Practical Aspect-Graph Derivation Incorporating Feature Segmentation Performance

Andrew W. Fitzgibbon
Department of Artificial Intelligence, Edinburgh University
5 Forrest Hill, Edinburgh EH1 2QL

Robert B. Fisher
Department of Artificial Intelligence, Edinburgh University
5 Forrest Hill, Edinburgh EH1 2QL

Abstract

A procedure is described for the automatic derivation of aspect graphs of surface-based geometric models. The object models are made of finite, typed, second order surface patches – allowing the representation of a large number of complex curved objects while retaining ease of recognition. A new representation, the detectability sphere, is developed to encode feature detectability constraints. The detectability metric is directly related to the performance of the imaging system, allowing the generated aspect graph to more truthfully represent the scene's relationship with the vision system.

An algorithm is described which fuses information from several views of the object to produce a small number of characteristic views which cover some desired portion of the viewsphere, and annotates these fundamental views with pose-verification hints. The procedure is compared with previous analytic and approximate solutions to the aspect-graph problem regarding relevance to the vision process, range of applicability, and computational complexity.

1 Introduction

An important problem in model-based 3D computer vision systems is the complexity and number of interpretations possible within a given scene. Even with a limited number of models in the system's modelbase, the many distinct 2D appearances that a complex 3D object may have can give rise to many thousands of possibilities. To categorise these views and their interconnections, Koenderink and van Doorn [13] defined the **aspect graph**, linking 2D object views (nodes) and visual events (arcs). Given a model's aspect graph, the matching process is aided by the constraints placed on the initial generation of hypotheses (invocation) and the ability to perform detailed symbolic verifications once pose has been determined.

Since its introduction, much effort has been expended in analytically deriving exact aspect graphs [3, 14, 15], which yield a complete description of the object's viewsphere. Analytic methods, however, have been limited to simple models and may be expensive if the model contains many features. Gigus and Malik [10] describe an algorithm for nonconvex polyhedra which has a worst case complexity of $O(n^8)$ in the number of vertices. Ponce's algorithm [15] applies to a general class of object but no implementation is reported, and no complexity measure is given.

Moreover, the exact aspect graph is generally too detailed to be useful in model matching with real data (Gigus and Malik suggest up to $O(n^6)$ distinct views). Koenderink and van Doorn recognised that many nodes in the aspect graph will correspond to "unstable" views where an infinitesimal camera motion will change the topological properties of the view, and exact aspect graphs are usually generated without such views, or they are pruned on completion of the graph. In a real application, not only these views, but also many where a non-infinitesimal but suitably small camera movement changes the aspect must be pruned. Ben-Arie [2] introduces the "probability sphere" where the individual and joint probabilities of feature visibility are represented as the areas of regions on the gaussian sphere, and this to some extent considers this situation.

However, the opposite may also happen: poor sensor performance may cause a view which is theoretically unstable to actually cover a significant portion of the viewsphere. For example, the face-on view of a cube under orthographic projection will be classified as unstable by an exact algorithm, whereas with certain range sensors, it will often be the only face visible for up to 15 degrees in any direction.

In summary, exact aspect graphs do not provide the information required for real scene analysis applications.

This report describes a practical approach which combines the geometric object models with empirically-derived sensor models to provide a probability of successful detection (**detectability**) measure for each model component. The significance of this approach is that thereby a viewsphere is derived containing precisely the features that will be viewable when the object is observed. By incorporating the sensor model we can generate fewer, more significant views which are directly usable in the matching process. Because sensor performance is described using experimentally measured performance graphs and rules, the technique is applicable to a very general class of sensors, and can encode not only the physical sensor, but also the effects of the image processing stages (smoothing, segmentation) that generally precede the symbolic matching.

2 Context

This work was done in the context of the IMAGINE II 3D vision system, described in [7]. This system uses a three-stage approach to the model-matching task:

Invocation pairs scene features with model subparts in an $m \times d$ network [6]. Hypotheses generated at this stage consist of pairings between model visibility groups and scene perceptual groupings. Using coarse visibility groupings at this stage allows constraints such as "Surfaces A and B will never be seen together" to be cheaply encoded, simply because they never appear in the same visibility set. A priori knowledge such as "The object is sitting on its bottom" and hints like "B might be visible but don't depend on it" can also be included, improving the accuracy and speed of recognition.

Interpretation tree pruning applies geometric constraints similar to those described in [11] to sets of hypotheses, further reducing the number of hypotheses. As this is a combinatorially explosive process, visibility information is used to reduce the numbers of features considered.

Geometric reasoning creates a position estimate for each assembly hypothesis, using a Kalman filter-based stochastic fitting process to minimise the effects of noise in the data [4, 1].

2.1 Object Models

The models considered for analysis are assemblies of non-infinite 2^{nd} order surfaces, described using the Suggestive Modelling System ([5], [9]). An **assembly** is a set of surfaces and reference-frame transformations:

$$\mathcal{A} \;=\; \left\{ (\mathcal{S}_i,\; {}^{\mathcal{A}}T_{\mathcal{S}_i}) \right\}_{i=1}^{m}$$

In this notation, the transformation ${}^{\mathcal{A}}T_{\mathcal{S}}$ transforms points in the surface's reference frame into the assembly's reference frame. The surface primitive $\mathcal{S}$ is on one side of the infinite quadric surface parameterized by a function $\mathbf{s}(u, v)$. The surface's finite extent is represented by a **parameter-space mask** — a subset of (u, v) space. Because the model stores the transformation from surface to assembly coordinates, the surfaces are represented in canonical positions. In practice we are additionally limited to those surfaces which can be reliably extracted from the sensor data. Currently this means restriction to Plane, Cylinder, Ellipsoid, Cone, and Elliptical and Hyperbolic Paraboloids.

3 Overview of the Method

An outline of the procedure for generating the visibility information for the matcher is as follows. Several sample viewpoints are chosen by tessellating the gaussian sphere into tesserae (typically around 800), the choice of tessellation being designed to maximize the coverage of the viewsphere while maintaining tessera connectivity and solid angle. Each tessera defines a camera position, from which the geometric model is raycast, generating visibility statistics for each subcomponent in each view. The amount of information which must be included in the visibility statistics is determined by the form of the detectability rules used in the next stage.

The visibility statistics are then analyzed on a per-feature basis, using empirically derived detectability rules to generate **detectability spheres** for each subcomponent, which encode the likelihood that the subcomponent will be correctly sensed and segmented from each sampled viewpoint.

Combining the per-feature detectability spheres and applying an overall desired reliability criterion yields a set of "reliably detectable" surfaces at each viewpoint. Merging views from which the same subcomponents are so classified divides the viewsphere into regions, labelled by the list of features which are detectable from within the region.

Collecting the viewsphere regions gives a list of visibility groups which is pruned based on stability and likelihood criteria to give a smaller list of "viewgroups" which is added to the SMS model.

3.1 Choice of Tessellation

We now consider the problem of choosing viewpoints from which to take sample views. Because the sensor used (a laser range finder) produces orthographic images, only the direction of the viewing vector need be considered, reducing the problem to finding an appropriate tessellation of the gaussian sphere. The naive approach, dividing the normal elevation (ϕ) and azimuth (θ) evenly, produces a rectangular array of normals which is easy to deal with in a computer. This, however, produces a non-uniform tessellation, which means that near to the poles of the sphere, each direction represents a smaller proportion of azimuth than an equivalent at the equator.

A number of techniques to avert this effect have been suggested in the literature. The most commonly seen solution is to take a solid such as an icosahedron and recursively subdivide its faces until a desired resolution has been achieved [12]. This approach, while pleasing, still fails to generate a perfectly uniform tessellation. Alternative approaches include various random techniques which try to ensure that the solid angle subtended by each tessera remains constant.

The solution adopted in this work takes a similar, but simpler, view. First, what are the difficulties we will have with a naive regular tessellation? The connectivity of regions will still give us the same characteristic views, but they will be 'larger' near the poles (occupying more tesserae), and work will have been wasted in generating views at more than the required resolution. We can correct for both the size (in solid angle) and the amount of wasted work simply by reducing by a factor of $\sin(\phi)$ the number of subdivisions of azimuth at each elevation ϕ.

3.2 Ray Casting the Finite Surfaces

To determine the visibility of objects in a single view, the depth of each surface at each point in the scene is required. These depths are calculated using a technique similar to graphical ray tracing using an image plane divided into $X \times Y$ pixels. The line of sight passing through each pixel is intersected with each subcomponent of the model, producing a list of intersection reports of the form (i, z), saying that surface $\mathcal{S}_i$ intersected at depth z. The subcomponent which reports an intersection at the nearest point along the line of sight will be visible (the **front surface**), all others are occluded by the front surface.

3.3 Determining Detection Reliability for One Surface

Analysis of the information produced by the raycasting algorithm uses the concept of a detectability sphere to represent feature visibility over the range of possible camera positions. The detectability sphere for a single surface $\mathcal{S}_i$ is the distribution

$$P\{\mathcal{S}_i\}(\hat{\mathbf{v}})$$

which defines, for each viewing direction, the probability that $\mathcal{S}_i$ will be correctly sensed, segmented and classified. This probability is on a binary event: "Will the symbolic data description produced by the early vision processing be good enough to allow us to modelmatch?" To answer this question, we first note that successful object recognition is a function of several factors:

Occlusion alters the size of a surface and its 2D appearance. Three dimensional shape is not affected but significant occlusion can hamper the segmentation process, impacting the reliability of shape parameter estimation. Cylinder radii, for example, are estimated poorly as less of the cylinder is seen.

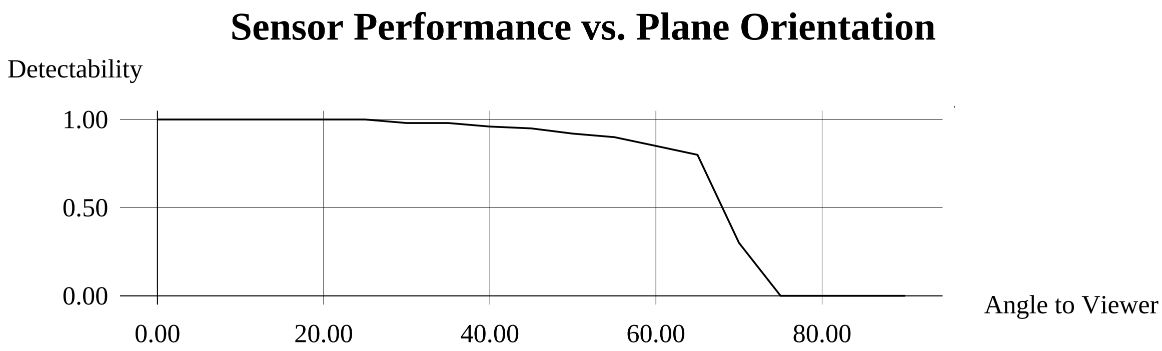

Figure 1: Graph relating detectability to the angle between the line of sight and a planar surface normal. The material used was anodized aluminium, the data acquired using a stereo laser striper in a 1m^3 workspace. Note the sudden catastrophic failure at about $65°$.

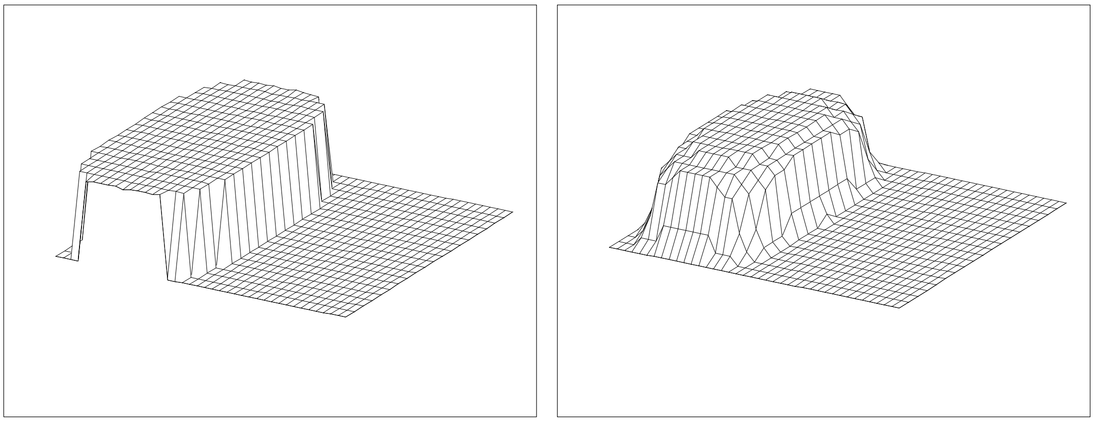

Figure 2: Detectability sphere (left) for a plane based on ratio of visible area to model area. The detectability sphere has been flattened onto a rectangular grid, indexed by camera azimuth and elevation, with height above the plane representing likelihood of correct segmentation. Azimuth varies between $-\pi$ and π. Elevation is between 0 and π. The figure on the right is the detectability sphere based on probability of correct classification.

The **quality of sensor data** depends on surface finish, shape and position with respect to the sensor. Figure 1 illustrates the variation of planar surface detectability over a range of orientations. For cylindrical patches a similar dependency is seen, but on the orientation of the cylinder axis.

The choice of **algorithm parameters** will affect system reliability at all stages of processing. For example, the HK curvature-based range data segmentation algorithm described in Trucco [16] has performance curves relating cylinder radius R and the H_0 and N_c segmentation parameters to the number of region pixels correctly classified. Hence, given R, H_0 and N_c for a particular cylinder, we can predict the percentage of image pixels that will be correctly classified, and thus the likelihood that the surface patch itself will be detected. At a higher level, if a simple search-tree pruning constraint uses a χ^2 test to compare model and data areas, say, then the choice of confidence interval is a parameter which will alter the detectability of surfaces whose area measurements are deviate significantly from the true values.

In addition, some general criteria can be applied — very small patches (as measured by raw pixel area) will not segment, and patches with a high compactness number ($perimeter^2/area$, again measured in raw pixel values) will also prove difficult. These criteria apply to all surface types and play an important role in removing from consideration small, rarely detectable subparts which will not aid recognition.

3.4 Detectability Rules

To represent these constraints in our algorithm, "detectability rules" are encoded for each surface type which relate the surface's pose, the parameters of the segmentation algorithm, and the segmentation performance data to give a probability that the feature will be correctly classified. The rules are represented as functions of the single-view, per-surface, visibility statistics:

• The total number of rays which intersected the forward-facing portion of the finite surface.

Occluded Surface \ Front Surface	1	2	3	4	…
1	–	7			
2	9	–	300		
3			–		
4				–	
⋮					

Table 1: Example occlusion matrix for a single view. The 300 figure is the number of rays cast where both surface 3 and surface 2 were intersected, but surface 3 occluded surface 2. The 7 and 9 illustrate quantisation noise caused by surfaces 1 and 2 sharing a boundary. Also, the diagonal is empty as a second-order surface cannot self-occlude (Consider the gradients in the case where two intersections are reported.)

- The number of rays along which the surface was the nearest to the camera. This defines the actual surface visibility, taking account of occlusion.
- The visible surface area. This is calculated at each image pixel, by correcting the pixel area for foreshortening by the dot product of the surface normal and line of sight.

3.4.1 Representing Self-Occlusion for Matching

The per-surface data indicates how much a particular surface has been occluded by all other surfaces in the view. For pose verification, however, it is useful to know in greater detail which surfaces have occluded others. This information is supplied in an **occlusion matrix** (Table 1), where the entry at cell (i, j) is the number of rays along which surface $\mathcal{S}_i$ was the front surface, and occluded a forward-facing portion of surface $\mathcal{S}_j$.

Occlusion information is useful in a quick model verification process once pose estimation is complete [7, 5]. The verification process can use the pose to index information about feature visibility (nearly fully visible, partially obscured, tangential) and surface ordering ("Surface A partially obscures surface B and hence expect a depth discontinuity boundary between the two.").

4 Merging Information from Multiple Views

The process of deriving the characteristic views may be separated into two subprocesses. First, each model surface $\mathcal{S}$ is taken individually and its detectability sphere, $P\{\mathcal{S}\}(\hat{\mathbf{v}})$ is generated. The individual surface probabilities are then combined to produce group probabilities, giving codetectability likelihoods for groups of subcomponents.

The second stage is to process the plausible groupings in order to filter out pathological or unstable views, before augmenting the SMS model with the new visibility information.

4.1 Joint Detectability for Each View

Applying the detection rules to each model surface in a particular view produces a set of detection probabilities $P\{\mathcal{S}_i\}$. These are combined to obtain the likelihood of codetectability for a set of surfaces $\{\mathcal{S}_{i_j}\}_{j=1}^{n}$. In general, we want to find the largest set whose joint probability exceeds some reliability threshold. We would expect, then, to have to calculate joint probabilities for all sets of surfaces in the view. However, because the probability that a surface will segment correctly depends only on the viewing direction, and not on the detectability of the other surfaces, the joint probability calculation simply amounts to taking the minimum of the individual probabilities[1]:

$$P\{\mathcal{S}_1, ..., \mathcal{S}_n\} = \min_{i=1}^{n} P\{\mathcal{S}_i\}$$

This in turn means that the joint probability of a group of surfaces will exceed the reliability threshold iff each individual probability does so. Thus, the surfaces can be individually thresholded and the successful surfaces grouped into a "reliably detectable" set (called a **viewgroup** in SMS). A connected-components algorithm then divides the viewsphere into regions from which groups of subcomponents are reliably detectable.

4.2 Pruning Insufficiently Stable Viewgroups

The list of viewgroups produced by the reliability thresholding will still contain undesirable views which are too unstable or contain too few surfaces. Unstable views are identified as those which are represented by very "thin" regions on the flattened viewsphere. This corresponds to a discretisation of the usual concept of viewpoint instability; essentially we are now defining an unstable group as one from which a small, but no longer infinitesimal, camera movement will change the object's aspect. Thin regions are pruned by finding their bounding box at several orientations. Regions whose minimum width over all orientations is less than $\sqrt{2}$ are regarded as pathological views and are rejected.

After this pruning, the viewgroups are converted to SMS format and added to the supplied geometric model.

5 Results

When evaluating the system, our first consideration is to see how well its predicted viewgroups agree with the views previously chosen by hand. The part considered is shown in Figure 3. The views shown are the views expected when the part is observed from above, its base being flat on a workbench.

Comparing the automatic and hand generated versions, views A, B and C correspond, but views D_1 and D_2 in the hand-edited model have been split into three by the program. This is explained by the diagnostic that D_1 and D_2, when evaluated by the program, proved to occupy a very small portion of the viewsphere. Instead, the three groups Z_1, Z_2 and Z_3 have been created, explaining significantly more of the sampled views. These three views are all the possible two-surface subgroups of the erroneous three-surface groups.

[1] This is similar to Ben-Arie's [2] use of set intersection on the gaussian sphere in the exact case.

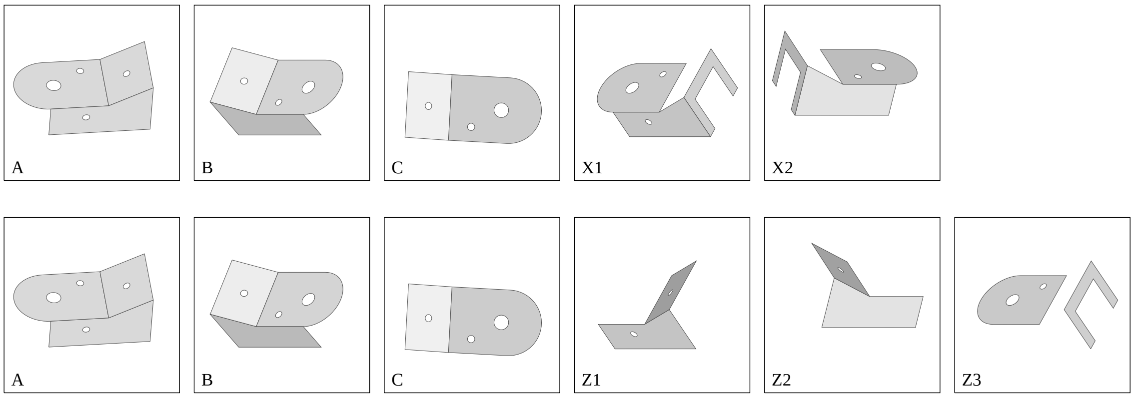

Figure 3: The B.Ae. Widget. This part is taken as an example of a moderately complex object for industrial inspection tasks. The upper sequence shows the original hand-selected viewgroups, with only the visible subcomponents drawn. The lower sequence shows the groups selected by the algorithm. Also, only the views from above are shown, as the widget is assumed to be constrained in this case to lie on its base.

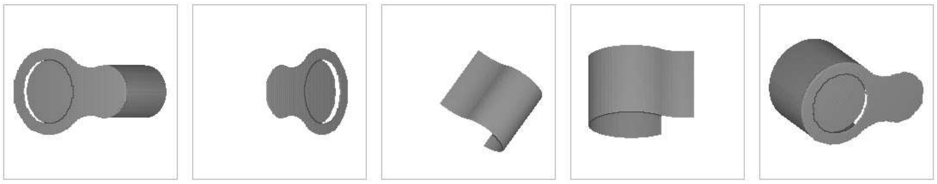

Figure 4: Renault part. Due to the object's rotational symmetry, only half of the generated views are shown.

The algorithm was also run on the partially symmetric object shown in Figure 4, again producing a sensible list of views. However, because the object is symmetric, the program has produced two copies of each group, where two model surfaces have the same shape, but are separately described.

6 Conclusions

The presented algorithm shows a number of advantages over exact techniques. Using second order surface patches as the basic primitive allows the incorporation of many more types of models, and leads to a viewsphere with a reasonably small number of important views. In contrast, exact edge based systems produce tremendously large viewspheres, which are difficult to index, and which may consist of many pathologically unlikely views. For example the B.Ae. part, with 45 modelled edges and 22 surfaces, would be expected to generate 8×10^9 views, as opposed to the 13 generated by our algorithm.

Using actual performance rules empirically derived from the segmentation software allows the viewgroups to more accurately represent what will be detected from the scene. For example, under orthographic projection, the top view (view 'C') in figure 3 should be deemed unstable, as a small camera movement will reveal the other surfaces. The segmentation performance, however, is such that the top view covers quite a large section of the viewsphere before the sides will pass the reliability threshold.

The ability to decide the threshold on detectability gives an easy tradeoff between reliability and speed in the model matcher – setting a high threshold means more distinct viewgroups and longer matching times. The modelmatcher with which the system is currently used is now being placed under knowledge-based control, which means that this is a useful property.

7 Further Work

The described algorithm works well for 'flat' (single-assembly) models with a small number of surfaces. When the number of surfaces becomes large, or when articulated parts are considered, the viewsphere quickly becomes very complex. This complexity is significantly reduced by using hierarchical object models. Fisher [8] shows that for a sample articulated part, the number of viewgroups is reduced to $O(n)$ in the number of surfaces by converting a flat assembly of surfaces to a hierarchical assembly of assemblies. The program could be extended to automatically deduce subcomponent hierarchies which would reduce the complexity of the viewsphere.

Symmetric objects cause problems in that too many viewgroups are produced when the object has separately modelled, but similar, subcomponents. Currently, matcher performance is improved by hand-editing the model file to remove 'duplicate' groups. The program could possibly detect such symmetries and remove them automatically.

References

[1] N. Ayache and O.D. Faugeras. Maintaining representations of the environment of a mobile robot. In *Robotics Research 4*, pages 337 – 350. MIT Press, USA,

1988.

[2] J. Ben-Arie. Probabilistic models of observed features and aspects with applications to weighted aspect graphs. *Pattern Recognition Letters*, 11(6):421 – 427, June 1990.

[3] D. Eggert and K. Bowyer. Perspective projection aspect graphs of solids of revolution: an implementation. In *Proceedings, 7^{th} Scandinavian Conference on Image Analysis*, pages 299 – 306, 1991.

[4] R.B. Fisher and M.J.L. Orr. Geometric reasoning in a parallel network. *International Journal of Robotics Research*, 10(2):103–122, 1991.

[5] Robert B. Fisher. SMS: A suggestive modelling system for computer vision. *Image and Vision Computing*, 5(2):98–104, May 1986.

[6] Robert B. Fisher. Model invocation for three dimensional scene understanding. In *Proceedings, 10th International Joint Conference on A.I.*, pages 805–807, 1987.

[7] Robert B. Fisher. *From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis.* John Wiley, UK, 1989.

[8] Robert B. Fisher. Reducing viewsphere complexity. In *Proceedings, European Conference on Artificial Intelligence*, pages 274–276, 1990.

[9] Andrew W. Fitzgibbon. *The Suggestive Modelling System: Reference Manual for Version 2.* University of Edinburgh, 1990.

[10] Z. Gigus and J. Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE T-PAMI*, 12:113 – 122, 1990.

[11] W. Eric L. Grimson and Tomas Lozano Pérez. Model-based recognition and localization from sparse range or tactile data. *Image and Vision Computing*, 3(3), 1984.

[12] K. Ikeuchi. Recognition of 3d objects using the extended gaussian image. In *Proceedings 7th IJCAI*, pages 595–600, 1981.

[13] J.J. Koenderink and A.H. van Doorn. The internal representation of solid shape with reference to vision. *Biological Cybernetics*, 32:211 – 216, 1979.

[14] D.J. Kriegman and J. Ponce. Computing exact aspect graphs for curved objects: solids of revolution. *International Journal of Computer Vision*, 5(2):119 – 135, November 1990.

[15] J. Ponce and D.J. Kriegman. Computing exact aspect graphs for curved objects: parametric patches. In *Proceedings, 8^{th} National Conference on Artificial Intelligence (AAAI)*, pages 1074 – 1079, 1990.

[16] E. Trucco and R. B. Fisher. Shape- and discontinuity-preserving segmentation of range images using curvature sign estimates. In *Proceedings, IEEE International Conference on Image Processing*, 1992.