Automated Assembly Inspection Using a Multiscale Algorithm Trained on Synthetic Images

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

Purdue University
School of Electrical & Computer Engineering
1285 Electrical Engineering Building
West Lafayette, Indiana 47907-1285

Abstract— **An important aspect of robust automated assembly is an accurate and efficient method for the inspection of finished assemblies. This article presents a novel multiscale assembly inspection algorithm that is used to detect errors in an assembled product. The algorithm is trained on synthetic images generated using the CAD model of the different components of the assembly. The CAD model guides the inspection algorithm through its training stage by addressing the different types of variations that occur during manufacturing and assembly. Those variations are classified into those that can affect the functionality of the assembled product and those that are unrelated to its functionality. Using synthetic images in the training process adds to the versatility of the technique by removing the need to manufacture multiple prototypes. Once trained on synthetic images, the algorithm can detect assembly errors by examining real images of the assembled product. The effectiveness of the system is illustrated on a typical mechanical assembly.**

I. Introduction

At a time when quality and cost are becoming even more important in the manufacturing process, accurate and efficient inspection is critical. However, the complexity of electrical and mechanical assemblies has reached a point where human inspection can be fatiguing, unreliable, and expensive. This has prompted many manufacturers to implement automated inspection systems, particularly for the inspection of two-dimensional components such as printed circuit boards [1]. These systems are typically driven by the information available in the CAD model of the circuit to be implemented.

Unfortunately, efforts to achieve the advantages of CAD-driven inspection systems for three-dimensional assemblies have been largely unrealized. While a large body of research exists in the area of CAD-driven vision [2], this work has been primarily focused on the recognition of three-dimensional objects as opposed to their inspection. For object recognition tasks, one typically needs to index into a large or complex database of possible object models. Discrete object features, such as edges and corners, have been widely used in these recognition approaches because they allow for efficient indexing into the model database [3].

In contrast to feature-based techniques, template-based approaches directly compare gray-scale image data to a predefined model, or template. For inspection, the additional information present in gray-scale data makes template-based approaches potentially more sensitive. In fact, direct comparisons have shown that template-based methods can outperform feature-based methods in some applications [4]. Traditionally, template matching has been viewed as being computationally expensive; however multiresolution processing can make these approaches computationally efficient [5].

Our approach to automated inspection is grounded in multiresolution template matching, however, the templates are stochastic models of the expected gray-scale variations in the image [6],[7]. These models are built by examining computer-generated images that exhibit the allowable variations in a correct assembly. The process of obtaining a model is conceptually illustrated in Fig. 1 using the example assembly shown in Fig. 2. The process starts by using information from the CAD model to generate multiple ray-traced images of the assembly within the range of acceptable tolerances [8], [9]. Connectivity information calculated from the CAD model is used to create an object tree that describes how assembly errors can manifest themselves in the image [10], [11]. The synthetic images and object tree are then used by the training algorithm to determine a stochastic model of the acceptable gray-scale and positional variations in an image of a real assembly. The training is performed using the Expectation Maximization (EM) algorithm [12] with the resulting parameters normalized using a single real image.

A key feature of our approach is the utilization of computer-generated synthetic images in the training process. This approach has the advantages of avoiding the manufacture of physical prototypes while providing

This work was supported by the NEC corporation, National Science Foundation grant number CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems, National Science Foundation grant number MIP93-00560, and an AT&T Bell Laboratories PhD Scholarship.

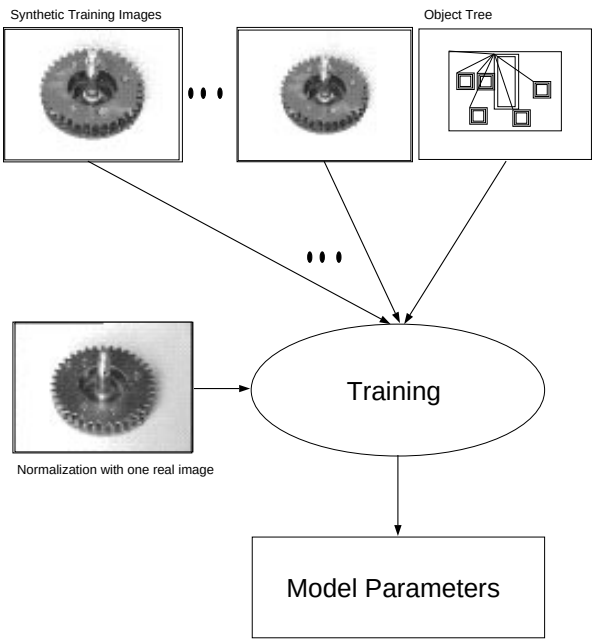

Fig. 1. To obtain a model of what a correctly assembled product looks like, the system first determines optimal viewing parameters. It then generates numerous synthetic images of the assembly which include acceptable variations that do not affect its function. The locations of potential errors are automatically identified from the CAD model and passed to the training procedure using a data structure referred to as an object tree. The model parameters which describe the statistical gray scale and positional variations are obtained using maximum likelihood estimation. A single real image is used to adjust for the lighting conditions in the testing environment.

Fig. 2. A video image of a pattern wheel assembly that is used as a representative example of a typical mechanical assembly for which the inspection algorithm was implemented.

a wider variety of acceptable variations. This not only results in a more robust inspection algorithm but also facilitates concurrent design of the assembly and its inspection system. Since each step in the inspection design process is automated, inspection becomes feasible for small batches.

The remainder of this article is organized as follows. An overview of the multiscale inspection algorithm is presented in section II. Section III describes the CAD database and its role in guiding the inspection algorithm. The synthetic image generation process is then addressed in section IV. Some experimental results are presented in section V followed by conclusions in section VI.

II. Multiscale Object Detection

We approach automated inspection as a problem in object detection, where we assume the inspection algorithm must make decisions based on a monochrome image of the object. Our multiscale detection algorithm is based on a stochastic object model, which is tailored to a specific object by adjusting the model structure and changing model parameters. The model generation and parameter estimation is driven by a CAD model of the object as described in the following sections.

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. We represent the two dimensional position and orientation of each subassembly as a state vector X. Since the position of a subassembly varies from image to image we model it as a random vector. The state density function for a node depends only on the state of its parent node in the object tree and on a set of node specific parameters ϕ. Thus, the states form a Markov chain along any path from the root of the object tree to a leaf node.

Fig. 3 shows an object tree, where the superscript (i) is used to denote quantities specific to node i. The image data Y associated with each node is modeled as a set of random variables with density functions parameterized by a template θ that indicates the expected appearance of the subassembly as well as the expected data variability. The data values will also depend on the position of the subassembly in an image, so the overall object model is as shown in the Fig. 3, with the arrows indicating conditional dependence.

We use the multiresolution Haar transform of each image as our data (see Fig. 4) and use a corresponding multiresolution template at each node of the object tree (see Fig. 5). This allows us to model each node at resolutions

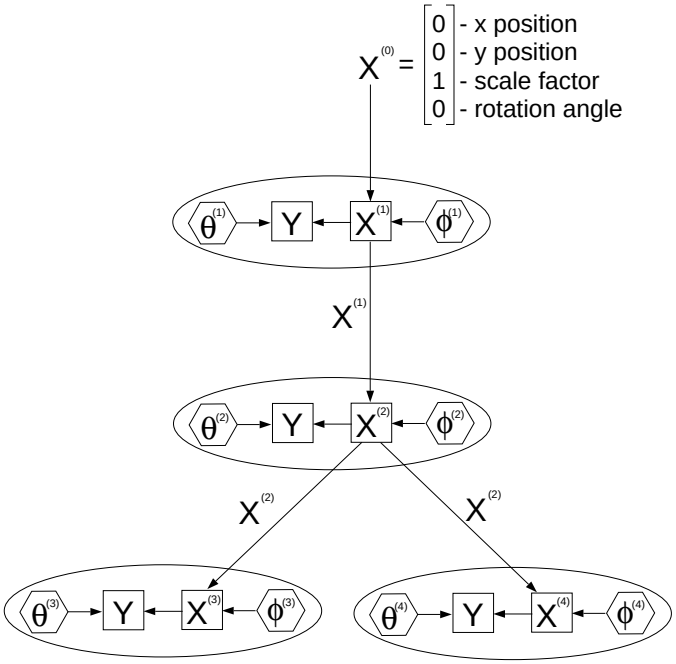

Fig. 3. An example of an object tree that identifies the significant locations within a training image as well as their relationship to each other. The state X of each node is modeled as a random vector with density functions that depend on the state of the parent node and a set of node specific parameters ϕ. The image data Y of each node consists of the wavelet transform of the image and is modeled as a random multiscale field with density functions parameterized by a template θ.

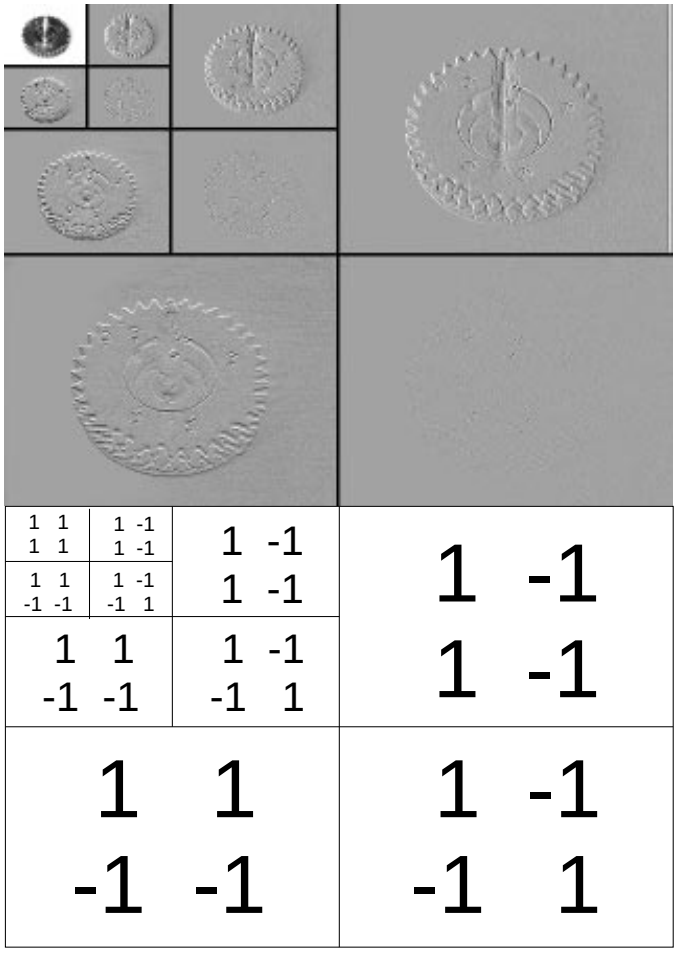

Fig. 4. An illustration of the Haar transform applied recursively to the video image of the pattern wheel shown in Fig. 2. The upper portion of the figure shows the actual transformed image at three different resolutions. The lower portion of the figure illustrates the pixel calculations that are recursively applied to create the multiresolution transformed image.

appropriate to the important features in the subassembly. It also permits us to search for the subassemblies via a fast multiscale search technique. We use recent results from the theory of multiscale random processes to aid in the analysis and construction of the model [13, 6].

The search for the most likely position X of a subassembly begins at a coarse resolution and progresses to finer resolutions. For a given resolution and candidate state we use the image data and templates at that and coarser resolutions to compute the log likelihood ratio between the hypothesis that the subassembly is present and the hypothesis that it is not. The states with the largest log likelihood ratio are investigated at the next finer resolution. The search continues in this fashion until the largest log likelihood ratio exceeds a predefined decision threshold β, at which point the search returns the associated state as the position of the subassembly. The search will terminate in a "no match" condition if it reaches a point where all remaining candidate states have log likelihood ratios less than a rejection threshold α. Thus, the search takes the form of a sequential likelihood ratio test, where the search progresses in scale rather than time. A more detailed discussion of the algorithm can be found in [7].

The inspection algorithm searches for the object in an image by using this fast multiscale search technique at each node of the object tree. The search traverses the object tree from the root to the leaves, performing a sequential MAP (SMAP) estimate of the state at each node [6] and passing this estimated state to the child nodes. The object passes inspection only if all subassemblies are located and found to match the model. This procedure hinges on our ability to compute a log likelihood ratio at each state and resolution, which in turn relies upon a knowledge of the parameters θ and ϕ for each node. These parameters, as well as the structure of the object tree, are determined using the CAD model of the object. These parameters are adjusted at the actual inspection site by using a single real image to compensate for variations in lighting environments.

We first use the CAD model to identify the important subassemblies and generate the object tree. The CAD model then generates a series of training images, each of which contains a properly assembled object, with all subassemblies and viewing conditions within their allowed tolerances. The object tree is "overlaid" on one of the

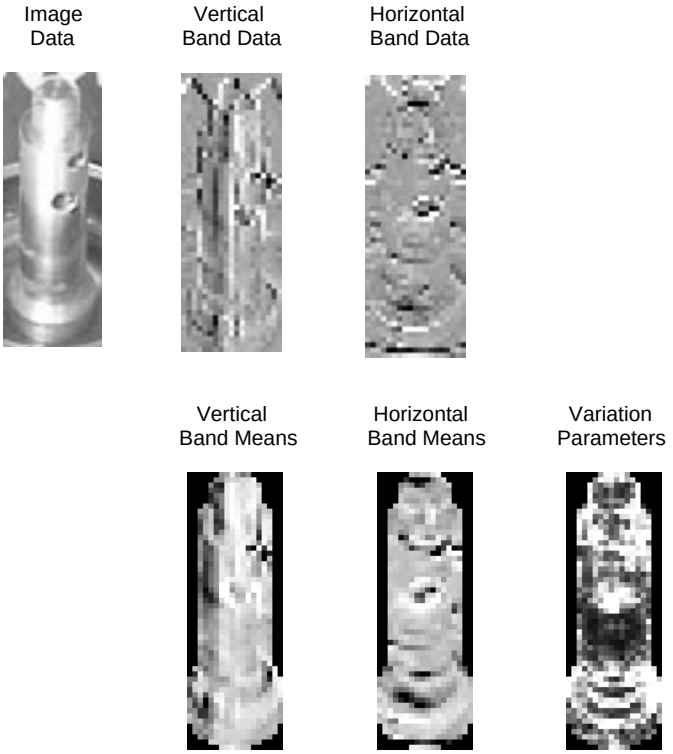

Fig. 5. The single monochrome image of the assembly under inspection is processed using the multiple resolution Haar transform illustrated in Fig. 4. An example of the results of this procedure is provided in the upper half of the figure. In this example the node of the object tree that corresponds to the shaft of the pattern wheel is shown at maximum resolution. This transformed version of the inspection image is compared to the statistical model of what we expect to see at this node for a correctly assembled pattern wheel. The statistical model shown in the lower half of the figure is automatically calculated from the training images created from the CAD model of the assembly. The dark regions in the model represent a mask which excludes extraneous information from the identification process.

training images, identifying the location and orientation of each subassembly in that image. This information is used to initialize the model parameters, which are then determined from the full set of training images via the iterative expectation maximization (EM) algorithm [6, 12].

III. CAD Database Analysis

As discussed in the preceding section, data obtained from an analysis of the CAD model of the assembly is used to guide the inspection algorithm in the training process by addressing the different variations possible and identifying an appropriate object tree in the training images. To be able to produce a large number of synthetic images with the required variations, a suitable CAD model has to be generated for the desired assembly and its components. The relationship among the different components of the assembly must be known to provide the inspection algorithm with an appropriate object tree.

For the purposes of illustration, the pattern wheel assembly pictured in Fig. 2 is used as a simple example of a realistic assembly used in a small batch manufacturing environment. For each component of the pattern wheel assembly, a CAD model was created using the TWIN Solid Modeling Package developed by the CADLAB of Purdue University. TWIN is a boundary representation solid modeler but also accepts Constructive Solid Geometry (CSG) models as input. Therefore, components are created in the CSG format and then converted to boundary representation for internal calculations [8].

Three CSG models are created for every component to account for component variations that naturally occur in a manufacturing environment. One model represents the component at its maximum tolerance limit while another represents it at its minimum tolerance limit. Both models are created by using the maximum and minimum dimensions that allow the component to function as designed. These are the Maximum Material Condition (MMC) and Least Material Condition (LMC) of the component with the region in between forming the component's tolerance zone [9]. Creating a CAD model of the component that lies within its tolerance zone generates an instance of the component that is within its allowed tolerance. Fig. 6 shows an example of the tolerance zone of a component. A new random instance of the component is generated by tracing the CSG tree of the LMC and the CSG tree of the MMC models of the component and selecting from a uniform distribution of the geometric parameters of the primitives in the CSG trees. The third CSG model represents the component at its optimal dimensions which can then be used, along with the other two CSG models, for generating a nonuniform distribution of random components that are within tolerance.

The locations of all components within the nominal assembly are specified by homogeneous transformation matrices that are used to accomplish two tasks. First, the variations between the relative locations of different components is specified in terms of a tolerance zone for these homogeneous matrices [10] so that different instances of properly functioning assemblies can be created. Second, when the components are transformed as specified by these homogeneous matrices, the contact relationship between any two components can be found by querying the modeler. The identification of the bounding faces between any components that are mated is used to form a graph in which the vertices represent the different components of the assembly and the edges between the vertices represent liaisons between the components, as connections or as contacts. This graph is called the liaison diagram [11] and it shows the interaction among the different components of the assembly. This process is demonstrated using Fig. 7 which shows an exploded view of the pattern wheel assembly. This figure illustrates the order of operations required to complete the assembly and highlights the single common axis of insertion for the pins and the shaft. Using this information along with the correct final state of the assembly the system determines which surfaces will be in contact. For example, the single cylinder pins (the high density and the unlatch pins) are inserted only into wheel-a1 and wheel-a2 and therefore form a close rela-

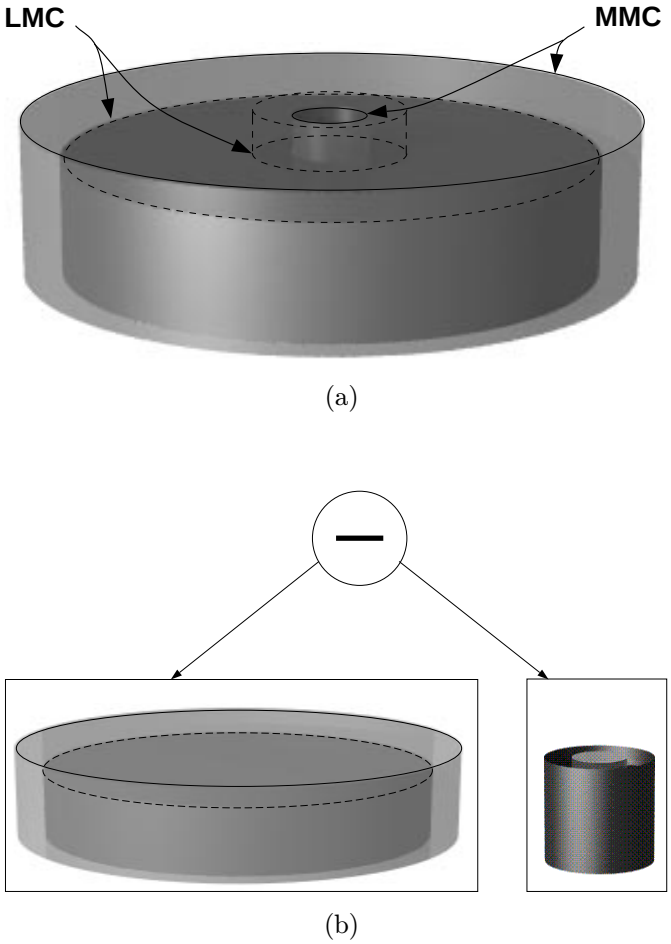

Fig. 6. This figure presents an illustration of tolerance zones for a simple component depicted in (a). Any instance of this component that lies within the shaded region bounded by the MMC and LMC surfaces is within tolerance and will function as designed. Instances of components within the tolerance zone are created by traversing the CSG trees of the MMC and LMC models of the component, as illustrated in (b), where each primitive is selected from within its tolerance zone.

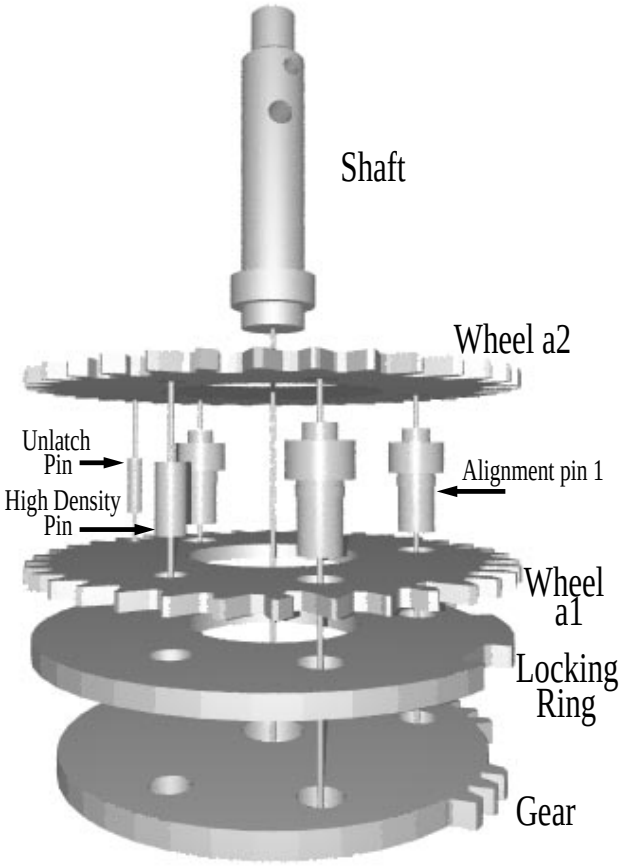

Fig. 7. An exploded view of the pattern wheel assembly generated from the information in the CAD model. This view illustrates the order of assembly as well as the single common insertion axis for all of the pins. Individual surface contacts between different components are used to create the liaison diagram shown in Fig.8.

tionship with the wheels through their side surfaces. The shaft is only inserted into the center hole of the gear. The multi-cylinder pins (the alignment pins), however, are in contact with the holes in the gear, the locking ring, and the wheels and functionally are required to maintain a precise separation between the two wheels as well as between wheel-a1 and the locking ring which rests on the gear. This last piece of information can not be clearly deduced from the exploded view but is clear in the resulting liaison diagram shown in Fig. 8 which is formed from the contact information explained above. Each circle in the diagram represents a surface on an individual component with the arrows indicating a contact between surfaces of different components.

The liason diagram forms the basis from which the object tree is generated. Contacts between different components imply that they were brought together by the assembly process and that errors in that process would result in misaligned or missing contacts. Thus to check for errors in the insertion of an individual component, the liason diagram is traversed starting with the vertex associated with that component. This process is illustrated for the high density pin in Fig. 9 resulting in a data structure that we refer to as an error tree. The error trees for all components of interest can then be combined to create the complete object tree (see Fig. 3) that guides the inspection process.

All relevant information needed by the inspection algorithm that can be gleaned from the CAD model is now available. The remaining step in the training process is the actual creation of realistic images that include the object tree and the different environmental variations that may occur in the actual assembly workcell. This is the topic of the next section.

IV. Synthetic Image Generation

There are two image generation algorithms used to create synthetic images from the CAD model of the assembly. The first is a fast rendering technique that uses only a simple local illumination model and takes advantage of special purpose VLSI hardware for performing geometrical calculations. These draft images are used to identify an

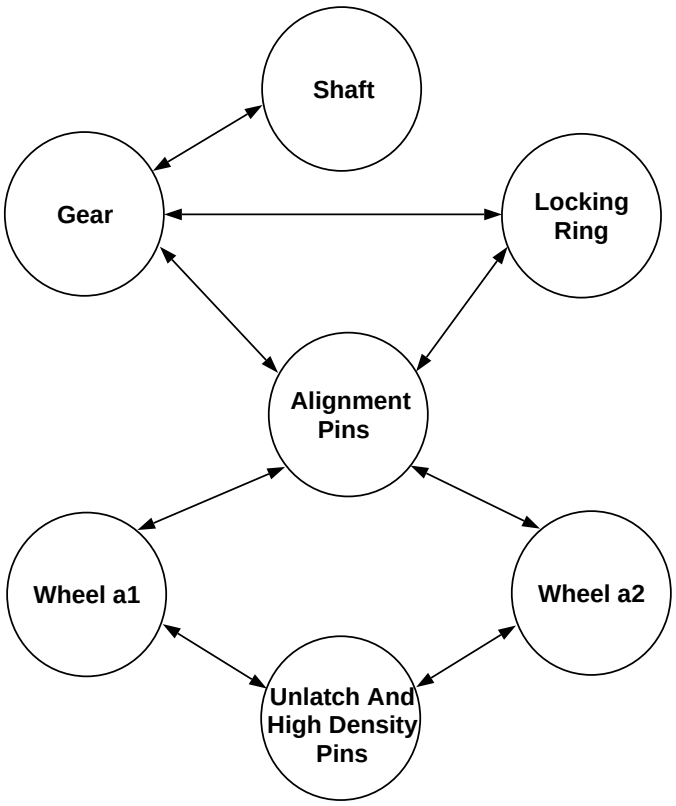

Fig. 8. A liaison diagram for the pattern wheel assembly showing the contact relationships among the different components. Each circle represents the structure of a particular component. The double arrows indicate a contact between some surfaces within two components.

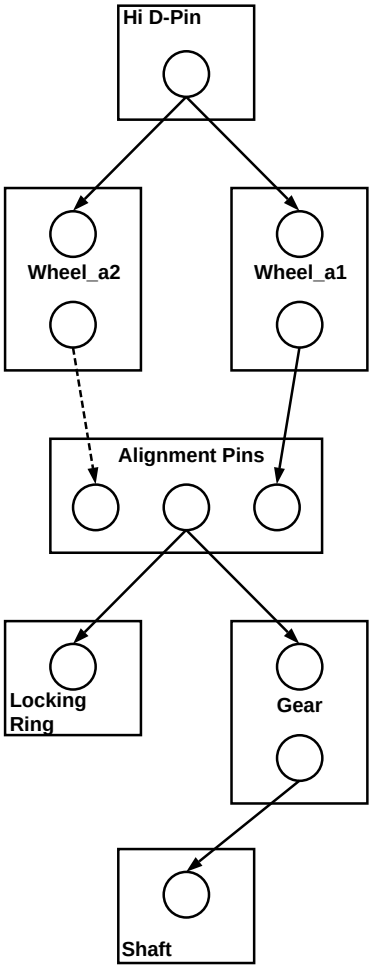

Fig. 9. The error tree for the High Density Pin that is generated from the liaison diagram. This tree illustrates the possible propagation of misaligned surfaces within the assembly. The circles represent different surfaces of a component. (For simplicity the three alignment pins are considered together and the unlatch pin is not shown.) The dashed arrow represents an edge that exists in the liason diagram but that is not included in the error tree. This allows several error trees to be combined to form the object tree used by the multiscale detection algorithm.

optimal camera location and to further refine the object trees used by the inspection algorithm. The location of the camera relative to the assembly under inspection will greatly affect the sensitivity of identifying possible errors. Our approach to computing an optimal camera location is to try and maximize the amount of information in the resulting image that is relevant to evaluating contacts between components. To do this we tag each of the contact surfaces in the error trees with a unique ambient color with all other surfaces set to black. We then render the assembly using a graphics workstation equipped with a Z-buffer using only the ambient intensity of the polygons. The resulting image contains the number of visible pixels for each surface of interest. With hardware support this procedure takes only a fraction of a second so it can be repeated for a large number of potential camera locations. The optimal camera location is selected as that which maximizes the number of visible pixels and surfaces. The information from this image created from the optimal camera location is also used to identify the location and size of the object nodes required by the inspection algorithm. To simplify processing, all object nodes are rectangular, however, a mask is used to identify the regions within the node that correspond to error tree surfaces. Only this region is used in building the statistical model of the node. This prevents irrelevant background information from affecting the sensitivity of the inspection process. An illustration of creating this mask information is provided in Fig. 10. These masks are also visible in the templates depicted in Fig. 5.

The second rendering technique is used to create more physically realistic synthetic images that are required to build the statistical model of what a correctly assembled product should look like (see Fig. 1). To obtain sufficiently realistic images, light-object interaction must be modeled. Although graphics workstations available today can generate shaded images at video rates, the illumination models used to generate these images typically only deal with the very first reflection from an object's surface. These so called first-order or local models do not include the effects of light reflecting from several objects or being transmitted through objects. These global models need to be considered in order to obtain more realistic images that can model different types of materials, particularly those that are highly reflective such as polished metals [14]. The only established rendering techniques that at-

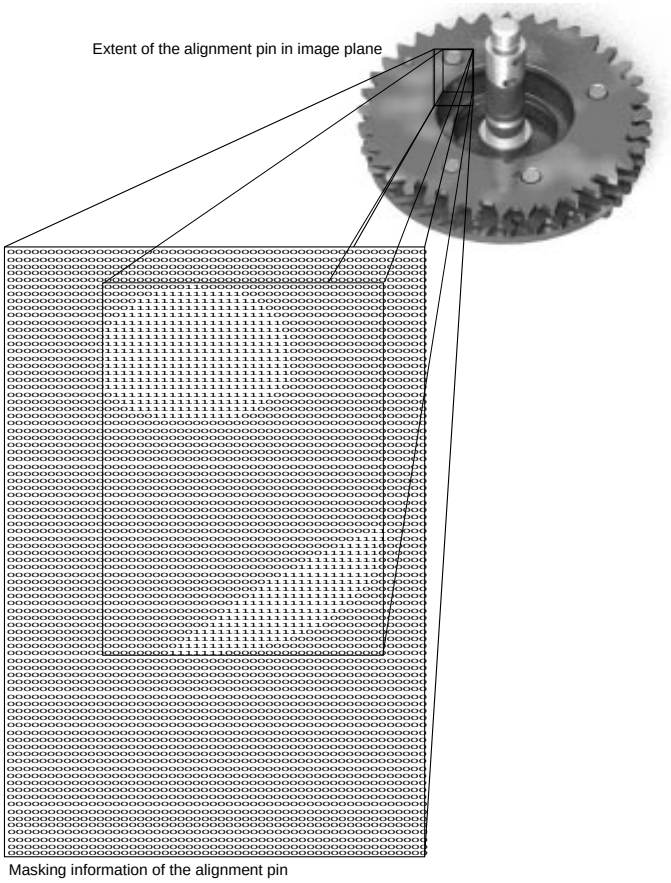

Fig. 10. The outer rectangle represents the bounding box of the projection of an alignment pin onto the image plane. The inner rectangle is the bounding box of the visible portion of this alignment pin. This bounding box is passed to the inspection algorithm as an object node along with the mask that identifies the region which corresponds to the alignment pin. This allows the inspection algorithm to ignore gray-scale variations from surfaces that are unrelated to the assembly process. This mask information is obtained using Z-buffer hardware.

tempt to model global lighting effects are ray tracing and radiosity. Because specular effects are very hard to model with radiosity, metallic parts are hard to simulate in images that are rendered using radiosity techniques. As a result, we selected ray tracing as the rendering technique for this application.

V. An Example

This section illustrates the implementation of the entire assembly error inspection system for the simple example of the pattern wheel presented in Fig. 2. First, the CAD model of the assembly is used to create instances of the different components that have simple variations within their allowed tolerances. This is required in order to produce training images that include acceptable levels of variations. Next, as explained in section III, the liaison diagram of the pattern wheel assembly is generated automatically from surface contact information (see Fig. 8). The exploded view of Fig. 7 suggests that the pattern wheel assembly can be completely assembled with only a single common axis of insertion. This identifies the pins as potential sources of errors that can occur due to misalignment. From the liaison diagram the system automatically generates the error trees of the pins. These error trees, like the example in Fig. 9, are used as the object trees (see Fig. 3) which identify the important locations in the images for various types of assembly errors. The number of levels included in the tree is determined by the visibility of the root node, i.e., if the single view used by the inspection algorithm results in the root node being partially occluded then evidence of misassembly should be corroborated by its child nodes to which the error may be propagated.

Ray tracing is then used to generate a group of synthetic images from the assembly CAD model. Six images were used to run this particular example. Parameters for the illumination model are specified from the material types of the different pattern wheel assembly components. Light and camera placements are automatically evaluated to provide maximum visibility of contact surfaces representing potential locations of assembly errors. The synthetic images along with the information from the object trees are then passed to the multiscale inspection algorithm for training. Fig. 11 illustrates one such synthetic image where the wheel-a2 error tree is shown superimposed onto the image (see Fig. 8). The object tree consists of the "boxed in" areas. The upper left corners of the nodes are connected to one another to indicate the connectivity of the tree, and the number of boxes around each node indicates that node's level in the tree. Note that up to this point no real images of the assembly are required so that this system can also be used to evaluate prospective designs for their inspectability without even building a prototype. Once the system has been trained it is ready to perform its function of inspecting real assemblies. Prior to automated inspection, a single real image of a correctly assembled product must be obtained from the actual inspection station to compensate for variations in lighting environments.

The system was tested on numerous real video images of correctly assembled pattern wheels, which can be safely assumed to comprise the vast majority of manufactured pieces, in various positions, orientations, and lighting conditions uniformly distributed in the range specified by the training set. In all cases the inspection algorithm produced no false negatives, despite the large variations in the resulting image, thus illustrating the robustness of the technique. The algorithm consumed an average cpu-time of 12 seconds on a Sun Sparc-10 workstation to identify a correct assembly. Next the system was tested with real assemblies that were misassembled due to missing or misaligned pins. Fig. 12 shows an example of a video image of one such defective assembly in which the alignment pin is missing. The performance of the algorithm is graphically illustrated in the figure by the boxes that are superimposed on the image. The algorithm first identifies the gross location of the assembly within the image. The box around the entire assembly indicates where the algorithm located the part. Note that the tilt of the box indicates that the

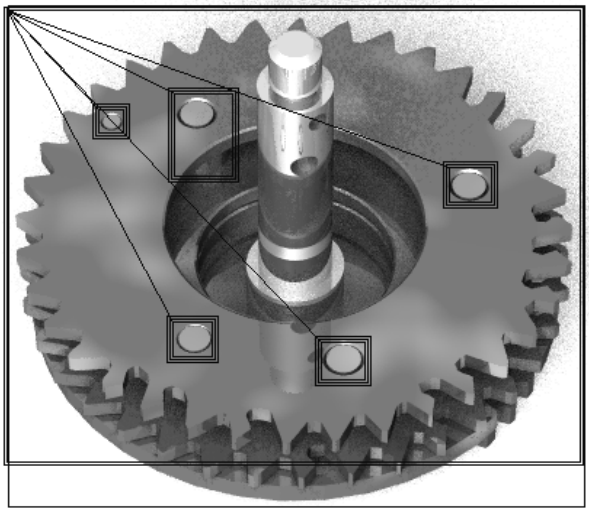

Fig. 11. A synthetic image of the pattern wheel assembly with the wheel-a2 error tree denoted by the connected boxes. This tree is used as the object tree required by the inspection algorithm to guide its analysis of the image. The number of boxes around each object represents the object's level in the tree. The boxes are automatically generated by calculating the visible portions of the components in the error tree with the first level box including the entire assembly.

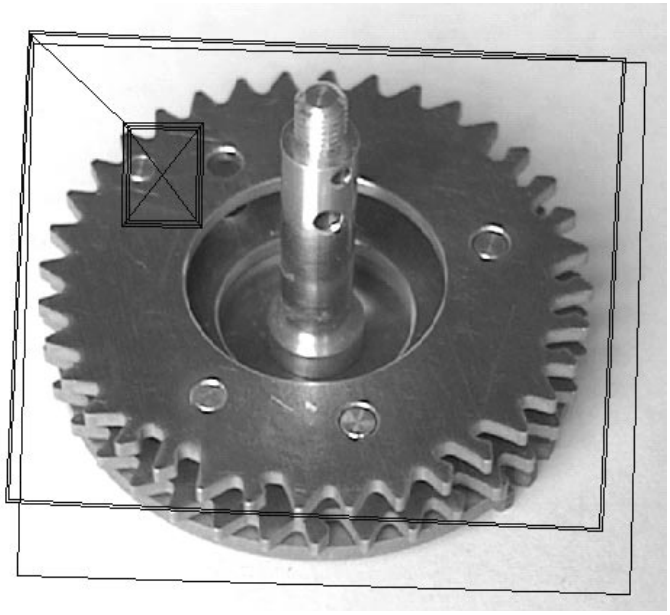

Fig. 12. A real video image of a defective assembly with the error correctly identified by the inspection algorithm. The "X" in the box identifies the location in the image in which a mismatch with the training images was found thus indicating the missing alignment pin. Note that the slightly different position, orientation, and scale of the real image as opposed to the synthetic training images as indicated by the large bounding box does not adversely affect the robustness of the algorithm.

algorithm was able to adapt to the different orientation of the part in the real image as compared to the training images, once again illustrating its robustness. The algorithm then "zooms in" to look for the pin, guided by the information in the error trees. At this point it detects a mismatch which is indicated in the image by an "X" in the area where it expected to find the pin.

Fig. 13 shows an example of a more subtle error resulting from wheel-a2 being misplaced on the pins. Note that the error can not be detected from just considering wheel-a2. However, once the algorithm descends to the second level of the wheel-a2 error tree, which includes all the pins (see Fig. 8), it detects an error in the node associated with one of the alignment pins. This is again indicated in the image by an "X" in the area where a problem was detected.

VI. Conclusions

This article has discussed the implementation of an assembly inspection system that uses a multiscale algorithm to detect errors in assemblies after being trained on images of correctly assembled products. It has been shown that synthetic images generated by using the CAD models of the assembly and computer graphics ray tracing rendering techniques can be effectively used to train the algorithm. The use of synthetic images has the advantages of simplifying the training process and automating the image selection and the object tree allocation procedure. In addition, the problem of addressing the different variations that can occur in the assembly workcell is simplified. The use of synthetic images generated directly from CAD models also allows the fine tunning of the inspection algorithm early in the design process thus allowing the assembly and inspection processes to be designed concurrently.

References

[1] H. Yoda, Y. Ohuchi, Y. Taniguchi, and M. Ejiri, "An automatic wafer inspection system using pipelined image processing techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 1, pp. 4–16, January 1988.

[2] *IEEE Workshop on Directions in Automated CAD-Based Vision*, Maui, HI, June 2-3 1991.

[3] C. H. Chen and A. C. Kak, "A robot vision system for recognizing 3-D objects in low-order polynomial time," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1535–1563, November/December 1989.

[4] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, October 1993.

[5] P. Burt, "Smart sensing within a pyramid vision machine," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 1006–1015, 1988.

[6] C. Bouman and M. Shapiro, "A multiscale random field model for bayesian image segmentation," *IEEE Transactions on Image Processing*, vol. 3, no. 2, pp. 162–177, March 1994.

[7] D. Tretter, C. Bouman, K. Khawaja, and A. A. Bouman, "A multiscale stochastic image model for

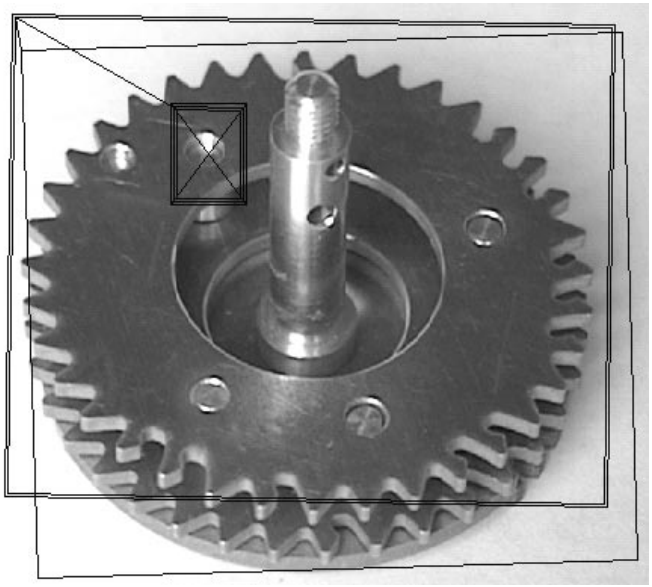

Fig. 13. A real video image of a defective assembly resulting from misplacing the top wheel (wheel-a2). The algorithm detected no errors in trying to locate the assembly and the first level of the wheel-a2 error tree as indicated by the single and double line squares. A defective assembly was identified when the inspection algorithm descended to the second level of the error tree.

automated inspection," *IEEE Transactions on Image Processing*, vol. 4, no. 12, December 1995.

[8] A. A. G. Requicha, "Representations for rigid solids: Theory, methods, and systems," *ACM Computing Surveys*, vol. 12, no. 4, pp. 437–464, December 1980.

[9] A. A. G. Requicha and S. C. Chan, "Representation of geometric features, tolerances, and attributes in solid modelers based on constructive geometry," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 3, September 1986.

[10] D. E. Whitney and O. L. Gilbert, "Representation of geometric variations using matrix transforms for statistical tolerance analysis in assemblies," *in Proc. 1993 International Conference on Robotics and Automation*, pp. 314–321, (Atlanta, Georgia), May 2-6 1993. IEEE.

[11] T. L. DeFazio, A. C. Edsall, R. E. Gustavson, J. Hernandez, P. M. Hutchins, H. W. Leung, S. C. Luby, R. W. Metzinger, J. L. Nevins, K. Tung, and D. E. Whitney, "A prototype of feature-based design for assembly," *Journal of Mechanical Design*, vol. 115, no. 4, pp. 723–734, December 1993.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[13] M. Bassevile, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky, "Modeling and estimation of multiresolution stochastic processes," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 766–784, March 1992.

[14] J. T. Kajiya, "The rendering equation," *Computer Graphics*, vol. 19, pp. 143–150, August 1986.