A Multiscale Stochastic Image Model for Automated Inspection

Daniel Tretter, *Member, IEEE*, Charles A. Bouman, *Member, IEEE*, Khalid W. Khawaja, and Anthony A. Maciejewski

Abstract—**In this paper, we develop a novel multiscale stochastic image model to describe the appearance of a complex three-dimensional object in a two-dimensional monochrome image. This formal image model is used in conjunction with Bayesian estimation techniques to perform automated inspection. The model is based on a stochastic tree structure in which each node is an important subassembly of the three-dimensional object. The data associated with each node or subassembly is modeled in a wavelet domain. We use a fast multiscale search technique to compute the sequential MAP (SMAP) estimate of the unknown position, scale factor, and 2-D rotation for each subassembly. The search is carried out in a manner similar to a sequential likelihood ratio test, where the process advances in scale rather than time. The results of this search determine whether or not the object passes inspection. A similar search is used in conjunction with the EM algorithm to estimate the model parameters for a given object from a set of training images. The performance of the algorithm is demonstrated on two different real assemblies.**

I. Introduction

FORMAL mathematical image models have long been used in the design of image processing algorithms for applications such as compression, restoration, and enhancement [1]. Such models are traditionally low level stochastic models of limited complexity. In recent years, however, important theoretical advances and increasingly powerful computers have led to more complex and sophisticated image models. Depending on the application, researchers have proposed both low-level and high-level models.

Low-level image models describe the behavior of individual image pixels relative to one another. Markov random fields and other spatial interaction models have proven useful for a variety of applications, including image segmentation and restoration [2], [3]. Bouman and Shapiro [4], along with Willsky, Benveniste, and their associates [5], [6], have developed multiscale stochastic models for image data.

High-level models are generally used to describe a more restrictive class of images. These models describe larger structures in the image explicitly, rather than describing individual pixel interactions. Grenander and his associates, for example, propose a model based on deformable templates to describe images of nonrigid objects [7], while Kopec and his colleagues model document images using a Markov source model for symbol generation in conjunction with a noisy channel [8], [9]. Our image model is primarily high level, although we do model individual pixel statistics within the context of larger structures. In addition, we combine the image model with a fast multiscale search procedure to form an object detection algorithm for use in the particular application of automated inspection. Since the detection process is based on a formal model of the image data, it can be carried out in a consistent manner using well-known stochastic estimation techniques.

A number of different approaches to the object inspection problem have been taken in the past. Much of the early work in this area concentrated on special purpose algorithms to inspect specific objects [10]. More recently, inspection has often been viewed as only one of a number of related machine vision tasks, so general object recognition systems are used for inspection. Examples of this approach include Brooks' ACRONYM system [11], as well as the systems of Flynn and Jain [12] and Mehrotra and Grosky [13], which perform three-dimensional pose estimation and use a multiple object database. Most object recognition techniques, however, are not based on a formal probabilistic model of the data. Instead, they generally extract features of some sort from the data and match these to corresponding object characteristics.

The image model proposed in this paper was constructed with the inspection application in mind. It therefore incorporates several concepts and features that have proven useful in other object detection algorithms. For instance, many approaches to object detection and shape representation use multiresolution processing to reduce computation while retaining robust results. Rosenfeld and his associates use multiscale template matching for object detection [14], [15], while other researchers use multiresolution descriptions to represent shape [16], [17]. Some researchers have combined multiscale approaches with a hierarchical description of object structure to further reduce computation. Burt uses a Laplacian pyramid data representation in conjunction with a tree structure that divides the object into various components [18]. Ettinger divides the object contour into subparts, which he searches using a coarse-to-fine recognition scheme [19]. Our object inspection algorithm incorporates similar concepts, so the

Manuscript received May 22, 1994; revised February 14, 1995. This work was supported by an AT&T Bell Laboratories Ph.D. Scholarship, the NEC corporation, National Science Foundation grant number MIP93-00560, and National Science Foundation grant number CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems. The associate editor coordinating the review of this paper and approving it for publication was Dr. Michael Unser.

D. Tretter is with Hewlett-Packard Laboratories, Palo Alto, CA 94304-1126 USA

C. A. Bouman, K. W. Khawaja, and A. A. Maciejewski are with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907-1285 USA.

IEEE Log Number 9415094.

1057–7149/95$04.00 © 1995 IEEE



Fig. 1. General model structure for a subassembly. The state is the (random) location, orientation, and scale factor of the subassembly. The image data is the (random) wavelet transform image. The parameters are deterministic quantities estimated from training data.

image model is based on an object component hierarchy and a multiscale data representation.

In this paper, which builds on the work presented in [20], [21], and [22], we develop a model-based inspection algorithm designed to detect assembly errors in a rigid object from a single monochrome image of the object. Since the algorithm is designed specifically for automated inspection, we can take advantage of the highly structured viewing conditions typically found in a factory environment. For example, since the object to be inspected is known in advance, the algorithm is only trained to be sensitive to this one object; anything else in the field of view is taken to be extraneous to the inspection task. Also, the regions of the object at which assembly errors are most likely to be visible are known, so the algorithm concentrates most of its attention on those object regions. Finally, the approximate location and pose of the inspected part will often be known [23], [24]. The algorithm is, therefore, designed to be robust to limited changes in viewing conditions, but it does not allow for arbitrary object orientation.

As viewing conditions change, the apparent shape and appearance of an object will alter, so the object model must be flexible enough to allow some degree of distortion. Each of the important features, or subassemblies, of the object is therefore modeled separately, and their relative positions in the image are permitted to vary randomly to a certain degree. The subassemblies are linked together in a stochastic tree structure, where the position, or state, of each subassembly is taken to be a random quantity dependent on the state of the parent subassembly in the tree. The states thus form a Bayesian network on the object tree [25].

Each subassembly is modeled separately using the structure shown in Fig. 1, where the arrows indicate conditional dependence. A subassembly's location, scale, and orientation in the image are expressed as a random state vector $\mathbf{X}$, where the component distributions are determined by the allowed viewing conditions. The exact distribution of $\mathbf{X}$ is dependent on the deterministic parameter set ϕ, which will remain the same for all images. The parameters are estimated from a set of training images, allowing the model to adapt to specific viewing conditions.

The data associated with each subassembly, which is taken to be a multiresolution wavelet decomposition of the original grayscale image, is modeled as a multiscale random field. Data values depend on the deterministic parameter θ, which can be thought of as a multiresolution template describing the appearance of the subassembly. The multiscale data model was developed with concepts and results from the theory of multiscale random processes in mind [4]–[6].

The inspection algorithm locates an object and all of its subassemblies in an image by estimating the state of each node of the object tree. The states are estimated based on the image data, which is modeled as a set of noisy measurements dependent on the underlying states. Thus, since the states form a Bayesian network on the object tree, the state estimation procedure is exactly analogous to state estimation for a hidden Markov model. The state estimation takes the form of a multiscale search at each node, progressing from the root of the object tree to its leaves. Each subassembly is inspected in turn, and the estimated state of the parent node is used to guide the multiscale search. The search at each node results in an approximation to the maximum a posteriori state estimate for the associated subassembly given the estimated parent state and the image data. The estimation procedure is therefore the sequential MAP (SMAP) procedure of Bouman and Shapiro [4]. This gives a noniterative, computationally efficient formulation for locating and identifying the desired object.

A similar multiscale search procedure is used during the training phase of the algorithm, where we estimate the model parameters from a set of training images. The parameter estimates are computed using the iterative expectation maximization (EM) algorithm [26].

The paper is organized as follows: In Section II, we define the tree structure making up the object model and specify the model associated with each subassembly. This model is then used in Section III to develop the multiscale search procedure for state estimation. Finally, Section IV discusses our parameter estimation procedure, which is used to adapt the algorithm to the particular object of interest. Simulation results are presented in Section V, and we end the paper with concluding remarks in Section VI.

II. The Model

In this section, we will specify a formal stochastic image model that can be used to describe the appearance of a general class of complex three-dimensional objects. The model has two distinct levels to its structure: The object tree and the subassembly. Each node of the object tree will be used to represent the relative position and orientation of the important object features, called subassemblies. Each subassembly will then be modeled using a wavelet transform of the associated image region.

Object Tree Model

Fig. 2 shows an example of an object tree for a complex three-dimensional object. Each box represents a subassembly or node of the tree, and is drawn around a feature of interest in the object's image. The boxes are connected together into a tree structure using lines, and the level of each node in the tree is represented by the number of lines making up the box. In general, the subassemblies will consist of various object components important for locating the object and for detecting assembly errors. Typically, nodes near the root of the tree are associated with larger parts of the object and represent the object's gross structure. These nodes also prove useful in locating the object in an image. Nodes further down the tree "zoom in" on smaller features that contain significant fine detail.

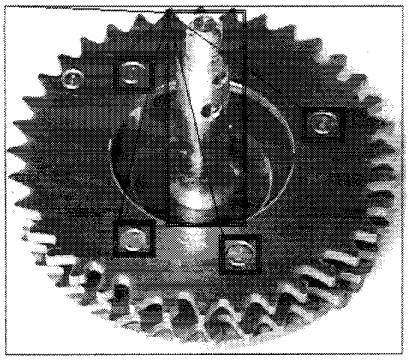


Fig. 2. An initialization image is used to define the object tree. The boxes indicate the subassemblies associated with the nodes of the tree, and the lines connecting the boxes show the parent-child links.

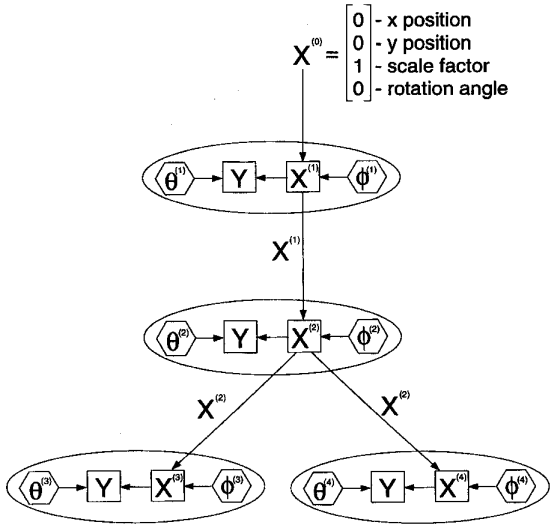


Fig. 3. Model structure of complete object assembly. At each node, Y is the image data; $X^{(c)}$ is the state containing the position, orientation and scale of the subassembly; $\theta^{(c)}$ is a set of data parameters which describes the appearance of the subassembly; and $\phi^{(c)}$ is a state parameter vector describing the variation in subassembly position.

Fig. 3 illustrates the structure and conditional dependencies in an object tree. Each node is represented by an oval containing four quantities, $X^{(c)}$, $\phi^{(c)}$, Y, and $\theta^{(c)}$, where c is the index of the node, and arrows indicate conditional dependency. We will use uppercase letters to denote random quantities and lowercase for nonrandom sample realizations.

The random state $X^{(c)}$ contains the position, orientation, and scale of the subassembly. $X^{(c)}$ is assumed random since the geometry of the camera and object may vary from image to image. In general, however, the position of a subassembly will depend on the position of its parent node in the object tree. This conditional dependence is indicated by the arrows between nodes. Since the observed image depends on the location and orientation of the object and its components, the image data Y in Fig. 3 depends on each of the states, $X^{(c)}$.

In addition to random quantities, each node contains two deterministic parameter vectors, $\phi^{(c)}$ and $\theta^{(c)}$. These parameter vectors are used to adapt the model to a wide variety of possible object behaviors and imaging environments. $\phi^{(c)}$ determines the mean and variation of a node's state given the parent node's state, and $\theta^{(c)}$ determines the mean and variation of image pixels given the node's state. Intuitively, one might think of $\theta^{(c)}$ as containing an image template for the subassembly, but we will see that $\theta^{(c)}$ actually contains more information than a simple template.

Since subassemblies only depend on each other through their positions, the node states $X^{(c)}$ form a Markov chain along any path from the root to a leaf of the tree. This tree dependent structure captures the interdependencies among the subassemblies while remaining amenable to efficient computational schemes [4], [6], [27]. If we index the nodes from 1 to M, then this Markov relationship may be stated as

$$p(x^{(1)}, \cdots, x^{(M)} | \phi^{(1)}, \cdots, \phi^{(M)}) = \prod_{c=1}^{M} p(x^{(c)} | X^{(p)} = x^{(p)}, \phi^{(c)}) \tag{1}$$

where p denotes the parent of node c, and the parent state for the root node of the object tree is the deterministic state vector $x^{(0)}$. Notice that the state of the subassembly $X^{(c)}$ depends on both the state parameters $\phi^{(c)}$ and the state of the parent node $X^{(p)}$.

The density functions given in (1) must next be defined. The subassembly state has components $X^{(c)} = [S^t, Z, R]^t$ where $S = [S_v, S_h]^t$ is vertical and horizontal position, Z is scale factor, and R is angle of rotation in radians. The state $X^{(c)} = x^{(c)} = [(s^{(c)})^t, z^{(c)}, r^{(c)}]^t$ defines a transformation of the subassembly from the image coordinate system to a normalized coordinate system with scale factor 1 and rotation angle 0. This normalized coordinate system is essentially used for data registration; the distortions in a particular image are undone, and the subassembly data is mapped to a common location. Each image pixel location i at resolution l will transform to a normalized location i', where

$$i' = \mathbf{T}^{(c)}(i - 2^{-l} s^{(c)}) ,$$
$$\mathbf{T}^{(c)} = \frac{1}{z^{(c)}} \begin{bmatrix} \cos r^{(c)} & \sin r^{(c)} \\ -\sin r^{(c)} & \cos r^{(c)} \end{bmatrix} .$$

We will use the matrix $\mathbf{T}^{(c)}$ to simplify our model notation.

The state parameter vector $\phi^{(c)}$ has the components

$$\phi^{(c)} = \begin{bmatrix} m^{(c)} \\ \gamma^{(c)} \end{bmatrix} = [(m_s^{(c)})^t, m_z^{(c)}, m_r^{(c)}, \gamma_s^{(c)}, \gamma_z^{(c)}, \gamma_r^{(c)}]^t ,$$

where $m^{(c)}$ and $\gamma^{(c)}$ play the role of a mean and variance vector, respectively. Given this notation, the state vector has a Gaussian distribution with the form

$$p\left(x^{(c)} | X^{(p)} = x^{(p)}, \phi^{(c)}\right) = \frac{|\mathbf{B}|^{-\frac{1}{2}}}{(2\pi)^2} \cdot \exp\left\{ -\frac{1}{2} (x^{(c)} - x^{(p)} - \mathbf{A} m^{(c)})^t \mathbf{B}^{-1} (x^{(c)} - x^{(p)} - \mathbf{A} m^{(c)}) \right\} \tag{2}$$

where $\mathbf{A}$ is a matrix determined by the parent state $x^{(p)}$ through the transformation $\mathbf{T}^{(p)}$, and $\mathbf{B}$ is a matrix determined

$$\text{a)}\ \begin{matrix} +1 & +1 \\ +1 & +1 \end{matrix} \qquad \text{b)}\ \begin{matrix} +1 & -1 \\ +1 & -1 \end{matrix}$$

$$\text{c)}\ \begin{matrix} +1 & +1 \\ -1 & -1 \end{matrix} \qquad \text{d)}\ \begin{matrix} -1 & +1 \\ +1 & -1 \end{matrix}$$

Fig. 4. Basis functions for the Haar transform. Notice that a) is the average, b) is the vertical edge gradient, c) is the horizontal edge gradient, and d) is only responsive to thin diagonal lines.

by $\phi^{(c)}$ and $x^{(p)}$.

$$\mathbf{A} = \begin{bmatrix} (\mathbf{T}^{(p)})^{-1} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} \left(z^{(p)}\right)^2 \gamma_s^{(c)} & 0 & 0 & 0 \\ 0 & \left(z^{(p)}\right)^2 \gamma_s^{(c)} & 0 & 0 \\ 0 & 0 & \gamma_z^{(c)} & 0 \\ 0 & 0 & 0 & \gamma_r^{(c)} \end{bmatrix}.$$

Note that the vertical and horizontal offset means depend on the matrix $\mathbf{T}^{(p)}$, which is a function of the scale factor $z^{(p)}$ and the rotation $r^{(p)}$. Therefore, the vertical and horizontal distance between subassemblies will scale with object size and change as the assembly rotates. For simplicity we assume the vertical and horizontal positions have the same variance. This assumption makes the variances independent of rotation angle. The root node does not have an actual parent node, so for this node we define the parent state $X^{(0)}$ to be $x^{(0)} = [0,\ 0,\ 1,\ 0]^t$.

Subassembly Model

In this section, we will present the model used for each subassembly or node of the object tree. This model determines the distribution of the image pixels in the region of each subassembly.

The subassembly model is based on a wavelet transform of the image. The wavelet transform has two important advantages in modeling the image. First, since the transform may be thought of as approximately separating the image into distinct spatial frequency bands, it tends to decorrelate the image data [28]. We will see that this decorrelation removes undesirable mismatches caused by small shifts in average gray scale. The decorrelation also results in a transformed image with the natural interpretation of vertical and horizontal edge bands. The second advantage of using the wavelet transform is the dramatically reduced computation which results from processing data at multiple scales [18]. In Section III, the object search is formulated as an optimization problem in a high-dimensional space. The key to the efficient solution of this optimization will be a structured search which exploits the multiresolution structure of the wavelet transform.

The wavelet transform uses the Haar basis functions illustrated in Fig. 4. Fig. 5 shows an image resulting from this Haar wavelet decomposition. Notice that at each resolution, two of the bands have the interpretation of being the horizontal or vertical edge gradients. This structure will be used to make the image model sensitive to both region (average gray scale) and edge (gradient magnitude) information. Another advantage of the Haar basis functions is the computational simplicity resulting from coefficients of ± 1.

We will generally assume that Y is the wavelet transformed image. The wavelet transform is an invertible, orthogonal transformation, so the transformed image contains all of the information in the original data. Also, since the Jacobian of the transformation is unity, the value of the density functions are equal for the original and transformed data.

For simplicity, our algorithm uses only the vertical and horizontal gradient information in the wavelet representation; we do not model the diagonal band information. This allows us to represent the data at each pixel location as a gradient vector. At each resolution l, define $Y_l = [Y_{lv}, Y_{lh}]$ where Y_{lv} and Y_{lh} are the vertical and horizontal bands of the wavelet transform. Generally, $0 \le l \le L-1$ where $l = 0$ is the finest resolution and $L-1$ is the coarsest. Each pixel in Y_l is denoted by $Y_l(i) = [Y_{lv}(i), Y_{lh}(i)]$ where $i = [i_1, i_2]^t$ is a vector index. Intuitively, this index corresponds to the physical position $[v, h] = [i_1 2^l + 2^{l-1},\ i_2 2^l + 2^{l-1}]$.

The pixels $Y_l(i)$ are assumed to be conditionally independent given the state $X^{(c)}$ and the data parameters θ. This is a reasonable assumption since the wavelet transform decorrelates the image data. Intuitively, the pixel value $y_l(i)$ represents the local gradient of the image at location i. Since image derivatives are known to be accurately modeled as Laplacian distributed [29], we choose a density function similar to the Laplacian density for our data distribution. In particular,

$$p(y_l(i)|X^{(c)} = x^{(c)}, \theta^{(c)}) = \frac{1}{2\pi(\lambda^{(c)}\tilde{\sigma}_l(i))^2} \cdot \exp\left\{-\frac{\|y_l(i) - \tilde{\mu}_l(i)\|}{\lambda^{(c)}\tilde{\sigma}_l(i)}\right\} \quad (3)$$

where $\|\cdot\|$ is the Euclidean norm and $\tilde{\mu}_l(i)$ and $\tilde{\sigma}_l(i)$ are model parameters determined by $x^{(c)}$ and θ. The redundant parameter $\lambda^{(c)}$, which also depends on the state $x^{(c)}$ and the resolution l, has been added to explicitly account for local variation in image brightness. Note that this model differs slightly from the Laplacian density, which uses a 1-norm in place of the 2-norm.

The mean vector $\tilde{\mu}_l(i)$ of (3) is just the average local gradient at pixel location i. This characterizes grayscale behavior, including edge polarity and sharpness. The variation parameters $\tilde{\sigma}_l(i)$ indicate the areas of greatest uncertainty in the template, which will generally occur near edges. Thus, the model is sensitive to both region-based and edge-based information, with the relative importance of each information type determined by the model parameters. Note that the variation parameter is common to both the vertical and horizontal wavelet bands. In this way, a rotation of the subassembly can be modeled by simply rotating each mean vector $\tilde{\mu}_l(i)$.

To define the relationship between the parameters of (3) and $X^{(c)}$ and θ, we must first precisely define the components for θ. For node c of the object tree, the components for $\theta^{(c)}$ are $\theta_l^{(c)}(i) = [\mu_l(i), \sigma_l(i)]$, where $\mu_l(i) = [\mu_{lv}(i), \mu_{lh}(i)]$ is the average gradient at template location i, and i is a vector index which takes values in $W_l^{(c)}$. The set $W_l^{(c)}$ may be thought of as a window containing the subassembly in the normalized

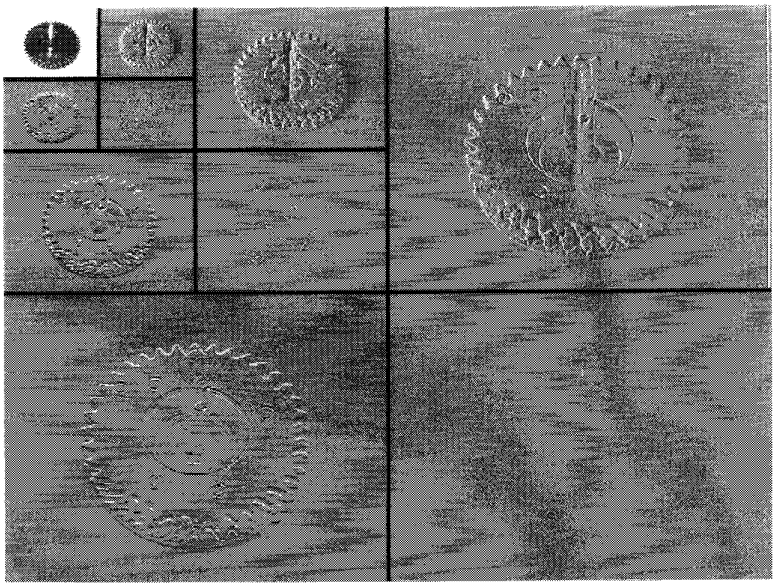


Fig. 5. Wavelet decomposition using the Haar basis functions. The transformation generates separate vertical and horizontal bands at each resolution.

coordinate system. In order to eliminate spurious results due to insufficient data, we define $W_l^{(c)}$ to be empty for resolutions l at which this window contains fewer than 4×4 pixels. In Fig. 2, these windows correspond to the rectangular boxes.

The effect of the state $x^{(c)} = [(s^{(c)})^t, z^{(c)}, r^{(c)}]^t$ is to transform and distort the template of parameters $\theta^{(c)}$ and its associated window $W^{(c)}$. Therefore, to compute the parameters of a pixel we will determine the θ parameters that transform to the pixel location. Unfortunately, this coordinate transformation will generally yield noninteger positions in the coordinates of the template. We solve this problem by using bilinear interpolation to compute parameter values between grid points. The variation parameters form a scalar template that undergoes an affine transformation while the mean vectors can be thought of as a local gradient field under the same transformation. The parameters of (3) are thus given by

$$\tilde{\mu}_l(i) = \mu_l\left(\mathbf{T}^{(c)}\left(i - 2^{-l}s^{(c)}\right)\right)\mathbf{T}^{(c)} \qquad (4)$$
$$\tilde{\sigma}_l(i) = \sigma_l\left(\mathbf{T}^{(c)}\left(i - 2^{-l}s^{(c)}\right)\right)$$

where the noninteger arguments of $\mu_l(\cdot)$ and $\sigma_l(\cdot)$ are interpreted as bilinear interpolation. Of course, (4) is only defined when i transforms to template locations contained in $W_l^{(c)}$. Therefore, this transformed window is defined to be

$$\tilde{W}_l^{(c)} = \left\{i : \mathbf{T}^{(c)}\left(i - 2^{-l}s^{(c)}\right) \in W_l^{(c)}\right\}.$$

Combining these ideas yields the complete data model at each resolution l.

$$p(y_l|X^{(c)} = x^{(c)}, \theta^{(c)}) = \prod_{i \in \tilde{W}_l^{(c)}} \frac{1}{2\pi(\lambda^{(c)}\tilde{\sigma}_l(i))^2} \cdot \exp\left\{-\frac{\|y_l(i) - \tilde{\mu}_l(i)\|}{\lambda^{(c)}\tilde{\sigma}_l(i)}\right\}. \qquad (5)$$

We should note that the model presented has a minor inconsistency. If the windows of the various subassemblies overlap, then there is more than one way in which the pixel parameters may be computed. Theoretically, this inconsistency could be eliminated by assigning a priority ordering to the nodes. For example, nodes closest to leaf nodes could occlude nodes higher in the tree. However, for computational simplicity we ignore this inconsistency and assume that the overlap of nodes in space and scale will not have a significant effect.

Also notice that pixels outside of the subassembly windows are not explicitly modeled. In practice, we will always compute ratios of density functions so the contribution due to these unmodeled pixels will cancel out. Kopec and Chou use this same idea in their model for document images [9].

III. State Estimation

To compare a given image to our model, we must first locate each of the object subassemblies in the image. This is equivalent to estimating the four-dimensional state vector associated with each node of the object tree. The states will be estimated using the sequential maximum a posteriori (SMAP) procedure of Bouman and Shapiro [4]. This technique simplifies the estimation problem by allowing the state of each node in the object tree to be estimated separately.

This section presents a multiscale technique to search the state space for the most likely position and orientation of a subassembly. Since the search algorithm must be performed for every new image, it should be as efficient as possible. Computational efficiency is achieved by using the log likelihood at coarse resolutions to guide the search at finer resolutions.

SMAP Estimation

The SMAP method starts at the object tree's root and progresses to its leaves. At each node of the tree, the maximum a posteriori (MAP) estimate of the state $X^{(c)}$ is computed given the image data y and the estimated state at the parent node, $\hat{x}^{(p)}$. In order to simplify computation and avoid a

recursive implementation, we modify the SMAP algorithm by ignoring data terms from descendants of the node c. Using these assumptions, the SMAP state estimate for node c is given by

$$\hat{x}^{(c)} = \arg\max_{x^{(c)}} \left\{ \log p(y|X^{(c)} = x^{(c)}, \theta^{(c)}) + \log p(x^{(c)}|X^{(p)} = \hat{x}^{(p)}, \phi^{(c)}) \right\}$$

To simplify computation, we will use likelihood ratios to compute $\hat{x}^{(c)}$. Let $p_0(y)$ be some as yet undefined density function for the data when the subassembly c is not present. Note that since $p_0(y)$ does not depend on $X^{(c)}$,

$$\hat{x}^{(c)} = \arg\max_{x^{(c)}} \left\{ \log \frac{p(y|X^{(c)} = x^{(c)}, \theta^{(c)})}{p_0(y)} + \log p(x^{(c)}|X^{(p)} = \hat{x}^{(p)}, \phi^{(c)}) \right\}. \quad (6)$$

A multiscale search procedure will be used to perform the optimization in (6), so we need to define a multiresolution version of the expression in (6). With this in mind, the log likelihood ratio for resolutions coarser than l is defined to be

$$L(x^{(c)}, l) = \log \left(\prod_{m=l}^{L-1} \frac{p(y_m|X^{(c)} = x^{(c)}, \theta^{(c)})}{p_0(y_m)} \right) + \log p(x^{(c)}|X^{(p)} = \hat{x}^{(p)}, \phi^{(c)}).$$

This expression, which we wish to maximize, is the sum of a data term and a prior term. The data term indicates how well the data at this state and resolution matches the subassembly model. The prior term gives the prior likelihood of the subassembly appearing at this location and orientation.

The prior term of the log likelihood ratio is computed using the prior state density function in (2), but the data term must still be precisely defined. For pixels $i \notin \tilde{W}_l^{(c)}$, the presence or absence of the subassembly is irrelevant. Therefore, for all $i \notin \tilde{W}_l^{(c)}$, $p_0(y_l(i)) = p(y_l(i)|X^{(c)} = x, \theta)$. If subassembly c is not present at state $x^{(c)}$, we have no a priori expectations for the pixel values in the window $\tilde{W}_l^{(c)}$. We therefore assume that these pixels are independent and identically distributed. Since y_l is a bandpass signal with no dc component, we assume the values are zero mean with distribution

$$p_0(y_l) = \prod_{i \in \tilde{W}_l^{(c)}} \frac{1}{2\pi \left(\lambda_0^{(c)}\right)^2} \exp\left\{ -\frac{\|y_l(i)\|}{\lambda_0^{(c)}} \right\}, \quad (7)$$

where $\lambda_0^{(c)}$ is the local average variation of the image data. Putting this model together with (5) yields the result

$$\log \left(\prod_{m=l}^{L-1} \frac{p(y_m|X^{(c)} = x^{(c)}, \theta^{(c)})}{p_0(y_m)} \right) = \sum_{m=l}^{L-1} \sum_{i \in \tilde{W}_m^{(c)}} \left(2 \log \frac{\lambda_0^{(c)}}{\lambda^{(c)} \tilde{\sigma}_m(i)} - \frac{\|y_m(i) - \tilde{\mu}_m(i)\|}{\lambda^{(c)} \tilde{\sigma}_m(i)} + \frac{\|y_m(i)\|}{\lambda_0^{(c)}} \right). \quad (8)$$

We estimate the unknown parameter $\lambda_0^{(c)}$ by maximizing (7) with respect to this parameter, while the value of $\lambda^{(c)}$ is estimated by maximizing (5). This gives the final expression

$$L(x^{(c)}, l) = 2\hat{N} \log \frac{\hat{\lambda}_0^{(c)}}{\hat{\lambda}^{(c)}} - \sum_{m=l}^{L-1} \sum_{i \in \tilde{W}_m^{(c)}} 2 \log \tilde{\sigma}_m(i) + \log p(x^{(c)}|X^{(p)} = \hat{x}^{(p)}, \phi^{(c)}), \quad (9)$$

where

$$\hat{\lambda}_0^{(c)} = \frac{1}{2\hat{N}} \sum_{m=l}^{L-1} \sum_{i \in \tilde{W}_m^{(c)}} \|y_m(i)\| \hat{\lambda}^{(c)} = \frac{1}{2\hat{N}} \sum_{m=l}^{L-1} \sum_{i \in \tilde{W}_m^{(c)}} \frac{\|y_m(i) - \tilde{\mu}_m(i)\|}{\tilde{\sigma}_m(i)}$$

$$\hat{N} = \sum_{m=l}^{L-1} \sum_{i \in \tilde{W}_m^{(c)}} 1 .$$

Note that the estimates $\hat{\lambda}_0^{(c)}$ and $\hat{\lambda}^{(c)}$ depend on the resolution l and the subassembly state $x^{(c)}$, which determines the windows $\tilde{W}_m^{(c)}$. The log likelihood ratio in (9) can now be computed at any candidate state $X^{(c)} = x^{(c)}$ and resolution l.

Multiscale Search for Subassembly

We next devise a procedure for searching the states, $x^{(c)}$, and resolutions, l, in an efficient manner. The possible subassembly positions, $x^{(c)}$, must be sampled at discrete points, and computation is saved by sampling $x^{(c)}$ more coarsely for large values of l corresponding to coarse resolution. Rotation and scale changes should also be sampled more finely for large templates. To do this, define the constant $d^{(c)}$ to be the diameter of the smallest circle containing the template at scale factor $z^{(c)} = 1$ and resolution $l = 0$. Then the sampling period of $z^{(c)}$ and $r^{(c)}$ should be inversely proportional to $d^{(c)}$. Using this approach, define $k = [k_1, k_2, k_3, k_4]^t$ to be a vector of integer indexes, and let $x(k, l)$ be the vector function

$$x(k,l) = \left[k_1 2^{l-1} + 2^{l-2},\ k_2 2^{l-1} + 2^{l-2},\ \frac{k_3 2^l + 2^{l-1}}{d^{(c)}},\ \frac{k_4 2^l + 2^{l-1}}{d^{(c)}} \right]^t .$$

The function $x(k, l)$ gives the candidate states at each resolution l, which we link to those at the next finer resolution by defining the neighbors of (k, l) to be

$$\text{next}(k,l) = \{(n, l-1) \mid n_i = 2k_i \text{ or } n_i = 2k_i + 1\} .$$

The state indexes k_1, k_2, k_3, and k_4 correspond to vertical and horizontal position, scale factor, and rotation angle, respectively. The index k_3 must therefore be nonnegative since only positive scale factors are possible, while the rotation angle must be between $-\pi$ and π, setting limits on the possible values of k_4 at each resolution l. The vertical and horizontal position are nominally unconstrained, although in practice indexes k_1 and k_2 are limited such that the position falls within the image boundaries. Fig. 6 illustrates this sampling scheme

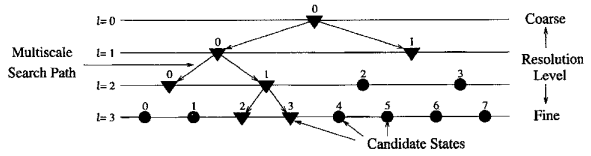


Fig. 6. Multiscale sampling for a one-dimensional state space. The index $k$associated with each sample is as labeled. A multiscale search procedure is carried out on these samples to compute the state estimate.

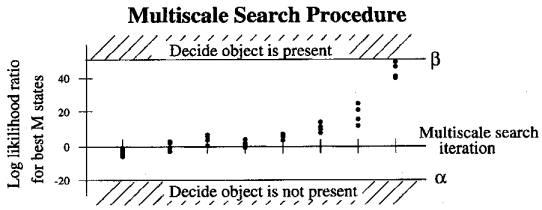


Fig. 7. An example search procedure for $M = 4$. The search terminates when it encounters a candidate state whose log likelihood ratio exceeds β or when the heap has been exhausted (all remaining candidate states have log likelihood ratios less than α).

for a single state component. Note that the candidate states form a binary tree that densely samples the space of possible states.

The multiscale search procedure is defined on this tree structure, and it proceeds based on the log likelihood ratio $L_d(k,l) \equiv L(x(k,l),l)$ associated with each sampling index k and resolution l. We initialize the search for a subassembly c by computing the log likelihood ratios over all vector indexes $k \in \mathcal{X}^{(c)}(\alpha,l)$ where

$$\mathcal{X}^{(c)}(\alpha,l) = \left\{k : \log p\Big(x(k,l)|X^{(p)} = \hat{x}^{(p)}, \phi^{(c)}\Big) > \alpha\right\}$$

and α is a user-defined rejection threshold. The initialization takes place at resolution $l = \max(l_{M_0}, l_0^{(c)})$, where l_{M_0} is equal to the coarsest resolution at which $\mathcal{X}^{(c)}(\alpha,\cdot)$ contains at least M_0 elements, and $l_0^{(c)}$ is the finest resolution to which the search is permitted to proceed. The constant M_0 is used to make sure the search is initialized with a reasonable number of points, and the finest resolution $l_0^{(c)}$ is set during training using the heuristic procedure described in section 4.

The initial candidate states and their associated log likelihood ratios are stored in a data structure known as a heap. This structure allows efficient insertion of new values and extraction of the pairs (k,l) with the largest log likelihood ratios.

After initialization, the search locates the Mmost promising search paths and expands them to the next finer resolution by computing the log likelihood ratios $L_d(\cdot)$ associated with their neighbors. If any of these log likelihood ratios fall below a rejection threshold α, the algorithm discards the corresponding state, thereby pruning the search space. If any of the log likelihood ratios exceed an acceptance threshold β, the corresponding state is returned as the state estimate $\hat{x}^{(c)}$. Candidate states with $L_d(\cdot)$ between α and β are stored on the heap. The algorithm then extracts the M best states from the updated heap and the process repeats. Since the best candidate states can occur at any resolution, the multiscale search can backtrack to coarser resolutions if necessary to investigate additional search paths. We improve robustness by choosing $M > 1$ and investigating multiple search paths simultaneously.

As illustrated in Fig. 7 the search takes the form of a sequential likelihood ratio test in which β and α represent acceptance and rejection thresholds. If these thresholds are not exceeded, the search process continues to finer resolutions where more data is obtained. If the search reaches a point at which all M candidate states are at the finest resolution, then a decision is made by comparing the log likelihood to a third threshold, β_0.

The search is implemented as described in Fig. 8. For our simulations we use the values $\alpha = -15, \beta = 100, \beta_0 = 20, M = 16$, and $M_0 = 100$. If the search for a particular subassembly terminates in a rejection (no match), that subassembly is declared missing, and the SMAP procedure is terminated for descendents of that node.

In some cases this search procedure will terminate with a match at a resolution $l^{(p)} > l_0^{(p)}$ for node p. The resulting coarse state estimate $\hat{X}^{(p)}$ can be viewed as a quantized version of the actual state, which we take to be at resolution $l_0^{(p)}$, so

$$\hat{X}^{(p)} = X^{(p)} + Q\,.$$

This quantization error will increase the uncertainty in the location of subassembly c, a child node of p. This increased uncertainty is accounted for by changing the covariance matrix of (2) to

$$\tilde{\mathbf{B}} = \mathbf{B} + \mathbf{B}_Q(\hat{X}^{(p)}, \phi^{(c)}, l^{(p)}, l_0^{(p)})$$

where $\mathbf{B}_Q(\cdot)$ is a diagonal matrix computed in Appendix A.

IV. Training Algorithm (Parameter Estimation)

An iterative procedure based on the expectation maximization (EM) algorithm is used to estimate the model parameters θ and ϕ from a set of training images. The first training image $Y(\cdot,0)$ is distinct from the rest because the states, X, are assumed known. This image, which we will refer to as the initialization image, defines the regions associated with each subassembly and will also be used to initialize the model parameters.

Ideally, we would like to compute the maximum likelihood estimates of θ and ϕ given the N training images $Y(\cdot,0)\ldots Y(\cdot,N-1)$. However, this would require a joint optimization over the entire object tree, which is too computationally complex. Instead, the estimates of $\theta^{(c)}$ and $\phi^{(c)}$ are computed at each node c using the N images and $\hat{x}_n^{(p)}$, the estimated parent state for image n.

$$(\hat{\theta}^{(c)}, \hat{\phi}^{(c)}) = \arg \max_{(\theta^{(c)},\phi^{(c)})} \prod_{n=0}^{N-1} p(y(\cdot,n)|X_n^{(p)} = \hat{x}_n^{(p)}, \theta^{(c)}, \phi^{(c)}). \quad (10)$$

As with the SMAP state estimation of section II-A, data information from descendants of node c is ignored.

Notice that (10) may be implemented as a sequence of optimizations at individual nodes. Since each optimization

```
1.  set l = max(l_{M_0}, l_0^{(c)})
2.  for all k ∈ X^{(c)}(α, l)
3.          compute L(k, l) and store (k, l) on the heap
4.  while the heap is nonempty
5.          extract the M largest likelihood ratios L(k_(1), l_(1)) ... L(k_(M), l_(M)) from the heap
6.          if for all i, l_(i) == l_0^{(c)}
7.                  if L(k_(1), l_(1)) > β_0 stop with match (k_(1), l_(1))
8.                  else stop with no match
9.          for i = 1 to M
10.                 if l_(i) > l_0^{(c)}
11.                         for all (k, l) ∈ next(k_(i), l_(i))
12.                                 compute L(k, l)
13.                                 if L(k, l) > β, stop with match (k, l)
14.                                 if L(k, l) > α, store (k, l) on the heap
15.                 else
16.                         store (k_(i), l_(i)) on the heap
17. stop with no match
```

Fig. 8. Multiscale search algorithm for inspection. Lines 1–3 initialize the heap data structure. Lines 6–8 check to see if all candidate nodes are at the finest resolution and, if they are, compare the maximum ratio to β_0. Lines 10–16 search children of candidate nodes.

depends on the estimated parent states $\hat{x}_n^{(p)}$, this sequence must proceed in order from root to leaves.

The difficulty in computing (10) is the missing state information $X_n^{(c)}$. Without this state at each image, we cannot determine the best state parameters $\phi^{(c)}$, or the template parameters $\theta^{(c)}$. The EM algorithm is specifically formulated to solve such "missing data" problems.

EM Algorithm

The EM algorithm works by computing a sequence of parameter estimates which converge to a local maximum of (10). The EM update equation is given by

$$(\hat{\theta}_{new}^{(c)}, \hat{\phi}_{new}^{(c)}) = \arg \max_{(\theta^{(c)}, \phi^{(c)})} \cdot \sum_{n=0}^{N-1} E[\log p(y(\cdot, n), X_n^{(c)} | X_n^{(p)} = \hat{x}_n^{(p)}, \theta^{(c)}, \phi^{(c)}) \, | \Gamma_n]$$

where

$$\Gamma_n = \{Y(\cdot, n) = y(\cdot, n), X_n^{(p)} = \hat{x}_n^{(p)}, \hat{\theta}_{old}^{(c)}, \hat{\phi}_{old}^{(c)}\}$$

and $\hat{\theta}_{old}^{(c)}$ and $\hat{\phi}_{old}^{(c)}$ are the parameters from the previous iteration. Using Bayes rule and noting that data parameters must be estimated for all subassembly resolutions $\leq l_0^{(c)}$, we get two separate update equations.

$$\hat{\theta}_{new}^{(c)} = \arg \max_{\theta^{(c)}} \sum_{n=0}^{N-1} \sum_{l=l_0^{(c)}}^{L-1} E[\log p(y_l(\cdot, n) | X_n^{(c)}, \theta^{(c)}) \, | \Gamma_n] \tag{11}$$

$$\hat{\phi}_{new}^{(c)} = \arg \max_{\phi^{(c)}} \sum_{n=0}^{N-1} E[\log p(X_n^{(c)} | X_n^{(p)} = \hat{x}_n^{(p)}, \phi^{(c)}) \, | \Gamma_n] \, . \tag{12}$$

Consider the state parameter update of (12). The update equations for the components of $\phi^{(c)} = [(m^{(c)})^t, (\gamma^{(c)})^t]^t$ can be computed by using the prior state density in (2), and then setting the derivative with respect to $\phi^{(c)}$ to zero. The update for the state means is given by

$$\hat{m}_{new}^{(c)} = \mathbf{A}^{-1} \frac{1}{N} \sum_{n=0}^{N-1} E[(X_n^{(c)} - \hat{x}_n^{(p)}) | \Gamma_n] \, .$$

The EM update equations will all contain expected values over the posterior-state density for node c in each training image. Each of these expectations can be approximated as a weighted sum over the sampled states at resolution $l_0^{(c)}$. For example,

$$\begin{aligned} E[(X_n^{(c)} - \hat{x}_n^{(p)}) | \Gamma_n] &\approx \frac{\sum_k (x(k, l_0^{(c)}) - \hat{x}_n^{(p)}) \, p\Big(x(k, l_0^{(c)}) | \Gamma_n\Big)}{\sum_k p\Big(x(k, l_0^{(c)}) | \Gamma_n\Big)} \\ &= \frac{\sum_k (x(k, l_0^{(c)}) - \hat{x}_n^{(p)}) \, \exp\Big\{L_d(k, l_0^{(c)}, n)\Big\}}{\sum_k \exp\Big\{L_d(k, l_0^{(c)}, n)\Big\}} \end{aligned}$$

where $L_d(k, l_0^{(c)}, n)$ is the log likelihood ratio associated with the pair (k, l) in training image n. While these sums may be computed, in our experience the likelihood ratio associated with the most likely state typically dominates by orders of magnitude, particularly for larger subassemblies and subassemblies containing fine detail. Note that even a modest difference in log likelihood ratios $L_d(\cdot)$ leads to a large difference in likelihood ratios $\exp\{L_d(\cdot)\}$. Thus, the expected values are approximated by the values corresponding to the most likely state, which is the state found by the multiscale search procedure. Formally, this approximation can be stated as

$$E[(X_n^{(c)} - \hat{x}_n^{(p)}) | \Gamma_n] \approx \hat{x}_n^{(c)} - \hat{x}_n^{(p)} \, .$$

This same approach is often taken when solving analogous expressions in speech and text recognition[9]. The update equation for the state means is then given by

$$\hat{m}_{new}^{(c)} = \mathbf{A}^{-1}\frac{1}{N}\sum_{n=0}^{N-1}(\hat{x}_n^{(c)} - \hat{x}_n^{(p)}) . \quad (13)$$

A similar method is used to compute the updates for the variance parameters $\gamma^{(c)}$. These updates are given by

$$\left[\hat{\gamma}_{s,new}^{(c)}, \hat{\gamma}_{z,new}^{(c)}, \hat{\gamma}_{r,new}^{(c)}\right]^t = \left[\frac{1}{2N}\sum_{n=0}^{N-1}\frac{\|\delta_s(n)\|^2}{\left(\hat{z}_n^{(p)}\right)^2}, \frac{1}{N}\sum_{n=0}^{N-1}\delta_z^2(n), \frac{1}{N}\sum_{n=0}^{N-1}\delta_r^2(n)\right]^t$$

where

$$[\delta_s(n), \delta_z(n), \delta_r(n)]^t = \hat{x}_n^{(c)} - \hat{x}_n^{(p)} - \mathbf{A}\hat{m}_{new}^{(c)}.$$

Now we need to compute the update equations for the data parameters from (11). Recall that a parameter $\lambda^{(c)}$ is used in the data model to account for intensity scaling of image regions, which is necessary for the log likelihood ratio computations. During training, however, all data variability among the training images is incorporated into the variability parameter estimates, $\hat{\sigma}_l(\cdot)$, so $\lambda^{(c)}$ becomes an arbitrary constant, which we set to one.

The template components $\mu_l(\cdot)$ and $\sigma_l(\cdot)$ can be expressed in terms of the parameters $\tilde{\mu}_l(\cdot)$ and $\tilde{\sigma}_l(\cdot)$ of equation (5) by performing the inverse of the transformations in (4). However, the transformations of (4) may not be strictly invertible, since the size of the transformed window $\tilde{W}_l^{(c)}$ may not be the same as the size of the untransformed window $W_l^{(c)}$. We avoid this problem by using bilinear interpolation on the data values to approximate the inverse of the bilinear interpolation in (4). Since each expectation in (11) is approximated by the value at the most likely state $\hat{x}_n^{(c)} = [(\hat{s}_n^{(c)})^t, \hat{z}_n^{(c)}, \hat{r}_n^{(c)}]^t$ for each training image n, the template components are computed as

$$\mu_l(i) \approx \tilde{\mu}_l\left((\mathbf{T}_n^{(c)})^{-1}i + 2^{-l}\hat{s}_n^{(c)}, n\right)(\mathbf{T}_n^{(c)})^{-1}$$
$$\sigma_l(i) \approx \tilde{\sigma}_l\left((\mathbf{T}_n^{(c)})^{-1}i + 2^{-l}\hat{s}_n^{(c)}, n\right)$$

where $[\tilde{\mu}_l(i,n), \tilde{\sigma}_l(i,n)]^t$ are the parameters corresponding to pixel $y_l(i,n)$ of training image n, and $\mathbf{T}_n^{(c)}$ is the transformation matrix evaluated at $\hat{x}_n^{(c)}$.

The image pixel values at the template component locations can be approximated via the same transformation. Let

$$\tilde{y}_l(i,n) = y_l\left((\mathbf{T}_n^{(c)})^{-1}i + 2^{-l}\hat{s}_n^{(c)}, n\right)(\mathbf{T}_n^{(c)})^{-1}.$$

Each expectation in (11) can now be approximated by the value at the most likely state $\hat{x}_n^{(c)}$, yielding a sum over the pixels in the window $\tilde{W}_l^{(c)}$. If this sum is thought of as an approximation to an integral over the window, a simple change of variables leads to a second approximation as a sum over the untransformed window $W_l^{(c)}$. Ignoring terms that do not depend on $\theta^{(c)}$, this gives (14) (at the bottom of this page), where the invariance of the 2-norm under rotation is used to obtain the final expression.

Substituting (14) into (11), the EM updates for the template parameters are given by

$$\hat{\mu}_{l,new}(i) = \arg\min_{\mu_l(i)} \sum_{n=0}^{N-1}(\hat{z}_n^{(c)})^3 \|\tilde{y}_l(i,n) - \mu_l(i)\| \quad (15)$$

$$\hat{\sigma}_{l,new}(i) = \frac{\sum_{n=0}^{N-1}(\hat{z}_n^{(c)})^3 \|\tilde{y}_l(i,n) - \mu_{l,new}(i)\|}{2\sum_{n=0}^{N-1}(\hat{z}_n^{(c)})^2}. \quad (16)$$

The computation of (15) would require a recursive implementation, so we approximate the update by assuming the scale factors are all near unity and replacing the 2-norm with a 1-norm. This gives the update

$$\hat{\mu}_{l,new}(i) = \text{Median}\{\tilde{y}_l(i,0), \ldots \tilde{y}_l(i,N-1)\}.$$

Since the EM algorithm is only guaranteed to converge to a local maximum of the likelihood equation, the final estimates can vary considerably depending on the initial starting point. We have devised a heuristic technique to compute initial parameter estimates. The state means are simply initialized to the known state values of the initialization image $Y(\cdot,0)$, with the scale factor and rotation angle defined to be unity and zero, respectively, for this image.

$$\hat{m}^{(c)} = x_0^{(c)} = [(s_0^{(c)})^t, 1, 0]^t.$$

Each template mean is initialized to one half the corresponding data value in the initialization image. Thus,

$$\hat{\mu}_l(i) = 0.5\tilde{y}_l(i,0).$$

$$\begin{aligned} E[\log p(y_l(\cdot,n)|X_n^{(c)}, \hat{\theta}^{(c)})\,|\Gamma_n] &\approx \sum_{i\in\tilde{W}_l^{(c)}}\left\{-2\log(\lambda^{(c)}\tilde{\sigma}_l(i,n)) - \frac{\|y_l(i,n) - \tilde{\mu}_l(i,n)\|}{\lambda^{(c)}\tilde{\sigma}_l(i,n)}\right\} \\ &\approx |(\mathbf{T}_n^{(c)})^{-1}| \sum_{i\in W_l^{(c)}}\left\{-2\log\sigma_l(i) - \frac{\|\tilde{y}_l(i,n)(\mathbf{T}_n^{(c)})^{-1} - \mu_l(i)(\mathbf{T}_n^{(c)})^{-1}\|}{\sigma_l(i)}\right\} \\ &= (\hat{z}_n^{(c)})^2 \sum_{i\in W_l^{(c)}}\left\{-2\log\sigma_l(i) - \frac{\hat{z}_n^{(c)}\|\tilde{y}_l(i,n) - \mu_l(i)\|}{\sigma_l(i)}\right\}, \end{aligned} \quad (14)$$

1. initialize parameter estimates $\hat{\theta}^{(c)}$ and $\hat{\phi}^{(c)}$
2. initialize $l_0^{(c)} = L - 1$
3. set *EM-iteration* $= 0$
4. while *EM-iteration* $< N_{EM}$ and $(\hat{\theta}_{new}^{(c)}, \hat{\phi}_{new}^{(c)}) \neq (\hat{\theta}_{old}^{(c)}, \hat{\phi}_{old}^{(c)})$
5. for $n = 1$ to $N - 1$
6. use multiscale search to compute state estimates $\hat{x}_n^{(c)}$ at resolution $l_n \leq l_0^{(c)}$
7. if *EM-iteration* $== 0$ and c is a leaf node, set $l_0^{(c)} = l = 0$
8. else set $l = \max_{n>0} l_n$
9. if $l > 0$, update $\hat{\theta}^{(c)}$ to resolution $l - 1$ using EM update equations
10. else update $\hat{\theta}^{(c)}$ to resolution l using EM update equations
11. if *EM-iteration* > 0
12. set $l_0^{(c)} = l$
13. update $\hat{\phi}^{(c)}$ using EM update equations
14. multiply $\hat{\gamma}_s^{(c)}$, $\hat{\gamma}_z^{(c)}$, and $\hat{\gamma}_r^{(c)}$ by $\left(\frac{2N}{2N-1}\right)$, $\left(\frac{N}{N-1}\right)$ and $\left(\frac{N}{N-1}\right)$, respectively, to remove bias
15. store $\hat{\theta}^{(c)}$ and $\hat{\phi}^{(c)}$ as model parameters for this subassembly, with $l_0^{(c)}$ as the finest model resolution

Fig. 9. EM algorithm for training. This procedure adapts the model to the variations seen in the training set.

In this way, the expected grayscale gradients $\mu_l(i)$ have the same direction as in the initialization image, but the gradient magnitudes can be smaller.

The remaining parameters characterize the expected variability of the data and state values, so their estimation from a single image is not so straightforward. The template variation parameters $\sigma_l(\cdot)$ are normally larger near edges in the image. These regions will also contain pixel values farther from zero, so the template σ_l is initialized to a smoothed version of the pixel norms in the initialization image. The following smoothing operation blurs the edges in the initialization image.

$$\hat{\sigma}_l = \{\hat{\sigma}_l(i)\}_{i \in W_l^{(c)}} = h(i) * (0.25 \, \|\tilde{y}_l(i,0)\|)$$

where $h(i)$ is a separable lowpass filter with one-dimensional filter weights of [0.25, 0.5, 0.25] and "$*$" denotes a two-dimensional convolution. This is useful since edge locations will typically vary slightly among images.

Finally, the state variances $\gamma^{(c)}$ are initialized to

$$\hat{\gamma}_s^{(c)} = \max\left(\left(\frac{\text{minimum template dimension}}{3}\right)^2, 16\right)$$

$$\hat{\gamma}_z^{(c)} = \hat{\gamma}_r^{(c)} = (0.04)^2 .$$

Note that this initialization implicitly assumes that the scale factor and rotation angle of the object will only vary to a limited extent from image to image.

The initial template values are based on only a single image, so they may be quite poor estimates of the parameters. The first EM iteration is therefore used to refine the estimates of the data parameters, but the state parameters and the value of $l_0^{(c)}$ are not changed during this iteration. In this way, the first iteration is essentially an initialization stage, giving a reasonable estimate of the template parameters based on the full set of training images.

The EM update scheme proceeds as shown in Fig. 9. We set $N_{EM} = 4$. The algorithm tends to converge to a fairly stable set of parameters by this point. Note that the finest model resolution $l_0^{(c)}$ is set to 0 for leaf nodes. These nodes are normally associated with subassemblies that are important for proper detection, so we force the algorithm to model these subassemblies at the finest resolution. The finest resolution for other nodes is initialized to $L-1$ and is set in a monotonically nonincreasing fashion during the training procedure.

Multiscale Search During Training

The search procedure used during the training phase differs in several ways from the procedure of Section III. We want to be sure the search returns the best possible state during training, so the magnitudes of the acceptance and rejection thresholds are increased to yield a slower, more conservative multiscale search. In particular, the acceptance threshold during training is set to $\beta_1 = 500$ and the rejection threshold to $\alpha_1 = -40$.

The object is assumed to be present in each of the training images, so the search is forced to terminate in a match. Ideally, this could be accomplished by setting the rejection threshold to negative infinity. In this case, however, none of the search paths would be pruned, and all candidate states would be added to the heap. During the early training iterations, the log likelihood ratio may never exceed β_1, so the search could be required to examine every possible candidate state before terminating. Consequently, we prune the search space by dynamically adjusting the rejection threshold during the search.

The search procedure during training is described in Fig. 10. The search terminates when it encounters a state at fine enough resolution with log likelihood ratio greater than β_1, or when all search paths have been completely examined or discarded. The state corresponding to the best likelihood ratio is returned as the node location.

After the training algorithm has converged, the variance estimates $\hat{\gamma}^{(c)}$, which correspond to maximum likelihood estimates, are biased. This bias is removed by multiplying each variance estimate by the appropriate fraction, $2N/(2N-1)$ for $\hat{\gamma}_s^{(c)}$ and $N/(N-1)$ for $\hat{\gamma}_z^{(c)}$ and $\hat{\gamma}_r^{(c)}$.

1. set $l = \max(l_{M_0}, 0)$
2. for all $k \in \mathcal{X}^{(c)}(\alpha_1, l)$
3. compute $L(k, l)$ and store (k, l) on the heap
4. set $L_{max} = -\infty$
5. while the heap is nonempty
6. extract the M largest likelihood ratios $L(k_{(1)}, l_{(1)}) \ldots L(k_{(M)}, l_{(M)})$ from the heap
7. for $i = 1$ to M
8. if $l_{(i)} > 0$
9. for all $(k, l) \in \text{next}(k_{(i)}, l_{(i)})$
10. compute $L(k, l)$
11. if $L(k, l) > L_{max}$ and $l \leq l_0^{(c)}$
12. set $L_{max} = L(k, l)$
13. set $(k_{max}, l_{max}) = (k, l)$
14. if $L(k, l) > \beta_1$ and $l \leq l_0^{(c)}$, stop with match (k, l)
15. if $L_{max} > 0$, $\tilde{\alpha} = 0.25^{l - l_{max}} L_{max} + \alpha_1$
16. else $\tilde{\alpha} = L_{max} + \alpha_1$
17. if $L(k, l) > \tilde{\alpha}$, store (k, l) on the heap
18. stop with match (k_{max}, l_{max})

Fig. 10. Multiscale search for training. The search is guaranteed to terminate in a match.



(a) (b)

Fig. 11. Training images for gear assembly. The boxes indicate the subassembly locations determined during training.

V. Simulation Results

We have used this algorithm to inspect two different real assemblies. The images used are eight bit monochrome NTSC images obtained from a standard camcorder. All simulations were run on a Sparc 10 workstation.

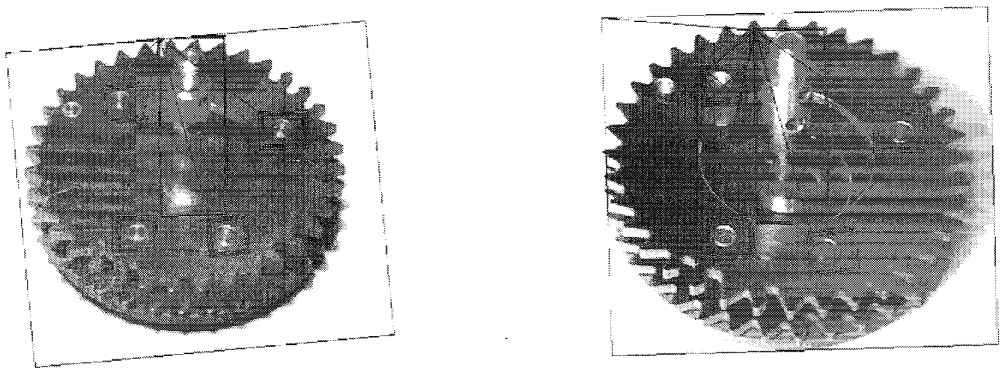
Fig. 2 shows the initialization image for the first assembly inspected. The subassemblies making up the object tree are drawn as boxes in the figure, and the lines connecting the boxes illustrate the object tree connections. The object tree contains six nodes in all for this assembly. The algorithm was first trained using this initialization image and five additional training images. Two of the training images are illustrated in Fig. 11, where the boxes indicate the state estimates during the last EM iteration. As the figure shows, the training algorithm located each of the subassemblies correctly.

Fig. 12 illustrates the output of the algorithm for two different test images. The object in Fig. 12(a) was assembled correctly, and the algorithm located each of the subassemblies, so this object passes inspection.

Fig. 12(b) shows an incorrectly assembled object. For this image the top plate of the assembly is not fully seated on the pins, but is tilted slightly toward the camera. This error causes the top of the left rear pin to be lower than the surface of the plate. Since the appearance of this pin does not match the training images, this image fails inspection. The algorithm indicates the error point by drawing a box with an "X" through it at the expected subassembly position.

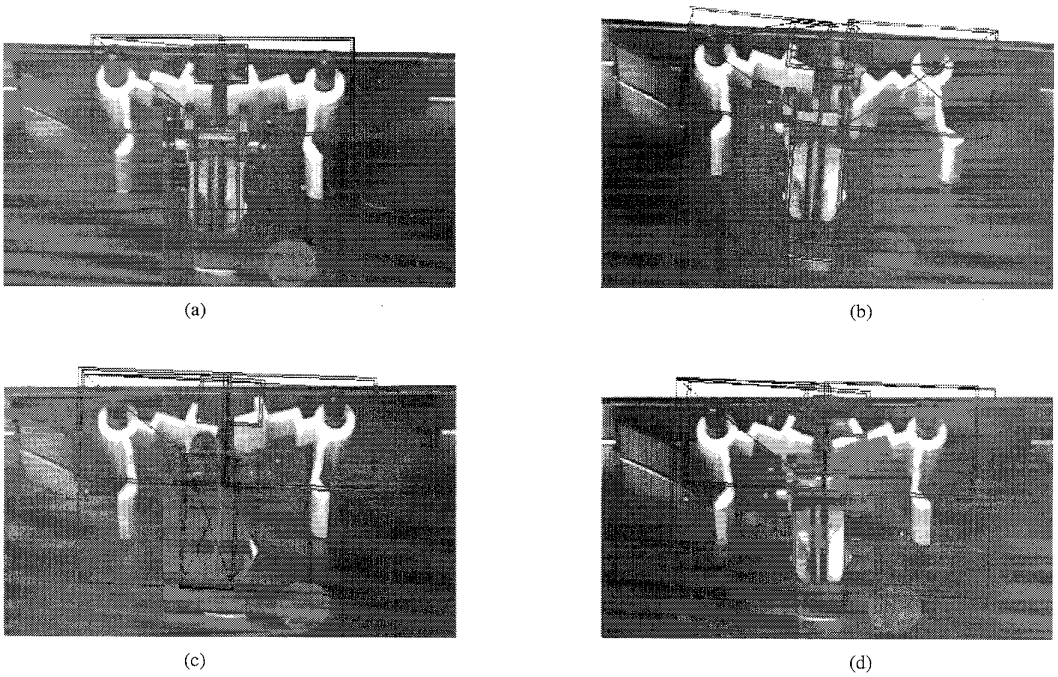
The second assembly to be inspected is a VHS video cassette. We zoomed in on the portion of the cassette that we wish to inspect and took a number of images of correctly and incorrectly assembled pieces. The manually constructed initialization image is shown in Fig. 13(a). For this assembly the object tree contains five nodes.

The algorithm was trained for this assembly using a total of five training images. The resulting model was then used to inspect a variety of images. Fig. 13(b)–(d) show the algorithm results for three images of incorrectly assembled cassettes. For the images of Fig. 13(b) and (c), the inspection algorithm correctly locates and flags the error.



(a) (b)

Fig. 12. Test images for gear assembly. (a) correctly assembled part; (b) misassembled part.



(a) (b)

(c) (d)

Fig. 13. VHS cassette images. (a) initialization image; (b)–(d) test images. Note that the error in image (d) goes undetected.

The image in Fig. 13(d) contains a more subtle error. The subassembly enclosed by the smallest box is a small metal spring. In Fig. 13(d), this spring is not fully inserted into the proper slot. However, since the part is quite small and largely occluded by the other assembly parts, the inspection algorithm fails to detect this error. The primary features inside this small box are the edges of the occluding parts and the edge of the cassette. Since these features match the model, the small errors caused by the misinserted spring are disregarded. In general, the algorithm can have difficulty detecting small features that have no sharp edges, particularly if they lie in areas of high activity in the image. This tendency can be reduced to some extent by using feature-shaped subassembly windows, but for small features this will reduce the number of pixels in the window even more, and any occlusion problems will remain. The algorithm also tends to overestimate the scale factor for small features. This tendency is illustrated by Table I, where node 4 corresponds to the small spring. Note that all other scale and rotation errors are quite modest. The error statistics in Table I were computed from a set of test images obtained by rotating the initialization image through a number of known angles ranging from $-10°$ to $10°$. In this way, the nominal scale factor and rotation angle were known for each test image, so the estimation errors could be computed.

Our algorithm implicitly assumes that the large majority of test images will contain correctly assembled objects. For a misassembled object, the search must discard all candidate search paths before it can terminate with no match. Thus, the amount of computation required for an object with assembly errors is typically much greater than that required for a

$$E[(X^{(c)} - \hat{x}^{(p)} - \hat{\mathbf{A}}m^{(c)})^t(X^{(c)} - \hat{x}^{(p)} - \hat{\mathbf{A}}m^{(c)})|\hat{X}^{(p)} = \hat{x}^{(p)}, \phi^{(c)}]$$
$$= E[\mathbf{B}|\hat{X}^{(p)} = \hat{x}^{(p)}, \phi^{(c)}] + E[QQ^T|\hat{X}^{(p)} = \hat{x}^{(p)}] +$$
$$E[(\mathbf{A} - \hat{\mathbf{A}})m^{(c)}(m^{(c)})^t(\mathbf{A} - \hat{\mathbf{A}})^t|\hat{X}^{(p)} = \hat{x}^{(p)}, \phi^{(c)}] \quad (17)$$

TABLE I
SCALE AND ROTATION ERRORS FOR VHS CASSETTE THE INITIALIZATION IMAGE WAS ROTATED THROUGH DIFFERENT ANGLES RANGING FROM $-10°$ TO $10°$ AND USED AS A TEST IMAGE TO COMPUTE ERROR STATISTICS FOR SCALE FACTOR AND ROTATION ANGLE. ALL TEST IMAGES HAD A NOMINAL SCALE FACTOR OF ONE

		Node 0	Node 1	Node 2	Node 3	Node 4
Scale	Mean	0.0517	0.0983	0.0475	-0.0192	0.3422
Factor	Std Dev	0.0714	0.1044	0.0539	0.0469	0.3429
Rotation	Mean	0.0000	-0.0063	-0.0063	-0.0044	0.0043
(radians)	Std Dev	0.0447	0.0230	0.0230	0.0239	0.0219

TABLE II
COMPUTATION FOR MULTISCALE SEARCH

Object	Avg CPU Time	Avg # Touches per Pixel
Gear Assembly	34.6 seconds	3.49
VHS Cassette	44.8 seconds	3.16

correctly assembled object. However, we do not consider this to be a problem since for our application, most of the inspected objects should be correctly assembled.

We measure the required computation for the algorithm in two ways. The first measurement is a simple recording of the required CPU time for testing the algorithm on a correctly assembled part. A second complexity measure is to count the average number of times each image pixel is "touched" during the multiscale search. This number is incremented for a particular pixel each time the pixel contributes to the log likelihood ratio computation. In this way, we get a measure of complexity that indicates the average number of times each image pixel is used, which is independent of the particular architecture on which the algorithm is implemented.

The inspection algorithm was run on a total of ten images of correctly assembled objects, none of which were included in the training set, for each of the two assemblies. The average CPU time and average number of times each pixel is touched are given in Table II.

VI. CONCLUSION

Stochastic model-based techniques can be effective for object detection, particularly in a highly structured environment. The procedure presented here demonstrates some of the principal characteristics of such a system, but the algorithm could be improved in a number of ways. The multiscale search and the parameter estimation procedure could both be made more efficient, and more accurate models could improve performance.

VII. APPENDIX A

In this appendix we derive the state variance correction term $\mathbf{B}_Q(\hat{X}^{(p)}, \phi^{(c)}, l^{(p)}, l_0^{(p)})$. We will assume the component quantization errors in Q are zero mean, uniform, and independent.

The adjusted variances in $\hat{\mathbf{B}}$ are the diagonal elements of (17), at the top of this page, where $\hat{\mathbf{A}}$ is the $\mathbf{A}$ matrix evaluated at $\hat{x}^{(p)}$.

The first two terms can be computed in a straightforward fashion to give

$$E[\mathbf{B}|\hat{X}^{(p)} = \hat{x}^{(p)}, \phi^{(c)}]$$
$$= \text{Diag}\left(\left[\left(\hat{z}^{(p)}\right)^2 + q\right]\gamma_s^{(c)}, \left[\left(\hat{z}^{(p)}\right)^2 + q\right]\gamma_s^{(c)}, \gamma_z^{(c)}, \gamma_r^{(c)}\right)$$
$$E[QQ^T|\hat{X}^{(p)} = \hat{x}^{(p)}] = \text{Diag}\left(\left(\frac{d^{(p)}}{2}\right)^2 q, \left(\frac{d^{(p)}}{2}\right)^2 q, q, q\right)$$

where

$$q = \frac{4^{l^{(p)}} - 4^{l_0^{(p)}}}{12(d^{(p)})^2}.$$

The nonlinear cosine and sine terms in $\mathbf{A}$make the third term of (17) more difficult to compute. We therefore approximate the diagonal components of this term using a Taylor series expansion and ignoring all powers of $(r^{(p)} - \hat{r}^{(p)})$ larger than two. After performing this calculation, the state variance correction term is given by

$$\mathbf{B}_Q(\hat{X}^{(p)}, \phi^{(c)}, l^{(p)}, l_0^{(p)}) = \text{Diag}(b_{Q,v}, b_{Q,h}, b_{Q,z}, b_{Q,r})$$

where

$$b_{Q,v} = q\gamma_s^{(c)} + \frac{(d^{(p)})^2}{4}q + \frac{1}{(\hat{z}^{(p)})^2}$$
$$(q(1-q)\hat{C}_v^2 + q((\hat{z}^{(p)})^2 + q)\hat{C}_h^2)$$
$$b_{Q,h} = q\gamma_s^{(c)} + \frac{(d^{(p)})^2}{4}q + \frac{1}{(\hat{z}^{(p)})^2}$$
$$(q(1-q)\hat{C}_h^2 + q((\hat{z}^{(p)})^2 + q)\hat{C}_v^2)$$
$$b_{Q,z} = q$$
$$b_{Q,r} = q$$

and $[\hat{C}_v, \hat{C}_u]^t = (\hat{\mathbf{T}}^{(p)})^{-1}m_s^{(c)}$, with $\hat{\mathbf{T}}^{(p)}$ equal to the transformation matrix $\mathbf{T}$ evaluated at $\hat{x}^{(p)}$.

REFERENCES

[1] A. K. Jain, "Advances in Mathematical Models for Image Processing," in *Proc. IEEE*, vol. 69, no. 5, pp. 502–528, May 1981.
[2] R. Chellappa and R. L. Kashyap, "Digital Image Restoration Using Spatial Interaction Models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 30, no. 3, pp. 461–471, June 1982.

[3] J. Besag, "On the Statistical Analysis of Dirty Pictures," *J. Roy. Statist. Soc. B.,* vol. 48, no. 3, pp. 259–302, 1986.
[4] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. Image Processing,* vol. 3, no. 2, pp. 162-177, Mar. 1994.
[5] A. Benveniste, R. Nikoukhah, and A. S. Willsky, "Multiscale system theory," in *Proc. 29th Conf. Decision and Control,* Dec. 1990, pp. 2484–2489.
[6] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky, "Modeling and estimation of multiresolution stochastic processes," *IEEE Trans. Inform. Theory,* vol. 38, no. 2, pp. 766–784, Mar. 1992.
[7] Y. Amit, U. Grenander, and M. Piccioni, "Structural image restoration through deformable templates," *J. Amer. Stat. Assoc.,* vol. 86, no. 414, pp. 376–387, June 1991.
[8] A. C. Kam and G. E. Kopec, "Heuristic image decoding using separable source models," in *Proc. Int. Conf. Acoust., Speech, Signal Processing,* vol. 5, Adelaide, Australia, Apr. 19–22, 1994, pp. 145–148.
[9] G. E. Kopec and P. A. Chou, "Document image decoding using Markov source models," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 16, no. 6, pp. 602–617, June 1994.
[10] R. T. Chin and C. A. Harlow, "Automated visual inspection: A survey," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 4, no. 6, pp. 557–573, Nov. 1982.
[11] R. A. Brooks, "Model-based three-dimensional interpretations of two-dimensional images," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PMI-5, no. 2, pp. 140–150, Mar. 1983.
[12] P. J. Flynn and A. K. Jain, "CAD-based computer vision: From CAD models to relational graphs," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 13, no. 2, pp. 114–132, Feb. 1991.
[13] R. Mehrotra and W. I. Grosky, "Shape matching utilizing indexed hypotheses generation and testing," *IEEE Trans. Robotics Automat.,* vol. 5, no. 1, pp. 70–77, Feb. 1989.
[14] A. Rosenfeld and G. J. Vanderbrug, "Coarse-fine template matching," *IEEE Trans. Syst., Man, Cybern.,* pp. 104–107, Feb. 1977.
[15] A.D. Gross and A. Rosenfeld, "Multiresolution Object Detection and Delineation," *Computer Vision, Graphics, Image Processing,* vol. 39, pp. 102–115, 1987.
[16] J. L. Crowley and A. C. Sanderson, "Multiple resolution representation and probabilistic matching of 2-D gray-scale shape," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 9, no. 1, pp. 113–121, Jan. 1987.
[17] S. Morita, T. Kawashima, and Y. Aoki, "Pattern matching of 2-D shape using hierarchical descriptions," *Syst. Computers in Japan,* vol. 22, no. 10, pp. 40–49, 1991.
[18] P. J. Burt, "Smart sensing within a pyramid vision machine," in *Proc. IEEE,* vol. 76, no. 8, Aug. 1988, pp. 1006–1015.
[19] G. J. Ettinger, "Large hierarchical object recognition using libraries of parameterized model sub-parts," in *Proc. Computer Soc. Conf. Computer Vision Pattern Recognition,* pp. 32–41, Ann Arbor, MI, June 5–9, 1988.
[20] D. R. Tretter and C. A. Bouman, "Multiscale stochastic approach to object detection," *SPIE Visual Commun. Image Processing '93,* pp. 1219–1230, Cambridge, MA, Nov 8–11, 1993.
[21] D. R. Tretter, K. W. Khawaja, C. A. Bouman, and A. A. Maciejewski, "A CAD driven multiscale approach to automated inspection," in *Proc. Int. Conf. Acoust., Speech, Signal Processing,* vol. 5, Adelaide, Australia, Apr. 19–22, 1994, pp. 397–400.
[22] ———, "Automated assembly inspection using a multiscale algorithm trained on synthetic images," in *Proc. IEEE Int. Conf. Robotics Automat.,* vol. 4, San Diego, CA, May 8–13, 1994, pp. 3530–3536.
[23] K. T. Gunnarsson and F. B. Prinz, "CAD model-based localization of parts in manufacturing," *Computer,* vol. 20, no. 8, pp. 66–74, Aug. 1987.
[24] H. Shariat, "A model-based method for object recognition," in *Proc. IEEE Int. Conf. Robotics Automat.,* Cincinnati, OH, May 13–18, 1990, pp. 1846–1851.
[25] H. R. Keshavan, J. Barnett, D. Geiger, and T. Verma, "Introduction to the special section on probabilistic reasoning," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 15, no. 3, pp. 193–195, Mar. 1993.
[26] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B.,* vol. 39, no. 1, pp. 1–38, 1977.
[27] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann, 1988.
[28] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 11, no. 7, pp. 674–693, July 1989.
[29] R. J. Clarke, *Transform Coding of Images.* Orlando, FL: Academic Press, 1985.



Daniel Tretter (S'87–M'88–S'90–S'93–M'95) received the B.S. degree in electrical engineering and mathematics from Rose-Hulman Institute of Technology, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, in 1988 and 1994, respectively.

From 1989–1990, he was a member of the technical staff at The Aerospace Corporation. He is currently employed at Hewlett-Packard Laboratories, Palo Alto, CA. His current research interests include multiscale image processing and image modeling.



Charles A. Bouman (S'86–M'89) received the B.S. degree in electrical engineering from the University of Pennsylvania, in 1981, and the M.S. degree in electrical engineering from the University of California at Berkeley, in 1982. In 1987 and 1989, respectively, he received the M.A. and Ph.D. degrees in electrical engineering from Princeton University under the support of an IBM graduate fellowship.

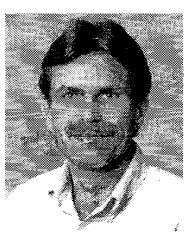
From 1982–1985, he was a staff member in the Analog Device Technology Group at the Massachusetts Institute of Technology, Lincoln Laboratory. In 1989, he joined the faculty of the School of Electrical Engineering at Purdue University as an assistant professor. His research interests include statistical image modeling, multiscale processing, and the display and printing of images. He is particularly interested in the applications of statistical signal processing techniques to problems such as color half-toning, tomographic reconstruction, multispectral segmentation, and fast image search. He has performed research for numerous government and industrial organizations including the National Science Foundation, U.S. Army, Hewlett-Packard, NEC Corporation, Apple Computers, Xerox, and Eastman Kodak.

Dr. Bouman was also an NEC Faculty Fellow, from 1991–1993. He is a member of SPIE and IS&T professional societies. He has been both chapter chair and vice chair of the IEEE Central Indiana Signal Processing Chapter. Currently, he is an associate editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, and a member of the 1996 Image and Multidimensional Signal Processing organizing committee.



Khalid W. Khawaja was born in 1967. He received both the B.S. degree in computer and electrical engineering and the M.S. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1990, respectively. He is currently completing a Ph.D. degree in electrical engineering at Purdue University.

His research interests include computer graphics applications in industrial automation, computer animation, and CAD.



Anthony A. Maciejewski received the B.S., M.S., and Ph.D. degrees in electrical engineering from Ohio State University, Columbus, OH, in 1982, 1984, and 1987, respectively.

Since 1988, he has been with the School of Electrical Engineering at Purdue University, West Lafayette, IN, where he is currently an associate professor. His primary research interests center on the simulation and control of kinematically redundant robotic systems.