Automatic Sensor Placement for Model-Based Robot Vision

S. Y. Chen *Member, IEEE* and Y. F. Li*, *Senior Member, IEEE*

***Abstract*—This paper presents a method for automatic sensor placement for model-based robot vision. In such a vision system, the sensor often needs to be moved from one pose to another around the object to observe all features of interest. This allows multiple 3D images to be taken from different vantage viewpoints. The task involves determination of the optimal sensor placements and a shortest path through these viewpoints. During the sensor planning, object features are resampled as individual points attached with surface normals. The optimal sensor placement graph is achieved by a genetic algorithm in which a min-max criterion is used for the evaluation. A shortest path is determined by Christofides algorithm.**

A *Viewpoint Planner* is developed to generate the sensor placement plan. It includes many functions, such as 3D animation of the object geometry, sensor specification, initialization of the viewpoint number and their distribution, viewpoint evolution, shortest path computation, scene simulation of a specific viewpoint, parameter amendment. Experiments are also carried out on a real robot vision system to demonstrate the effectiveness of the proposed method.

***Index Terms*--Sensor placement, Viewpoints, Robot vision, Hierarchical genetic algorithm, Christofides algorithm.**

I. Introduction

With the rapid growth of automation in manufacturing industry, computer vision now plays an important role in product inspection, assembly, and design in reverse engineering, etc. Since a vision sensor can only sample a portion of an object from a single viewpoint, multiple 3D images need to be taken and integrated from different vantage points to enable all features of interest to be measured. Sensor placement which determines the viewpoints and viewing strategy thus becomes critically important for achieving full automation and high efficiency in such a process.

Sensor placement has been an active area of research in recent years. The relevant work can be classified into two application categories: model based and non-model based. Typical non-model based applications include 3D object reconstruction and modeling. Model based applications are widely used in product assembly, inspection, object recognition, dimension measurement, etc. where the object's geometry and a rough estimate of its pose are known. Previous approaches to sensor placement mainly focused on modeling of sensor constraints and calculating a "good" viewpoint to observe one or several features on the object. Little consideration is given to the overall efficiency of a generated plan with a sequence of viewpoints. The early work on sensor placement focused mainly on the analysis of placement constraints, such as resolution, focus, field of view, visibility, and conditions for light source placement in 2-D space [1]. More extensive surveys of the early work can be found in [2-5].

Sensor placement applying in object modeling was addressed for deciding the portion of the viewing volume to be scanned [6-8]. Hutchinson and Kak [9] planned the sensing strategies by a hypothesis and assessing method. Allen et al. extended their earlier sensor planning work [3, 10] to urban scene planning [11]. Comparing with an octree representation of a 3D scene [12], Banta and Abidi used uniformly sized voxels to represent the viewing volume [13]. The next best view (NBV) [14, 15] was identified as the one that would sample the most nonempty voxels and achieve maximum information gain in the object model.

Varying the view parameters will cause the observed features to undergo measurable local transformations which can be used to simplify and constrain the computation of unknown scene parameters. Kutulakos and Dyer [16] exploited the differential properties of smooth surfaces to model local changes in the appearance of an occluding contour due to camera motion. This knowledge enabled them to position the camera, first to extract occluding contours from an edge map, and then to use the extracted contours to sweep out the complete 3D shape. Arbel and Ferrie [17] showed how entropy maps can be used to guide an active observer along an optimal trajectory and how a gaze-planning strategy can be formulated by using entropy minimization as a basis for choosing a next best view.

Sensor planning for autonomous navigation [18-22] and active object recognition [23-26] has also been actively studied recently. For active 3-D object recognition, the system is an iterative active perception system that executes the acquisition of several views of the object, builds a stochastic 3D model of the object, and takes a decision where the best next view is. In this aspect, Okamoto et al. [23] conducted

Manuscript received March 26, 2002; revised March 15, 2003. This work was fully supported by the Research Grants Council of Hong Kong [Project No. CityU 1136/98E]. This paper was presented in part at the IEEE International Conference on Robotics and Automation, Washington, DC, May 2002.

S. Y. Chen and Y. F. Li (*author for correspondence) are with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong. E-mail: {mesychen, meyfli}@cityu.edu.hk.

research based on entropy measurement.

For model-based applications, attempt was made in determining an object's manufacturing accuracy using range sensors [27] or intensity cameras [10]. Tarbox and Gottschlich [28] presented an Integrated Volumetric Inspection System (IVIS) and proposed three algorithms for inspection planning. Trucco et al [4] present a model-based approach to camera placement. Tarabanis et al [3] developed a model-based sensor planning system, the machine vision planner (MVP), which works with 2-D image obtained from a CCD camera. Compared with other sensor planning systems, the MVP system uses a synthesis rather than a generate-and-test approach. A general automatic sensor planning system (GASP) was reported in [4] for computing optimal positions for inspection tasks using a known imaging sensor and feature-based object models. This exploits a feature inspection representation which outputs off-line an explicit solution for the sensor position problem. GASP computes visibility with an approximate model. The reliability of the inspection depends on the physical sensors used and the processing software. In order to obtain accurate 3-D measurements, Olague and Mohr [29, 30] proposed to use genetic algorithm to determine the optimal sensor placements. Prieto et al. also suggested to improve the accuracy by positioning the sensor's head according to a strategy for optimum 3D data acquisition [27], which guaranteed that the viewpoints satisfied the accuracy requirement in the scanning process.

A critical problem is still not well solved in the sensor placement: the global optimization of sensor planning. When multiple features need to be observed and multiple viewpoints need to be planned, the minimum number of viewpoints needs to be determined. To achieve high efficiency and quality, the optimal spatial distribution of the viewpoints should be determined too. These are also related to the sensor configurations and environmental constraints. Furthermore, to make it flexible in practical applications, we need deal with arbitrary object models without assumptions on the object features.

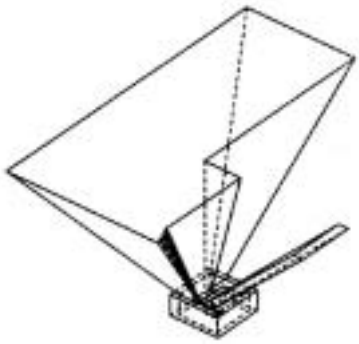

Fig. 1 Typical previous method to determine the admissible domain of viewpoints (after [3])

In model-based vision tasks, researchers have made efforts to find an admissible domain of viewpoints to place the sensor (Fig. 1) to look at one or several object features [1,3]. This method is difficult to be applied in a multi-feature-multi-viewpoint problem as it can not determine the minimum number of viewpoints and their relative distribution.

This paper is dedicated to developing a method for planning model-based vision tasks, e.g. inspection, with both optimal viewpoint distribution and sensing sequence. In such tasks, the procedure of plan generation includes the following:

(a) Input the object's geometric information from a model database;
(b) Give the specifications of the vision tasks and sensor configurations;
(c) Generate a sensor placement graph with the fewest viewpoints;
(d) Search a shortest path for robot; and
(e) Output the sensing plan.

Therefore, the problem of sensor placement for model-based vision tasks is to search an optimal placement graph and a shortest path for achieving the sensing operations. In this paper, the geometric information of the object is loaded from a 3D CAD data file. A strategy is developed to automatically determine a group of viewpoints for a specified vision-sensor with several placement parameters such as position, orientation, and optical settings. Each viewpoint should satisfy multiple constraints due to the physical and optical properties of the sensor, scene occlusion, and robot reachability in the environment etc. The sensing plan is evaluated by a min-max criterion, which is achieved by a hierarchical genetic algorithm (HGA). The shortest path through the viewpoints is determined by Christofides algorithm. Combining the two algorithms results in a complete solution to the model-based sensor placement problem.

II. Sensor Placement for Robot Vision

A. Vision Sensors

This paper assumes that a stereo vision sensor is used for acquisition of the 3D surface information, whilst the method can be extended to other types of sensors. The stereo vision sensor has the following parameters (positional and optical):

(a) three degrees of freedom of the sensor's position: (x, y, z);
(b) three degrees of freedom of the sensor's orientation: the pan, tilt, and swing angles (α, β, γ); and
(c) optical parameters including (d, f, a): the back principal point to image plane distance, d; the entrance pupil diameter, a of the lens; and the focal length f of the lens.

We assume that the two cameras are parallel and the baseline (the distance between the cameras' centers) is fixed. Hence the sensor's state (viewpoint) is given as a vector:

$$\boldsymbol{v} = (x, y, z, \alpha, \beta, \gamma, d, f, a). \quad (1)$$

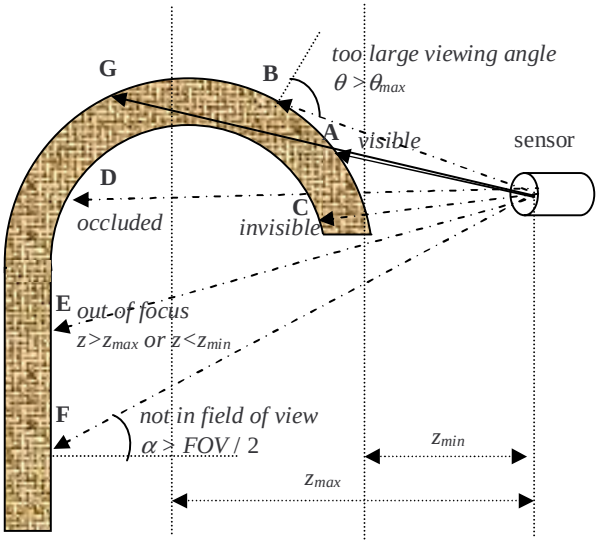

Fig. 2 Some sensor placement constraints

B. Viewpoint and Constraints

All viewpoints must be planned in the nine-dimensional space (viewpoint space):

$$\boldsymbol{V} = \{\boldsymbol{v_i} \mid \boldsymbol{v_i} \in \{(x, y, z, \alpha, \beta, \gamma, d, f, a)\}\}. \quad (2)$$

A point in the above space is defined as a "generalized viewpoint" [31]. Moreover, an admissible viewpoint must satisfy multiple sensor placement constraints (Table 1), including the geometrical (G_1, G_2, G_6), optical (G_3, G_5, G_8), reconstructive (G_4, G_6), and environmental (G_9) constraints. These common constraints are listed in the following table and each constraint condition needs to be satisfied in most vision tasks.

TABLE 1 SENSOR PLACEMENT CONSTRAINTS

Satisfaction	Constraint
G_1	Visibility
G_2	Viewing angle
G_3	Field of view
G_4	Resolution constraint
G_5	In-focus or viewing distance
G_6	Overlap
G_7	Occlusion
G_8	Image contrast (affect (*d, f, a*) settings)
G_9	Kinematic reachability of sensor pose

Denote *A* as a point on the object surface, **n** as its normal, *S* as the vision sensor, **v** as its pose, $\mathbf{v}_a$ as the viewing direction from *S* to *A*. We say that point *A* is visible if the dot product of its normal and the viewing direction is negative. That is

$$\mathbf{G}1: \quad \mathbf{n} \cdot \mathbf{v}_a < 0, \quad (3)$$

This means that the point is visible if the angle (θ) between its normal and the view direction is less than 90°. However, we should set a limit ($\theta_{\max}$) for this angle as the sampling will not be reliable when it is close to 90°. According to the above equation, we have

$$\mathbf{G}2: \quad \theta = \pi - \cos^{-1}\frac{\mathbf{n} \cdot \mathbf{v}_a}{\| \mathbf{n} \| \times \| \mathbf{v}_a \|} < \theta_{\max}. \quad (4)$$

Most CCD cameras have a field-of-view limited by the size of the sensor area and the focal length of the lens [10]. A surface point beyond the sensor's field of view will be projected outside the sensor area and will not be detectable. The locus, which satisfies the field of view constraint for a set of surface features enclosed by a circumscribing sphere, is given by the following equation:

$$\mathbf{G}3: \quad \mathbf{v} \cdot \mathbf{v}_a - \| \mathbf{v} \| \cdot \| \mathbf{v}_a \| \cos(\alpha/2) \geq 0 \quad (5)$$

where α is the field-of-view angle of the sensor.

The resolution constraint ensures that the object is sampled with a minimum accuracy requirement. Pixel resolution can be used to determine the minimum scene feature size resolvable by the vision system. If considering the spatial resolution as the size each pixel represents in the real world, the following can be formulated:

$$\mathbf{G}4: \quad \sigma_{resol} = \left(\frac{z}{Nf} - \frac{1}{N}\right)\frac{1}{\cos\theta} < \sigma_{acceptable}, \text{ (mm/pixel)} \quad (6)$$

where *z* is the distance from the lens to object surface and *N* is the number of total pixels in an image's scanning line.

If a point is imaged to a blur circle of a given size *c*, it is considered sufficiently in focus for a given application. The system is then focused for a range of depths from D_1, the far limit of the depth of field, to D_2, the near limit [10]. Assume that a digital image acquired by the vision sensor has a size of $N \times N$ and the blur radius is restricted in on-pixel length (c=2a/N). If the focus distance, *d*, is adjustable from $d_{\min}$ to $d_{\max}$, the object can be placed between $z_{\min}$ and $z_{\max}$:

$$\mathbf{G}5: \quad z_{\min} < z < z_{\max}, \quad (7)$$

where

$$z_{\min} = \frac{Nfd_{\max}}{Nd_{\max} - Nf + 2f}, \; z_{\max} = \frac{Nfd_{\min}}{Nd_{\min} - Nf - 2f}, \text{ and}$$

$$f < d_{\min} < d_{\max} < 2f.$$

During the reconstruction of a 3-D object surface, the overlap constraint ensures that the re-sampled part of the object is already seen. Because the registration algorithms [23, 32] and many algorithms for integrating range data perform best when the range data overlaps, registration and

integration impose an overlap constraint on the choice of the next best view. The size of overlapping area is dependent on the image-merging algorithm. Here we assume that a minimal width is required:

G6: $w>w_{min}$ (8)

If vision sensor is mounted on a robot end-effector and the hand-eye system is calibrated in advance, w_{min} may be set to zero and the overlap constraint can be removed.

Occlusion is an important scene attribute useful to the sensor planning process [33]. The target A is visible if no other entities are in front of it. Here the 'entity' means any geometrical element $\boldsymbol{e_j}$, such as line, surface, or solid object. Thus we have

$$\mathbf{G}7:\ A=\begin{cases} visible: & if\ ((L_{SA}\cap(\bigcup_{j=1}^{n}e_j))=\phi) \\ occluded: & otherwise \end{cases}, \quad (9)$$

where L_{SA} is the straight line connecting the sensor center and point A, ϕ is an empty set of the entity intersection.

Contrast is a criterion of image quality. It may affect the position and optimal settings of vision sensor, e.g. the diameter of the aperture of a CCD sensor. The kinematic reachability of a sensor pose and robot-environment collision constraint should also be considered in a real robot system. A sensor pose must be in the reachable space.

Fig. 2 illustrates some sensor placement constraints (G_1, G_2, G_3, G_5, G_7). Considering the 6 points (A - F) on the object surface, only point A satisfies all the 5 constraints, while all other points violated one or more of the constraints.

III. Cost Evaluation of a Sensor Placement Plan

A. Previous Approaches

The task of viewpoint planning for model-based vision is to find a set of admissible viewpoints in the viewpoint space, which satisfy all of the sensor placement constraints and can well finish the vision task. In most of the related work, the constraints in sensor placement are expressed as a cost function with the aim to achieve the minimum cost. This cost function should have a value approaching infinity associated to the direction of a pose that the sensor cannot assume and a unitary value associated to the direction of a pose that is possible for the sensor to assume [23].

The term "best-next-view" (BNV) was defined as the next sensor pose which would enable the greatest amount of previously unseen three-dimensional information to be acquired [13, 15]. [3] [34] chose to formulate the probing strategy as a function minimization problem. The optimization function is given as a weighted sum of several component criteria, each of which characterizes the quality of the solution with respect to an associated requirement separately. Thus the optimization function is written as:

$$h=\max(\alpha_1 g_1+\alpha_2 g_2+\alpha_3 g_3+\alpha_4 g_4) \quad (10)$$

subject to $g_i\geq 0$, to satisfy four constraints, i.e. the resolution, focus, field-of-view, and visibility.

In [35], the strategy of viewpoint selection took into account three factors: 1) the new observed area volume $G(\phi_{t+1})$, 2) the cost function F in order to reduce the total camera displacement $C(\phi_t,\phi_{t+1})$, and 3) constraints to avoid unreachable viewpoints and to avoid positions near the robot joint limits $B(\phi)$. The cost function F_{next} to be minimized is defined as a weighted sum of the different measures:

$$F(\phi_{t+1})=A(\phi)+a_1 g(\phi_{t+1})+a_2 C(\phi_t,\phi_{t+1})+a_3 B(\phi). \quad (11)$$

[36] evaluated the suitability of all potential viewpoints of the NBV by using a rating function

$$f(\theta,\phi)=w_\varepsilon f_\varepsilon(\theta,\phi)+w_o f_o(\theta,\phi)+w_s f_s(\theta,\phi) \quad (12)$$

where θ and ϕ are two parameters on the viewpoint sphere; fe, fo, fs are factor functions rating on some physical or heuristic constraints, and w_e, w_o, w_s are the weighting coefficients. The viewpoint with the largest value of $f(\theta,\phi)$ will be chosen as the NBV.

[37] considered the total cost via a function:

$$T[F]=\sum_{i=1}^{k}t_o(f_i) \quad (13)$$

where the cost $t_o(f)$ gives the total time needed to manipulate the hardware to the status specified by f, to take a picture, to update the environment and register the space, and to run the recognition algorithm. The effort allocation $F=\{f_1, ..., f_k\}$ gives the ordered set of operations applied in the search. The probability of detecting the target by the allocation is:

$$P[F]=P(f_1)+...+\{\prod_{i=1}^{k-1}[1-P(f_i)]\}P(f_k) \quad (14)$$

where $P(f)$ is the probability of detecting the target.

Then the next action is selected that maximizes the term

$$E(f)=\frac{P(f)}{\Delta_T(f)},\qquad \Delta_T(f)=t_o(f). \quad (15)$$

Triggs et al [38] gave a method of function optimization technique to minimize their viewpoint evaluation function. They divided the search space into a set of local regions and

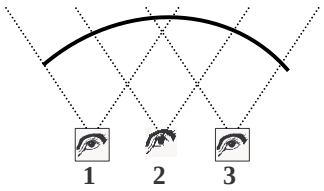

Fig. 3 A redundant viewpoint

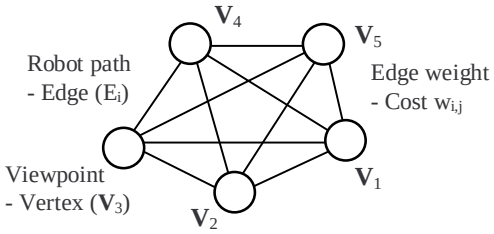

Fig. 4 A sensor placement graph

built a probabilistic function interpolation. These distributions can be used to choose which region to refine and where to subdivide it. The goal was to optimize the function, so that a sample only "succeeds" if it improves on the best currently known function value f_{best}. If the probability density for the function value at some point is $p(f)df$, the expected gain or improvement to f_{best} from a sample placed at that point is

$$< gain >= \int_{-\infty}^{f_{best}} (f_{best} - f)p(f)df \qquad (16)$$

In the previous work on sensor placement, the evaluation of a viewpoint is normally achieved by direct computation. Such an approach is normally formulated for a particular application and is therefore difficult to be applied to general tasks. In this paper, we also minimize the cost, but by evolutionary computing, so that it is applicable to different vision tasks. The object and sensor parameters can be given by the user at the beginning of computation. A shortest path for robot execution is also given based on graph theory.

B. Lowest Travelling Cost

In this paper, we assume a priori model of the object. The procedure for generating a sensor placement plan is summarized as follows:

(a) Generate a number of viewpoints.
(b) Reduce redundant viewpoints.
(c) If the placement constraints are not satisfied, increase the number of viewpoints.
(d) Construct a graph corresponding to the space distribution of the viewpoints.
(e) Find a shortest path to optimize robot operations.

Generating a large number of viewpoints will most likely satisfy all constraints and finish the vision task, but it will also increase the cost. To achieve an optimal solution, we need eliminate all possible redundant viewpoints. Fig. 3 shows that the 2nd viewpoint is redundant because it does not increase information on the object model.

A plan of viewpoints is mapped onto a graph $G = (V(G), E(G), \psi_G, w_E)$ with weight $\boldsymbol{w}$ on every edge $\boldsymbol{E}$, where the vertices $\boldsymbol{V}_i$ represent viewpoints. Edge $\boldsymbol{E}_{ij}$ represents a shortest collision-free path between viewpoint $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$, and weight $\boldsymbol{w}_{ij}$ represents the corresponding distance. Such a graph is termed as ***sensor placement graph G*** in this paper.

Fig. 4 shows an example topology of a viewpoint plan. A practical solution to sensor placement problem must provide a number of viewpoints reachable by the robot and there must exist a collision free path between every two acceptable viewpoints.

For a sensor placement graph $\boldsymbol{G}$ with n viewpoints, its order and size are $o(G) = n$, $\partial(G) = \frac{1}{2}n(n-1)$, respectively. The shortest path for taking all views is a Hamilton cycle which is a sequence of vertices: $\boldsymbol{C}$=(x_1, x_2, …, x_n, x_1) where $xi \neq xj, xi \in V(G), i \in [1, n]$. The path length is

$$l_c = w(x_n, x_1) + \sum_{i=1}^{n-1} w(x_i, x_{i+1}), xi \in V(G). \qquad (17)$$

The time consumed in generating a viewpoint plan includes:

(a) n*t_1 – the time needed to acquire a view and to transfer it to a 3D local model. This includes image digitalization, image preprocessing, 3D surface reconstruction, etc.

(b) n*t_2 – the time for fusion and registration, i.e. merging the local model with the previous partial model.

(c) t_3 – the time needed to perform the viewpoint planning. In doing this, a plan of viewpoints and optical settings of the sensor are determined and a path is generated for the robot to move to the next pose.

(d) t_4 – the time needed for the robot to perform the task of moving from one viewpoint to another.

Here t_3 is subject to the constraints listed in Table 1. If n viewpoints of image acquisition are needed to finish the task, the total needed time is $[(t1+t2)*n+t3+t4]$. Since there exists a priori model of the object, the planning strategy may run offline and t_3 is eliminated from the above equation. Assuming that t_1 and t_2 are constants and t_4 is proportional to the path length, the task time becomes $T_{cost} = (T1+T2)*n + l_c\kappa$.

It is obvious that reducing the number of viewpoints will improve the vision perception performance. Therefore, the objective here is to achieve the lowest traveling cost T_{cost} through the planned viewpoints. In fact, if both the object model and the robot environment are specified, the length of the shortest path for taking the views will not vary much and the traveling cost will be proportional to the number of

viewpoints. Hence the objective becomes minimizing the number of viewpoints. An optimal solution of the sensor placement contains the fewest number of viewpoints and the corresponding graph has a lowest order. This will be determined by HGA in the next section.

c_1	c_2	…	…	c_{nmax}	V_1	V_2	…	…	V_{nmax}

In this paper, our goal is to minimize the sensing cost, but not by a combined function as in a traditional approach. We will achieve it by (1) minimizing the number of viewpoints subject to task completion, (2) optimizing the viewpoint distribution, and (3) finding a shortest travelling path.

IV. Optimal Sensor Placement Graph

A. HGA Representation

In this paper, we use a hierarchical GA to determine the optimal topology in the sensor placements which will contain minimum number of viewpoints with the highest accuracy while satisfying all the constraints. The hierarchical chromosome can be regarded as the DNA that consists of the parametric genes and control genes. In this work, parametric genes (V_i) mean the sensor poses and optical settings and control genes (c_i) mean the topology of viewpoints. To show the activation of the control gene, an integer "1" is assigned to each control gene being enabled whereas "0" indicates a state of turning off. When "1" is signaled, the associated parameter genes associated with that particular active control gene are activated in the lower level structure. However, the inactive genes always exist within the chromosome even when "0" appears.

For the sensor placement problem, a chromosome in GA represents a group of viewpoints with a specific topology. Fig. 5 illustrates the structure of the hierarchical chromosome corresponding to the plan of viewpoints. Here $V_i = (x, y, z, \alpha, \beta, \gamma, a, f, d)$ represents a variable viewpoint, where $x, y, z \in R$, $\alpha, \beta, \gamma \in [-\pi, \pi]$, $a \in [a_{\min}, a_{\max}]$, $f \in [f_{\min}, f_{\max}]$, and $d \in [d_{\min}, d_{\max}]$, and the corresponding $c_i = \{0,1\}$ represents a control gene which is a binary variable.

B. Min-Max Objective and Fitness Evaluation

In this paper, a plan of sensor placements is evaluated by a min-max criterion, which includes three objectives and a fitness evaluation formula.

The order of a graph $\boldsymbol{G}$ is equivalent to the number of occurrences of "1" in the control level genes. To plan a group of viewpoints with minimum order in the sensor placement graph, the first objective is given as:

$$\textbf{Objective 1}: \text{minimize } o(G) = \sum_{i=1}^{n\max} c_i . \tag{18}$$

Assume that the accuracy of vision inspection is proportional to the surface resolution of the vision sensor and consider m features to be acquired. The second objective is to improve the average accuracy via

$$\textbf{Objective 2}: \text{maximize } \eta(F) = \frac{1}{m}\sum_{j=1}^{m} \frac{w_{j,image}}{l_j} \tag{19}$$

where w_{image} is the size of a feature on the sensor image and l_j is its actual length.

On the other hand, an admissible viewpoint is subject to 9 constraints in the sensor placement space, i.e. resolution, in-focus, field of view, visibility, viewing angle, overlap, occlusion, contrast, and reachability. We set up a penalty scheme to handle these constraints such that invalid chromosomes become low performers in the population. The constrained problem is then transformed to an unconstrained condition by associating the penalty with all the constraint violations. We use a vector of penalty coefficients to combine the nine constraints:

$$\boldsymbol{K} = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \delta_7, \delta_8, \delta_9) . \tag{20}$$

where δ_i is constant weight representing the importance of that constraint. If the constraint does not need to be satisfied (e.g. G_6-overlap), the weight will be set to zero.

Define a binary function

$$\varphi_i = \begin{cases} 0, & \text{if the constraint is satisfied} \\ 1, & \text{if the constraint is violated} \end{cases}$$

and construct another vector of constraints:

$$\boldsymbol{Q}(\boldsymbol{l}, \boldsymbol{V}) = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9) , \tag{21}$$

where $\boldsymbol{l}$ is an object feature and $\boldsymbol{V}$ is a viewpoint.

Therefore the third objective is to minimize the total penalties for the constraints:

$$\textbf{Objective 3}: \text{minimize } penalty = K \cdot Q^T . \tag{22}$$

If there are m features and n viewpoints, the average penalty for a viewpoint topology is:

$$penalty = \frac{1}{m}\sum_{i=1}^{m} \min[K \cdot Q(l_i, V_j)^T \mid j = 1,2,...,n] . \tag{23}$$

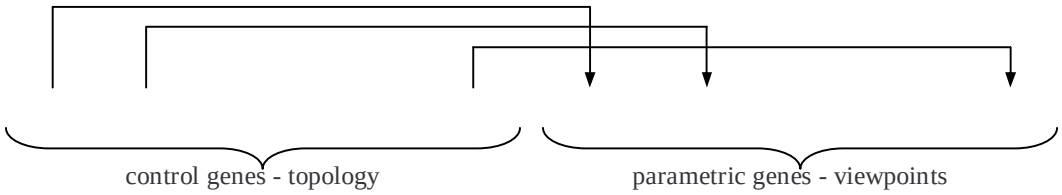

Fig. 5 Hierarchical chromosome structure for sensor placement

Finally, we combine the penalty scheme with the two objective functions to derive the fitness function:

$$\textbf{Fitness}: f(\mathbf{G}) = (an_{\max} + b\ell_{\max} + |K|) - ao(G) - \frac{b}{\eta(F)} - K \cdot Q^T, \quad (24)$$

where $|K| = \sum_{i=1}^{m} \delta_i$, $(an_{\max} + b\ell_{\max} + |K|)$ is the maximum possible value that produces positive fitness, $\ell_{\max}$ is the maximum possible resolution, and *a* and *b* are two adjustable scaling factors.

C. Evolutionary Computing

According to the characteristics of the sensor placement problem, the following genetic parameters and operations are adopted:

(a) Chromosome length:
$2n$, where n is the maximum viewpoints and determined according to the object size and sensor configurations,

(b) Crossover method:
control level genes: one-point crossover if n<10, two-point crossover if n>=10; probability of crossover $p_c = 0.25$;
parametric level genes: Heuristic crossover with ratio=0.8. Here the parameters of sensor pose and optical settings are real numbers.

(c) Mutation method:
control level genes: bit-flip mutation; probability of mutation $p_m = 0.01$.
parametric level genes: $g = g + \phi(\mu, \sigma)$ where ϕ is Gaussian distribution function, μ and σ are the mean and variance, respectively.

(d) Selection method: Roulette-Wheel selection method;

(e) Replacement: Steady State without duplicates;

(f) Population size: 30-100, based on the length of chromosome;

(g) Initial population: randomly generated on a sphere around the object.

V. Determination of a Shortest Path

A. The Viewpoint Distance

For a sensor placement graph, there may exist more than one path with the shortest (or approximately shortest) length. To determine a shortest path through all these viewpoints, we firstly need compute the distance between each pair of viewpoints (V_i and V_j), called pose distance $w(V_i, V_j)$. With different types of robots and different control modes, the distance should be computed in different ways correspondingly:

1) Tool Level Control

To achieve robot-independent representation of the location of the robot tool or hand, the control program often defines the locations in terms of a Cartesian reference frame fixed to the base of the robot or workspace. If the robot moves at a constant speed, the execution time is proportional to the 3-D position difference (Euclidean distance) or 3-axis orientation difference, i.e.

$$w(V_i, V_j) = \max(\mathrm{Inp}({}^{xyz}V_i - {}^{xyz}V_j)/\mu, \mathrm{Inp}({}^{\alpha\beta\gamma}V_i - {}^{\alpha\beta\gamma}V_j)/\nu), \quad (25)$$

where μ and ν represent the translational speed and rotational speed respectively, ${}^{xyz}V$ and ${}^{\alpha\beta\gamma}V$ are the three position and orientation components of *V*, respectively. Inp(•) represents the function of vector inner product.

2) Asynchronous Joint Control

If the robot is controlled in joint space to change its pose, the distance is computed by

$$w(V_i, V_j) = \sum_{t}^{n_{dof}} \mu_t ({}^tV_i - {}^tV_j) \quad (26)$$

where n_{dof} is the robot's DOF number, μ_t is the execution speed of joint *t*, and ${}^tV$ is the joint location at pose *V*.

3) Synchronous Joint Control

When the robot is controlled in joint space with all joints moved simultaneously, the distance is determined by the maximum one

$$w(V_i, V_j) = \max\{\mu_t ({}^tV_i - {}^tV_j) \mid t \in [1, n_{dof}]\}. \quad (27)$$

If the sensor's optical settings (e.g. zoom, focus, Iris, etc.) are under motorized control, it will also take time to change the sensor configuration from one viewpoint to another. Then the viewpoint distance should be the larger of this distance (time equivalent) and the above robot pose distance.

With n viewpoints obtained by generic algorithm, a symmetrical distance matrix can be generated by computing each pair of the viewpoints: $\mathbf{W} = \{w_{ij}\}$, which will be used to determine the shortest path.

B. Determination of a shortest path

We may assume that the robot should resume its initial state after completing the vision task (since it needs be ready for inspection of the next workpiece). Given a specified graph, now another fundamental task is to find an optimal closed chain that is the shortest (or approximately shortest) one of all the possible chains.

Obviously a sensor placement graph satisfies the triangle inequality, i.e.

$$w(V_i, V_j) \le w(V_i, V_k) + w(V_k, V_j), \forall V_k \in V(G) \setminus \{V_i, V_j\}, \quad (29)$$

where the "=" holds if the position of $\boldsymbol{V}_k$ is on the path l_{ij} and the orientation of $\boldsymbol{V}_k$ is the middle angle between Ω_i and Ω_j.

Because a sensor placement graph $\boldsymbol{G}$ is finite, connected, and complete, the optimal closed chain is the optimal Hamilton cycle. Furthermore a complete graph must contain Hamilton cycles, i.e. there exist cycles which contain all the vertices once. In graph theory, it has been proved that if $\boldsymbol{G}$ is complete and satisfies the triangle inequality, the optimal chain $\boldsymbol{C"}$ in a connected and weighted graph $\boldsymbol{G"}$ corresponds to an optimal cycle $\boldsymbol{C}$ in its complete and weighted graph $\boldsymbol{G}$. That is, $C" \leftrightarrow C \quad and \quad w(C") = w(C)$, where $w(X)$ is the length of chain or cycle X.

To plan a sequence of robot operations or to find an optimal Hamilton cycle, we have to decompose $\boldsymbol{G}_n$ into the union of some edge-disjoint Hamilton cycles. There are total n vertices and $\partial(G) = \frac{1}{2}n(n-1)$ edges in the graph $\boldsymbol{G}_n$. A Hamilton cycle $\boldsymbol{C}$ must contain n edges too. Let a Hamilton cycle be a sequence of vertices: $\boldsymbol{C}$=(x_1, x_2, …, x_n) where $xi \ne xj, xi \in V(G), i \in [1, n]$. The problem might be solved by enumerating all possible Hamilton cycles $\boldsymbol{C}_i$ in the graph, by comparing their summed weighs $w(\boldsymbol{C}_i)$, and then finding out the smallest one *cost*=min[$w(\boldsymbol{C}_i)$]. However, there are totally $o(C) = \frac{1}{2}(n-1)!$ Hamilton cycles. When n is large, this will give unacceptable computations, e.g. $o(C) = 6 \times 10^{16}$ when n=20. This is a non-deterministic polynomial complete (NPC) problem in graph theory and must be solved by an approximation algorithm.

Here we use an approximation algorithm developed by Christofides. The procedures of this algorithm for finding an optimal Hamilton cycle is described as:

Step 1. Construct the distance matrix **W** from graph (***G***, *w*).
Step 2. Find the smallest tree **T** in **W** using *Prim algorithm*
Step 3. Find the odd degree set **V** in **T** and calculate the *perfect matching* **M** of the smallest weighs in ***G***[***V***] using *Edmonds-Johnson algorithm*.
Step 4. Find an Euler circuit $\boldsymbol{C}_0$=(*x1, x2, x3, …, xn*) in ***G****=**T**+**M** using *Fleury algorithm*.
Step 5. Start from *x1* and go along $\boldsymbol{C}_0$, remove each multi-occurrence vertex from $\boldsymbol{C}_0$ except for the last *x1* and finally form a Hamilton cycle ***C*** of graph ***G***. This gives the approximated optimal cycle.

The resulting Hamilton cycle is an approximate solution. It has been proven that the error ratio does not exceed 0.5 even in the worst case. If L_0 is the optimal solution (sum of weighs) and L is the approximate solution by Christofides algorithm, we have $1 \le L / L_0 \le 1.5$. In this algorithm, the total computation cost is $O(n^3)$. In contrast, using direct search method takes $O(n!)$.

VI. Implementation Considerations

A. Geometry Scripts

The object model is usually extracted from CAD file. However, directly using these data may result in prohibitive computation for planning the sensor placements. For example, Prieto et al. [27] chose to import the CAD model with IGES format, which contains the NURBS representation of object surfaces. This data format must be converted to 3-D voxels so that they could search for a viewpoints set. However, even with a very simple object, a large number of 3-D voxels will result, making the computation too costly.

In this paper, we define a format of "Geometry Scripts" (GS) in which the whole object is constructed with some geometric primitives, such as surfaces, solid boxes, cylinders, spheres, etc. These primitives are used to build higher-level geometries by CSG (Constructive Solid Geometry) operations, such as AND ("*"), OR ("+"), and SUB ("-"). Using geometry scripts has two advantages: (1) it is intuitive and easy to understand. Users may directly write (instead of import from CAD file) the scripts to describe what needs to be inspected. (2) The more important advantage is that it is very convenient for computing the sensor placements. A point can be checked if it satisfies the placement constraints within a short period of time.

B. Inspection Features

In this research, any geometry elements/entities, such as cylinders, freeform surfaces, curves, and individual points, can be specified as the features which need be inspected in the vision task. However, for computation simplicity, all these features will firstly be converted into individual points. In our system, this is accomplished by a re-sampling method and usually the normal of each point can also be determined automatically. Fig. 6 illustrates an object sampled as about one thousand points. The sampling rate was determined automatically according the sensor configuration and surface

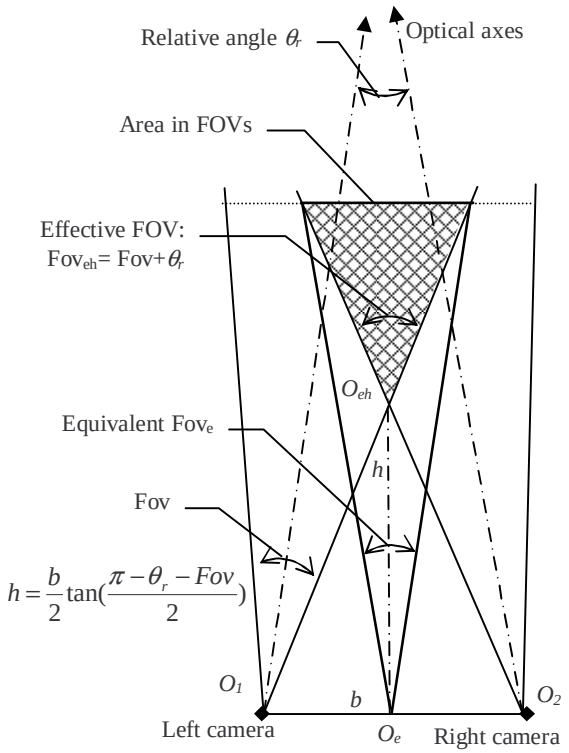

Fig. 7 Equivalent sensor's field of view

size.

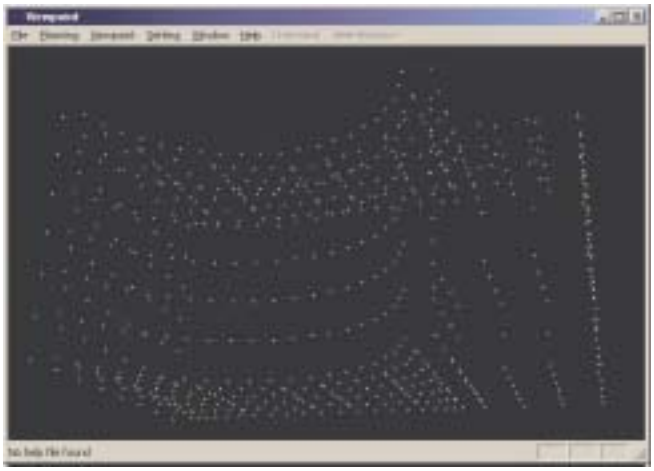

Fig. 6 Sampling of inspection features

C. Sensor Structure

For a stereo sensor composed of two cameras, the sensor placement constraints should be met by each individual camera. To reduce the computation complexity, we may rebuild an equivalent 3-D sensor to the stereo sensor.

Fig. 7 illustrates a stereo with a small angle between the two cameras. It is equivalent to a camera placed at O_{eh} and with effective field of angle $Fov_{eh}= Fov+\theta_r$. More conveniently, it is equivalent to be placed at O_e with FOV:

$$Fov_e = 2\tan^{-1}[(1-\frac{h}{d})\tan(Fov+\theta_r)]$$
$$= 2\tan^{-1}[\tan(Fov+\theta_r)-\frac{b}{2d}-\frac{b}{2d\cos(Fov+\theta_r)}] \quad (30)$$

where θ_r is the relative angle between the two cameras, b is the baseline length, and d is the viewing distance.

D. Constraints

In principle every constraint has to be satisfied for a viewpoint in a plan. If the viewpoint violates any of the constraints, it should be rejected (during the evolution it may also be kept according to the overall fitness). The calculation order of the constraints is important for improving the computation efficiency. This paper uses the following computation order: visibility, field of view, viewing angle, in-focus, occlusion, and others. When one of them is violated, all other constraints will not be checked. Of all sensor placement constraints, the visibility has the lowest computation, just involving the calculation of $\mathbf{N}_i \bullet \mathbf{V}_i < 0$. It is thus checked as the first constraint.

The occlusion constraint is the most complex one and takes most of the computation time. To test if a point is occluded by other geometry elements, we need to check if the line segment between the point and the sensor intersects with these elements. For some regular geometry elements, such as ball, circle, square, box, etc., this can be tested in a simpler way. Take the example of squares, this can be achieved via the following steps:

1) Assume:
   object point $\mathbf{Q} = (x_q, y_q, z_q)$, with normal $\mathbf{N} = (l_q, m_q, n_q)$;
   sensor position $\mathbf{S} = (x_s, y_s, z_s)$, orientation $\mathbf{T} = (l_s, m_s, n_s)$;
   square vertex $\mathbf{P}_i$, i=1, 2, 3, 4.
2) Translate the coordinate system to origin **Q**:
   $\mathbf{Q}' = 0$
   $\mathbf{S}' = \mathbf{S} - \mathbf{Q}$
   $\mathbf{P}_i' = \mathbf{P}_i - \mathbf{Q}$ (i=1,,4)
3) Transform the z-axis to point it along **QS**:
   $\mathbf{V}_{QS} = \mathbf{S}' / \| \mathbf{S}' \| = (a, b, c)$.
   $\mathbf{P}_i'' = \mathbf{R}\mathbf{P}_i'$,
where

$$\mathbf{R} = \begin{bmatrix} ac/\sqrt{1-c^2} & bc/\sqrt{1-c^2} & -\sqrt{1-c^2} \\ -b/\sqrt{1-c^2} & a/\sqrt{1-c^2} & 0 \\ a & b & c \end{bmatrix}. \quad (31)$$

4) Check the occlusion:
   $b_{occ} = (\rho_{12}\ \rho_{34} > 0)$ AND $(\rho_{23}\ \rho_{41} > 0)$, (32)
where $\rho_{ij} = \mathbf{P}_i''(x)\ \mathbf{P}_j''(y) - \mathbf{P}_i''(y)\ \mathbf{P}_j''(x)$.

For freeform geometries, the occlusion has to be determined by "object projection" with "depth test" or "bounding volume test", which will be computationally expensive.

E. Viewpoint initialization

A good initial population will improve the efficiency of the evolutionary computing. In this paper, the initial guess of the maximum number of viewpoints and their space distribution is made via the following two steps.

1) Estimation of viewpoint number

We first compute the object's geometric center and find a sphere to surround it. The maximum number of viewpoints is estimated by:

$$N = \frac{2S_{object}}{S_{view}} = \frac{2\pi^2 R^4}{d^2 \tan^2(\frac{Fov}{2})} \tag{33}$$

where S_{obj} is the object surface area, S_{view} is a single view size, R is the sphere radius, d is the average viewing distance.

2) Uniform distribution on a sphere

Since a uniform distribution of an arbitrary number of viewpoints on a sphere cannot be described by a general mathematical formula, it needs to be handled by a special method. Here we adopt an artificial physics method to solve this problem. Take the viewpoints as particles with the same electric charge and randomly sprinkle them on the sphere. Each particle will repel every other particle. The system will lead to a stable (minimum energy) configuration where each particle is equidistant from all the others and each particle is maximally separated from its closest neighbors by the electric repulsive forces.

The sensor orientation is set to look inward to the sphere center, which is described by two parameters, θ and φ based on the sphere coordinate system. They can be expressed as a unit direction vector:

$$\vec{P}_d = (x\vec{\mathbf{i}}, y\vec{\mathbf{j}}, z\vec{\mathbf{k}}), \text{ where } \begin{cases} x = \cos\varphi\cos\theta \\ y = \cos\varphi\sin\theta \\ z = \sin\varphi \end{cases}. \tag{34}$$

Fig. 8 illustrates an example of the initial distribution of viewpoints on a sphere. The maximum number is determined according the object and sensor configuration. The distribution is uniform (with similar minimum distance) on the sphere, but the positions are still random.

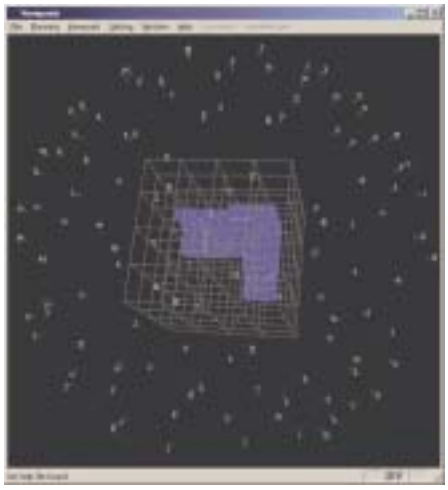

Fig. 8 Initial viewpoint distribution

VII. Experiments

A. The Viewpoint Planner

We developed a 3D animation system, called *Viewpoint Planner*, which includes the following functions:

* 3-D geometry input: using specially defined scripts with CSG logic operations;
* Selective display of object, lighting effect, 3D grids, sampled inspection features, viewpoint distribution, etc.;
* Illustration of the shortest path through the viewpoints;
* Simulation of an arbitrary viewpoint look through;
* Acquisition of the 3D map of the current view (simulation of scene depth map);
* Configuration of sensor specifications;
* Configurable scene apparent effects: with lighting, texture, or color rendering;
* Amendment of the parameters of a specific viewpoint
* Initial estimation of the maximum number of viewpoints for an object (according to the object size and sensor configuration);
* Uniform generation of the initial viewpoints on a sphere around the object;
* Evolutionary Computation of optimal sensor placement graph;
* Determination of a shortest path through the viewpoints;
* Viewpoint sequence export to a file.

B. Examples of Planning Results

This section presents three examples of sensor placements which were carried out by the *Viewpoint Planner*. In operations, the user only needs to give the object model scripts and sensor configurations. The system will automatically generate the initial guess, perform the evolutionary computation of optimal sensor placements, and give a shortest path.

The first two examples have the same sensor configurations,

but different object models and inspection tasks. The second and third examples have the same object model, but with different sensor settings and inspection tasks.

The planning results are given below and illustrated in Fig. 9-11. The naive path length is an arbitrary path length not optimized for comparison with the shortest path found by the proposed method. We obtained these results on a PC with 750MHz CPU, 128MB RAM, using Windows 2000

1) Example one

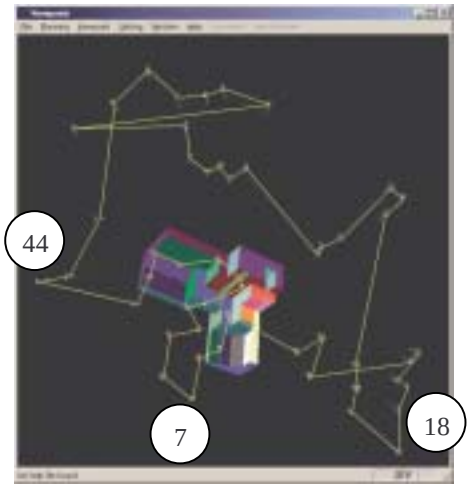

Fig. 9 View planning for object 1: the viewpoint distribution and a shortest path

Input:

Object size: 325×254×183 (mm)

Inspection task: full observation of all surfaces except for the bottom;

Sensor configuration: 25mm lens, 26.5602mm focal length, 2/3" sensor, F2.8, 11.66° Fov, 393.5-460.2mm field depth;

Planning result:

Initialization: 118 viewpoints,

Finial optimized sensor placement plan: 50 viewpoints,

Naive path length: 425.604993,

The shortest path: 129.499801,

2) Example Two

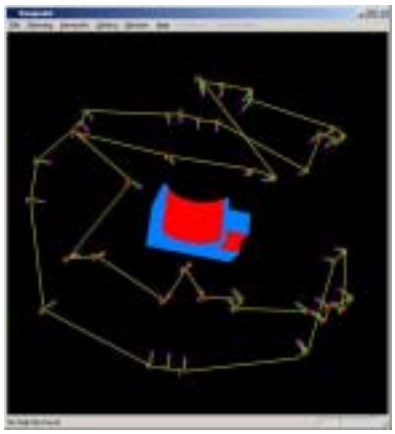

Fig. 10 View planning for object 2: the viewpoint distribution and a shortest path

Input:

Object size: 300×150×180 (mm)

Inspection task: all surfaces except for the bottom;

Sensor configuration: 25mm lens, 26.5602mm focal length, 2/3" sensor, F2.8, 11.66° Fov, 393.5-460.2mm field depth;

Planning result:

Initialization: 80 viewpoints,

Finial optimized sensor placement plan: 44 viewpoints,

Naive path length: 165.889252,

The shortest path: 72.510986,

3) Example Three

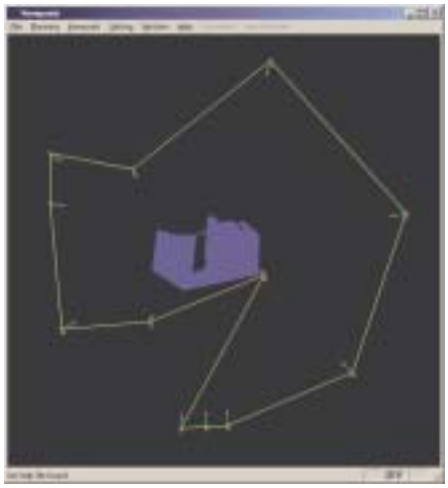

Fig. 11 View planning for object 3 (with lighting effect) : the viewpoint distribution and a shortest path

Input:

Object size: 300×150×180 (mm)

Inspection task: all surfaces, except for the bottom, the left side, and the back side;

Sensor configuration: 16mm lens, 16.5mm focal length, 2/3" sensor, F2.8, 18.63° Fov, 488.2-570.9mm field depth;

Planning result:

Initialization: 20 viewpoints,

Finial optimized sensor placement plan: 12 viewpoints,

Naive path length: 66.705595,

The shortest path: 43.698211,

The computation time for finding the optimal view plan in Example 1, 2, and 3 was 15 hours, 9 hours, and 45 minutes respectively, whereas the time taken for finding the shortest path was 9 minutes, 5 minutes, and 1 second respectively. The above computation time (averaged by multiple runs) is for reference only since the software was running in a debug mode. The actual speed should be much higher. However, as our plan was generated off-line, this computation cost is not important here.

4) Viewpoint Observation

With the *Viewpoint Planner*, after evolutionary computing we may check what can be seen from an individual viewpoint. Fig. 12 shows the simulated scene observed from viewpoint No. 44 (in Fig. 9), with the specified sensor configurations (especially the field of angle). The views from point No. 7 and No. 18 are given in Fig. 14 and Fig. 16, respectively. Every viewpoint can be observed and the corresponding 3-D depth map can also be generated for comparison with real situations.

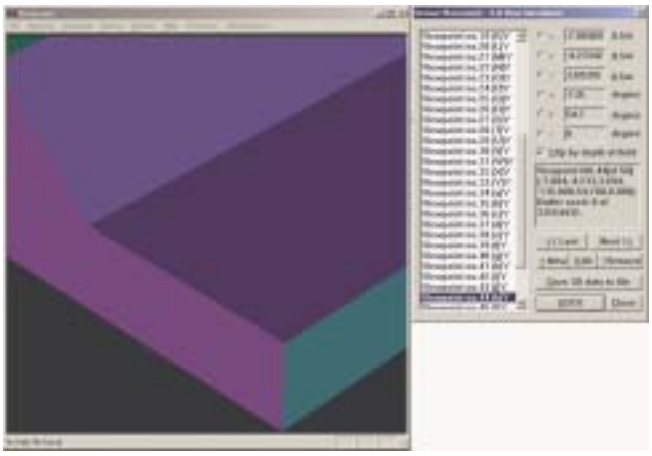

Fig. 12 View from Viewpoint No. 44 (in example one)

C. Examples of Implementation with a Real Robot

We also conducted experiments on a real robot to verify the planning results. Fig. 13 illustrates our experiment setup. It includes a 6DOF robot (STAUBLI RX-90B) with ±0.02mm repeatability, a high-speed vision system, the object (with its model illustrated in Fig. 9), a light source, and a stereo head. The stereo was composed of two identical cameras (Pulnix TM-765, 756×581 resolution, 2/3" CCD), with the baseline of 160mm and a small relative angle 9 degrees. We used two sets of lens, 25mm (for examples one and two) and 16mm (for example three). When using the 25mm lens, the equivalent field of angle is about 11.5 degrees. When using the 16mm lens, the equivalent field of angle is about 18 degrees.

Assume that the sensor placement graph has been generated at the offline stage by the *Viewpoint Planner*. In this stage, the robot is controlled to move to the specified viewpoints and the scene images are captured by the vision sensor.

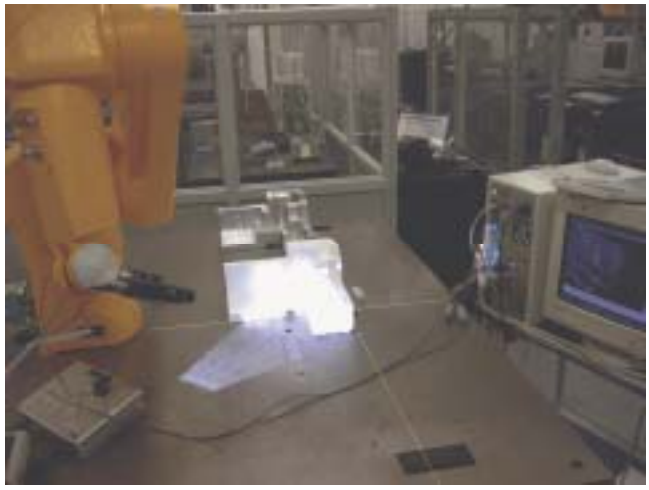

Fig. 13 Testing with a real robot

Here we give the results of two typical viewpoints (No. 7 and No. 18 in Fig. 9) in example one. Fig. 14 illustrates what can be seen at viewpoint No. 7 with the Viewpoint Planner. Fig. 15 illustrates two images captured by the left and right cameras. Here masked illumination was used to facilitate finding the correspondences. The overlapped area will be used for 3-D reconstruction. Fig. 16 and Fig. 17 shows another example at viewpoint No. 18.

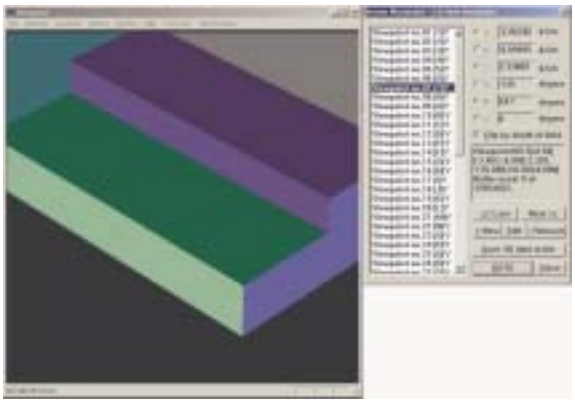

Fig. 14 The scene at viewpoint No. 7

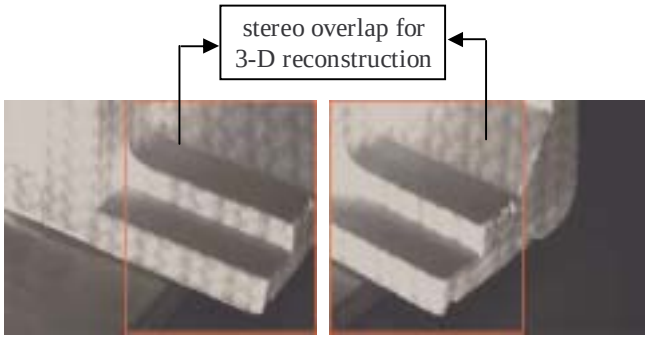

Fig. 15 The left and right images captured by the stereo vision

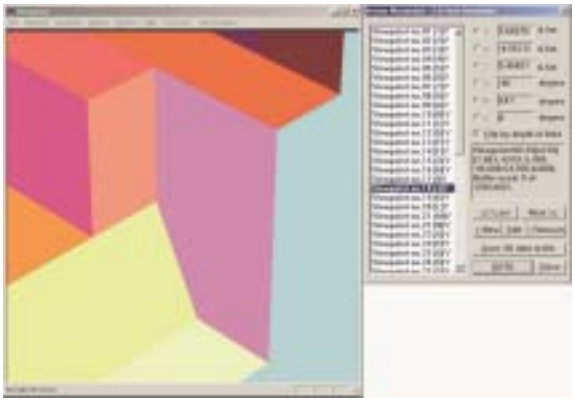

Fig. 16 Expected view at viewpoint No. 18

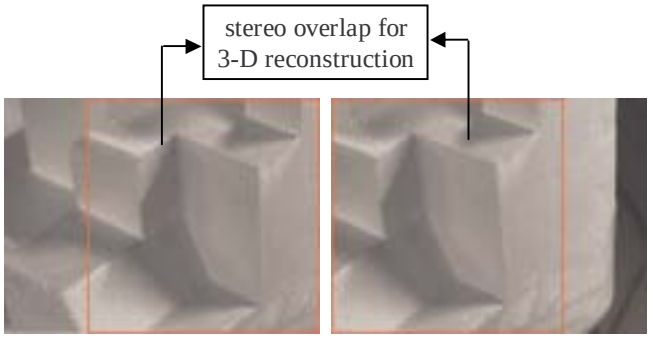

Fig. 17 Scene captured by the stereo vision

The experiments show that the results in real implementation match the simulation results well. Hence the sensor placement plan generated by the *Viewpoint Planner* and performed by the robot demonstrated satisfactory performance in real operations. However, the planning results may be not the best plan. Here the results are considered acceptable or satisfactory if the viewpoints in the plan do not violate any sensor placement constraints and the redundant viewpoint number is sufficiently small. To achieve a more optimal plan with even a slight improvement in the result, much longer time would be needed in the evolutionary computation. This may not pay off in practical implementations.

VII. Conclusions

This paper proposed a method for automatic sensor placement including the optimal sensor placements and the shortest path through these viewpoints. The plan in the sensor placements is evaluated with three conditions: low order, high precision, and satisfying all constraints.

Using the conventional methods, it is difficult to achieve an optimal sensor placement graph because of the complex, large scale, highly nonlinear characteristics of the problem. As a numerical optimizer, HGA generates the solutions that are not mathematically oriented, but possesses an intrinsic flexibility and the freedom for choosing desirable optima according to task specifications. The search for the shortest path through a number of viewpoints is an NP-Complete problem and an approximation algorithm has to be used for determining the viewing sequence. The Christofides algorithm is an effective one that can guarantee minimum error even in the worst case.

Compared with the previous approaches, our method can deal with complex objects with multiple inspection features and viewpoints, generate minimum number of viewpoints, and lead to global optimization. It provides a stable and complete solution for model-based vision tasks, including viewpoint decision, constraint satisfaction, optimization of viewpoint distribution, planning of robot operation sequence. All these techniques are integrated into the software, the *Viewpoint Planner*, to make it useful in practical applications. The experimental results show that the real situations match the planned results well.

References

[1] C. K. Cowan and P. D. Kovesi, "Automatic sensor placement from vision task requirements", IEEE Trans. on pattern analysis and machine intelligence, vol.10, no.3, May 1988, pp. 407-416.

[2] K. A. Tarabanis, P. K. Allen, R. Y. Tsai; "A survey of sensor planning in computer vision", IEEE Trans. on Robotics and automation RA-11:1, 1995, pp. 86-104.

[3] K. A. Tarabanis, R. Y. Tsai, P. K. Allen, "The MVP sensor planning system for robotic vision tasks", IEEE Trans. on robotics and automation RA-11:1, 1995, pp. 72-85.

[4] E. Trucco, M. Umasuthan, A. M. Wallace, V.Roberto, "Model-based planning of optimal sensor placements for inspection", IEEE Trans. on Robotics and automation RA-13:2, 1997, pp. 182-194.

[5] T.S. Newman and A.K. Jain, "A survey of automated visual inspection", Computer Vision and Image Understanding, 61(2), 1995, pp. 231-262.

[6] S. Y. Chen, Y. F. Li, and R. S. Lu, "Sensor Placement for Active Modeling of Unknown Objects", Int. Conf. on Computer Vision, Pattern Recognition and Image Processing, North Carolina, USA, Mar 2002, pp. 749-752.

[7] R. Pito, "A solution to the next best view problem for automated surface acquisition", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol 21 no.10, Oct. 1999, pp. 1016 -1030.

[8] C. J. Banta, L. M. Wong, C. Dumont and M. A. Abidi, "A Next-Best-View System for Autonomous 3-D Object Reconstruction", IEEE Trans. on Systems, Man and Cybernetics, vol. 3, no. 5, 2000, pp. 589-598.

[9] S.A. Hutchinson and A.C. Kak, "Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities," IEEE Trans. on Robotics and Automation, vol. 5, no. 6, 1989, pp. 765-783.

[10] S. Abrams, P. K. Allen, K. Tarabanis, "Computing camera viewpoints in an active robot work cell Abrams", Int. J. of Robotics Research, vol.18, no.3, Mar 1999, pp. 267-288.

[11] I. Stamos, P. K. Allen, "Interactive sensor planning", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Jun. 1998, pp. 489-494.

[12] N. Ahuja, J. Veenstra, "Generating octrees from object silhouettes in orthographic views", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11 no. 2, Feb. 1989, pp. 137 -149.

[13] J. E. Banta, M. A. Abidi, "Autonomous Placement of a Range Sensor for Acquisition of Optimal 3-D Models," Proc. of the 22nd Int'l Conf. on Ind. Elec., Contr., and Inst. Taipei, R.O.C., 1996, pp. 1583-1588.

[14] D. Papadopoulos-Orfanos, F. Schmitt, "Automatic 3-D digitization using a laser rangefinder with a small field of view", Conf. on Recent Advances in 3-D Digital Imaging and Modeling, 1997, pp. 60 -67.

[15] Y. F. Li and Z. Liu, "Uncertainty-Driven Viewpoint Planning for 3D Object Measurements", *Proc. **IEEE** International Conference on Robotics and Automation*, Taipei, Taiwan, May 2003.

[16] K. N. Kutulakos, C. R. Dyer, "Global surface reconstruction by purposive control of observer motion", IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 1994, pp. 331 -338.

[17] T. Arbel, F. P. Ferrie, "Viewpoint selection by navigation through entropy maps", Proc. of the Seventh IEEE Int. Conf. on Computer Vision. vol 1, 1999, pp. 248 -254.

[18] S. Uppala, D.R. Karuppiah, M. Brewer, S.C. Ravela, R.A. Grupen, "On viewpoint control", Proc. IEEE Int. Conf. on Robotics and Automation, vol. 4, May 11-15, 2002, pp. 4334-4339.

[19] J.R. Spletzer and C.J. Taylor, "Sensor planning and control in a dynamic environment", Proc. IEEE Int. Conf. on Robotics and Automation, vol. 1, May 11-15, 2002, pp. 676-681.

[20] M. Albalate, M. Devy, and J. Marti, "Perception Planning for an Exploration Task of a 3D Environment", Proc. Int. Conf. on Pattern Recognition ICPR2002, Quebec, Canada, Aug. 2002.

[21] P.P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich, "Viewpoint Selection Using Viewpoint Entropy", Vision, Modeling, and Visualization, Stuttgart, Germany, 2001.

[22] Vivek Sujan, "Optimum Camera Placement by Robot Teams in Unstructured Field Environments", Proc. IEEE 2002 Int. Conf. on Image Processing, New York, USA, Sept. 22-25, 2002.

[23] J. Okamoto, M. Milanova, U. Bueker, "Active perception system for recognition of 3D objects in image sequences", 5th Int. Workshop on Advanced Motion Control, Coimbra, 1998, pp. 700-705.

[24] H. Bulthoff, C. Wallraven, and A. Graf, "View-Based Dynamic Object Recognition Based on Human Perception", Proc. Int. Conf. on Pattern Recognition, Quebec, Canada, Aug. 2002.

[25] D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker, "View-Based 3-D Object Recognition Using Shock Graphs", Proc. Int. Conf. on Pattern Recognition ICPR2002, Quebec, Canada, Aug. 2002.

[26] A.J. Briggs and B. R. Donald, "Visibility-Based Planning of Sensor Control Strategies", Algorithmica, vol.26, no. 3-4, 2000, pp. 364-388.

[27] F. Prieto, T. Redarce, et al, "CAD-based range sensor placement for optimum 3D data acquisition", Proc. Second Int. Conf. on 3-D Digital Imaging and Modeling, 1999, pp. 128 - 137.

[28] G.H. Tarbox and S.N. Gottschlich, "Planning for complete sensor coverage in inspection", Computer Vision and Image Understanding, 61(1), 1995, pp. 84-111.

[29] G. Olague, R. Mohr, "Optimal camera placement to obtain accurate 3D point positions", Proc. Fourteenth Int. Conf. on Pattern Recognition, vol.1, 1998, pp. 8 -10.

[30] G. Olague, R. Mohr, "Optimal Camera Placement for Accurate Reconstruction", Pattern Recognition, vol. 35, no. 4, 2002, pp. 927-944.

[31] C. Giraud, B. Jouvencel, "Sensor selection: a geometrical approach", Int. Conf. on Intelligent Robots and Systems, Pennsylvania, Aug., 1995, pp. 555-560.

[32] K. Higuchi, M. Hebert, K. Ikeuchi, "Building 3-D models from unregistered range images", Graphical models and image processing, vol.57, no.4, July, 1995, pp. 315-333.

[33] M. K. Reed, P. K. Allen, I. Stamos, "3-D modeling from range imagery: an incremental method with a planning component", Int. Conf. on Recent Advances in 3D Digital Imaging and Modeling, 1997, pp. 76-83.

[34] X. G. Gu, M. M. Marefat, F. W. Ciarallo, "A robust approach for sensor placement in automated vision dimensional inspection", Proc. IEEE Int. Conf. on Robotics and Automation, 1999, pp. 2602-2606.

[35] E. Marchand, F. Chaumette, "Active vision for complete scene reconstruction and exploration", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 21 no.1, Jan. 1999, pp. 65-72.

[36] H. Zha, K. Morooka, T. Masegawa, "Next best viewpoint (NBV) planning for active object modeling based on a learning-by-showing approach", Lecture Notes in Computer Science 1352, 1998, pp. II-185.

[37] Y. Ye, J. K. Tsotsos, "Where to look next in 3D object search", IEEE Int. Symp. for CV, Florida, 1995, pp. 539-544.

[38] B. Triggs, C. Laugier, "Automatic camera placement for robot vision tasks", IEEE Int. Conf. on Robotics and Automation, 1995, pp. 1732-1737.

[39] S. Y. Chen and Y. F. Li, "A Method of Automatic Sensor Placement for Robot Vision in Inspection Tasks", Proc. IEEE Int. Conf. on Robotics and Automation, vol. 3, Washington D.C., May 2002, pp. 2545-2550.