Visibility Scripts for Active Feature-Based Inspection

E. Trucco, E. Thirion, M. Umasuthan and A.M. Wallace
Department of Computer Science, Heriot-Watt University,
Edinburgh, Scotland

Abstract

We report the first stage of a project aimed at computing *visibility scripts* for active inspection applications, in which a robot-mounted sensor observes a known object from different viewpoints. Visibility scripts describe the optimal sensor position for a given inspection task and may involve different visibility requirements, e.g. achieving optimal visibility of a single object feature or simultaneous visibility of a set of features. We discuss also *stereo visibility*, or the optimal placement of a stereo head within the visibility region of a feature. Stereo visibility is a novel feature in the panorama of comparable systems and may prove nontrivial in some situations. Script generation is based on an approximate visibility space, the *property sphere*. We take into account several constraints imposed by most real systems, for instance the limited workspace of a real sensor or the desired resolution at which a feature must be observed.

1 Introduction

This paper addresses the problem of *optimal sensor placement* for inspection applications. The class of applications considered involves inspection systems which can observe an object from different viewpoints by moving either the sensor or the object. The sensor is required to acquire an optimal image of one or more features, or a sequence of images. Optimality is defined by various factors, noticeably feature visibility and reliability of feature detection. Since the workspace of any robot is constrained in practice, one might have to contend with suboptimal sensor placements. We call the sequence of optimal sensor placements for a given inspection task a *visibility script.*

The first issue in computing visibility scripts is *feature visibility*: from which region of the 3-D space around an object is a feature visible. This problem goes hand-in-hand with the topics of *viewer-centered representations*, of which *aspect graphs* are perhaps the most popular form (see [6] for a recent survey and introduction). Research in the field has considered mostly image models based on line drawings with edges as main features ([7],[14],[8]); a few surface-based [9] and component-based [15] aspect graph algorithms have been reported recently. The main problems are that implementations of exact techniques are few and the algorithms confined to rather limiting shapes: complexities are very high, up to $O(n^9)$ in the number of object features [6]; very few authors consider surfaces, which are interesting features for inspection in practice; and exact aspect graphs can be redundant in practice.

Notice also that aspects are maximally connected set of viewpoints, whereas feature visibility regions can contain holes or be disconnected. In practical applications, therefore, *approximate visibility representations* are adopted instead of exact aspect graphs. Approximate representations ([3],[4],[16],[6]) restrict the set of possible viewpoints to a sphere of large but finite radius, centered around the object. The approximate space is a discrete grid of viewpoints, obtained by computing a *quasi-regular tessellation*, or *geodesic dome*, of the sphere. Raytracing is used to compute visibility from each viewpoint. The main reason for using approximate representations in applications is that they are a well-understood class of methods, applicable to every object shape. The price to be paid is that there is no guarantee that every significant view is captured given the number of viewpoints (the *resolution* of the tessellation). The particular representation we consider in this paper is the *property sphere* ([3],[4]), briefly detailed in Section 3.

Many inspection tasks are *feature-oriented*: one is interested in inspecting object parts which correspond to model features. One would also like to predict *how reliably* a feature will be detected from a given viewpoint. This leads to the definition of *optimal viewpoint* for a feature inspection task ([10],[11],[1],[12]). We have designed a representation which expresses explicitly the visibility region of a feature, associates an optimality coefficient to each viewpoint and makes it possible to access information by feature index. *Constraints* on the extension of the visibility space are imposed by the characteristics of the sensing devices, by the feature detection techniques and by the workspace of the robot on which the sensor is mounted ([1],[2],[11],[13],[12]). Sometimes the sensor adopted is a stereo camera system, for example as part of a triangulation-based range finder. Computing the optimal sensor placement for a stereo head so that visibility is guaranteed from both cameras can prove nontrivial. Although important for active inspection, this problem has not received much attention in the literature of visibility-based sensor placement. We describe a technique for computing *stereo visibility* and finding the optimal placement for a stereo head in Section 7.

2 Definitions

A few key terms and concepts are defined at this point.
Sensors: the techniques described in this paper apply to several types of sensors, including cyclopean cameras, stereo heads and range finders. In the following, the sensor's type and geometry is explicitly mentioned when necessary. We will refer for simplicity to the case of a mobile sensor moving around a fixed object, although the object could be moved instead.
Features: the features considered are *surface patches*, corresponding to planar and curved object faces.
Models: we generate models with the RoboSolid solid modelling package. The examples in this paper use the model of a widget shown in Figure 1, a moderately complex industrial part.

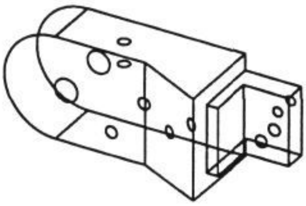

Figure 1: Line-drawing rendering of the widget model.

3 Building the approximate visibility space

The approximate visibility space adopted in this work is the *property sphere*, introduced in [3]. A property sphere is built by subdividing each face of an icosahedron in four equilateral triangles and "pushing out" the new vertices obtained onto the surface of the sphere circumscribed to the icosahedron. By iterating this operation on each new facet, a set of 20 quadtrees can be generated. The resulting sphere tessellation is *quasi-regular* in the sense that it approximates the regularity properties of the platonic polyhedra. The depth of the quadtrees, which is assumed the same for all the faces, is called the *resolution* of the dome. If the resolution of the initial icosahedron is 0, the total number of facets is $20 * 4^{res}$. The main question about geodesic domes is what resolution should be used. Unnecessarily high resolutions result in a wastage of memory; too low resolutions may miss important views. Typical resolutions used in the literature are 2 and 3 (320 and 1280 dome facets respectively).

4 The FIR representation

Inspection tasks require information about the visibility region of a feature and the optimality of the viewpoints inside the region. The representation adopted must make such information explicit and easily accessible. We have designed such a representation, called *FIR* (for Feature Inspection Representation). Computing the FIR solves directly the optimal single-feature inspection problem for *all* features in the model and makes other tasks easy by maintaining explicitly the desired information for all the features. The FIR consists of an array of *feature visibility region descriptors* (henceforth FVRDs). Each FVRD refers to one feature and consists of two components. The first is a *list of viewpoints* from which the feature is visible, which specifies the feature's visibility region under perspective projections. The second is the region's *stability*, which depends on the percentage of the property sphere covered by the region and is given by $\frac{r}{N}$, where r is the number of viewpoints in the region and N the total number of viewpoints in the property sphere. Unstable regions are poor candidates for sensor positioning even if their feature visibility is satisfactory. Each viewpoint descriptor in the FVRD list includes the viewpoint's cartesian and spherical coordinates as well as the following attributes. All attribute values are in the range $[0, 1]$.

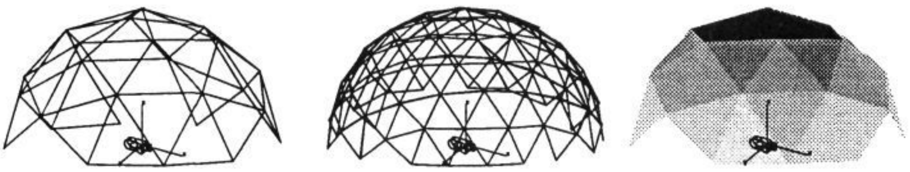

Figure 2: Visibility region for the top plane of the widget at resolution 1, 2 and shaded according to viewpoint optimality (resolution 1, the darker the better). The widget is oriented as in Figure 1.

Visibility: the absolute visibility of the feature in pixels, normalized by the image resolution.
Reliability: the expected reliability with which the feature will be detected from the viewpoint. Computation of this coefficient depends on the characteristics of the sensor and feature detector adopted: for instance, low-curvature cylindrical patches might be confused with planes and assigned low reliability.
Optimality: the global merit of the viewpoint, obtained by combining visibility v and reliability r:

$$o = o(v, r) = k_v v + k_r r$$

where the weights k_v, k_r satisfy $k_v, k_r \in [0, 1]$ and $k_v + k_r = 1$. These weights express the relative importance of visibility and reliability according to the particular task. For instance, if a sequence of images must be acquired to be inspected by an operator (no feature detection involved), a convenient choice is $kr = 0, k_v = 1$.

The essential algorithm for computing a FIR involves generating a geodesic dome and raytracing (perspective projections) from each viewpoint on the dome, from which visibility and reliability are computed for each feature. If too few pixels of a feature are visible from a viewpoint, no viewpoint descriptor is created. Finally, the the algorithm evaluates the stability of each FVRD.

The size of a FIR depends on the object considered and on the resolution of the property sphere. For the widget model in Figure 1 the size is about 20k bytes at resolution 2 (320 viewpoints) and about 80k bytes at resolution 3 (1280 viewpoints). Table 1 in Section 8 gives further examples. The time taken varies with the object, the dome resolution and the image resolution adopted. With 64x64 images, building the FIR for the widget took about 25 mins with 320 viewpoints and 2.5 hours with 1280 viewpoints on a SPARC workstation. With 128x128 images, the time is about 3 hours with 320 viewpoints. Notice that an approximate aspect graph is also computable from the FIR by using a region-growing algorithm on the viewsphere (see for instance [4]).

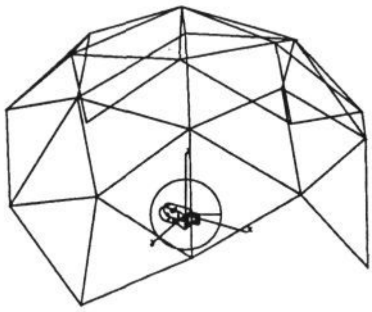

Figure 3: Covisibility region for front cylindrical patch and top L-shaped plane of the widget (referring to Figure 1) at dome resolution 1.

5 Basic visibility scripts: optimal feature visibility

The basic visibility script consists of moving the sensor to the optimal viewpoint for observing a given feature. Our representation has been designed to make this task particularly easy; computing the representation solves directly the basic visibility problem for *all* features. It suffices to pick the best viewpoint in the interesting feature visibility region. Figure 2 shows the visibility region for the side plane of the widget (top plane with hole in Figure 1) as a partial geodesic dome. By associating optimality weights to viewpoints, the FIR supports also the inspection of sets of features from optimal positions. It is sufficient to identify the set of optimal viewpoints for all the features involved. The sensor trajectory can then be planned under appropriate constraints, e.g. that the total distance covered by the sensor is minimum.

6 Optimal covisibility

Knowledge of covisibility is essential whenever several features must be observed simultaneously. The shape of a covisibility region is easily found, thanks to the FIR, by intersecting the visibility regions of all the features involved. The problem reduces to list intersection. Figure 3 shows the covisibility region of the front cylindrical patch of the widget and the side L-shaped plane facing up in Figure 1. The ***stability*** of a covisibility region is the same as that of a single-feature visibility region. The definition of the region's ***optimality*** requires more attention. The optimality o_i^{cov} of a viewpoint i belonging to the covisibility region of features $\{f_1, \dots f_N\}$ is computed as a function of the optimalities o_{ij} of viewpoint i for single-feature visibility of feature j. Any candidate function for o_i^{cov} must meet two requirements. First, the same number of pixels should be visible simultaneously for all features: it is no good to see the whole of feature f_1 and nothing of feature f_2. Second, o_i^{cov} should increase with the number of pixels visible. We assume $o_{ij} \in [0, 1]$ for all i and j. The function satisfying these requirements we adopted is

$$o_i^{cov} = \sqrt{\bar{o}_{ij}^2 - \sigma_o^2}$$

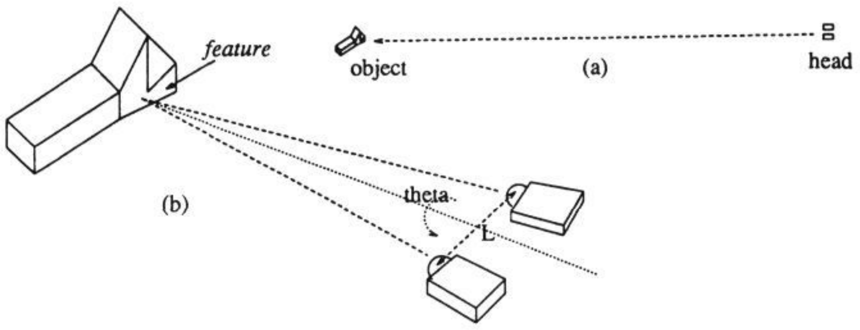

Figure 4: Stereo visibility can require consideration of the sensor geometry (b) or not (a). See text.

where $\bar{o}_{ij}$ and σ_o^2 are respectively the mean and variance of the o_{ij}.

7 Stereo visibility

Stereo visibility can be necessary when the sensor used is a stereo head. Computing stereo visibility requires that a few conditions are satisfied. Firstly and obviously, a given feature must be observable from both cameras: therefore both cameras must lie inside the visibility region of the feature. Secondly, it may be required that the camera-to-camera distance (or *interocular* distance) must be compatible with the distance between two adjacent viewpoints on the property sphere. Thirdly, constraints imposed by the robot used might limit the possible attitudes that the stereo head can assume. All or some of these constraints must be considered according to the task at hand, as discussed below.

In some cases the stereo system can be approximated with one point and we can assume that both cameras observe the same image. This happens usually for tasks requiring global visibility i.e. that the *whole* object is visible from all viewpoints, as for instance when checking for missing subparts. The conditions are that the object-sensor distance is much greater than both the interocular distance and the object size (see Figure 4a). In this case the probability that both cameras are in the same visibility region is high and the problem can be reduced to one of single-camera visibility. This assumption is adopted implicitly in [2].

In some tasks, however, the sensor cannot be approximated by a point. This happens when the probability that the two cameras end up in different visibility regions is not negligible. This can occur when the the interocular distance is comparable with the camera-object distance and therefore with the radius of the viewsphere (Figure 4b), as is the case in close inspection tasks. In such cases positioning the stereo head can be nontrivial.

If objects are known *a priori*, it might be possible to predefine an *optimal inspection direction*. For a planar patch, for instance, this can be the normal to the plane taken through the feature's baricentrum. However, the use of a single direction can be unsatisfactory: for instance, range-based HK curvature estimators might distort cylindrical patches according to the angle formed by the local patch normal and the viewing direction [5]. Moreover, for a stereo head, a single direction does not guarantee visibility from *both* cameras. Our approach is to guarantee stereo visibility

Bytes allocated	Saved	Threshold
84432	62160 (42%)	4
82632	63960 (44%)	6
80784	65808 (45%)	8
76368	70224 (48%)	10

Table 1: Minimum visibility constraint: size of the allocated FIR in bytes, memory saved by visibility thresholding in bytes and as a percentage of the size of the unthresholded representation, threshold enforced in pixels. The widget model was raytraced at a resolution of 64x64 from all viewpoints of a medium-resolution property sphere (320 nodes).

by finding the optimal position of the stereo head inside the visibility region of a feature to be inspected, as described below.

First the radius of the property sphere is determined according to the constraints imposed by the task (see Section 8) and a property sphere generated. If the sensor is to be placed at the minimum distance from the object which guarantees no collision, the radius of the property sphere is taken to be the radius of the minimum sphere enclosing the object. Then the visibility region of the desired feature is computed as described in Section 4. Notice that the region of space from which it is necessary to raytrace can be rather small and only a partial dome is generated at close distance from a feature.

We then try to find the optimal unconstrained position for the stereo head within the visibility region. To do this, the stereo head is approximated with a linear segment of length L equal to the camera-to-camera distance (see Figure 4). The algorithm selects pairs of viewpoints which are distant $L \pm \varepsilon$ from each other, where ε expresses the tolerance introduced by the approximate visibility space and depends on the resolution of the property sphere. We adopted Korn and Dyer's algorithm [4] (complexity $O(f)$, f number of viewpoints in the property sphere) for finding all viewpoints at a fixed distance from a given one. The *combined optimality* of the pairs is then evaluated. Optimalities are combined as described for covisibility (Section 6). The viewpoint pair maximising the combined optimality is the optimal sensor position.

Finally, the solution is checked against workspace constraints, which restrict the possible head rotations around its axis (angle θ in Figure 4). If the optimal unconstrained solution does not satisfy the constraints, the first suboptimal solution which does is chosen. We assume that it is always possible to adjust the cameras' vergence so that they point to the centre of the feature being inspected (as shown in Figure 4). In this case, we can assume that the images actually acquired by the stereo head differ from the images predicted by the FIR only by a rotation.

Notice that only a limited number of candidate viewpoint pairs is usually considered by the algorithm, thanks to the combined effect of the head geometry constraint and the close distance implying small partial domes.

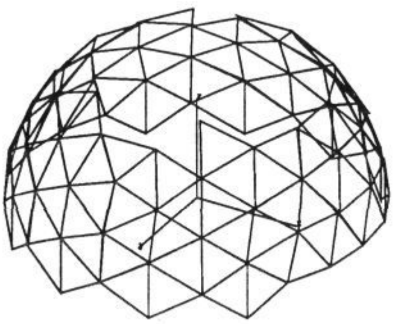

Figure 5: Constrained visibility space for a turntable-based inspection setup.

8 Constraints

In most practical applications, several constraints are imposed on sensor placement. Constraining factors include the sensor's geometry, the robot's workspace, the feature detectors adopted. Constraints lead to a reduction of the number of viewpoints to be considered in any task, but imply additional computation to be enforced. We describe here only the constraints adopted in the present prototype implementation: *workspace, minimum resolution* and *minimum visibility*. More constraints will be added to the system in the future.

Robot workspace. Depending on the characteristics of the robot adopted, certain regions of space will not be accessible to the robot-mounted sensor. The radius and reachable areas on the property sphere must be constrained accordingly, and sensor placement computed within the resulting constrained area. At the moment workspace contraints are expressed through systems of inequalities in spherical coordinates. As an example, Figure 5 shows the area of the property sphere satisfying the workspace constraints imposed by an inspection system whereby the object sits on a turntable and is observed by a camera free to move around the object but in a limited elevation range. The resulting workspace can be easily described in spherical coordinates by the inequalities $\Theta_{min} < \theta < \Theta_{max}$, where Θ_{min} and Θ_{max} depend on the installation and θ is the elevation (spherical coordinates).

Minimum visibility. The minimum visibility constraint imposes that the number of pixels of any feature visible from any viewpoint cannot be less than a threshold specified by the user. The threshold depends on the requirements of the task considered. This constraint is enforced during the construction of the FIR (see Section 4) and inhibits the allocation of viewpoint descriptors for which feature visibility is unacceptably low. The benefit introduced by this simple constraint in terms of memory saved is remarkable. Table 1 shows some figures obtained in our experiments with the widget model.

Minimum resolution. In some tasks it is desirable to ensure that the interesting feature appears in the image at a minimum resolution. for instance for close inspection or because the feature detector would not yield reliable results at coarser resolution. This leads to an upper bound for the distance between the feature and the camera which can be expressed as follows. Consider the geometry shown in Figure 6 and let D be the maximum linear dimension of the interesting feature in millimiters, d the distance between the camera and the feature in millimeters, and

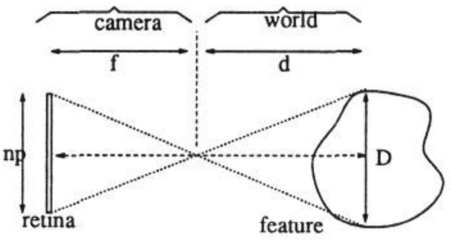

Figure 6: The minimum resolution constraint geometry.

n the feature's size on the retina in pixels. Then the requirement that the feature's linear resolution in pixels on the retina is at least n implies $d \leq k_{cam} \frac{D}{n}$ where $k_{cam} = \frac{f}{p}$ is a constant defined by the camera parameters, f the focal length and p the linear resolution of the retina in millimiters per pixel (supposed homogeneous for rows and columns). Images taken from a property sphere generated around the center of the desired feature, with a radius satisfying the above inequality, will meet the minimum resolution constraint.

9 Conclusions

We have presented some initial developments of a project aimed at generating inspection scripts, or sequences of optimal sensor positions for feature inspection under the constraints imposed by a real setup. The problem is of considerable interest for several active inspection applications. We see the main attractive features of this work in its explicit consideration of stereo visibility spaces and of a set of constraints which, although partial at present, do occur in real installations. Extensions are planned in order to cope with more complex workspaces, to incorporate further constraints, to obtain optimal scripts for sequential inspection of lists of features.

Acknowledgments

Thanks to Bob Fisher, John Hallam and Tim Newman for useful discussions and suggestions. This work was supported by the LAIRD project (Location and Inspection with Range Data), a collaboration between British Aerospace, Bae SEMA Ltd, the National Engineering Laboratory, Heriot Watt University and the Universities of Edinburgh and Surrey. The project is led by British Aerospace and is funded by the SERC/Information Engineering Directorate (GR/F38327:1551).

References

[1] S. Sakane, M. Ishii and M. Kakikura: *Occlusion Avoidance of Visual Sensors Based on a Hand-Eye Action Simulator System: HEAVEN*. Advanced Robotics **2**, pp. 149 - 165, 1987.

[2] S. Sakane and T. Sato: *Automatic Planning of Light Source and Camera Placement for an Active Photometric Stereo System.* Proc. IEEE Int. Conf. on Robotics and Automation, 1991, pp. 1080 –1087.

[3] G. Fekete and L.S. Davis: *Property Spheres: a New Representation for 3-D Object Recognition*, Proc. IEEE Workshop on Computer Vision, Representation and Control, pp. 192 – 201, 1984.

[4] M.R. Korn and C.R. Dyer: *3-D Multiview Object Representations for Model-Based Object Recognition*, Pattern Recognition **20**, pp. 91 – 103, 1987.

[5] E. Trucco: *On Shape-Preserving Boundary Conditions for Diffusion Smoothing.* Proc. IEEE Int. Conf. on Robotics and Automation, 1992, pp. 1690 – 1694.

[6] K. Bowyer and C.R. Dyer: *Aspect Graphs: an Introduction and Survey of Recent Results*, International Journal of Imaging Systems and Technology **2**, 1990, pp. 315 – 328.

[7] Z. Gigus, J. Canny and Seidel: *Efficiently Computing the Aspect Graph of Polyhedral Objects*, Proc. IEEE International Conference on Computer Vision, 1988, pp. 30 – 39.

[8] S. Petitjean, J. Ponce and D. Kriegman: *Computing Exact Aspect Graphs of Curved Objects: Algebraic Surfaces.* Tech. Rep. UIUC-BI-AI-RCV-92-02, University of Illinois at Urbana-Champaign, 1992.

[9] Kaiser, K. Bowyer and Goldgof: *On Exploring the Definition of a Range-Image Aspect Graph*, Proc. 7th Scandinavian Conference on Image Analysis, 1991, pp. 652 – 656.

[10] J. Ben-Arie: *Probabilistic Models of Observed Features and Aspects with Applications to Weighted Aspect Graphs*, Pattern Recognition Letters **11**, 1990, pp. 421 – 427.

[11] K. Ikeuchi and T. Kanade: *Modeling Sensors: Toward Automatic Generation of Object Recognition Program*, Computer Vision, Graphic and Image Processing **48**, 1989, pp. 50 – 79.

[12] H.-S. Kim, R.C. Jain and R.A. Volz: *Object Recognition Using Multiple Views*, Proc. IEEE Conference on Robotics and Automation, 1985, pp. 28 – 33.

[13] C.K. Cowan and P.D. Kovesi: *Automatic Sensor Placement for Vision Task Requirements*, IEEE PAMI **10**, 1988, pp. 407 – 416.

[14] D. Eggert and K. Bowyer: *Perspective Projection Aspect Graphs of Solids of Revolution: an Implementation.* Proceedings 7th Scandinavian Conference on Image Analysis, 1991, pp. 299 –306.

[15] S. Dickinson, A. Pentland and A. Rosenfeld: *From Volumes to Views: An Approach to 3-D Object Recognition*, Proc. IEEE Workshop on Advances in CAD-Based Vision, Hawaii, 1991, pp. 85 – 96.

[16] T. M. Silberberg, L. Davis and D. Harwood: *An Iterative Hough Procedure for Three-Dimensional Object Recognition.* Pattern Recognition **17**, 1984, pp. 621 – 629.