Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

Camera and Light Placement for Automated Assembly Inspection

K. W. Khawaja, A. A. Maciejewski, D. Tretter, and C. A. Bouman

Purdue University
1285 Electrical Engineering Bldg.
West Lafayette, Indiana 47907-1285

Abstract

Visual assembly inspection can provide a low cost, accurate, and efficient solution to the automated assembly inspection problem, which is a crucial component of any automated assembly manufacturing process. The performance of such an inspection system is heavily dependent on the placement of the camera and light source. This article presents new algorithms that use the CAD model of a finished assembly for placing the camera and light source to optimize the performance of an automated assembly inspection algorithm. This general-purpose algorithm utilizes the component material properties and the contact information from the CAD model of the assembly, along with standard computer graphics hardware and physically accurate lighting models, to determine the effects of camera and light source placement on the performance of an inspection algorithm. The effectiveness of the algorithms 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 visual inspection systems. Unfortunately, efforts to achieve the advantages of CAD-driven visual inspection systems for three-dimensional assemblies have been largely unrealized. One impediment to achieving this goal is the automatic determination of camera positions and lighting environments that facilitate the inspection process.

The general area of sensor planning has received significant attention from the computer vision research community [1]. Optimal camera and light placement algorithms have been designed by considering the visibility of specific object features [2], [3],[4],[5]. Illumination models have primarily focused on Lambertian surfaces [6],[7] since the primary motivation has been object detection. While the assembly inspection application has analogous constraints, the appearance of various object features is used for inferring improper functionality due to errors in assembly. Thus one is more concerned with how features vary in their appearance and additional information is available to guide the selection of optimal camera and light placements.

In this work, we employ a multiscale image processing inspection algorithm, developed previously [8], that uses a statistical model of what a properly assembled component should look like. The statistical model is generated from synthetic images derived from the CAD model of the assembly and information about component tolerances [9]. Naturally the sensitivity of the statistical model, and therefore of the inspection algorithm's ability to identify assembly errors, is highly dependent on the camera placement and lighting in the inspection environment. Thus the focus of the work described here is to develop an algorithm for determining camera and light locations that provide maximum sensitivity for identifying a class of assembly errors.

The remainder of this article is organized as follows. A short description of the visual inspection algorithm used in this work is introduced in section II. The issues related to the rendering techniques are addressed in section III. The camera placement algorithm is then described in section IV followed by a description of the light placement algorithm in section V. The generate-and-test approach is then outlined in section VI. Experimental results are shown in section VII and finally, conclusions are presented in section VIII.

II. Multiscale Object Detection

Automated inspection is approached in this work as a problem in object detection, where it is assumed that the inspection algorithm must make decisions based on a monochrome image of the object. A multiscale detection algorithm based on a stochastic object model, which is tailored to a specific object by adjusting the

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.

model structure and changing model parameters, is used. The model generation and parameter estimation is driven by a CAD model of the object. The CAD model of a simple example assembly is illustrated in Fig. 1.

The inspection algorithm models an object as a stochastic tree referred to here on as the object 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. The position and orientation of each node in the object tree is modeled as a random state vector with density function depending only on the state of the parent node and on a set of node specific parameters created during the training stage from a set of synthetic images [9]. For example, the object tree automatically generated from the CAD model of the assembly in Fig. 1 is illustrated in Fig. 2. The data associated with each node is modeled as a set of random variables with density functions parameterized by a template 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. A multiresolution Haar transform of each image is used as the data along with the corresponding multiresolution template at each node of the object tree. The search for the most likely position of a node starts at a coarse resolution and progresses to finer resolutions. For a given resolution and candidate position and orientation the image data and templates at that and coarser resolutions are used to compute a log likelihood ratio between the hypothesis that the node 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. The details of this algorithm are provided in [8].

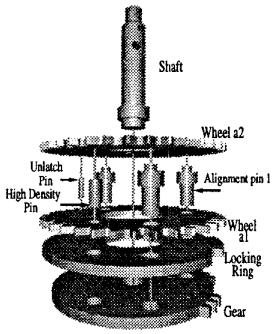


Fig. 1. An exploded view of a typical mechanical assembly generated from the information in the CAD model. This view illustrates the order of assembly as well as the single common insertion axis for all of the pins.

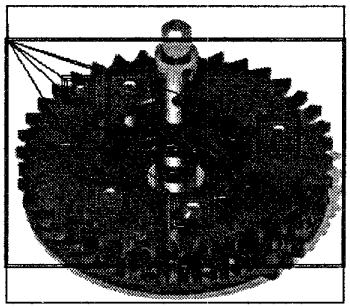


Fig. 2. A synthetic image of the pattern wheel assembly with an object tree denoted by the connected boxes and calculated using the CAD information of the inserted pins. This tree is required by the inspection algorithm to guide its analysis of the image. The number of boxes around each object represents the object's level in the tree. The boxes are automatically generated by calculating the visible portions of the components in the tree with the first level box including the entire assembly.

III. Synthetic Image Generation

There are two image generation algorithms used to create synthetic images from the CAD model of the assembly. The first is the standard fast scan-line rendering technique that uses only a simple local illumination model and takes advantage of special purpose VLSI hardware for performing geometrical calculations. This rendering process is primarily for determining the visibility constraints used to optimize camera and light source placement. The second rendering technique uses more computationally expensive, but also more physically realistic models, to generate the synthetic images that are required to build the statistical model of the appearance of a correctly assembled product. It is also used to determine the variation of that appearance, due to assembly errors, as a function of the camera and lighting environment.

A. Fast Rendering Algorithm

Fast rendering algorithms running on special purpose graphics workstations are used to create draft images of the assembly. These draft images are used to accomplish two main tasks. The first is to further refine the object trees used by the inspection algorithm. The information calculated for the image created from the optimal camera location is used to identify the location

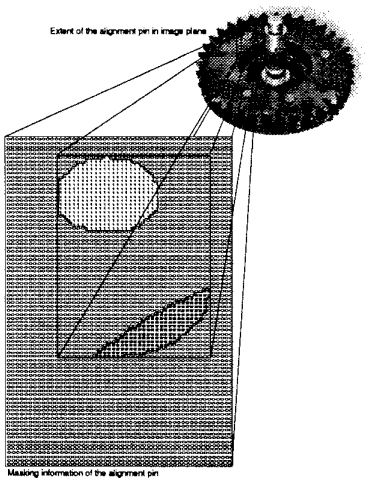


Fig. 3. The outer rectangle represents the bounding box of the projection of an alignment pin in the assembly onto the image plane. The inner rectangle is the bounding box of the visible portion of this alignment pin. This bounding box is passed to the inspection algorithm as an object node along with the mask that identifies the region which corresponds to the alignment pin. Also, visible faces of the component are identified along with the amount visible. This information is obtained using Z-buffer hardware.

and size of the object nodes. To simplify processing, all object nodes are rectangular, however, a mask is used to identify the regions within the node that correspond to related component 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. The second purpose of these draft images is to identify the extent to which surfaces of interest are visible. The surfaces of interest are determined from the contact information in the CAD model [9] and are an important factor in determining an optimal camera location (See sections IV and V). Both of these two tasks are essentially hidden surface problems and can utilize the Z-buffer hardware available in most 3D graphics workstations. This is done by tagging each surface of interest with a unique ambient color with all other surfaces of other components set to black. The assembly is then rendered using a standard scan-line algorithm available on any 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. This process is illustrated in Fig. 3.

B. Accurate Rendering Algorithm

To build an accurate statistical model of the gray scale appearance of an assembly, the techniques used to generate the synthetic images must accurately simulate the physics of light-object interaction. This precludes the use of the standard scan-line algorithms available in graphics workstations that only use approximate empirical models and are limited to so-called "local" reflections. To deal with the multiple light reflections, i.e. "global reflections", that are typical of metallic components we use standard ray tracing techniques along with the physically realistic Cook-Torrance model for local illumination.

The ray tracing paradigm has a long history but its application as a comprehensive rendering technique is generally attributed to Whitted [10]. The intensity of a ray is recursively defined as

$$I = I_l + k_{rg} I_r + k_{tg} I_t \tag{1}$$

where

I_l intensity due to direct (local) illumination
I_r intensity due to reflected light
I_t intensity due to transmitted (refracted) light
k_{rg} global bidirectional specular reflectance
k_{tg} global bidirectional transmission coefficient.

and I_r and I_t are calculated recursively by firing rays in the reflected and refracted directions.

In addition to the Lambertian model used in [10] to calculate I_l we include the physically accurate model of specular reflection known as the Cook-Torrance lighting model [11] with a Beckman distribution to describe surface roughness. The accuracy of this model for assemblies composed of polished metals, like that illustrated in Fig. 2, was experimentally verified by comparing the synthetically generated images with actual video images at various camera and light locations.

IV. Camera Placement

An analysis of the contact surfaces obtained from the CAD model of the assembly provides the information required to determine locations where assembly errors are likely [9]. These locations are what ultimately determine the object tree used by the inspection algorithm (see Fig. 2 for an example). Clearly, the size and visibility of the nodes in the object tree are heavily dependent on the viewing direction. To obtain a viewing direction which includes as much information as possible, a two step optimization is performed in which the criteria are to maximize the separation of the nodes in the object tree and to minimize occlusion. It is assumed that the viewing direction is always pointing at the center of the assembly and that the field of view and distance of the camera are selected so that all nodes are visible. This effectively constrains the camera to a hemisphere above the assembly as in [4], [5].

A. Object Node Separation

The contact information among the different components of the assembly is used to determine areas of interest within the image [9]. Maintaining a spatial separation between these components within the image improves the performance of the inspection algorithm by preventing interaction between object nodes. Thus, it becomes useful to see the distances between these components as close as possible to their true lengths.

To determine the view direction in which the apparent distances between the nodes are as close as possible to their true length, the singular-value decomposition (SVD) is used. Emphasis is placed on shorter distances by inversely weighting the component displacement vectors by their magnitude. Accumulating the displacement vectors $[x_i\ y_i\ z_i]$ into a matrix results in

$$A = \begin{bmatrix} \frac{x_1}{m_1^2} & \frac{y_1}{m_1^2} & \frac{z_1}{m_1^2} \\ \frac{x_2}{m_2^2} & \frac{y_2}{m_2^2} & \frac{z_2}{m_2^2} \\ \vdots & \vdots & \vdots \\ \frac{x_e}{m_e^2} & \frac{y_e}{m_e^2} & \frac{z_e}{m_e^2} \end{bmatrix} \in R^{e\times 3} \qquad (2)$$

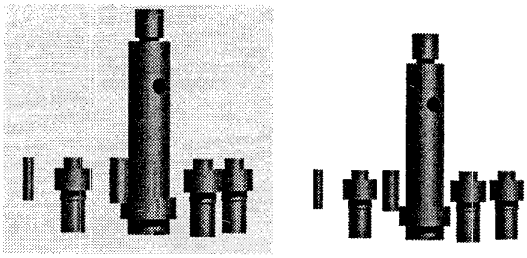
where $m_i^2 = x_i^2 + y_i^2 + z_i^2$, and n is the number of components of interest so that all combinations of displacement vectors results in $e = \frac{(n^2-n)}{2}$. The SVD of A,

$$A = \sum_{i=1}^{3} \sigma_i \hat{u}_i \hat{v}_i^T \qquad (3)$$

with $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$, provides quantitative information concerning the quality of various viewing directions. If maximizing separation were the only criteria then the viewing direction $\hat{v}_3$ gives the view direction from which the graph edges will be seen as close as possible to their true length (with preference given to shorter lengths). However, the effects of occlusion need to be considered. Therefore, rather than selecting the viewing direction as $\hat{v}_3$, the effects of occlusion are studied for candidate viewing directions that lie in the plane described by $\hat{v}_2$ and $\hat{v}_3$ as described in the following section. An illustration of the above procedure is presented in Fig. 4 for the example assembly given in Fig. 1. It is interesting to note how close the SVD calculation comes to a totally unoccluded view shown in Fig 4(b).

B. The Visibility Function ($\mathcal{V}$)

A visibility function $\mathcal{V}$ is used to quantify the quality of candidate viewing directions identified using the procedure described above. Clearly, the more areas of possible assembly errors are visible, the better performance one can expect from the inspection algorithm. As a result, the issue of visibility is addressed in terms



(a) (b)

Fig. 4. (a) Viewing the shaft and the pins along $\hat{v}_2$ obtained from (3) (b) Viewing the shaft and the pins using a totally unoccluded view direction in the plane spanned by $\hat{v}_2$ and $\hat{v}_3$.

of the number of visible components, the number of visible faces on each component, and the number of image pixels associated with each face:

$$\begin{aligned} \mathcal{V} = \ & c_1 N_c + c_2 \textstyle\sum_{i=1}^{N_c} (1 - \exp^{-F_i F_i^0}) \\ & + c_3 \textstyle\sum_{i=1}^{N_c} \frac{\sum_{j=1}^{F_i} (1 - \exp^{-P_{ij} P_{ij}^0})}{F_i} \end{aligned} \qquad (4)$$

where,

- N_c number of visible components of interest
- F_i number of visible faces on component i
- F_i^0 variation of surface normal on visible faces
- P_{ij} number of visible pixels on face j of component i
- P_{ij}^0 contact information for face j
- c_k empirically determined constants.

The motivation for using exponential functions in the various terms of $\mathcal{V}$ is due to the fact that errors in a component's assembly are propagated to the various surfaces of that component since it is a rigid body. Therefore additional surfaces on a single component simply provide more information about errors in that component whereas surfaces from other components can be used to broaden the range of errors that can be identified.

The value of F_i^0 controls the rate of increase in the exponential function with respect to F_i. Since faces that have widely varying surface normals will have wider shading variations, such surfaces are desirable since there is more information that will be available to the inspection algorithm. Therefore, F_i^0 is calculated to sense the degree to which the surface normals of the visible surfaces on component i vary. This is done by first associating with each face a dominant surface normal. For planar faces this dominant surface normal is simply the unique face normal. For curved faces, the dominant normal is calculated as the average surface normal weighted by the number of visible pixels that have that normal. To obtain a measure of how much

all of these dominant face normals vary over the entire component, they are concatenated into an F_i by 3 matrix denoted N. The SVD of N,

$$N = \sum_{m=1}^{3} \sigma_m \hat{u}_m \hat{v}_m^T \tag{5}$$

provides information about how the dominant surface normals are distributed over the entire component. This information is used to calculate the exponential coefficient F_i^0 using:

$$F_i^0 = \frac{\sum_{m=1}^{3} \sigma_m}{3\sigma_1} \tag{6}$$

so that $\frac{1}{3} \leq F_i^0 \leq 1$.

In an analogous manner, the coefficient P_{ij}^0 is used to emphasize faces that provide more information to the inspection algorithm. In this case, the displacement of faces that are in contact with other components are more likely to result in visible effects from assembly errors. Therefore, the more surfaces that are in contact with face j of component i, the larger its value of P_{ij}^0 .

V. Light Placement

To determine an optimal light source position, our approach is to experimentally determine the effects of light position on the inspection algorithms ability to distinguish both translational and rotational errors in assembled components. An analysis of these experimental results is then used to develop an algorithm that can automatically determine good light locations by evaluating a metric $\mathcal{L}$.

A. The Effect of Light Position on Performance

An experiment based on raytraced synthetic images and real video images was used to study the effect of light on the performance of the inspection algorithm. The simple test assembly, illustrated in Fig. 5, consists of a pin inserted in a hole. In the experiment the camera is placed at 45 degrees from the top of the pin in the X-Y plane, which was determined to be optimal based solely on the camera placement algorithm discussed above. Different light positions located 10 degrees apart in the X-Y plane are then tested to determine how accurately the inspection algorithm can detect errors in the pins location relative to the plane into which it was inserted. For each light position the inspection algorithm is trained on the correct assembly and then used to test assemblies that have various degrees of rotational (misalignment) and translational (misinsertion) errors. The log likelihood statistics from the inspection algorithm are a measure of how much the

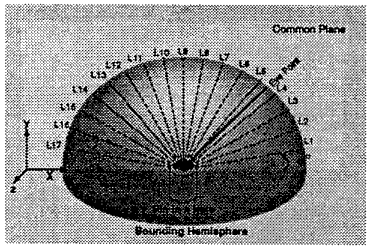


Fig. 5. Experiment setup used to test the effect of light on the performance of the inspection algorithm. At every light position synthetic images are used to train the algorithm. Then errors are introduced to the images. The effectiveness of detecting these errors shows the effect of light on performance.

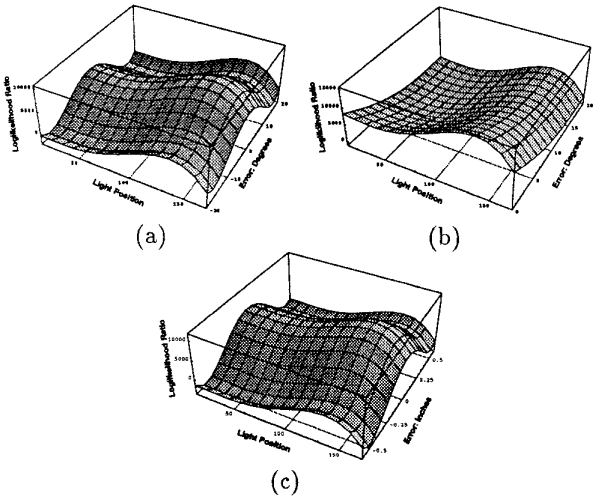


Fig. 6. (a) A quadratic fit among the log likelihood match results from the experiment shown in Fig. 5 with rotational errors around the Z axis. (b) Same as (a) with rotational errors around the X axis. (c) A quadratic fit among the log likelihood match results from the experiment shown in Fig. 5 with horizontal insertion errors.

incorrect assemblies match the images of the correctly inserted pin. These results are plotted as a function of both the light position and the degree of error in the insertion for both types of errors in Fig. 6. Fig. 6(a) shows the results for misalignment errors between -20 and 20 degrees around the Z axis. Fig. 6(b) illustrates misalignment errors between 0 and 20 degrees around the X axis (negative rotations around the X axis generate symmetrical images). Finally, Fig. 6(c) shows the results from inserting the pin to an incorrect depth, between ± 0.5 in. Note that in all three cases the algorithm is most sensitive to the errors when the light source is positioned at 135 degrees, the perfect specular direction for the top surface of the pin in its correctly assembled location.

B. The Light Source Placement Algorithm

Based on the above and similar experiments, it has been empirically determined that the statistical model built by the inspection algorithm is most sensitive in cases where variations in the surfaces of interest are located at orientations that correspond to perfect specular reflections. An analysis of the experimental data has allowed us to characterize why the appearances change rapidly due to assembly errors into the following three categories:

1. A visible surface is displaced such that the intensity of its specular reflection changes rapidly.
2. A surface with a normal different from the surrounding surfaces is covered or uncovered.
3. A surface is displaced in such a way that either casts or removes a shadow.

Clearly, placing the light source so that one receives a specular reflection from the surfaces of interest utilizes the first point (in much the same manner as a potential customer evaluates the paint job on an automobile). It is not as obvious, however, that it also utilizes the second category of appearance variation. This is true because specular highlights of polished surfaces tend to decrease rapidly as the surface normal varies. Therefore, it is statistically unlikely that uncovering a random surface will result in a high degree of specular reflection. Therefore, our light placement algorithm attempts to find light locations that attain a high degree of specular reflection from the surfaces of interest, i.e., those at which assembly errors are likely.

To accomplish this task, all of the visible surfaces are sampled and the resulting pixels, denoted p_{ij} for the jth face of the ith component, are used to determine a least-squares fit for the light location that maximizes specular reflection. For each pixel p_{ij}, a vector $\hat{h}_{ij}$ is calculated which represents a unit vector halfway between the surface normal, $\hat{n}_{ij}$, and the viewing direction, $\hat{v}_{ij}$. All of these $\hat{h}_{ij}$ vectors are concatenated into a matrix H, the SVD of which provides quantitative information about the dominant value of $\hat{h}$ and its variation. The average p_{ij} is then used to calculate the angle, θ, between $\hat{h}$ and the viewing vector. For optimal specular reflections the light direction $\hat{l}$ is then calculated by a clockwise rotation of $\hat{h}$ by 3θ in the plane containing the viewing vector.

C. The Illumination Function ($\mathcal{L}$)

To quantitatively evaluate the quality of a particular light source location, the following equation is used:

$$\mathcal{L} = \sum_{i=1}^{N_c} \left(\frac{F_i^v}{F_i} \left(\frac{\sum_{j=1}^{F_i} \sum_{k=1}^{n_{ij}} \hat{l} \cdot \hat{R}_{ijk}}{2F_i n_{ij}} + .5 \right) \right) \quad (7)$$

where,

N_c number of visible components of interest
F_i number of visible faces on component i
F_i^v number of F_i faces not in shadows
n_{ij} number of points from face j, component i
$\hat{R}_{kij}$ perfect specular direction for kth sample point p_{ij}.

The value of $\mathcal{L}$ is a measure of the effectiveness of a particular lighting direction based on the portion of the visible surfaces of an assembly component that are not shadowed from the light and how close it is to the perfect specular direction of these surfaces. The determination of whether the sampled points p_{ij} are shadowed from the light source is efficiently calculated by using the Z-buffer hardware in a manner analogous to that described in section III-A except that light location replaces the camera location.

VI. The Generate-and-Test Analysis

While the results of section IV can be used to determine a camera location and then used to apply the results of section V to determine a light location, this will not result in an optimal camera-light pair since the camera location affects the illumination function $\mathcal{L}$. To determine an optimal camera-light pair, the camera is first only constrained to lie in the plane determined from (3) and then a linear combination of the functions $\mathcal{V}$ and $\mathcal{L}$ is evaluated for camera locations that lie in this plane. The optimal value of this overall function $\mathcal{M}$, given by

$$\mathcal{M} = C_{\mathcal{V}}\mathcal{V} + C_{\mathcal{L}}\mathcal{L} \quad (8)$$

where $C_{\mathcal{V}}$, $C_{\mathcal{L}}$ are constants, is then used to determine the optimal camera-light placement pair. Results for the simple example used throughout this article are presented in the following section.

VII. Results

The camera and light placement algorithm was initially tested on simple assemblies to verify its performance. Then, it was tested on more complex assemblies. In this section the results from running the algorithm on the wheel assembly shown earlier in Fig. 1 are presented. The pins of the wheel assembly were used as the components of interest. The camera was constrained to lie on a semicircle as described in section IV-A. This set of valid camera locations was then sampled and the camera-light pair function $\mathcal{M}$ was evaluated. Fig. 7 plots the different components of $\mathcal{M}$. Curve (a) shows the second term of $\mathcal{V}$ (eq. (4)). It shows that the visibility of the components' faces diminishes at near horizontal and vertical views. On the

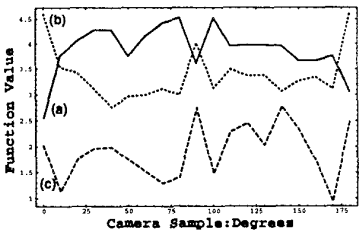
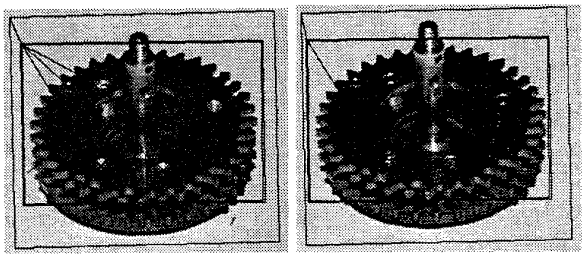


Fig. 7. The values of the different terms in $\mathcal{M}$ for the assembly shown in Fig. 2 when the camera is constrained to lie on the plane determined using (3). The first term of $\mathcal{V}$ is constant (not plotted), the second term is denoted (a), the third term (b), and (c) shows $\mathcal{L}$.



(a) (b)

Fig. 8. (a) Error in top wheel placement. The location of a mismatch is identified by a rectangle with an "X" mark. The tree shown in Fig. 2 is used here. (b) Error in high density pin insertion.

other hand, plot (b), the third term of $\mathcal{V}$, shows higher values at these views since equally large face areas are visible on the different assembly components of interest. Similarly, curve (c) of the $\mathcal{L}$ function shows better light positions at horizontal views because of the better match to the specular direction of the different visible faces. The combination of these terms in $\mathcal{M}$ lead to preferring the inclined views. For example, setting all the constants to unity except for c_3 which is set to .1 leads to selecting the view at 140 degrees shown in Fig. 2. Testing the real assembly from different views after training on synthetic images showed the advantages of using the inclined views around 140 degrees. Fig. 8 shows two examples. Fig. 8 (a) shows a detected error caused by misplacing the top wheel. This error passes undetected from a horizontal view. Fig. 8 (b) shows a detected pin insertion error which passes undetected from a vertical view.

VIII. Conclusion

This article has discussed automatic camera and light source placement for an assembly inspection system that uses a multiscale algorithm to detect errors in assemblies after being trained on synthetic images of correctly assembled products. It was shown that the performance of this inspection algorithm can be improved by optimizing an empirically determined function that describes the quality of an image based on the visibility and intensity of the components of interest.

References

[1] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai, "A survey of sensor planning in computer vision," *IEEE Trans. Robot. Automat.*, vol. 11, no. 1, pp. 86–104, Feb. 1995.

[2] K. Tarabanis, R. Y. Tsai, and P. K. Allen, "Automated sensor planning for robotic vision tasks," *in Proc. 1991 IEEE Int. Conf. Robot. Automat.*, pp. 76–82, (Sacramento, CA), April 1991.

[3] C. K. Cowan, "Automatic camera and light-source placement using CAD models," *Proc. IEEE Workshop on Directions in Automated CAD-Based Vision*, pp. 22 – 31, (Maui, HI), June 2-3 1991.

[4] S. Yi, R. M. Haralick, and L. G. Shapiro, "Automatic sensor and light source positioning for machine vision," *Proc. 10th Int. Conf. Pattern Recognition*, pp. 55–59, 1990.

[5] S. Sakane, M. Ishii, and M. Kakikura, "Occlusion avoidance of visual sensors based on a hand-eye action simulator system: Heaven," *Advanced Robotics*, vol. 2, no. 2, pp. 149–165, 1987.

[6] S. Sakane and T. Sato, "Automatic planning of light source and camera placement for an active photometric stereo system," *Proc. 1991 IEEE Int. Conf. Robot. Automat.*, pp. 1080–1087, (Sacramento, CA), April 1991.

[7] F. Solomon and K. Ikeuchi, "An illumination planner for Lambertian polyhedral objects," *Proc. 1995 Int. Conf. Robot. Automat.*, pp. 1719–1725, (Nagoya, Japan), May 21-27 1995.

[8] D. Tretter, C. A. Bouman, K. W. Khawaja, and A. A. Maciejewski, "A multiscale stochastic image model for automated inspection," *IEEE Trans. Image Proc.*, vol. 4, no. 12, Dec. 1995.

[9] K. W. Khawaja, A. A. Maciejewski, D. Tretter, and C. A. Bouman, "Automated assembly inspection using a multiscale algorithm trained on CAD-generated synthetic images," to appear in *IEEE Robot. Automat. Mag.*

[10] T. Whitted, "An improved illumination model for shaded display," *Comm. ACM*, vol. 23, no. 6, pp. 343–349, June 1980.

[11] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *ACM Trans. Graphics*, vol. 1, no. 1, pp. 7–24, Jan. 1982.