A CAD DRIVEN MULTISCALE APPROACH TO AUTOMATED INSPECTION †

Daniel Tretter, Khalid Khawaja, Charles A. Bouman, and Anthony A. Maciejewski
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907-1285

ABSTRACT

In this paper we develop a general multiscale stochastic object detection algorithm for use in an automated inspection application. Information from a CAD model is used to initialize the object model and guide the training phase of the algorithm. An object is represented as a stochastic tree, where each node of the tree is associated with one of the various object components used to locate and identify the part. During the training phase a number of model parameters are estimated from a set of training images, some of which are generated from the CAD model. The algorithm then uses a fast multiscale search strategy to locate and identify the subassemblies making up the object tree. We demonstrate the performance of the algorithm on a typical mechanical assembly.

1. INTRODUCTION

One important component of an automated assembly system is a fast, reliable automated inspection module. Automated assembly processes are often designed with the aid of a CAD model of the manufactured part. This model contains specifications for each of the subassemblies making up the part as well as information describing how the subassemblies fit together. The CAD model can provide useful parameters and tolerances to the inspection module, which can then be tailored to a specific inspection task with minimal human intervention.

We approach the inspection task as a problem in object recognition, where we wish to detect and distinguish error-free objects from those with assembly errors. In this work we assume that our inspection system is provided with a single monochrome image of each object to be inspected. We also desire to minimize the amount of human intervention required during the model building and training phase of the algorithm, preferring to allow the CAD system to drive the training. We feel that this approach is desirable to produce a general purpose algorithm that will work well under a variety of conditions. Also, by using the CAD model to guide the inspection process, the assembly and inspection processes can be designed concurrently[1].

A number of authors have developed a variety of approaches to automated inspection, but many of them are tailored to a specific application or do not rely on a probabilistic object model[2]. In this work, we develop a consistent stochastic approach to the problem of fast 2D object detection and recognition in monochrome images. Since we wish to develop a general framework that will apply in a variety of situations, our approach will involve a number of parameters that must be estimated from training data. We will use a CAD model of the object to guide the design and training of the object model.

We devise a multiresolution model of the object, where various object features are represented at different resolutions. We use some recent results in the study of multiscale random fields to aid us in our model development and analysis[3, 4]. Object detection is then performed as a multiscale search procedure, generally progressing from coarse object features to finer details. A similar search is employed during the training phase, in which model parameters are estimated. Finally, we implement this procedure on a gear assembly to demonstrate its performance.

2. MULTISCALE OBJECT DETECTION

Our inspection algorithm models an object as a stochastic tree, where the nodes of the tree represent various components, or subassemblies, of the object. These subassemblies contain the key features for discrimination and error detection. Nodes near the root of the tree typically model larger structures that aid in locating the object while nodes further down "zoom in" on the critical areas where assembly errors are likely to occur[5]. The object model structure is similar to the one we described in [6], but many of the details concerning the data model are different here.

Suppose the nominal appearance of a subassembly is given by a multiresolution template θ, which may also be thought of as a deterministic parameter vector for the data y. We will represent the position and orientation of the subassembly as a state vector S. Since generally the subassembly's position is not known, S will be considered random. Finally, a set of state parameters ϕ is used to characterize the allowed variations in the subassembly's position. Each box in Figure 1 represents a single subassembly. The relationship among y, S, θ, and ϕ is shown for each subassembly, with arrows indicating conditional dependence. Ignoring for the moment the dependence on other subassemblies, the relationship for a single subassembly can be written as

$$p(y, S|\phi, \theta) = p(y|S, \theta)p(S|\phi).$$

Note that our probability notation only distinguishes between random and deterministic quantities through context.

We speed the search for the subassembly by using the multiresolution Haar transform representation for both the image data and the template. Since the Haar transform represents horizontal and vertical frequencies separately, the template θ_l at each resolution l actually consists of templates for both the horizontal and vertical components of the data y_l at that resolution. Successive resolutions of the Haar transform are obtained by successive filtering and subsampling operations, so at resolution l the data in each component has been subsampled by 2^{l+1}. The coarsest res-

†This work was supported by an AT&T Bell Laboratories PhD 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.

0-7803-1775-0/94 $3.00

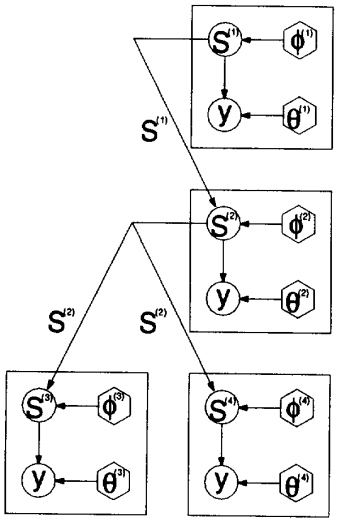

Figure 1. The object tree links together subassembly models

olution of the image or template is chosen so that the width and height have no fewer than 4 samples.

We will search for the most likely position S of the subassembly by starting at a coarse resolution and progressing to finer resolutions. At each resolution l we use the image data and templates $\{\bar{y}_l, \bar{\theta}_l\} = \{y_k, \theta_k : k \geq l\}$. We use this information at each position S and resolution l to compute the log likelihood ratio between the hypothesis that the subassembly is present and the hypothesis that it is not. If $p_1(\cdot)$ is the density function assuming the subassembly is present and $p_0(\cdot)$ is the density function assuming it is not, then we define the log likelihood ratio

$$L_l(S|\bar{\theta}_l, \phi) = \log\left\{\frac{p_1\left(\bar{y}_l, S|\bar{\theta}_l, \phi\right)}{p_0\left(\bar{y}_l\right)}\right\} = \log\left\{\frac{p_1\left(\bar{y}_l|S, \bar{\theta}_l\right)}{p_0\left(\bar{y}_l\right)}\right\} + \log p(S|\phi). \quad (1)$$

As in [6], we assume that S contains four conditionally independent Gaussian components: vertical position; horizontal position; scale factor; and rotation.

We compute the data term of (1) by assuming the data set $\{\bar{y}_l\}$ consists of independent Laplacian samples. Each location of the template θ_k contains a vertical band mean $\mu_v(i)$, a horizontal band mean $\mu_h(i)$, and a scale σ_i assumed to be common to that location for both the vertical and horizontal Haar bands at resolution k. Thus, if the subassembly is present, each data value can be normalized by subtracting the appropriate mean and dividing by the scale factor. The resulting values will then be identically distributed Laplacian random variables. If the subassembly is not present, the original data values are assumed to be identically distributed with zero mean. If we estimate the parameters of the two Laplacians by their maximum likelihood estimates, the data term of (1) becomes

$$\log\left\{\frac{p_1\left(\bar{y}_l|S, \bar{\theta}_l\right)}{p_0\left(\bar{y}_l\right)}\right\} = |\mathcal{W}_S^{(l)}| \log\left\{\frac{\hat{\lambda}_0}{\hat{\lambda}_1}\right\} - \sum_{i \in \mathcal{W}_S^{(l)}} \log \sigma_i \,,$$

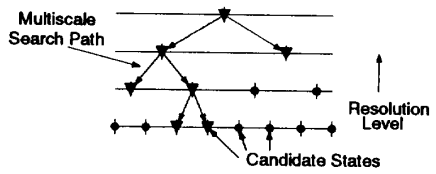

Figure 2. Sample state space more finely at fine resolution

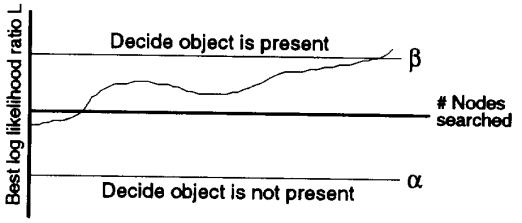

Figure 3. A sequential likelihood ratio test terminates when the log likelihood ratio exceeds the range $[\alpha, \beta]$

where $\mathcal{W}_S^{(l)}$ is the set of pixels at resolution $\geq l$ that lie within the template at position S, and

$$\hat{\lambda}_0 = \frac{1}{|\mathcal{W}_S^{(l)}|} \sum_{i \in \mathcal{W}_S^{(l)}} |y(i)| \,,$$

$$\hat{\lambda}_1 = \frac{1}{|\mathcal{W}_S^{(l)}|} \sum_{i \in \mathcal{W}_S^{(l)}} \frac{|y(i) - \mu_j(i)|}{\sigma_i} \,,$$

with $j \in (v, h)$. Intuitively, this means we should choose the hypothesis that gives a distribution with lower variance.

Using this log likelihood ratio as our "goodness of fit" measure, we perform a multiscale search of the state space to determine the position of the subassembly. The prior term of (1) is used to restrict the search space. The allowable state space is then sampled coarsely and searched completely at coarse resolution. After this, the search proceeds generally from coarse to finer resolutions, with the best matches at coarse resolution being expanded to the next finer resolution, as illustrated for a one dimensional state space in Figure 2. The state space is also sampled more finely at these resolutions, allowing the search to gradually refine its estimate of the subassembly's position.

Figure 3 shows how the search method is similar to a sequential likelihood ratio test that progresses in scale rather than time. Acceptance and rejection thresholds determine the stopping condition for the search. This search technique is discussed in more detail in [6].

Figure 1 illustrates our object model, which links the various subassembly models in a tree structure. Nodes near the root of the tree are typically associated with larger subassemblies, while those further down model the finer details of the object. The nodes are linked together in a top down fashion, with the state of each node dependent on the state of its parent node. Intuitively, this means that a node inherits the position and orientation of its parent node. Therefore, the node states S form a Markov chain along any path from the root to a leaf of the tree.

Figure 4 shows an example of an object tree for a gear assembly. Each box represents a subassembly or node of the tree. The level of each node is indicated by the number

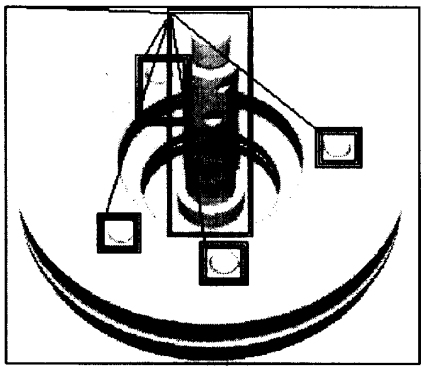

Figure 4. CAD generated synthetic image with object tree marked.

of lines making up the box, and connecting lines indicate parent-child relationships.

As before, we assume the state components of a given node are conditionally independent given the state of its parent node. We also assume each node has separate image data associated with it. This is not strictly true, since subassemblies can overlap. Overlapping subassemblies will generally be represented at different resolutions, however, since larger subassemblies often use only coarse resolutions. Under this assumption the only interaction between the nodes is through the state linkages of the object tree. Thus, we can generalize the log likelihood expression for node c with parent node p as

$$L_l(S^{(c)}|S^{(p)},\bar{\theta}_l^{(c)},\phi^{(c)}) = \log\left\{\frac{p_1\left(\bar{y}_l|S^{(c)},\bar{\theta}_l^{(c)}\right)}{p_0\left(\bar{y}_l\right)}\right\} + \log p\left(S^{(c)}|S^{(p)},\phi^{(c)}\right).$$

We estimate the states for the tree using Bouman and Shapiro's sequential MAP (SMAP) procedure[4]. Starting at the root of the tree, we estimate the state of each node in turn, passing the estimate to any child nodes as the parent state. This procedure is explained in more detail in [6].

3. TRAINING (PARAMETER ESTIMATION)

The object model is trained using images of the object and an initial tree structure as shown in Figure 4. As discussed in section 4, the tree structure and initial parameters are obtained from the CAD model. The model parameters $(\theta^{(c)},\phi^{(c)})$ are then estimated for each node c from a set of training images using an iterative procedure. As in the state estimation procedure, we estimate the parameters for each node in turn starting at the root of the object tree. Each iteration steps through the entire set of training images, estimating the position of the subassembly in each image. The parameter estimates are then updated based on the image data at these locations. After the parameter estimates have converged, the node state for each training image is passed to any child nodes as the parent state.

The estimation problem at each node can be viewed as a "missing data" problem, where the missing data is the state of the node $S_n^{(c)}$ for each training image n. We estimate the parameters for each node via the well known expectation maximization (EM) procedure, which is especially formulated to deal with such problems. The resulting iterative parameter updates are similar to those given in [6].

4. USE OF CAD MODEL FOR TRAINING

An automated assembly process is normally designed with the aid of a CAD model of the manufactured object. The CAD information can be used to generate synthetic images of the object under a variety of workcell conditions and component specifications[7]. In particular, the CAD model is first used to select lighting conditions and an effective viewing angle for the inspection. The model contains information concerning the acceptable tolerances of each object component, so training images can be generated with the components of the assembly at the limit of their acceptable dimensions. Relationships among the different components deduced from the CAD model can aid in the design of the object tree structure. These capabilities and the image generation procedure are discussed in more detail in [7].

We use the CAD model to first generate a nominal initialization image, as illustrated in Figure 4. The model specifies the location and extent of the object and its component parts, which are linked together to form the object tree. This image is also used to generate initial parameter estimates $(\hat{\theta},\hat{\phi})$ for the training algorithm.

We then run the training algorithm on a set of two training images. One of the images is the synthetic initialization image, for which the states are known. The other image is a real picture of the manufactured object. After the training algorithm has converged, the state estimates for the second image are taken to be the actual states for that image, and the new model parameters are used to initialize the algorithm for a larger training set. The new training set, which is a mixture of real and synthetic images, contains the original two images as well as a number of other training images. The initial parameter estimates are now characteristic of both real and synthetic images, so the training algorithm converges to good state estimates for all of the training images. After training, the final model parameters are stored and used for testing other real images.

We would like the object detection algorithm to be robust to changes in image brightness and noise level. In particular, real images will contain much more noise than synthetically generated images. In order to adjust for these image differences, we use overall image statistics to estimate a gain and noise parameter for each image. When computing log likelihood ratios during training or testing, the data parameters θ are adjusted for each image according to these parameters.

5. SIMULATION RESULTS

Figure 4 shows the initialization image used to define our object tree, with the various nodes of the tree indicated by the boxes. The algorithm first searches for the gear assembly at coarse resolution. The resulting location is then used to guide the search for the shaft, which proceeds to a finer resolution. Finally, the shaft position guides the search for the four pins.

The training algorithm is first run on only the initialization image and a single real image. Figure 5 illustrates the final positions found for each subassembly in the real image. The location of the root node is determined less precisely than the other subassemblies since the search for this node only proceeds at coarse resolutions.

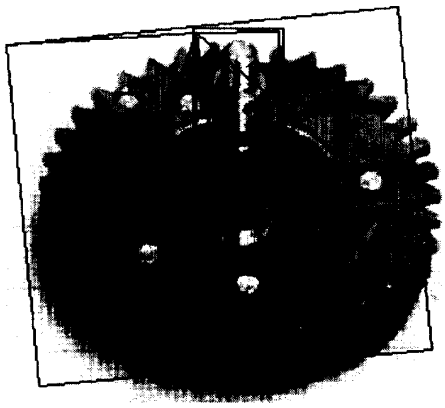

Figure 5. Real image used for training

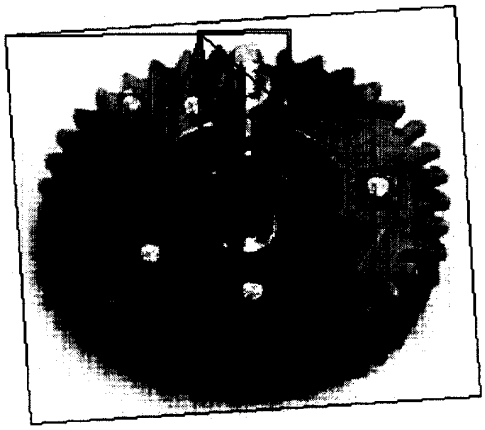

Figure 6. Results for correctly assembled test image

The parameter estimates obtained from this small training set are then used to initialize the algorithm for a larger training set. We ran the second stage of training on two different image sets. Training set A consists of the two original images plus two additional real images. Training set B contains the four images of training set A plus two additional synthetic images. The object's three-dimensional position and orientation changes slightly from image to image for each training set.

Each training set gives rise to a different set of model parameters, which are used on two new test images. The final results were comparable for the two training sets, but the algorithm ran about 5 to 10 times faster using set A. Since this training set consists primarily of real images, the associated data parameters match the test data more closely. This results in a faster, more efficient search for the subassemblies. The results shown here are for training set A.

Figure 6 shows the result on an image in which the object is properly assembled, while Figure 7 illustrates the result for an object in which one of the pins is missing. The algorithm indicates the missing node by drawing a box with an "X" through it at the expected node location. The test algorithm locates the correctly assembled part in 6.5 seconds

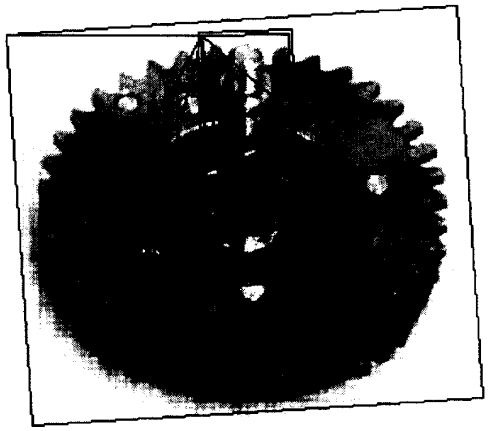

Figure 7. Results for test image with a missing pin

of CPU time on a Sparc 10 workstation. For the misassembled part, the search for the missing pin traverses many possible states before deciding that the node is missing. The algorithm takes a total of 22.6 seconds for this part.

6. CONCLUSION

We have developed a multiscale object detection algorithm for use in an automated inspection application. The algorithm is quite general, using CAD information to guide the model generation and training. After training, the algorithm can be used to detect assembly errors.

REFERENCES

[1] M. Marefat, S. Malhotra, and R.L. Kashyap, "Object-Oriented Intelligent Computer-Integrated Design, Process Planning, and Inspection," *Computer*, vol. 26, no. 3, pp. 54-65, March 1993.

[2] R. Hoffman and H.R. Keshavan, "Evidence-Based Object Recognition and Pose Estimation," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 173-178, Cambridge, MA, Nov 14-17, 1989.

[3] 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. on Information Theory*, vol. 38, no. 2, pp. 766-784, March 1992.

[4] C.A. Bouman and M. Shapiro, "A Multiscale Random Field Model for Bayesian Image Segmentation," accepted to appear in March 1994 *IEEE Trans. on Image Processing.*

[5] P.J. Burt, "Smart Sensing within a Pyramid Vision Machine," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 1006-1015, August 1988.

[6] D.R. Tretter and C.A. Bouman, "Multiscale Stochastic Approach to Object Detection," *SPIE Visual Communications and Image Processing '93*, pp. 1219-1230, Cambridge, MA, Nov 8-11, 1993.

[7] K.W. Khawaja, D.R. Tretter, A.A. Maciejewski, and C.A. Bouman, "Automated Assembly Inspection Using a Multiscale Algorithm Trained on Synthetic Images," *submitted to IEEE Int'l Conf. on Robotics and Automation, 1994.*