Automated Assembly Inspection Using a Multiscale Algorithm Trained on Synthetic Images

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

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

Abstract— **An important part of a robust automated assembly process is an accurate and efficient method for the inspection of finished assemblies. This paper 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 and control the lighting conditions. 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

One important component of intelligent manufacturing systems that are capable of producing high quality products in small batches and with short design cycle times is a method of on-line automated inspection. This paper discusses the implementation of a multiscale assembly inspection algorithm that is used to detect and distinguish error-free products from those with assembly errors after being trained on synthetic images of correctly assembled components. Generating real images of an assembly in order to train an assembly inspection algorithm is a demanding and time consuming operation. One must physically create scenarios that could occur in the workcell where the piece is assembled in order to generate training images. Those scenarios must address the different types of variations that could occur in the workcell. These variations can be of the kind that would affect the functionality of the assembled piece, e.g., variations in the dimensions of the different components of the assembly beyond acceptable tolerance, or of the kind that would not affect the functionality, like lighting and material properties. This would require the ability to generate the components of the assembly at the limit of their acceptable dimensions. Once the training images are generated, one must manually inform the inspection algorithm about any information in the images that is needed to run the algorithm, e.g., important features in an image.

In the approach presented here it is shown how using images that are synthetically generated directly from CAD models can greatly simplify this training process. This is done by using information gleaned from the CAD model to guide the inspection algorithm through its training phase. Because CAD models contain geometric information concerning tolerances and correct part matings, one can generate an unlimited number of synthetic images with the precise variations desired. Relationships among the different components of the assembly deduced from the CAD model can help in automating the generation of the information in the training images that is needed by the inspection algorithm. This not only greatly facilitates the training of image processing algorithms used for error detection, but also provides the algorithm with images much earlier in the design process. The CAD models are also analyzed to provide information pertaining to the design of the workcell such as camera and light placement.

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

The remainder of this paper is organized as follows. A short description of the multiscale inspection algorithm is introduced 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 below.

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 S. Since the position of a subassembly varies from image to image and is generally not known, 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. 1 shows an object tree, where the superscript (i) is used to denote quantities specific to node i. The 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 figure, with the arrows indicating conditional dependence.

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

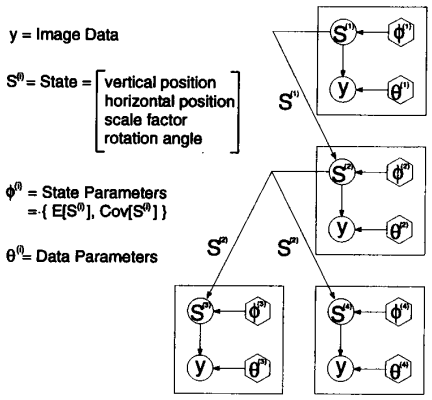

Fig. 1. An example of an object tree that identifies the significant locations within a training image as well as their relationship to each other.

the theory of multiscale random processes to aid in the analysis and construction of the model [1, 2].

We will search for the most likely position S of a subassembly by starting at a coarse resolution and progressing 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 [3].

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 [2] 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.

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 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 [2, 4].

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. Variations that would affect functionality are addressed in this section while variation that would not affect functionality are considered in section IV.

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 any needed data about an image.

For the purposes of illustration, the pattern wheel assembly pictured in Fig. 2 was used as a simple example of a realistic assembly used in a small batch manufacturing environment. This wheel assembly was used in Archimedes, a prototype system for automating mechanical assembly [5]. For each component of the pattern wheel assembly, a CAD model was created using the TWIN Solid Modeling Package [6]. 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.

Three CSG models were created for every component from the part's machine drawings to account for variations that naturally occur in a manufacturing environment. One 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 allowed dimensions for the component in its machine drawings. These are

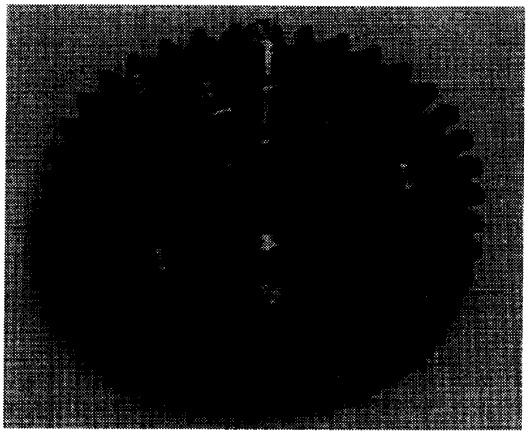

Fig. 2. A video image of a pattern wheel assembly that is used as a representative example of a typical mechanical assembly for which the inspection algorithm was implemented.

the Maximum Material Condition (MMC) and Least Material Condition (LMC) [7] of the object. Creating a CAD model of the object that lies within these limits generates a component that is within the allowed tolerance. 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 distribution of random components that are within tolerance [8]. Currently, a random object is generated by selecting from a uniform distribution of its geometric parameters within the object's LMC and MMC.

The location of all components within the assembly are specified by homogeneous transformation matrices so that the relationship between any two parts can be found by querying the modeler. The identification of the bounding faces between any parts that are mated is used to form a graph in which the nodes represent the different parts of the assembly and the lines between the nodes represent liaisons between the parts, as connections or as contacts. This graph is called the liaison diagram [9, 10] and it shows the interaction among the different components of the assembly. This process is demonstrated using Fig. 3 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 relationship with the wheels through their side surfaces. The shaft

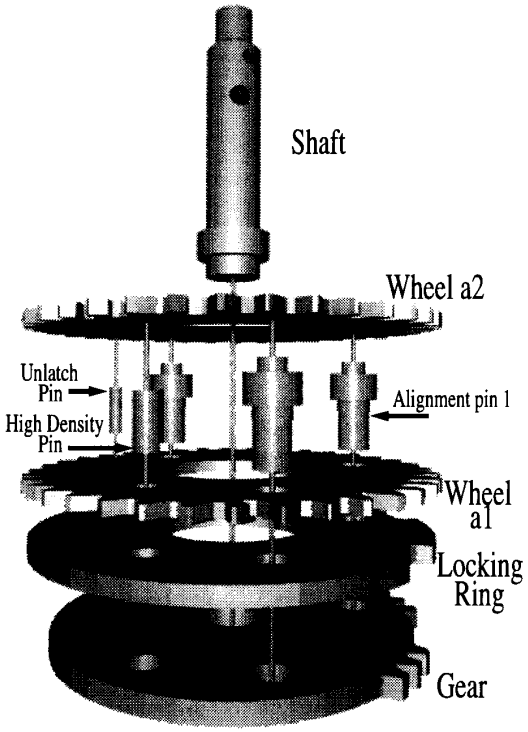

Fig. 3. An exploded view of the pattern wheel assembly generated from the information in the CAD model. This view illustrates the order of assembly as well as the single common insertion axis for all of the pins. Individual surface contacts between different components are used to create the liaison diagram shown in Fig.4.

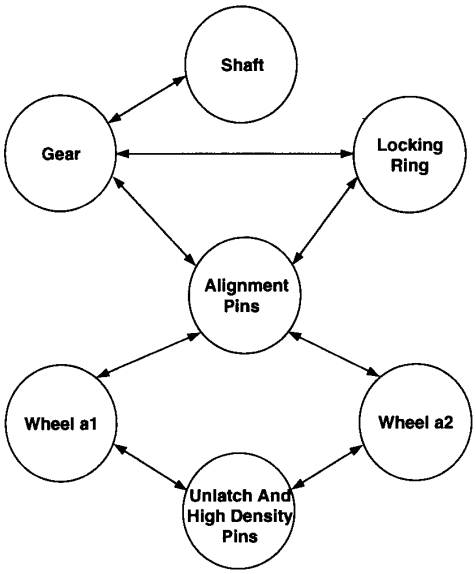

Fig. 4. A liaison diagram for the pattern wheel assembly showing the contact relationships among the different components. Each circle represents the structure of a particular component. The double arrows indicate a contact between some surfaces within two components.

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. 4 which is formed from the contact information explained above. Each circle in the diagram represents all surfaces of an individual component with the arrows indicating a contact between surfaces of different components.

A perspective view of the assembly can be generated from the CAD model described above using standard computer graphics techniques. The CAD model is used to automatically generate suitable view vectors. The field of view for an image is chosen such that the assembly is guaranteed to be within the screen limits. If any particular component is identified as a possible source of error, then the location of the visible portions of that component can be easily "boxed in" in the generated training image. As a result, the different states shown in Fig. 1 that are used for training are obtained for any object node. By tracing the liaison diagram, one can also identify what other components are going to be affected by that error and how close that effect is. This generates an error tree for a particular part. For example, Fig. 5 shows the error tree of one of the pins (the high density pin). The circles represent different surfaces within each object. It shows that the wheels are in direct contact with the high density pin which identifies the possibility of a close effect. The alignment pins, however, are connected to the wheels which identifies a more remote effect. The gear and the locking ring are in contact with the alignment pins which show yet a more remote possible effect. The shaft being at the bottom level of the tree represents the least likely affected component. This error tree along with the error trees of other components of interest are used in generating the object tree required by the inspection algorithm that was shown in Fig. 1.

All relevant information needed by the inspection

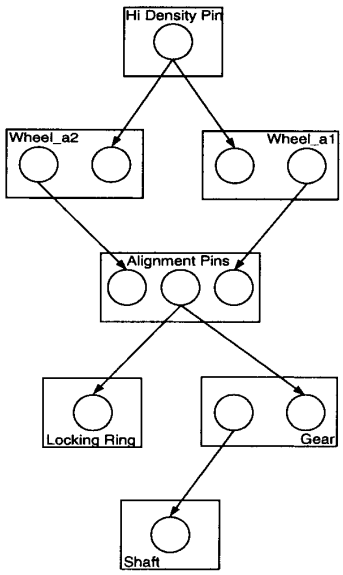

Fig. 5. The error tree for the High Density Pin that is generated from the liaison diagram. This tree illustrates the possible propagation of misaligned surfaces within the assembly. The circles represent different surfaces of an object. (For simplicity the three alignment pins are considered together and the unlatch pin is not shown.)

algorithm concerning the training images is now available. The remaining step in the training process is the actual creation of realistic images that include the different environmental variations that may occur in the actual assembly workcell. This is the topic of the next section.

IV. Synthetic Image Generation

To create the most realistic synthetic images of an assembly, computer graphics raytracing rendering techniques are used. The program Rayshade is used to create these ray-traced images by translating the CAD model of the assembly into Rayshade objects. The CAD system described above sets the limits on the viewing parameters for Rayshade. Other parameters that can also be varied include the characteristics of the lighting conditions, material properties, and various textures for the image backgrounds. Different kinds of materials can be used by choosing the right surface attributes [11]. A number of images are then selected to train the inspection algorithm. This number can vary depending on the case study and is automatically chosen such that the space of varying parameters is adequately represented.

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, shown in Fig. 4, of the pattern wheel assembly is generated automatically from surface contact information. The exploded view of Fig. 3 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. 5, are used as the object trees, shown in Fig. 1, 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.

Rayshade is then used to generate a group of synthetic images from the assembly CAD model. Three images were used to run this particular example. Input parameters to Rayshade are set such that the material type of the different pattern wheel assembly components are matched as close as possible. Light and camera placements are automatically evaluated to provide suggestions for workcell designs. The synthetic images along with the information from the object trees are then passed to the multiscale inspection algorithm for training. Fig. 6 illustrates one such synthetic image where one path down the right side of the high density pin error tree is shown superimposed onto the image. 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 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

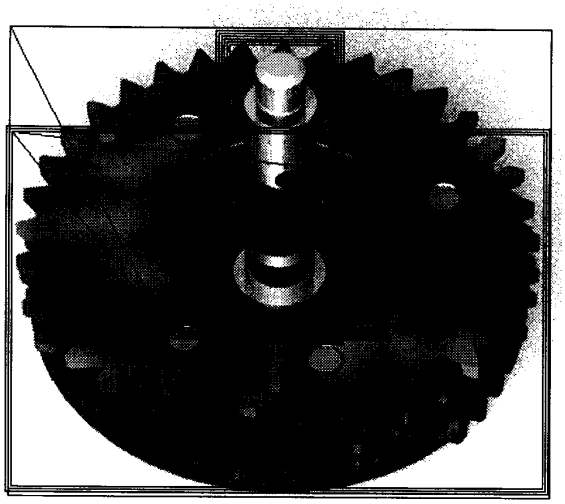

Fig. 6. A synthetic image of the pattern wheel assembly with one path down the high density pin error tree denoted by the connected boxes. The path shown is the rightmost path illustrated in Fig.5 starting from the high density pin and ending with the shaft. This tree is used as the object tree required 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.

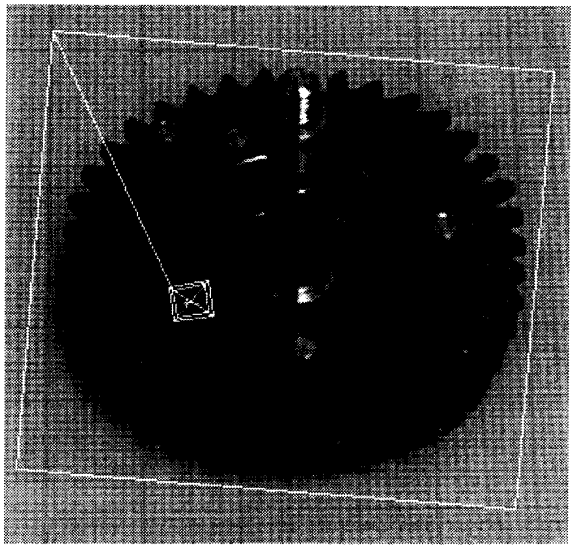

Fig. 7. A real video image of a defective assembly with the error correctly identified by the inspection algorithm. The "X" in the box identifies the location in the image in which a mismatch with the training images was found thus indicating the missing high density pin. Note that the slightly different 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.

perform its function of inspecting real assemblies. 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. 7 shows an example of a video image of one such defective assembly in which the high density 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 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. 8 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. 4), 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 paper 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 raytracing rendering techniques can be effectively used to train the algorithm. The use of synthetic images

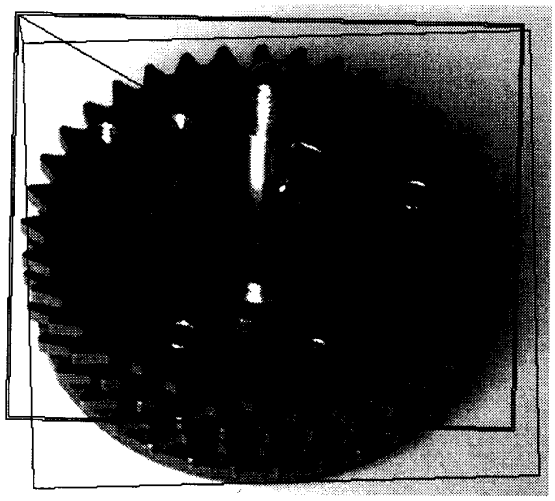

Fig. 8. A real video image of a defective assembly resulting from misplacing the top wheel (wheel-a2). The algorithm detected no errors in trying to locate the assembly and the first level of the wheel-a2 error tree as indicated by the single and double line squares. A defective assembly was identified when the inspection algorithm descended to the second level of the error tree. Note that the variation in specular reflection (see Fig.7) due to a different light position did not affect the performance of the algorithm.

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] M. Bassevile et al., "Modeling and estimation of multiresolution stochastic processes," *IEEE Trans. on Information Theory*, vol. 38, no. 2, pp. 766–784, March 1992.

[2] C. Bouman and M. Shapiro, "Multispectral image segmentation using a multiscale image model," *in Proc. 1992 IEEE Int. Conf. on Acoust., Speech, and Signal Processing*, pp. III–565 – III–568, (San Francisco, CA), March 23-26, 1992.

[3] D. Tretter and C. Bouman, "Multiscale stochastic approach to object detection," *in Proc. SPIE Visual Communications and Image Processing '93*, pp. 1219–1230, (Cambridge, MA), Nov 8-11, 1993.

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

[5] D. Strip and A. A. Maciejewski, "Archimedes: An experiment in automating mechanical assembly," *in Proc. Third Int. Symposium on Robotics and Manufacturing (ISRAM '92)*, pp. 605–611, (Vancouver, British Columbia), July 18-20, 1990.

[6] T. A. Mashburn, "A polygonal solid modeling package," Master's thesis, School of Mechanical Engineering, Purdue University, December 1987.

[7] A. A. G. Requicha, "Toward a theory of geometric tolerancing," *Int. J. of Robotics Research*, vol. 2, no. 4, Winter 1983.

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

[9] T. DeFazio et al., "A prototype of feature-based design for assembly," *in Proc. 1990 ASME Design Automat. Conf.*, vol. DE 23-1, pp. 9–16, (Chicago, IL), September 1990.

[10] J. L. Nevins and D. E. Whitney, *Concurrent Design of Products and Processes*, McGraw-Hill, New York, 1989.

[11] R. S. Hunter, "Appearance attributes of metallic surfaces," *in Appearance of Metallic Surfaces,ASTM Special Tech. Publ. 478*, pp. 3–21. American Society for Testing and Materials, Philadelphia, 1970.