan. 1986.

[2] R. Basri. Recognition by prototypes. In *Proc. Conf. Comput. Vision and Patt. Recognit.*, pages 161–167, 1993.

[3] G. J. Bierman. *Factorization Methods for Discrete Sequential Estimation.* Academic Press, New York, 1977.

[4] R. Brunelli and T. Poggio. HyperBF networks for real object recognition. In *Proc. Int. Joint Conf. Artificial Intell.*, volume 2, pages 1278–1284, 1991.

[5] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(10):1042–1052, Oct. 1993.

[6] J. H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intell.*, 31:159–183, 1987.

[7] R. L. Delanoy, J. G. Verly, and D. E. Dudgeon. Automatic building and supervised discrimination learning of appearance models of 3-D objects. In *SPIE Proc. Vol. 1708: Applications of Artificial Intell. X: Machine Vision and Robotics*, pages 549–560, 1992.

[8] B. Draper. *Learning Object Recognition Strategies.* PhD thesis, Univ. of Massachusetts, Amherst, Mass., May 1993.

[9] S. Edelman. On learning to recognize 3-D objects from examples. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(8):833–837, Aug. 1993.

[10] O. Fichera, P. Pellegretti, F. Roli, and S. B. Serpico. Automatic acquisition of visual models for image recognition. In *Proc. IAPR Int. Conf. Patt. Recognit.*, volume 1, pages 95–98, 1992.

[11] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[12] P. Gros. Matching and clustering: Two steps towards automatic object model generation in computer vision. In *Proc. AAAI Fall Symp.: Machine Learning in Comput. Vision*, 1993.

[13] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. J. Comput. Vision*, 5(2):195–212, Nov. 1990.

[14] A. K. Jain and R. Hoffman. Evidence-based recognition of 3-D objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 10(6):783–801, Nov. 1988.

[15] D. G. Lowe. The viewpoint consistency constraint. *Int. J. Comput. Vision*, 1:57–72, 1987.

[16] B. A. McArthur. *Incremental Synthesis of Three-Dimensional Object Models Using Random Graphs.* PhD thesis, Univ. of Waterloo, 1991.

[17] H. Murase and S. K. Nayar. Learning and recognition of 3-D objects from brightness images. In *Proc. AAAI Fall Symp.: Machine Learning in Comput. Vision*, pages 25–29, 1993.

[18] A. R. Pope and D. G. Lowe. Learning 3D object recognition models from 2D images. In *Proc. AAAI Fall Symp.: Machine Learning in Comput. Vision*, 1993.

[19] A. R. Pope and D. G. Lowe. Learning object recognition models from images. In *Proc. Int. Conf. Comput. Vision*, pages 296–301, 1993.

[20] A. R. Pope and D. G. Lowe. Modeling positional uncertainty in object recognition. Technical Report 94-32, Dept. of Computer Science, Univ. of B.C., Nov. 1994. WWW `http://www.cs.ubc.ca/tr/1994/TR-94-32`.

[21] A. R. Pope and D. G. Lowe. Vista: A software environment for computer vision research. In *Proc. Conf. Comput. Vision and Patt. Recognit.*, pages 768–772, 1994. WWW `http://www.cs.ubc.ca/nest/lci/vista/vista.html`.

[22] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Ann. Statist.*, 11(2):416–431, 1983.

[23] E. Saund. Labeling of curvilinear structure across scales by token grouping. In *Proc. Conf. Comput. Vision and Patt. Recognit.*, pages 257–263, 1992.

[24] J. Segen. Model learning and recognition of nonrigid objects. In *Proc. Conf. Comput. Vision and Patt. Recognit.*, pages 597–602, 1989.

[25] M. Seibert and A. M. Waxman. Adaptive 3-D object recognition from multiple views. *IEEE Trans. Patt. Anal. Mach. Intell.*, 14(2):107–124, Feb. 1992.

[26] K. Sengupta and K. L. Boyer. Information theoretic clustering of large structural modelbases. In *Proc. Conf. Comput. Vision and Patt. Recognit.*, pages 174–179, 1993.

[27] H. Shvayster. Learnable and nonlearnable visual concepts. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12(5):459–466, May 1990.

[28] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, 1986.

[29] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. Conf. Comput. Vision and Patt. Recognit.*, pages 586–591, 1991.

[30] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27:1134–1142, 1984.

[31] J. J. Weng, N. Ahuja, and T. S. Huang. Learning recognition and segmentation of 3-D objects from 2-D images. In *Proc. Int. Conf. Comput. Vision*, pages 121–128, 1993.

[32] A. K. C. Wong and M. You. Entropy and distance of random graphs with application to structural pattern recognition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 7(5):599–609, Sept. 1985.

Acknowledgments

This research was supported by the Institute for Robotics and Intelligent Systems (IRIS), one of the Canadian Networks of Centers of Excellence. Additional support was provided by the Natural Sciences and Engineering Research Council of Canada and the Canadian Institute for Advanced Research. We wish to thank Jeffrey Beis, Alun Evans, James Little, Alan Mackworth, Daniel McReynolds, and Robert Woodham for helpful discussions.

Index