2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Viewpoint Selection for Vision Systems in Industrial Inspection

Jose Luis Alarcon-Herrera, Xiang Chen and Xuebo Zhang

Abstract— **An automatic method for solving the problem of view planning in high-resolution industrial inspection is presented. The method's goal is to maximize the visual coverage, and to minimize the number of cameras used for inspection. Using a CAD model of the object of interest, we define the scene-points and the viewpoints, with the later being the solution space. The problem formulation accurately encapsulates all the vision- and task-related requirements of the design process for inspection systems. We use a graph-based approach to formulate a solution for the problem. The solution is implemented as a greedy algorithm, and the method is validated through experiments.**

I. Introduction

Automation of metrology and inspection tasks for quality control is becoming ubiquitous in all types of industries. From food to pharmaceutics and most commonly in the automotive industry, vision systems are being deployed often enough that automation of the design process of the vision system itself is now required. Visual inspection in industrial settings is usually performed by processing images at resolutions on the order of tens of millimetres to tens of micrometers. Moreover, the inspection requirements often push the limits of modern devices when one considers the speeds at which these systems are expected to perform. Hence, the need to make the most out of design parameters such as the position and orientation of cameras becomes paramount.

The design process itself is, in most cases, executed by a vision specialist, who will integrate a solution using off-the-shelf cameras and optics. The crucial part of the design is to develop a camera layout, that is, deciding on the position and orientation of the camera or cameras such that the inspection scene is imaged with sufficient quality. Even when using the device's manufacturer's guidelines for positioning, these are only approximations that leave out important details about the vision system for the sake of generality and compatibility. In consequence the vision specialists find themselves performing trial and error to refine the solution. This is, at best, a difficult task and sometimes prohibitive because of the cost of implementing a camera layout for the sole purpose of testing it. Thus, it is desirable to have access to an automatic method for viewpoint selection. One advantage of industrial inspections is that, very often, the 3D model of the part or parts to be inspected is available to the design process. In this work we assume that such a model is available.

Jose Alarcon and Xiang Chen are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, Ontario N9B 3P4, Canada {alarconj,xchen}@uwindsor.ca. Xuebo Zhang is with the Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin 300071, China. zhangxb@robot.nankai.edu.cn

The objective of this work is to provide vision specialists with a tool for automatically producing camera layouts for industrial inspection. For simplicity of presentation, at this point we refer to this tool as a black box with the following inputs and outputs, which will be described in more detail in Section IV.

TABLE I
Design Tool

Inputs	Outputs
CAD model	List of camera poses
List of cameras	Expected performance metric
List of lenses	
Task requirements	

The originality of this work lies on the simple yet clever use of graph tools for solving the problem of viewpoint selection. First, we introduce some important concepts such as the performance metric and a formal description of the problem in Section II. Second, in Section III we describe a viewpoint generation process similar to that used by Tarbox and Gottschlich [1]. In Section IV we present our viewpoint selection algorithm, together with experimental results in Section V. Finally, we offer directions for future work and some concluding remarks in Section VI.

A. Related Work

The literature can be subdivided in several categories. According to the type of the target application, one can find viewpoint selection for visual surveillance and 3D active triangulation. In the former, the problem usually involves multiple cameras and multiple targets (see Zhao et al. [2]), and in the latter, the problem often involves single range cameras and single targets (see Pito [3]). This work targets industrial inspections with non-range cameras, also, our problem formulation considers multiple cameras and single targets. A different categorization of the literature can be made according to the approach used to solve the problem of viewpoint selection. Erdem and Sclaroff [4] tackle the problem of viewpoint selection and cost minimization for specific task requirements, and propose a reduction of the problem to the set covering problem by mapping the field of view to feasible regions in two dimensions and by mapping the task-based constraints of the system to area coverage constraints. Mittal and Davis [5] design a cost function for sensor planning based on the average capture quality. The authors implement a simulated annealing algorithm to minimize the cost function. Reddy and Conci [6] solve for automatic camera positioning using a particle swarm optimization algorithm, which operates in two dimensions.

Scott [7] proposes a solution to the viewpoint selection problem by first defining a solution space and generating a matrix that measures visual coverage according to some custom metric. The solution is found by a greedy search that operates over the entries of the matrix, where the rows represent points in the scene and the columns represent viewpoints. The solution is then to cover as many rows with as few columns as possible.

II. The Viewpoint Selection Problem

A. Modeling the Vision System

In this work we make use of a coverage model. This model considers four visual components: resolution, focus, view angle, and visibility. The visual requirements of the task are implemented as a set of parameters, listed in Table II and defined in units of length (l), pixel units (px), and/or angle units (a). In the first three parameters, the subscript i stands for "ideal" and a stands for "acceptable", the latter should be set to the limit at which the task produces no useful information, and the former should be set to the point beyond which the task performance does not increase. The values of these task parameters can be set by the client's specifications, or empirically.

Resolution measures the number of units of length that are represented by a pixel in the image sensor. The blur circle requirement captures the need for sharpness in the image by setting a threshold on its diameter (focus). The view angle parameter measures the extent to which the task can be performed in the presence of skewed images. Finally, the image boundary padding is used to account for the visibility requirement of some vision tasks, which is that these perform better when there is a small neighbourhood around feature points in the image plane.

TABLE II
Task Parameters

Parameter	Description
R_i, R_a	minimum ideal/acceptable resolution (l/px)
c_i, c_a	maximum ideal/acceptable blur circle (px)
ζ_i, ζ_a	maximum ideal/acceptable view angle (a)
γ	image boundary padding (px)

The coverage model of a vision system models multiple cameras, the environment, and the task. The camera system is approximated using the pin hole model, which accounts for the sensor's properties, and the coverage model additionally considers the lens' properties. The geometric environment model is a set of three-dimensional surfaces and it accounts for static occlusion in the scene. The task is modeled using a set of directional points, which are three-dimensional points with a direction component, and a set of task parameters. The directional points model the target to an arbitrary degree of precision, and the task parameters model the visual requirements of the task. The coverage function $C(v, p)$ is bound to the range $[0, 1]$ and it represents the grade of coverage at a point p as seen from viewpoint v. In order to represent the grade of relevance of each point to the task, the relevance function $R(p)$ is used and it is also bound to the range $[0, 1]$. Finally the coverage performance of the sensor system with respect to the task is given by

$$F(C, R) = \frac{\sum_{p \in \langle R \rangle} C(p)R(p)}{\sum_{p \in \langle R \rangle} R(p)} \tag{1}$$

in this case $C(p)$ represents the coverage at point p in the multi-viewpoint case. $\langle R \rangle$ is a finite, discrete task point set induced by R. p is a directional point in camera coordinates.

The coverage model was developed by Mavrinac et al. [8]. Since we will use the coverage model's metric as the objective function in our proposed algorithm, we need to make sure that the model is reliable. To this end we refer the reader to the validation of the model, which is provided in Alarcon et al. [9], and Mavrinac and Chen [10].

B. The Problem

Given an ordered list of directional scene-points p and an ordered list of viewpoints v representing feasible camera poses, find the minimum number of viewpoints that maximizes the coverage of the scene-points. The formal problem is described as an integer programming problem.

$$\text{minimize} \sum_{j=0}^{|v|-1} x_j, \quad \text{subject to} \tag{2}$$

$$\sum_{j=0}^{|v|-1} \sum_{i=0}^{|p|-1} \frac{C(v_j, p_i)x_j}{|p|} \geq 1 \tag{3}$$

$$x_j \in \{0, 1\}; \quad j = 0, 1, \ldots, |v| - 1 \tag{4}$$

Equation (4) is a binary condition over the list v, meaning that the value of x_j is 1 if viewpoint v_j is in the candidate solution, and 0 otherwise. Equation (3) sets a constraint on the coverage strength for all points so that the coverage performance be 1. Please note the we reserve subscript i for scene-points and subscript j for viewpoints, and that $|\cdot|$ represents the length of the list.

III. The Solution Space

A. Scene Discretization

From the previous problem formulation we can see that the size of the solution space is indeed very large, and is in fact $2^{|v|}$. Although we could reduce the size of the solution space by removing all binary combinations of the elements in x that are larger than the minimum necessary number of viewpoints, we have no *a priori* way of computing this number. In addition, a brute force algorithm that tests all possibilities would be untraceable because its order of growth is, at best, exponential with respect to the size of the viewpoint list.

In order to devise a solution to the problem in Section II-B, we first discretize the scene. To this end we import the CAD file of the part to be inspected, and extract the triangular mesh. Then, we compute the centroid of each triangle and the normal vector to the plane where the triangle lies. Using each of the triangles' centroids and normals we construct an ordered list of directional points. In this work we assume that

the list of directional points effectively models the geometry of the part for the purposes of inspection.

B. Viewpoint Generation

We are particularly interested in populating the solution space only with feasible viewpoint candidates. Thus, we use an approach similar to that used by Tarbox and Gottschlich [1]. For every point in the scene point list, an optimal viewpoint v_j can be generated using the following guidelines:

- A viewpoint along the line that extends from the scene-point's normal provides visibility coverage strength of 1.0.
- A viewpoint with optical axis aligned with the line that extends from the scene-point's normal provides angle of view coverage strength of 1.0.
- A viewpoint located at the standoff distance away from the scene-point provides resolution coverage strength of 1.0.

The explanation for these guidelines is as follows. First, a scene-point located along the optical axis maps to the centre of the image, hence ensuring maximum visibility. Second, let $\mathcal{P}$ be the plane tangent to the surface where the scene-point lies, and let $\mathcal{I}$ be the image plane associated with the viewpoint. Then, a scene-point with a normal that is aligned with the optical axis of the viewpoint is equivalent to having planes $\mathcal{P}$ and $\mathcal{I}$ be parallel with respect to each other and thus avoid skewed images. Third, since resolution measures the units of length represented by a pixel in the image, resolution is a function of distance. The standoff distance is the furthest a viewpoint can be before resolution falls bellow a predetermined threshold. Equations (5) to (7b) show how to compute the viewpoint space.

The standoff distance.

$$d_S = R \ \min\left(\frac{w}{\tan\alpha_l + \tan\alpha_r}, \frac{h}{\tan\alpha_t + \tan\alpha_b}\right) \tag{5}$$

where d_S is the standoff distance. R is the desired resolution substituting for R_i or R_a (see Table II). w and h are the width and height, in pixels, of the camera sensor. α_l, α_r, α_t, and α_b are the left, right, top, and bottom angles of the field of view of the camera, respectively.

The position component.

$$v_x = p_x + d_S \ sin(p_\rho) \ cos(p_\eta) \tag{6a}$$
$$v_y = p_y + d_S \ sin(p_\rho) \ sin(p_\eta) \tag{6b}$$
$$v_z = p_z + d_S \ cos(p_\rho) \tag{6c}$$

The direction component.

$$v_\rho = p_\rho + \pi \tag{7a}$$
$$v_\eta = p_\eta \tag{7b}$$

where viewpoint v is defined as $v = [v_x, v_y, v_z, v_\rho, v_\eta]$, and similarly for scene-point p. In both cases subscripts x, y, and z represent the position, and subscripts ρ and η represent the direction formatted in spherical coordinates.

C. Graph-Based Data Structure

Given the ordered list of scene-points p and the associated list of viewpoints v, we build a data structure that will help us formalize our viewpoint selection algorithm latter in Section IV. Note that since every scene-point yields a viewpoint then $|p| = |v|$, hereafter we use n to denote the size of either p or v. The interaction topology between the scene-points and the viewpoints is represented using a graph $G = (V, E, w)$ with the set of vertices $V = \{p_i, \cdots, p_n, v_j, \cdots, v_n\}$ for $i = 0, \cdots, n-1$ and $j = 0, \cdots, n-1$, and edges $E \subseteq V \times V$. An edge exists between two vertices if one of the vertices is visible from the other. Additionally, the weight of the edge is equal to the coverage strength resulting from evaluating the pair of vertices in the edge, $w = C(v_j, p_i) \ \ i, j \ \in \ n$. It is evident by now that G is a directed graph because only viewpoints can have visibility, in other words, $C(p_i, v_j) = 0 \ \ \forall \ i, j \ \in \ n$ and $C(p_i, p_j) = 0 \ \ \forall \ i, j \ \in \ n$. To illustrate this, Figure 1 shows the adjacency matrix of vision graph G.

	$v_j, \ldots, v_n$	$p_i, \ldots, p_n$
$p_i, \ldots, p_n$	$C(v_j, p_i)$	0
$v_j, \ldots, v_n$	0	0

Fig. 1. Adjacency matrix of vision graph G

Note that although viewpoints could, in theory, see other viewpoints, we are not interested in a camera configuration where viewpoints are part of the scene. To avoid this we set $C(v_i, v_j) = 0 \ \ \forall \ i, j \ \in \ n$. Our data structure is then, the n by n resulting matrix. This matrix is very similar to the *measurability matrix* used by Scott [7]. One important difference between the work by Scott and this work, is that in [7] the viewpoint selection is solved by an algorithm that directly operates over the entries of the measurability matrix. In this work we use this matrix as a first step to building a graph of visual overlap (see Section IV). And so we leverage our work beyond this concept. For the remaining of this paper we will refer to this matrix as the vision matrix M.

In order to handle and extract some useful information from the vision matrix, we present some tools for computing the coverage of each viewpoint and the degree of overlap between viewpoints. The list of points that are covered from viewpoint v_j is the column vector $M[:, j]$. The coverage of a single viewpoint $c(v_j)$ is defined as the coverage strength provided by that viewpoint for all scene points.

$$c(v_j) = \frac{1}{n} \sum_{i=0}^{n-1} M[i, j] \tag{8}$$

The maximum coverage $c_{\max}$ for all viewpoints in M is

$$c_{\max} = \max\left(c(v_j)\right)\Big|_{\forall v_j \in v} \tag{9}$$

For any two viewpoints v_j and v_k the degree of coverage overlap is defined as the dot product of the two column

vectors $M[:,j]$ and $M[:,k]$, normalized by the maximum coverage $c_{\max}$.

$$o(v_j, v_k) = \frac{\frac{1}{n}\sum_{i=0}^{n-1}\left[M[i,j]\cdot M[i,k]\right]}{c_{\max}} \tag{10}$$

IV. Viewpoint Selection Algorithm

The objective in this work, as stated in Table I, is to output a list of camera poses for improved visual quality. In addition we also seek to minimize the number of cameras needed for inspection, as well as optimize the hardware selection, from a list of cameras and lenses. In this Section we describe how to perform these three tasks.

A. Viewpoint Placement

Given the ordered list of scene-points p and the associated viewpoints v, we compute the vision matrix M and fill its entries with the coverage strength $C(v_j, p_i)$ for each scene-point and viewpoint pair. The motivation for building this matrix is that it enables us to quantify the degree of coverage $c(v_j)$ that each viewpoint candidate can provide. Additionally, using the degree of visual overlap $o(v_j, v_k)$ we can generate the system's graph of coverage overlap. Let $O = (V, E, w)$ be the weighted undirected overlap graph, with the set of vertices $V = \{v_j, \cdots, v_n\}$ for $j = 0, \cdots, n-1$, and edges $E \subseteq V \times V$. There exists an edge between a pair v_j and v_k if the viewpoints simultaneously cover some of the scene-points. The weight w is equal to the overlap as given by (10).

Using both overlap graph O and the row vector of coverage $\left[c(v_j)\right]\big|_{\forall v_j \in v}$, we compute the compounded degree for each vertex in O. The compounded degree d^c of a vertex v_j is defined as the product between the sum of the weights of all the edges that contain said vertex and the corresponding coverage $c(v_j)$ and is given by

$$d_j^c = c(v_j) \sum_{k=0}^{|h^j|-1} w_{jk} \tag{11}$$

where k is used to iterate over the list h^j of neighbours of v_j, and w_{jk} is the weight of the corresponding edge.

The premise of our viewpoint selection method is the following. Let $o(v_j, v_k) = 1.0$ be the overlap shared by viewpoints v_j and v_k. Since the overlap is non-zero, any viewpoint of the two performs as good as the other at covering the shared scene. Furthermore, if v_j has non-zero overlap with other viewpoints, then v_j becomes a desirable viewpoint for selection since it can replace its neighbours. It is important to be careful when using this criteria for viewpoint selection since a high degree of overlap does not guarantee a high degree of coverage. This is why, in (11), we multiply the sum of the weights by the coverage strength.

A greedy algorithm is implemented in order to perform viewpoint selection. A list of the compounded degrees of all vertices in O is ordered in descending order with respect to d^c. The algorithm starts by selecting the viewpoint with the highest degree, and proceeds to eliminate from the list all the other viewpoints that are direct neighbours in the overlap graph. This is repeated until the original list of compounded degrees is empty. The motivation for eliminating neighbouring viewpoints is that this reduces redundancies (i.e. excessive overlap), and thus maximizes the area being covered by the viewpoints. Algorithm 1 shows the viewpoint placement process.

Algorithm 1 Viewpoint Placement

Require: p
From p, generate v as given by (5) to (7b)
for $i = 0 \rightarrow n$ **do**
  for $j = 0 \rightarrow n$ **do**
    $M[i,j] = C(v_j, p_i)$
  end for
end for
$V \leftarrow v$
pairs $\leftarrow$ non-repeated combinations of V
for pair $\in$ pairs **do**
  append $E \leftarrow$ pair
  append $w \leftarrow o$(pair), as given by (10)
end for
$O = (V, E, w)$
for $v_j \in V$ **do**
  append $D^c \leftarrow d_j^c$, as given by (11)
end for
sort D^c in descending order
while $|D^c| > 0$ **do**
  pivot $\leftarrow D^c[0]$
  append $S \leftarrow pivot$
  remove $D^c[0]$ from D^c
  N $\leftarrow$ neighbors of pivot in O
  for neighbor $\in$ neighbors **do**
    if neighbor $\in D^c$ **then**
      remove neighbor from D^c
    end if
  end for
end while
return S

B. Viewpoint Minimization

After performing viewpoint placement, we proceed to minimize the number of viewpoints for inspection. We employ a binary search algorithm that operates over the list of selected viewpoints as given by the algorithm in Algorithm 1. The binary search works as follows. Given a list of viewpoints, we cut the list in half and test its coverage performance $F(C, R)$, if the performance remains within a predefined threshold ε then we continue to cut the list by half. On the other hand, if the performance falls below ε then we increase the cut (i.e. selecting more than half of the viewpoints). The algorithm stops when the minimum number of viewpoints that can perform relatively close to the original is found. It is important to note that a binary search can be used only if the list of viewpoints is ordered with respect to the coverage performance. That is, there is a direct linear relationship between the number of cameras and the coverage performance of the list. This is true in this case because the viewpoint placement algorithm sorted the list of viewpoints based on the compounded degree, which is directly related to the coverage performance since it accounts for coverage strength. The binary search is implemented in Algorithm 2.

C. Hardware Selection

At this stage we have all the tools we need to optimize viewpoint selection over a list of cameras and lenses. To this end, we prepare a list of devices which is the combination

Algorithm 2 Viewpoint Minimization

Require: S
best $\leftarrow$ $F(S)$, as given by (1)
bestNumber $\leftarrow$ $|S|$
cut $\leftarrow$ 0.5
temp $\leftarrow$ $S[0:(|S|\cdot cut)]$
while $|temp| \neq$ bestNumber **do**
  $f \leftarrow F(temp)$
  if $abs(\text{best} - f) < \varepsilon$ **then**
    $S \leftarrow S[0:(|S|\cdot cut)]$
    cut $\leftarrow$ 0.5
    bestNumber $\leftarrow$ $|S|$
  else
    cut $\leftarrow$ cut $\cdot 1.1$
  end if
  temp $\leftarrow$ $S[0:(|S|\cdot cut)]$
end while
return $temp$, f

of cameras and lenses. Given a list of cameras L^C and a list of lenses L^L, the list of devices $L^D = [L^C \times L^L]$such that $|L^D| = |L^C| \cdot |L^L|$. To choose the best camera-lens combination we simply test all the elements in L^D by computing the corresponding $F(C, R)$ and select the combination with the highest performance. Although testing all elements in L^D may seem computationally expensive, one must keep in mind that in industrial environments, the list of cameras and lenses is initially narrowed by the vision engineer who performs the design. Hence, L^D is usually present in small sizes (i.e. 5 to 10 camera-lens combinations).

V. Experimental Results

We first test our viewpoint selection method using simulations over the CAD models of a door and hood parts. We input a list of one camera and a list of five lenses for a total of five hardware combinations. The list of lenses varies in focal length from 4 to 16mm, and all lenses have an $f\#$ of 2.8. The task parameters were selected as follows: $R_i = R_a$, which is selected individually for each part, $c_i = 1.0$ and $c_a = 2.5$, $\zeta_i = \zeta_a = 1.5707$, and $\gamma = 0$.

Figure 2 shows the visualization of the output of the design tool. Figure 2a shows a visualization of the directional points located at the centroid of each triangle of the CAD models, and Figure 2b shows the field of view of one of the selected cameras as a red squared frustum. The results are presented in Table III. The table shows the name of the shape, the number of directional points in the CAD model, the desired resolution for inspection expressed in units of millimetres per pixel, the number of selected/optimized cameras, the total execution time in minutes, and the expected coverage performance of the selected arrangement.

TABLE III
Simulation Results

Shape	Points	Resolution	Selected	Time	$F(C, R)$
Door	342	1.6	43/12	2.60	0.9880
Hood	274	1.4	14/7	2.02	0.9841

In this experiment we use the coverage performance $F(C, R)$ as the metric for evaluating the effectiveness of our viewpoint selection method. And we encourage the reader to verify the validity of this coverage metric and the coverage model by referring to [8]–[10].

In order to provide a point of reference with existing methods for viewpoint selection, we take the work of Scott [7] and compare it to our own. As stated in Sections I-A and III-C, both methods use a similar data structure: the vision matrix. Furthermore, both methods use a scalar in the $[0, 1]$ range as a metric for performance. Hence, a direct comparison is possible, which is shown in Table IV.

TABLE IV
Comparison

Method	Shape	Points	Time	Performance
Our Method	Hood	274	0.404	0.9841
Scott [7]	Sphere	252	0.75	1.0

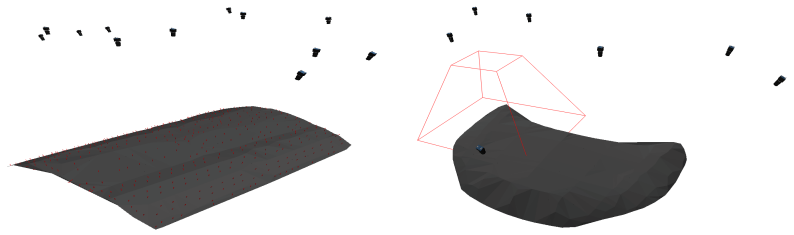
the results in Table IV are taken directly from [7], as reported by Scott. The hood shape was chosen for this comparison because it is the closest, in the size of the solution space, to the shape used by Scott in his experiment. Finally, the time is measured in minutes.

It is important to note that in [7] the author assumes that hardware has been selected and thus no camera-lens optimization is done. Our method on the other hand executes Algorithms 1 and 2 for each camera-lens combination (for a total of 2.02 minutes in the hood example). The time it takes our method to produce a solution, in this example, is in fact 0.404 minutes when considering only one camera-lens configuration, hence showing that our method outperforms that of Scott with respect to time while having a considerably good performance.

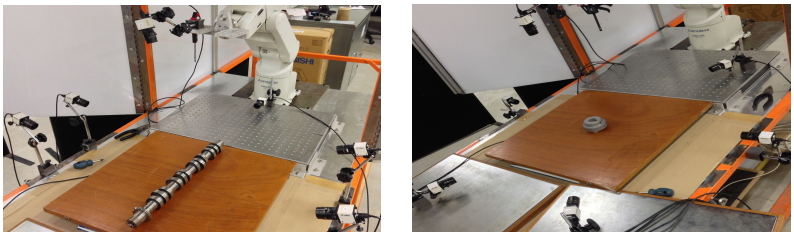
We now move away from the simulations and test our method using a physical apparatus. In this experiment we generate camera configurations to cover two mechanical parts: a crankshaft and a motor coupling, as seen in Figure 2. In this case only one type of camera (iCube NS4133BU) and one type of lens (NET SV-0813V) are used, and the task parameters were selected as follows: $R_i = R_a = 0.4$, $c_i = 1.0$ and $c_a = 5.0$, $\zeta_i = \zeta_a = 0.9599$, and $\gamma = 0$.

There is a problem inherent in the positioning of the cameras; unlike the simulation, we are unable to place the cameras at the exact location described by the generated camera configuration. The solution is to use extrinsic camera calibration as means of feedback and to repeat the process of positioning and calibrating until all cameras are close enough to their respective locations. Although the robotic manipulator in Figure 2c would have been an ideal solution to this problem, the reach of the robotic arm was too small to effect all camera positions. In this experiment we report that the mean euclidean error between the camera locations and the actual locations is 10mm.

Using image processing operations we are able to measure, from the images, some performance criteria. For example, the total area of the part that is visible to the cameras, the mean resolution at which the cameras image the part, and the greatest angle at which cameras can see non-parallel faces of the part. These measurements are reported in Table V



(a) Door (b) Hood



(c) Crankshaft (d) Motor Copuling

Fig. 2. Output of Design Tool and Physical Aparatus

under "Measured". The data reported under "Expected" is the result of the simulation after the camera configuration was generated.

TABLE V
RESULTS

Crankshaft	Expected	Measured
Visibility	96.58%	91.94%
Resolution	$0.4\text{mm}/px$	$0.35\text{mm}/px$
Angle	53.10°	51.98°

Copuling	Expected	Measured
Visibility	100%	98.21%
Resolution	$0.4\text{mm}/px$	$0.31\text{mm}/px$
Angle	54.74°	54.00°

From Table V we can see that both the visibility and the angle of view of the final configuration were lower than expected, although still close. On the other hand, the measured resolution outperformed the expected one. Thus, we confirm that our method for automatic viewpoint selection is able to perform while satisfying the visual requirement of resolution, which is most important for industrial inspections.

VI. Conclusions and Future Work

We have presented a novel method for viewpoint selection in industrial inspections. We implemented our method and devised a design tool that only needs as inputs: a CAD model, a list of cameras and lenses, and the desired resolution. The output is a list of the positions and orientations of the selected viewpoints. Experimental results have shown that the method performs well. One interesting feature of this method is that under the same input the method converges to the same solution under the same amount of time.

In the future we would like to relax some of the assumptions made in this work. For example, the scene is reduced to consider only one directional point per plane in the CAD model. Although this has worked well in practice, it is not always the case. This work was in part motivated by our collaboration with a local vision system's integrator. Thanks to our collaboration with our industrial partners, in the near future we plan to test our method and design tool in real production lines at the site of manufacturing.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada and by the National Council of Science and Technology of Mexico.

References

[1] G. Tarbox and S. Gottschlich, "Planning for Complete Sensor Coverage in Inspection," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 84–111, 1995.

[2] J. Zhao, S. C. Cheung, and T. Nguyen, "Optimal Camera Network Configurations for Visual Tagging," *IEEE Journal of Selected Topics in Signal Processing.*, vol. 2, no. 4, pp. 464–479, 2008.

[3] R. Pito, "A Solution to the Next Best View Problem for Automated Surface Acquisition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[4] U. M. Erdem and S. Sclaroff, "Automated Camera Layout to Satisfy Task-Specific and Floor Plan-Specific Coverage Requirements," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 156–169, Sept. 2006.

[5] A. Mittal and L. S. Davis, "A General Method for Sensor Planning in Multi-Sensor Systems: Extension to Random Occlusion," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 31–52, July 2008.

[6] K. K. Reddy and N. Conci, "Camera positioning for global and local coverage optimization." in *ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, 2012, pp. 1–6.

[7] W. R. Scott, "Model-Based View Planning," *Machine Vision and Applications*, vol. 20, no. 1, pp. 47–69, 2009.

[8] A. Mavrinac, J. L. Alarcon-Herrera, and X. Chen, "A Fuzzy Model for Coverage Evaluation of Cameras and Multi-Camera Networks," in *ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, 2010, pp. 95–102.

[9] J. L. Alarcon-Herrera, A. Mavrinac, and X. Chen, "Sensor Planning for Range Cameras via a Coverage Strength Model," in *IEEE/ASME Intl. Conf. on Advanced Intelligent Mechatronics*, 2011, pp. 838–843.

[10] A. Mavrinac and X. Chen, "Optimizing Load Distribution in Camera Networks with a Hypergraph Model of Camera Topology," in *ACM/IEEE Int. Conf. Distributed Smart Cameras*, 2011.