Automatic Generation of Object Recognition Programs

KATSUSHI IKEUCHI AND TAKEO KANADE, SENIOR MEMBER, IEEE

Invited Paper

This paper discusses issues and techniques to automatically compile object and sensor models into a visual recognition strategy for recognizing and locating an object in three-dimensional space from visual data. Historically, and even today, most successful model-based vision programs are handwritten; relevant knowledge of objects for recognition is extracted from examples of the object, tailored for the particular environment, and coded into the program by the implementors. If this is done properly, the resulting program is effective and efficient, but it requires long development time and many vision experts.

Automatic generation of recognition programs by compilation attempts to automate this process. In particular, it extracts from the object and sensor models those features that are useful for recognition, and the control sequence which must be applied to deal with possible variations of the object appearances. The key components in automatic generation are: object modeling, sensor modeling, prediction of appearances, strategy generation, and program generation.

An object model describes geometric and photometric properties of an object to be recognized. A sensor model specifies the sensor characteristics in predicting object appearances and variations of feature values. The appearances can be systematically grouped into aspects, where aspects are topologically equivalent classes with respect to the object features "visible" to the sensor. Once aspects are obtained, a recognition strategy is generated in the form of an interpretation tree from the aspects and their predicted feature values. An interpretation tree consists of two parts; a part which classifies an unknown region into one of the aspects, and a part which determines its precise attitude (position and orientation) within the classified aspects. Finally, the strategy is converted into an executable program by using object-oriented programming. One major emphasis of this paper is the sensors, as well as objects, must be explicitly modeled in order to achieve the goal of automatic generation of reliable and efficient recognition programs.

Actual creation of interpretation trees for two objects and their execution for recognition from a bin of parts are demonstrated.

Manuscript received November 1, 1987; revised March 29, 1988. This research was sponsored by the Defense Advanced Research Projects Agency, DOD, through ARPA Order 4976, and monitored by the Air Force Avionics Laboratory under Contract F33615-84-K-1520. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

The authors are with The Robotics Institute and Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA.

IEEE Log Number 8822790.

I. INTRODUCTION

A large class of practical vision problems is object recognition, that is, recognizing and locating objects in the scene by means of visual inputs. To name a few, visual part acquisition on a conveyor belt or from a bin of parts, target recognition in aerial images, and landmark recognition by a mobile robot, all belong to this class of problems. In most of these cases, we have some prior knowledge of the objects of interest, such as the shapes, sizes, reflective properties, and so forth. Model-based vision [1], [2] seeks to actively use such prior knowledge of objects for guiding the recognition process in order to achieve efficiency and reliability.

One of the critical issues in building a model-based vision system is how to quickly extract and organize the relevant knowledge of an object and to systematically turn it into a vision program. In particular, it is important to know what features of objects are useful for recognition, and what control is to be applied to deal with possible variations of the object appearances. In earlier vision systems, such knowledge of objects has been extracted from examples of the object, tailored for the particular environment, and coded into the program by the implementor. For example, in interpreting incomplete line drawings of polyhedra of known size and shape, Falk [3] analyzed failure patterns of line extraction and implemented strategies to cope with them. In fact, even today, most successful vision systems are developed based on the implementors' insight into the specific problems. Some representative examples include 3-D object recognition systems in range maps by Oshima and Shirai [4] and by Faugeras and Hebert [5], aerial photointerpretation systems by Nagao and Matsuyama [6] and by McKeown, Harvey and McDermott [7], bin-picking systems by Perkins [8] and Ikeuchi and Horn [9], and the NAVLAV mobile robot vision system by Thorpe *et al.* [10]. In these systems, features and recognition strategies to be used are selected by the researchers. Although the resulting system may be effective and efficient, this "hand-coding" method requires large amounts of time and deep vision expertise for building model-based vision systems.

Quite often, a geometrical model of the object is available which represents the three-dimensional shape information by means of polyhedra, generalized cylinders, or other

0018-9219/88/0800-1016$01.00 © 1988 IEEE

primitives. Given such an object model, visual recognition of an object amounts to determining its attitude (position and orientation) in space by using its various features which are observable in the images. In this view, one can imagine a generic model-based vision system which, given an input image or other sensory data, recognizes an object in it by means of a geometric reasoning mechanism which can deduce possible object attitudes from apparent object features. The historical and pioneering vision system by Roberts [11] can be viewed as such a generic approach. It reduced the problem of object matching to that of estimating the parameters of transformation (rotation, translation, size, and projection) by minimizing a matching error between model vertices and image joints.

Grimson and Lozano-Perez [12] have formulated the problem of object localization measurements (such as position) within a generic hypothesize-and-test search paradigm. When matching a set of observed surface points with a set of polyhedral object models, the possible matching pairs are expanded as a search tree. The matcher prunes this tree by using relational constraints between pairs of measurements which the object models impose if the matching is correct so far. The method has been applied to 2-D and 3-D object recognition using sparse range, touch, and orientation sensory input.

Probably, however, the most representative effort toward domain-independent model-based vision systems is ACRONYM by Brooks [13]. ACRONYM takes models of objects represented by generalized cylinders and their spatial relationships. Recognition or matching of the models to an input image is performed by using a symbolic algebraic reasoning system which reasons about projection and relational constraints on geometry. ACRONYM has succeeded in recognizing airplanes in aerial images.

When performing matching, a generic domain-independent model-based system relies on a generic reasoning mechanism: numerical optimization of some matching criterion, tree search by hypothesize-and-test, or constraint satisfaction by symbolic reasoning. The system uses the object model interpretively, that is, the knowledge is extracted from the model and transformed into an execution strategy at run time. As a result, the system may not be most efficient for the particular object in hand. This is a necessary price that an interpretive method must pay for its generality and flexibility.

One method for increasing efficiency is compilation. That is, the relevant knowledge in the object models is extracted and compiled into an object recognition strategy off-line so that as little computation as possible is spent at run time. Interestingly enough, we can regard some of the earlier vision work as examples of compilation. The generalized Hough transform by Ballard [14] and the direction coding method by Yoda, Motoike, and Ejiri [15] can be regarded as compiling the object shape in the appropriate transform so that the recognition reduces to peak finding in a histogram. However, these methods have limited applicability.

Bolles, Horaud, and Cain used a "local-feature-focus" recognition strategy for recognition of 3-D objects in a jumble [16]. The method involves selecting a class of "focus" features of similar shape on the object. Matching begins with the "focus" features. In selecting appropriate features for the strategy, they precomputed various feature values from a given CAD model of objects.

Goad [17] presented one of the first and most systematic methods for automatic generation of object recognition programs based on compilation. His method compiles visible edge of an object into an interpretation tree. Each branch of the tree is constructed to execute three stages: prediction, observation, and back-projection. In the prediction stage, a model edge is extracted from the node based on the current hypothesis of viewer direction, and the position and orientation of its projection in the image is predicted. In the observation stage, the list of image edges is checked to see whether any has the predicted qualities. In the back-projection stage, if an edge with predicted qualities was found in the prediction stage, then the match is extended to include this edge, and the measured position and orientation of the edge are used to refine the current hypothesis as to the location of the camera. During the compilation mode, stages and nodes which will become unnecessary at run time are detected and pruned. Various conditions and data structures to be used at run time are also computed. This way, much of the computation at run time is saved. The method for selecting the most efficient sequence of edges to be examined was not discussed, however.

Koezuka and Kanade [18] constructed an interpretation tree automatically from a model of a polyhedral object by using parallel edges as initial features to be used in matching. Parallel line features remains parallel over a wide range of viewing directions, but the direction and distance between a pair of lines still provide strong constraints on viewer direction, and thus can be used to create a reliable and efficient interpretation tree.

Ikeuchi [19] presented a compilation technique based on visible regions. The system classifies various views into aspects, where aspects are defined as topologically equivalent views. The interpretation tree is constructed so that an unknown view will be classified into an aspect and then its attitude will be determined precisely. He developed rules to generate an interpretation tree from a geometric model. The rules determine what kinds of features should be used in what order and generate an interpretation tree.

Automatic generation of recognition programs by compilation of object models tries to combine the merits of a hand-written system and those of a generic interpretive system. A general compilation program generates a tailored special program from a given 3-D model. A large portion of the computation needed for using the object model, such as analysis of the best recognition strategy, analysis of occlusion, and estimation of expected feature values, can be done at compile time, and the result can be compiled into the special program. In some cases, the object properties might be represented in the flow of the program rather than its data structure. As a result, the compiled special program to run on-line can be more efficient than generic programs. Yet, since the program is generated automatically, the development time could be reduced.

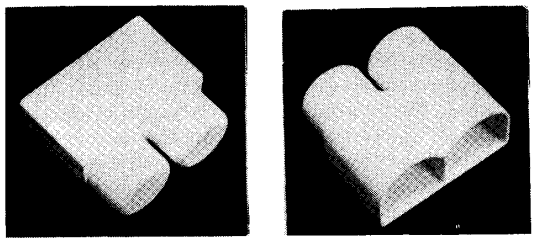
This paper discusses issues and techniques for automatic generation of recognition programs by compilation. The discussion will be based on our current approach [19]–[21], whose key steps are object modeling, sensor modeling, prediction of object appearances, strategy generation, and program generation. An object model describes geometric and photometric properties of an object to be recognized. A sensor model specifies the sensor characteristics in pre-

dicting object appearances and variations of feature values. The appearances can be systematically predicted and grouped into aspects, and a recognition strategy is generated in the form of an interpretation tree from the grouping and the predicted feature values. Finally, the strategy is converted into an executable program by using object-oriented programming. A major emphasis of this paper is that sensors, as well as objects, must be modeled explicitly in order to achieve the goal of automatic generation of reliable and efficient recognition programs. First, we will present our initial system for generating an interpretation tree for bin-picking using photometric stereo. This example system will introduce various concepts as well as issues.

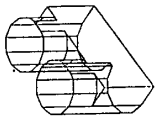
II. Compiling an Object Model into an Interpretation Tree

This section will present an example of compilation of a geometric object model into an interpretation tree. The example task is a bin picking task. The object shown in Fig. 1(a) and (b) is the sample object and the scene in Fig. 1(c) is a typical image from which the object must be recognized and located.

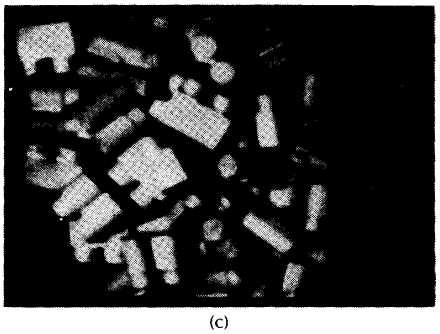
A 3-D object can give rise to an infinite number of 2-D shapes in an image. These apparent 2-D shapes of a 3-D object, however, can be grouped into a finite number of equivalence classes, called *aspects* [22], where each aspect contains the apparent shapes arising from the same set of visible features of objects, such as faces, edges or vertices,



(a)



(b)



(c)

Fig. 1. Object recognition example. (a) Photo of a sample object. (b) Geometric model of the object. (c) Sample scene.

with the same topological relationships among them. We can therefore distinguish two types of shape changes: one is shape change between aspects (called aspect change); the other is shape change within an aspect (called linear change).

We will divide a recognition stage into two substages: aspect classification stage and linear-change determination stage. Thus, our goal is to automatically develop an interpretation tree which first classifies the input image of an object into one of the possible aspects, and then calculates the exact attitude of the object. It should be noted that different features are most likely required to resolve aspect changes than are required to resolve linear changes. Also, in resolving linear changes, appropriate techniques and features might be different depending on the particular aspect in which the linear change occurs. Thus, it is essential for both competence and efficiency to compile a geometrical model into an interpretation tree so that the most appropriate features among all the available features are used at each aspect classification stage and linear-change determination stage.

A. Extracting Aspects

For object recognition purposes, aspects are defined as topologically equivalent classes with respect to the object features "visible" to the sensors. For example, aspects have been defined by visible lines [22], [23], by visible vertices [24], and by occluding boundaries [25]. As will be explained later, our example system will use photometric stereo [26], [27] as the major sensor. Photometric stereo determines surface orientations by illuminating the surface with three or more light sources. Thus we categorize the aspects based on visible faces for photometric stereo.

Viewer or camera configurations, which result in various appearances of a 3-D shape, consist of six degrees of freedom in general: three degrees of freedom in translation, and three degrees in rotation. However, in most industrial vision problems, such as bin picking, we can assume orthographic projection as the first approximation. This is because the camera is set up at a relatively far and fixed distance to the objects and the objects are imaged only near the center of the camera's field of view. This means that the three translations are either known or constant. Since a rotation around the camera optical axis results in a rotation of the image, not in a significant change of appearances, the two degrees of freedom which specify the viewer direction are the dominant ones in determining aspects.

We will thus explore changes of apparent shapes over the set of possible viewer directions. A viewer direction can be described as a point of the Gaussian sphere which is placed at the center of an object. Each apparent shape (thus, each point on the Gaussian sphere) can be characterized by those faces visible from that viewer direction. Suppose we have n faces, $S_1, S_2, \cdots, S_n$, where one face corresponds to either a planar surface or a curved surface which will be detected as a single surface patch in photometric stereo. Let the variable X_i denotes the visibility of face S_i, that is

$$X_i = \begin{cases} 1 & \text{face } S_i \text{ is visible;} \\ 0 & \text{otherwise.} \end{cases}$$

An n-tuple $(X_1, X_2, \cdots, X_n)$represents a label of an apparent shape in terms of face visibility. This label will be referred

to as a *shape label*, and we can characterize each viewer direction with this label.

The set of contiguous viewer directions that have the same *shape label* forms an *aspect*. There are two methods to enumerate possible aspects of a given object: an analytic method and an exhaustive method. Though precisely finding possible aspects by an analytic method is relatively easy for convex polyhedra, it becomes more complex and less tractable for concave objects and curved objects. For practical purposes, we favor the exhaustive method, in which we generate apparent shapes of the object under various viewer directions sampled on the Gaussian sphere, examine shape labels of the generated shapes, and classify them into aspects.

We tessellate the Gaussian sphere by using a geodesic dome which subdivides the sphere into many small spherical triangles [28], each of which reprsents a sampled viewer direction. At each sampled viewer direction, an apparent shape of the object is generated using a geometric modeler, and its shape label $(X_1, X_2, \cdots, X_n)$ is calculated. This way, all possible shape labels are calculated, evenly sampled over all possible viewer directions, and grouped into aspects.[1] Finally, a representative attitude is selected for each aspect chosen from the set of viewer directions which belong to the same aspect. Usually, the viewer direction which results in an appearance with the largest sectional area is selected as the representative attitude. The representative attitude is used to calculate the representative values of features to be used in discriminate aspects and to calculate the precise attitude within an aspect.

Fig. 2 shows the result of applying this method to the object of Fig. 1. The sample object has twelve component faces. Fig. 2(a) shows the geometric model of the object. Fig. 2(b) shows the Gaussian sphere tessellated into sixty small triangles using the one-frequency dodecahedron. Sixty different shapes corresponding to the tessellated triangles are generated as shown in Fig. 2(c), where the faces surrounded with bold lines are detectable using photometric stereo. Because of the geometry of the light sources, some faces visible to humans are not detectable by photometric stereo. Fig. 2(d) shows the larger eight component faces used for the shape label among the twelve faces of the object. Smaller regions under a certain threshold are regarded as nondetectable. Fig. 2(e) lists the five aspects obtained as the result of classification of the sixty appearances in Fig. 2(c). For aspects 1 to 5, five representative attitudes are generated as shown in Fig. 2(f).

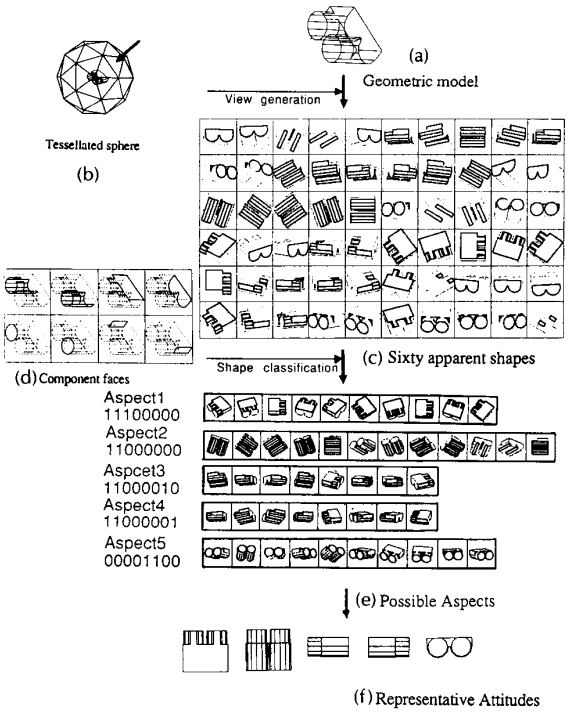


Fig. 2. Extraction of aspects. (a) Geometric model of an object. (b) Gaussian sphere tessellated into sixty triangles to represent viewer directions sampled. (c) Sixty appearances. The faces surrounded with bold lines are detectable by photometric stereo. Because of the geometry of the light sources, some faces visible to humans are not detectable by photometric stereo. (d) Eight component faces to be used for the shape label. (e) Five aspects obtained as the result of classification of sixty appearances by the shape label. Eight digits at each aspect represent the shape label of the aspect. The visible faces are indicated under each aspect. For example, faces 1, 2, and 3 are observable in aspect 1, whose shape label is 11100000. (f) Five representative attitudes.

B. Sensors and Features

This section will give a brief description of the sensors we used and then present how the aspects are described in terms of available features. In our example system, the major sensor is photometric stereo which provides surface orientations. In addition, we use dual photometric stereo to obtain depth information and an edge detector to locate fine features of objects.

Photometric Stereo [26]: Photometric stereo takes multiple images of the same scene from the same camera position under different illumination directions in order to determine surface orientations (p, q) based on differences in brightness. Since different images are taken from the same point, there is no disparity between the images as there is with binocular stereo, so no correspondence problem has to be solved. This makes photometric stereo very fast. By using photometric stereo we generate a needle map, which is a distribution of (p, q) over the image. From the distribution of (p, q) over a region, we can recover various geometric features of visible regions such as area and moment.

Dual Photometric Stereo [27]: Although photometric stereo can determine the surface orientation very fast, it cannot determine absolute depth. In order to determine absolute depth fast, we use the dual photometric stereo method which uses two camera and three light sources, and matches a needle map of the left camera photometric stereo and the other from the right camera. A needle map obtained by photometric stereo can be easily segmented into isolated regions using uninterpretable regions around objects. We will establish the correspondence between left regions and right regions by using three characteristics: vertical mass center positions, average surface orientation over the region, and region area. Since our method only checks correspondences between regions, the number of combina-

[1]Note that although possible omission of aspects may occur in this method, it would not hurt anyway because the aspects of narrow areas in the Gaussian sphere are seldom observed.

tions necessary to examine is small, so the system is very rapid. A depth map is obtained from each region's disparity and average surface orientation. The depth map will be used to determine the target region from which the recognition process begins.

Edge Detector: We also use an edge map which is obtained by differentiating brightness distributions and grouping edge points into line segments with a line. The edge map will be used as a supplementary source when the system cannot determine the object attitude completely using features from a needle map.

In summary, an input scene is described by a needle map, a depth map, and an edge map by using these three sensors. In describing aspects, we can use features available from these three representations of the input scene. Since surface orientation is obtained as the needle map, we can actually recover 3-D features of the original faces, instead of 2-D projected features, such as the area, shape, etc. Let (p, q) be the surface gradient of a region. Then, the matrix

$$T = \begin{bmatrix} \sqrt{1+p^2} & pq/\sqrt{1+p^2} \\ 0 & \sqrt{1+p^2+q^2}/\sqrt{1+p^2} \end{bmatrix} \quad (1)$$

gives the affine transformation to map from the 2-D image coordinates to the 2-D coordinates on the 3-D face. This transformation can be used to recover the 3-D features of the original face from 2-D features of the corresponding region in the image.

Each aspect is now described by using various features obtainable from the above sensors. In our example, features used include face moment, face relationships, face shape, edge relationships, extended Gaussian image (EGI), and surface characteristic distribution. Each of these features is discussed below.

Face Moment: The face moments are represented by the two principal moments, m_{xx} and m_{yy}, of a face.

Face Relationship: An object often appears as multiple separated regions in the image. This is especially true with nonconvex objects under photometric stereo. The relationships between regions are very useful features. For each visible face, relative position information is stored which tells where each of the other visible faces should appear in the aspect. The relationship is represented by a vector with respect to the local face coordinate system. The origin of the local coordinate system is the mass center. The z axis and x axis agrees with the surface orientation, the direction of the maximum moment, respectively.[2]

Face Shape: The face shape is described by the radial distance function $d = d(\theta)$, where d is the radial distance from the mass center of the face to its boundary, and θ is the angle from the x axis of the local coordinate system.

Edge Relationships: In some cases the needle map alone cannot determine the object attitude uniquely. In this case some of the prominent edge information is useful to reduce ambiguity. The locations of edges are stored by the start and end positions. As in other face information, these positions are represented in the local face coordinate system. When applying this information, the expected position of an edge is computed using the inverse of the affine transformation, (1) derivable from the surface orientation of the face. Then, the narrow stripe region connecting the converted start and end positions can be searched on the edge map to see whether or not there is actually an edge.

Extended Gaussian Image (EGI): An EGI of an object is nothing but a spatial histogram of its surface orientations [29]–[31]. The EGI has two nice properties. One is that the EGI is invariant to translation of the object, and the other is that when an object rotates, its EGI also rotates in exactly the same manner while not changing the relative EGI mass distribution.

Surface Characteristics Distribution: A surface patch can be characterized as planar, cylindrical, elliptic, or hyperbolic. The characteristics are defined in terms of the Gaussian curvature and the mean curvature [32] and are independent of the viewer direction and the rotation. Distribution of the characteristics are stored with respect to the local coordinate system, and are used in a similar way and for a similar purpose as prominent edges.

For each aspect extracted for an object, the features listed above are calculated.[3] The descriptions of all aspects thus obtained are now used to construct the interpretation tree with which input scenes will be recognized.

C. Generating an Interpretation Tree

An interpretation tree consists of two parts: the first part is used for classifying the input scene into one of the aspects, and the second part is used for calculating the exact attitude of the object within the aspect determined. In this subsection we will create an interpretation tree for our example object. First we discuss how to generate the classification part of the interpretation tree. The basic idea is a recursive examination of features of aspects to see whether or not they can discriminate a group of aspects into sub-groups of aspects. That is, starting with all the aspects as a single group, we check if a certain feature can divide the group into subgroups. If so, a branch node is created which registers the features as the discriminator and the subgroups divided are connected as descendant nodes. Then for each subgroup (descendant node), the process is applied recursively until a subgroup is made of a single aspect or equivalently a single aspect is assigned to a leaf node.[4]

We have used the following seven features for discrimination. In order of preference, they are: the original face moment, the original face shape, the extended Gaussian image (EGI), the surface characteristic distribution, the edge distribution, the region distribution, and the relationship between a particular edge and a particular surface characteristic distribution.

As an example, we apply this method to the object shown in Fig. 1(a). The object has five aspects, shown in Fig. 2(e), so the start node contains a group of five aspects, {$S1$, $S2$,

[2]This local coordinate has 180 degree ambiguity with respect to the x-axis direction. Also if the region has no unique maximum moment direction, for example, a circular region, only the direction of x axis is defined arbitrary. In this case, only the distance between the two regions is stored.

[3]An aspect represents a group of object appearances, while an aspect component represents a group of 2-D faces (or a group of several 2-D faces, if they have C^1 continuity across the connecting edges). Features are calculated at each aspect component.

[4]Actually, as an initial stage of the project, a "skeleton" of a tree was predesigned by considering the "distances" among aspects, and the decision as to whether or not a feature can divide the aspects at the node was made by human. For more details, see [19]. This human-assisted decision process has since been converted to an automatic decision process.

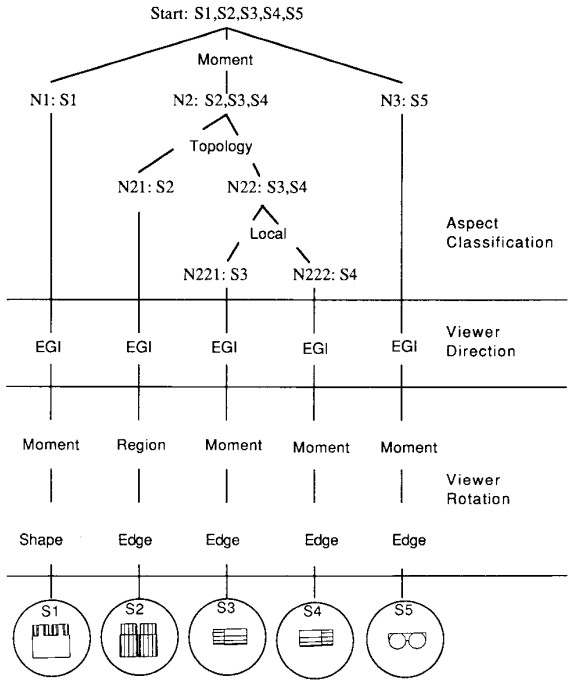


Fig. 3. Interpretation tree.

$S3, S4, S5\}$[5] (see Fig. 3). Since the original face moment can divide the aspect groups into three subgroups, $\{S1\}$, $\{S2, S3, S4\}$, and $\{S5\}$, it is adopted as a discriminator at the starting node, and three descendant nodes, $N1$, $N2$, and $N3$ are generated from the start node. Since node $N1$ and node $N3$ contain only one aspect, $S1$ and $S5$, respectively, the generation process terminates at these nodes. On the other hand, node $N2$ contains three aspects, so further processing is applied to the node. Neither the original face shape, the extended Gaussian image, the surface characteristic distribution, nor the edge distribution can not discriminate the aspect group $\{S2, S3, S4\}$. Since the region distribution divides the aspect group into two subgroups, $\{S2\}$ and $\{S3, S4\}$, this feature is adopted as a discriminator for node $N2$, and two descendent nodes, $N21$ and $N22$ are generated from $N2$. Node $N22$ still contains two aspects, and requires further processing. Because $S3$ and $S4$ have a different internal structure of regions, the region distribution feature is adopted as the discriminant to produce two nodes, $N221$ and $N222$. Now the complete aspect classification part of the interpretation tree has been obtained.

Once the aspect classification part is constructed, we will move on to generation of the part of the interpretation tree which determines the viewer direction and rotation. If a feature can reduce some of the remaining freedom in the viewer direction and rotation, it will be adopted into the tree. The decision as to whether or not a feature can reduce the freedom was made by a human at this point.[6]

[5]More precisely, one aspect component, having the largest area, is selected among aspect components of each aspect as the face from which recognition process begins. Thus, the later stages examine various features of the selected aspect components.

[6]This human-assisted decision process has since been converted to an automatic decision process.

We have used the following eight features for determination of the linear shape change. In order of preference, they are: the mass center of EGI distribution, the EGI, the position of observable region distribution, the inertia direction of original face, the original face shape, the position of the surface characteristics distribution, the position of the edges, and the position of the edges with respect to the position of the surface characteristics distribution.

The viewer direction and rotation are determined for each aspect using the most effective feature at each step. The selection depends on the aspect and the stage of the determining process. As an example, we will consider the case of node $N21$ or aspect $S2$. The other cases can be treated in the same way. Aspect $S2$ has two observable regions of cylindrical surfaces. The EGI mass center can determine viewer direction. Theoretically, the EGI distribution could have determined the viewer direction and the rotation uniquely in this aspect, but due to noise it would have been very unreliable. Thus, we will use other features to determine the viewer rotation.

Since aspect $S2$ has two observable regions, the region distribution feature is applicable and can constrain the viewer rotation up to two directions (up or down). Neither the moment direction, original face shape, nor surface characteristic feature can disambiguate one of the two remaining possibilities. However, the edge distribution feature can do. As a result, the EGI mass center, region distribution, and edge distribution have been adopted into the tree in this order. Fig. 3 shows the final interpretation tree obtained.

D. Applying the Interpretation Tree

In recognizing objects at run time with the interpretation tree created, the system uses three kinds of feature maps: an edge map, a needle map, and a depth map as shown in Fig. 4. An edge map is obtained by differentiating the camera intensity image. Each of two photometric stereo sensors, left and right, produce a needle map using three intensity images corresponding to the three lighting conditions. A depth map is constructed by the dual photometric stereo method [27]. An important advantage of these three maps is that they are registered in the same coordinate system; that is, all pixels having the same $i - j$ pixel coordinates correspond to the same physical point.

Our bin-of-parts example scene contains many instances of the object, while the interpretation tree specifies how to recognize a single object. Therefore we have to select a portion of an image where the interpretation tree is going to be applied. For this purpose, we choose the highest region (i.e., the region closest to the camera) as the target region to be interpreted. The interpretation tree extracts necessary features from the region. These features will be transformed and compared with the aspect model according to the procedures contained in the interpretation tree. Based on the decisions at each node, the target region is classified into one of the aspects, and then the precise attitude and position are determined.

Fig. 5 illustrates how the interpretation proceeds for the case of Aspect 2. The arrow in the picture (b) indicates the target region. According to the interpretation tree, the face moment of the region is to be calculated by using the shape and size of the region together with its spatial surface ori-

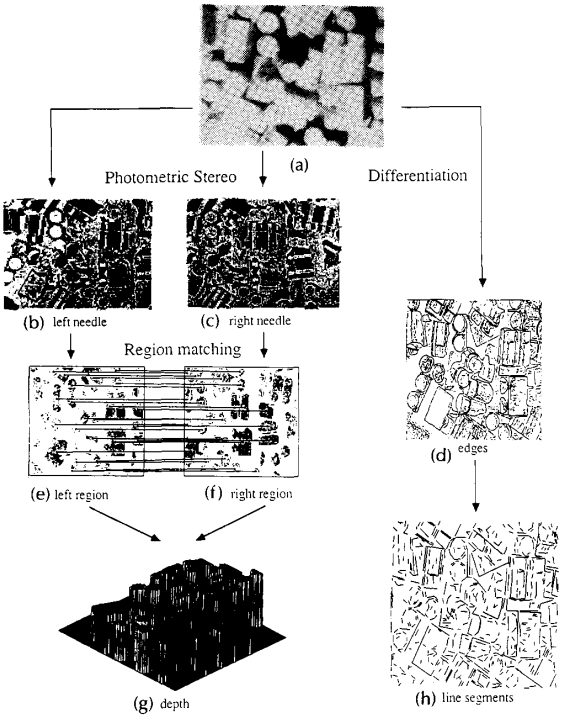


Fig. 4. Basic vision module. (a) Input scene. (b) Left needle map obtained by left photometric stereo. Surface orientations are depicted as small needles. (c) Right needle map obtained by right photometric stereo. (d) Edges obtained by Canny edge operator. (e) Left region map. A needle map obtained by photometric stereo can be easily segmented into isolated regions using uninterpretable regions around objects. Due to the arrangement of the light sources, a higher object projects shadows over the surrounding lower objects. Since the projected shadow areas becomes uninterpretable regions, a higher object is usually surrounded by uninterpretable regions. The left region map is obtained by segmenting the left needle map based on these uninterpretable regions. (f) Right region map. (g) Depth map obtained by dual photometric stereo. The correspondence between left regions and right regions is established by using three characteristics: vertical mass center position, average surface orientation over a region, and region area. A depth map is obtained by fitting a plane based on the depth at a mass center given from disparity and average surface orientation. (h) Line segments obtained by Miwa line finder.

entations from the needle map. The rectangle in Fig. 5(a) indicates the direction and magnitude of the moment value thus obtained. Based on the value of face inertia, the interpretation tree determines this region to belong to the group of aspects $S2$, $S3$, and $S4$.

The interpretation tree then distinguishes aspect $S2$ from the rest by determining whether a neighboring region exists having the same moment size and direction around the target region. The interpretation tree tries to find such a region. In this case it succeeds, as shown in Fig. 5(c). From this, the interpretation tree determines that the target and the neighboring regions come from the same object and belong to the aspect $S2$.

The rest of the processing is to verify the determined aspect and to calculate accurate object attitude, again following the interpretation tree. Comparison of the EGIs from the model and the scene determines the viewer direction (Fig. 5(d)). Next, the viewer rotation around the viewer direction must be determined. From the relationship between the two regions, the viewer rotation can be determined up to two directions (180° apart) (Fig. 5(f)), but more detailed analysis of the edge distribution is necessary to determine it uniquely. The interpretation tree examines the edge distributions in the two stripe regions which are predicted for the two possible rotations. In this way, by following the interpretation tree as shown by the bold line (Fig. 5(e)), the object has been recognized and its attitude has been calculated uniquely (Fig. 5(g)). Fig. 5(h) presents the recognition result by projecting the object model with the detected attitude on top of the depth map.

For different aspects, other parts of the interpretation tree are similarly executed. When the interpretation tree has been executed on various regions in an image for another scene, the combined interpretation results look like Fig. 6, in which 10 instances of objects have been located successfully.

III. Toward Systematic Methods of Compilation

The system presented in the previous section has compiled the object model into a recognition strategy in the form of an interpretation tree, and the resultant interpretation tree was successfully used to recognize the object instances in a cluttered bin-of-parts scene. In the off-line compilation stage, it automatically derived distinctive aspects from a geometrical object model, built feature descriptions of aspects by calculating expected feature values from the object model, and then, based on those descriptions, generated an interpretation tree for classifying the aspects and determining the attitude within each aspect. At on-line run time, the interpretation tree has controlled the localization process by using the predesignated most appropriate features at each stage. The recognized object position and attitude could be used for such tasks as bin-picking.

Though successful and promising, the system raises several important issues to be solved in order to develop a more systematic and general method of compiling recognition programs from models. We have found that one of the most crucial things is a more systematic way for modeling object appearances. So far, modeling has concentrated primarily on a geometric modeling of an object. However, the appearance of an object in an image, and the features of an object that can be reliably detected are determined not only by object properties, but also by sensor characteristics. The same object model in the same attitude can create different appearances and features when seen by different sensors. Edge-based binocular stereo reliably detects depth at edges perpendicular to the epipolar lines. Photometric stereo or a light-stripe range finder detects surface orientation and depth of surfaces which are illuminated and visible both by the light sources and by the camera.

Thus, in model based vision, it is insufficient to consider only an object model; it is essential to appropriately model sensors as well. Modeling sensors for model-based vision, however, has attracted little attention. In fact, the lack of explicit sensor models was the basic reason that the system in the previous section required human assistance. In order to make automatic and correct decisions, the system must

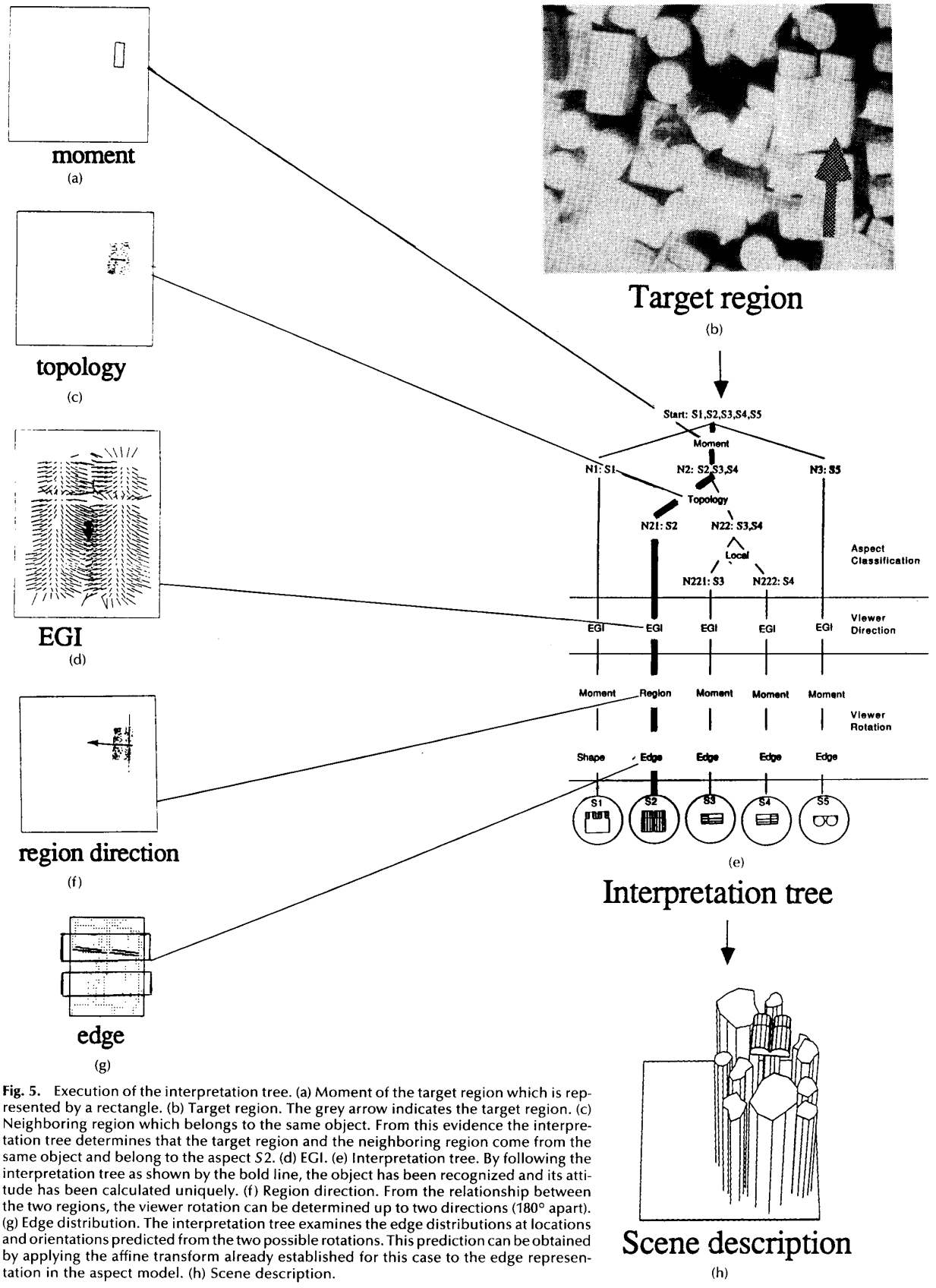
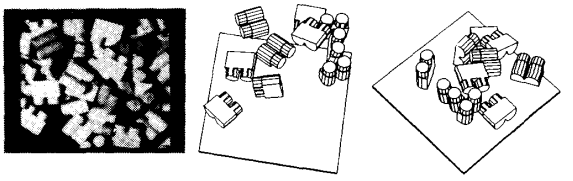


Fig. 5. Execution of the interpretation tree. (a) Moment of the target region which is represented by a rectangle. (b) Target region. The grey arrow indicates the target region. (c) Neighboring region which belongs to the same object. From this evidence the interpretation tree determines that the target region and the neighboring region come from the same object and belong to the aspect *S2*. (d) EGI. (e) Interpretation tree. By following the interpretation tree as shown by the bold line, the object has been recognized and its attitude has been calculated uniquely. (f) Region direction. From the relationship between the two regions, the viewer rotation can be determined up to two directions (180° apart). (g) Edge distribution. The interpretation tree examines the edge distributions at locations and orientations predicted from the two possible rotations. This prediction can be obtained by applying the affine transform already established for this case to the edge representation in the aspect model. (h) Scene description.



(a) (b) (c)

Fig. 6. Another interpretation result. (a) Input scene (top view). (b) Recognition result (front view). (c) Recognition result (side view).

correctly characterize object's appearances for the particular sensor in use, predict ranges of feature values, and develop a framework to convert those predictions into decision rules. In the following sections, we will discuss some of the issues toward this goal, including representation of sensor-object relationships, characterization of detectability and reliability of sensors, prediction of ranges of feature values, and generation of flexibile execution programs.

IV. Modeling Sensors

Different types of sensors are used in model-based vision. For our purpose, "sensors" are transducers which transform "object features" into "image features." For example, an edge detector detects edges of an object as lines in an image. Photometric stereo measures surface orientations of surface patches of an object. There are both passive and active sensors. Binocular stereo is passive, while a light-stripe range finder is an active sensor using actively controlled lighting. Table 1 gives a summary of various sensors in terms of what object features are detected in what forms.

Table 1 Summary of Sensors

Sensor	Vertex	Edge	Face	Active/Passive
Edge detector [33]	—	line	—	passive
Shape-from-shading [34]	—	—	region	passive
Synthetic aperture radar [35]	point	point/line	point	active
Time-of-flight range finder [36]	—	—	region	active
Light-stripe range finder [4]	—	—	region	active
Binocular stereo [37], [38]	—	line	—	passive
Trinocular stereo [39]	—	line	—	passive
Photometric stereo [26], [9]	—	—	region	active
Polarimetric light detector [40]	—	—	point	active

In addition to qualitative descriptions of a sensor, a sensor model must model two characteristics quantitatively: *detectability* and *reliability*. Detectability specifies what kind of features can be detected in what conditions. Reliability specifies the expected error in the value of a feature. Since these two characteristics depend on how the sensor is located relative to an object feature, we will first define a feature configuration space to represent the geometrical relationship between the sensor and the feature. Then, we will investigate the way to specify detectability and reliability over the space.

A. Feature Configuration Space

Whether and how reliably a sensor detects an object feature depend on various factors: distance to a feature, attitude of a feature, reflectivity of a feature, transparency of air, ambient lighting, and so forth. In most model-based vision problems, the attitude of a feature, that is, angular freedom in the relationship between a feature and a sensor, affects sensor characteristics most. For that purpose, we attached a coordinate system to an object feature and consider the relationship between the sensor coordinate system and the feature coordinate system. For example, for a face feature, we define a coordinate system so that the z axis of the feature coordinate system agrees with the surface normal and x-y axes lies on the face, but defined arbitrarily otherwise. For other features, we can define feature coordinates appropriately.

For the sake of convenience let us fix the sensor coordiante system and discuss how to specify feature coordinates with respect to it. The angular relationships from the sensor coordinate system to a feature coordinate system can be specified by three degrees of freedoms: two degrees of freedom in the direction of the z axis, and one degree of freedom in the rotation about the z axis. See Fig. 7(a).

We will define a sphere in which a feature coordinate system is specified as a point. Referring to Fig. 7(b), the direction from the sphere center to the point coincides with the z axis of the feature coordinate. The distance from the spherical surface to the point is determined by the angle of rotation (modulo 360°) around the z axis from the coordinate on the spherical surface. A point on the spherical surface represents a feature coordinate obtained by rotating the sensor coordinate around the axis perpendicular to plane given by the sphere center, the spherical point, and the north pole. The north pole of the sphere is made to correspond to the case when the feature coordinate is aligned completely with the sensor coordinate.[7] We will refer to this sphere as the feature configuration space.[8]

B. Constraints on Feature Detectability

Using the feature configuration space, we will represent in a general way the constraints on the attitude of a feature for it to be detected by a sensor. A sensor has two types of components in general: illuminators and detectors. In order for a feature to be detected by a sensor, it must satisfy cer-

[7]This representation will not create discontinuities around the north pole as opposed to the case in which Euler angles from the sensor coordinate frame to the feature coordinate frame are used to specify spherical points; this representation will instead create discontinuities at the center of the sphere and at the south pole. However, this is not harmful, because we mostly use the area around the north pole to discuss detectability and reliability.

[8]Note that this sphere is different from the Gaussian sphere used in the previous section. Previously, the Gaussian sphere represented the sensor coordinates (the viewer directions) with respect to the object coordinates and detectability of each feature was examined by an *ad hoc* method for each viewer direction. In contrast, here we are developing a tool to examine the detectability of a feature using the sphere to represent the feature coordinates with respect to the sensor coordinates. This tool will be applied to features of an object which is rotated with respect to the sensor coordinates.

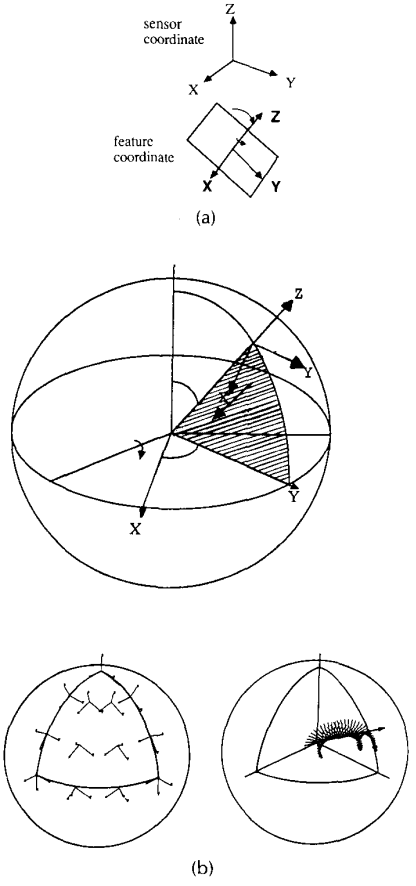


Fig. 7. Feature configuration space. (a) Relationship between sensor coordinate and feature coordinate. The feature coordinates can be specified by three degrees of freedom: two degrees of freedom in the direction of the z axis of a feature, and one degree of freedom in the rotation about the z axis of a feature. (b) Feature configuration space. One feature coordinate can be represented as a point in the sphere. The drawing at the bottom left depicts the coordinates corresponding to the points on the spherical surface, while the one at the bottom right depicts the coordinates corresponding to the points on one axis.

tain conditions on being illuminated by its illuminators and being visible from its detectors.

Once we define a local coordinate system on an object feature, we can compute configurations of a feature in which it is illuminated by each illuminator, and configurations in which it is visible by each detector. In this analysis, it should be noted that illuminators and detectors can be treated interchangeably. In [20] this concept was defined as generalized sources (G-sources). The illumination direction of an illuminator and the line of sight of a detector correspond to the G-source illumination directions, and both can be represented in the feature configuration space as a radial line from the sphere center. Also, illuminated configurations by an illuminator and visible configurations from a detector correspond to the G-source illuminated configuration, and both can be specified as a volume in the configuration space. Finally, we can obtain the constraints in which the feature is detectable by the sensor with AND and OR operations on illumination (line-of-sight) directions and illuminated (visible) configurations of all components of sensors.

Fig. 8 shows an example analysis of a face feature for a light-stripe range finder. A light-stripe range finder has two

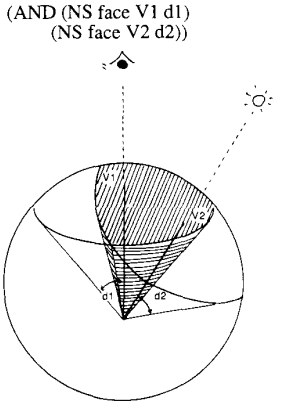


Fig. 8. Detectability constraints of a face for a light-stripe range finder.

G-sources (a TV camera and a light source): the direction denoted by $V1$ indicates the line of sight of the TV camera; $V2$ indicates the illumination direction of the light source. The illuminated configurations of a face are determined by the z axis (i.e., its surface normal), and are not dependent on its rotation. Therefore, illuminated configurations of a feature from a spherical cone whose axis is $V2$ and whose apex angle is $d2$. Similarly, the configurations of a feature visible from the TV camera form a spherical cone whose center direction is $V1$ and whose apex angle is $d1$. Since a light-stripe range finder detects the faces which are illuminated from the source and visible from the TV camera, the detectable configurations are the intersection of the two cones. Similarly we can analyze the detectable configurations of various features for various sensors. The results of the analysis are summarized in [20].

V. Modeling Appearances

Aspects have been defined as topologically equivalent classes with respect to the object features "visible" to the sensors. Classifying object appearances into aspects systematically raises several issues. First, since aspect is defined relative to sensors, the detectability of features by the particular sensors to be used must be incorporated. In the system of the Section II, however, we used the constraints of the surface visibility by the photometric stereo in an *ad hoc* manner. Now that we have developed a way to represent the detectable configurations of features, we can use it in generating appearances. Second, we will discuss how to represent object appearances and aspects in a systematic way. In the previous system, output from the geometric modeler is handled by a human-assisted process to analyze them and to generate a recognition strategy from them. This interactive process can handle any *ad hoc* representations. However, in the present system, a complete automatic process should handle the output and generate a recognition program. This requires a systematic representation of object appearances as well as aspects. Finally, it is useful to obtain

an estimate of the number of aspects in order to make sure that the recognition methods based on aspects are applicable to an object with a reasonable complexity. This section will discuss these three issues.

A. Appearance Generation from Constraints on Feature Detectability

To predict object appearances, we apply the constraints on feature detectability to each feature of the object. Each feature is detectable by the sensor if it satisfies the following two conditions:

1) None of the illumination (line-of-sight) directions are occluded by any other parts of the object;
2) The detectable configurations contain the configuration of the feature.

To check these conditions we use the constraints together with a geometric modeler. We rotate the object into a certain attitude to be examined, and then see whether its features satisfy the previous constraints.

Fig. 9 illustrates this process of predicting object appearances for a light-stripe range finder. Suppose an object is placed like Fig. 9(a). Fig. 9(b) shows the detectability constraints on a face for a light-stripe range finder. We will put this configuration space on each candidate face to examine whether the face is detectable. See Fig. 9(c). This amounts to checking the following conditions:

1) The light source direction is not occluded by other faces.
2) The line of sight of the TV camera is not occluded by other faces.
3) The local coordinate of a face, defined by the surface orientation (z axis) and the tangential plane (x-y axis), is contained in the detectable configurations.

Fig. 9(d) shows the result of this operation. The shaded area indicate those which satisfy the conditions and thus are detectable by the light-stripe range finder.

B. Describing Aspects

Appropriate descriptions of aspects must be defined so that they can be used in automatic generation of interpretation trees. The description of an aspect should include constituent appearances, a set of features extractable for the aspect, and the expected feature values. This description should have flexible and convenient forms for applying generation rules to them and for use in execution. We will represent aspects on frames by using a frame representation language, Framekit+, because it has a flexible structure and powerful demon facilities. Since an aspect is an abstract concept which represents a group of possible appearances, we will first consider how to represent each appearance in the frame. Then, we will represent aspects based on the representation of appearances.

A geometric modeler generates a possible appearance of an object under a given attitude. We will convert output data from the geometric modeler into representations in Framekit+. One appearance, for example *I0* in Fig. 10(a), is represented by one frame, which points to several appearance component frames representing visible 2-D faces,

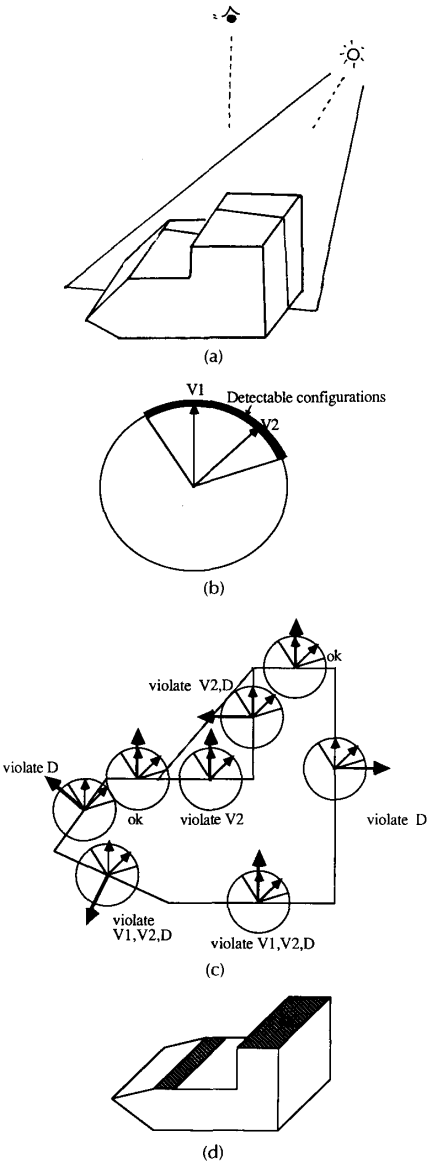


Fig. 9. How to use detectability constraints. (a) Light-stripe range finder. (b) Detectability constraints on a face for a light-stripe range finder. The constraints consist of detectable configurations and two G-source illumination directions, $V1$, $V2$. (c) Applying detectability constraints. (d) Detectable faces. The shaded areas indicate those which satisfy the conditions and thus are detectable by the light-stripe range finder.

IMAGE-COMP01, and *IMAGE-COMP02*.[9] Each frame corresponding to one visible 2-D face maintains various geometric properties of the face in slots. For example, face area and face moment are maintained in slots *AREA* and

[9]In this example, one 2-D face corresponds to one image component. If several 2-D faces have C^1 continuity across the edges, these faces are grouped and stored as one single image component. In this case, face area and face moment are calculated over the group of faces.

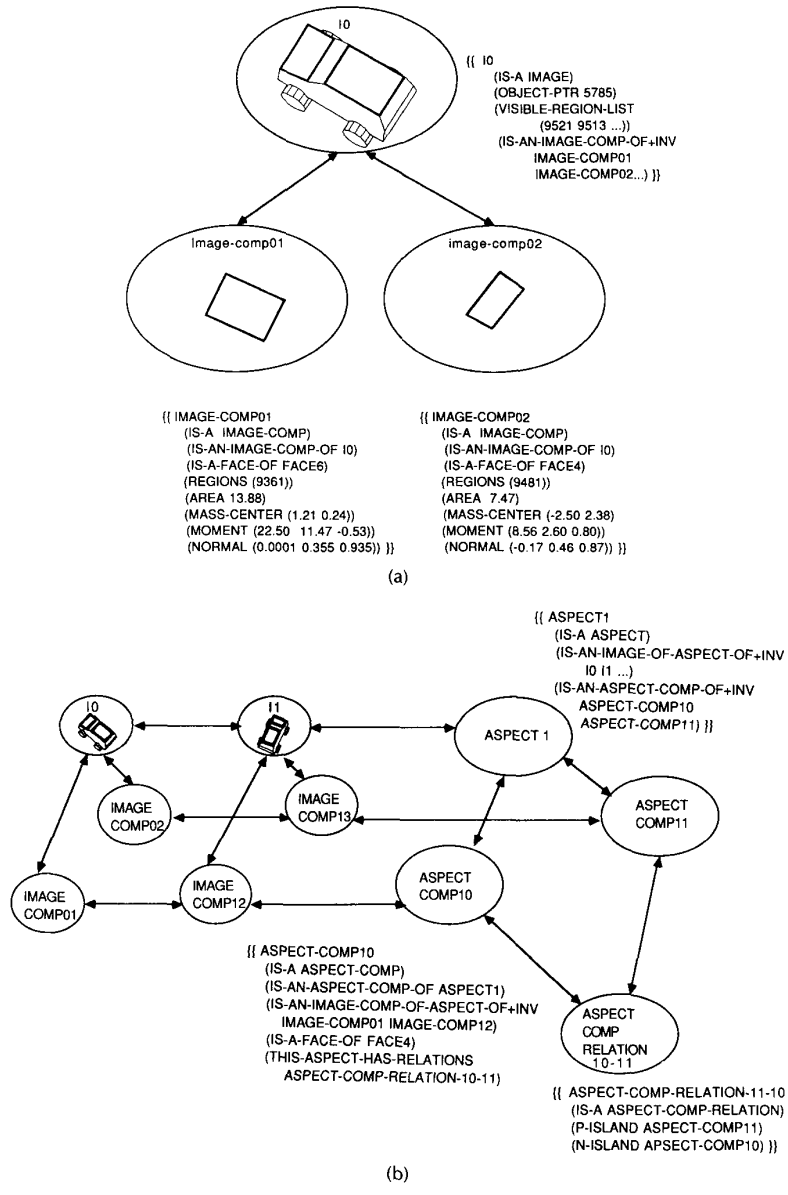


Fig. 10. Frame representation of aspects. (a) Image structure. Each image structure consists of a frame corresponding to an image and several frames corresponding to 2-D visible faces in the image. (b) Aspect structure. Each aspect structure consists of an aspect frame, aspect component frames, and aspect component relation frames.

MOMENT. The values of these features are obtained by using output data from a geometric modeler. Each frame representing a 2-D visible face has a backpointer to the 3-D face from which the 2-D face is projected. For example, the *IS-A-FACE-OF* slot of *IMAGE-COMP01* frame has a value *FACE6*.[10] An image structure consists of an image frame and image component frames.

Once image structures are represented, we can generate aspect structures in frames. Since an aspect is an abstract concept for a group of images (appearances), an aspect structure is similar to its constituent image structures. In order to construct aspect structures, shape labels of all image frames are examined one by one, where a shape label is the combination of visible 3-D faces as explained in Section II-A. The visible 3-D faces among a 2-D appearance can be retrieved by backpointers of 2-D faces to 3-D faces such as *FACE6* in *IS-A-FACE-OF* slot of *IMAGE-COMP01* frame, where *FACE6* is the frame name of a 3-D face of the object.

If an image structure cannot find any aspect structure with the same shape label among the already established ones, a new aspect frame is created together with aspect com-

[10]Each frame also contains array addresses of various geometric items such as 2D FACE, 2D EDGE, and 2D VERTEX in the database of the geometric modeler; for example, *9361* in *REGIONS* slot of *IMAGE-COMP01* frame. These allow us to access the original geometric data, if necessary.

ponent frames which correspond to image component frames: therefore, the aspect structure has the same structure as the image structure. Also, frames to represent the relationships between pairs of aspect components are created. If an image structure can find an aspect structure with the same shape label, the image frame is registered to the aspect frame as an instance and its frames of 2-D faces are registered to corresponding aspect component frames.

An example of an aspect structure is shown in Fig. 10(b). Aspect frame *ASPECT1* points to several aspect component frames, *ASPECT-COMP10, ASPECT-COMP11* with the *IS-AN-ASPECT-COMP-OF+INV* slot. It also points to its instance images *I0, I1* with *IS-AN-IMAGE-OF-ASPECT-OF+INV* slot, while its aspect component frame, *ASPECT-COMP10* points to its instance 2-D faces *IMAGE-COMP01, IMAGE-COMP12*. Frame *ASPECT-COMP-RELATION-11-10* is a relation frame which represents the relationship between *ASPECT-COMP10* and *ASPECT-COMP11*.

C. Estimating the Number of Aspects

An interesting and important question related to using aspects for object recognition is how many aspects an object will have. If this number is extremely large, it is impractical to classify an unknown scene into an aspect and then to determine the attitude within it.

One might think that the number of distinct aspects grows exponentially as the number of faces n in the object increases. However, the number of aspects grows much slower by a polynomial in n. To see this, let us consider the number of aspects $f_p(n)$ of a 2-D convex polygon with n edges seen in perspective. The sensor can be placed at any point on the 2-D plane. Each edge, when extended, divides the 2-D plane into two half-plane: when the sensor is located in the half-plane corresponding to the front side of the edge, then the edge is visible; otherwise it is invisible. Therefore, the problem of obtaining the number of distinct aspects $f_p(n)$ is equivalent to obtaining the number of regions into which n lines divide a 2-D plane. In fact, the visible/non-visible combinations attached to each region make up the shape label.

We can derive the formula of $f_p(n)$ by an inductive method. Suppose we add the nth line after $n-1$ lines have already been drawn. This new line intersects the existing $n-1$ lines at $n-1$ points (we are assuming the maximal case), which divide the new line into n segments. Each segment on the new line divides one old region into two regions. Thus, this operation adds n new regions. That is,

$$f_p(n) = f_p(n-1) + n.$$

By solving this, we obtain

$$f_p(n) = \frac{n^2 + n}{2} + 1$$

as the upper bound on the number of aspects of a 2-D convex polygon under perspective projection.

We can obtain the number of aspects $F_p(n)$ of a convex 3-D polyhedron with n faces in a very similar way. In this case, each face, when extended, divides a 3-D space into two 3-D half-spaces. We have to count the number of volumes that result when n planes divide a 3-D space. We can again use an inductive method. Assume that we have divided the 3-D space by $n-1$ planes. As shown in Fig. 11,

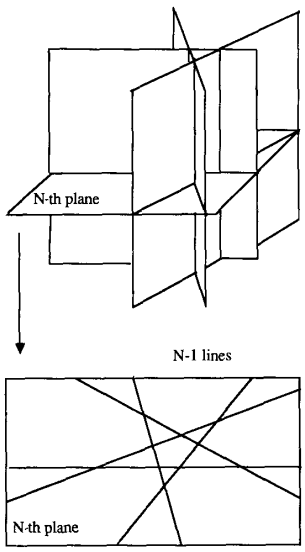


Fig. 11. Intersection of N planes.

if we add the nth plane, it intersects with the existing old $n-1$ planes, and generates $n-1$ intersection lines on it. Thus, on the nth plane there are $f_p(n-1)$ polygons, each of which divides an old volume into two. Therefore, addition of the nth plane adds $f_p(n-1)$ volumes:

$$F_p(n) = F_p(n-1) + f_p(n-1).$$

Thus

$$F_p(n) = n^3/6 + 5n/6 + 1$$

is the upper bound on the number of aspects of a 3-D convex polyhedron with n faces under perspective projection.

If we can assume orthographic projection, as we have done in our previous system, the number of aspects further reduces. Orthographic projection limits the possible sensor positions on the infinite sphere, and one occluding plane draws a great circle on the sphere to divide it into two hemispheres. We should count the number of regions on the sphere divided by n great circles. Since the nth great circle intersect with the previous $n-1$ great circles at $2(n-1)$ points and adds $2(n-1)$ new regions, we obtain the following recursive equation:

$$F_o(n) = F_o(n-1) + 2(n-1).$$

Thus

$$F_o(n) = n^2 - n + 2.$$

We notice that the upper bounds of the number of aspects grows as a quadratic function of the number of faces n. Moreover, for practical recognition purposes, n should be taken as the number of significantly large faces rather than including all the tiny faces.

Nonconvex polyhedra have more aspects, because aspects are determined not only by occluding planes due to faces but also occlusions due to edges. Suppose 3-D nonconvex polyhedron has n faces, o edges, and p vertices. In the worst case, we have to consider occlusion planes defined by all the pairs of edge and vertex: $o \times p$. Thus, $F_o(n + o \times p)$ provides the upper bounds. However, in reality,

the number of aspects must be much smaller, because a large fraction of pairs of vertex and edge either need not be considered or do not generate significant aspects to be taken into account for recognition.

VI. Predicting Ranges of Feature Values

In classifying an unknown scene into an aspect and determining its exact attitude, we need to select features with high reliability and discriminant power. The reliability and discriminant power of a feature depend not only on the norminal value that the aspect is expected to have, but also its expected variances over the aspect. For example, imagine a geometric feature whose nominal values for two aspects are calculated as 100 and 90 by a geometric modeler. If a sensor has an error range of plus/minus 1 for the feature, the feature is a reliable discriminator to separate the two aspects. On the other hand, if the error range of the sensor is plus/minus 20, the feature is not usable. Therefore, prediction of the range of values that a feature will take over an aspect is very important for strategy generation.

This section will discuss a method to predict the range of feature values. We must consider two levels of feature reliability. The first is that of sensory measurements by a sensor and this is obtained by analyzing the measurement mechanism of a sensor. In many cases, however, a geometric feature is derived from a set of sensory measurements and is used as a discriminator. We must also analyze the propagation of error from sensory measurements to a derived geometric feature in order to determine its reliability.

A. Errors in Sensory Measurements

As an example of predicting the range of sensory measurements, we will consider a depth measurement by a hypothetical light-stripe range finder. Let us assume that the main source of error in the depth measurement by this sensor comes from the ambiguity of the slit position on a surface due to the width of the light beam and angular errors in setting the light directions. The error model can be obtained analytically.

As shown in Fig. 12(a), let us denote the angular ambiguity of the light stripe by $\delta\theta$. The light is intercepted by an object surface, creating a slit pattern on it. The angular ambiguity $\delta\theta$ of the light direction results in ambiguity δy in the position on the surface.

$$\delta y = \frac{r\delta\theta}{\cos\alpha}$$

where r is the distance of the surface from the light source, and α is the angle between the light direction $\boldsymbol{S}$ and the surface normal $\boldsymbol{N}$. This positional ambiguity on the surface is observed as the slit position ambiguity (or "slit width") δi in the camera image. If β is an angle between the surface normal $\boldsymbol{N}$ and the viewer direction $\boldsymbol{V}$, then

$$\delta i = (\cos\beta)\delta y.$$

Finally, this ambiguity is transferred into the error of the depth measurement by triangulation. For simplicity, if we assume orthographic projection for the camera, the ambiguity in the image δi creates distance error δz.

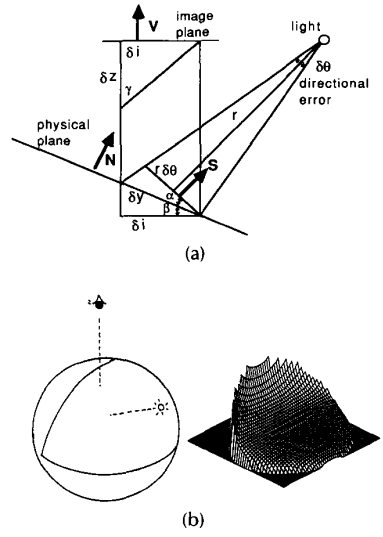


Fig. 12. Predicting uncertainty of a sensory measurement by a light-stripe range finder. (a) Detection mechanism. (b) Predicted uncertainty distribution of a depth measurement over the configuration space.

$$\delta z = \frac{\delta i}{\tan\gamma}$$

where γ is the angle between $\boldsymbol{V}$ and $\boldsymbol{S}$.

In total, by representing the angles α, β, and γ in terms of $\boldsymbol{V}$, $\boldsymbol{N}$ and $\boldsymbol{S}$, we obtain

$$\delta z = \frac{\cos\beta}{\cos\alpha\tan\gamma} r\delta\theta = \frac{(\boldsymbol{N}\cdot\boldsymbol{V})(\boldsymbol{S}\cdot\boldsymbol{V})}{(\boldsymbol{N}\cdot\boldsymbol{S})\sqrt{1-\boldsymbol{S}\cdot\boldsymbol{V}}} r\delta\theta.$$

Since r is roughly constant, the error distribution of this light-stripe range finder over the detectable area is governed by the factor $((\boldsymbol{N}\cdot\boldsymbol{V})(\boldsymbol{S}\cdot\boldsymbol{V}))/(\boldsymbol{N}\cdot\boldsymbol{S})\sqrt{1-\boldsymbol{S}\cdot\boldsymbol{V}}$. Fig. 12(b) plots this function.

B. Reliability of Geometric Features

Usually sensory measurements, such as depth detected by a sensor are further converted into object features such as area and inertia of a face. This process involves grouping pixels into regions, extracting some feature values and transforming them into another. Modeling the error generation and propagation in this process is difficult in general, but as an example of predicting the reliability range of a geometric feature, let us consider an area feature of a face detected as a region by our hypothetical light-stripe range finder. Fig. 13 shows the conversion process from depth values to the area of a face. The process includes three parts: obtain the area of the corresponding region in the image, compute the surface orientation of the region, and finally convert the image area into the surface area by the affine transform determined by the surface orientation. We will analyze how the errors are introduced and propagated in these three parts.

Suppose that a surface under consideration has the real area A and the surface orientation β (angle between the surface normal and the viewing direction). It should create a region of size n pixels where

$$n = A\cos\beta.$$

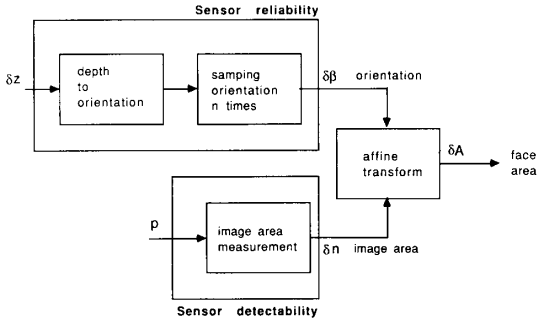


Fig. 13. Conversion process from a depth value to an area size of a face.

However, because of the imperfect detectability of the sensor, the sensor fails to find some of them, and the measured area will be different from the nominal area n. Let P denote the probability of detecting this surface.[11] Then, the process of measuring the area by sampling n pixels can be modeled by a binomial distribution with mean nP and variance $nP(1 - P)$. Assuming two standard deviations, the discrepancy in area measurement will be

$$\delta n = n - (nP - 2\sqrt{nP(1-P)}) = n(1-P) + 2\sqrt{nP(1-P)}.$$

Another error is also introduced in obtaining the surface orientation β from measured depths due to the sensor error δz. If we estimate the surface orientation at a pixel by differentiating depths of neighboring pixels, then the error will be $\cos^2 \beta \delta z$. However, since we have roughly n pixels in the region, the surface orientation will be averaged, reducing the error by a factor $\sqrt{n}$. Thus

$$\delta\beta = \frac{\cos^2 \beta}{\sqrt{n}} \delta z.$$

Finally, the estimation of the area of the face, $A + \delta A$, is obtained by converting the image area into 3-D space

$$A + \delta A = \frac{n + \delta n}{\cos(\beta + \delta\beta)}.$$

Thus, assuming that $\delta\beta$ is small, we see that

$$\delta A = A(1-P) + 2\sqrt{\frac{AP(1-P)}{\cos\beta}} + A \tan\beta\delta\beta$$

$$= A(1-P) + \sqrt{\frac{A}{\cos\beta}}\left(2\sqrt{P(1-P)} + \frac{\sin 2\beta}{2}\delta z\right).$$

In this way, we can predict what deviations from the nominal value of the area feature should be expected once we model the sensor and know its intrinsic detectability P and reliability δz.

C. Applying the Sensor Model to Aspect Structures

By using sensor model, we can predict the ranges of various feature values at each aspect. At each image, since a nominal value of a feature and its configuration with respect to sensor coordinates are given, we can predict the range of the feature value for each 2-D face of the image by using the formula described above. Then, the range of the feature value at an aspect component is obtained as a sum of ranges of the feature values over its registered image components which can be reachable along *IS-AN-IMAGE-COMP-OF-ASPECT-OF+INV*. The predicted range will be stored in the slot of an aspect component frame.

Fig. 14 shows slots for this purpose. For example, area ranges, moment ranges, and moment ratio ranges are calculated at each image components, *IMAGE-COMP01*, *IMAGE-COMP12* which can be retrieved along the link stored in slot *IS-AN-IMAGE-COMP-OF-ASPECT-OF+INV* of *ASPECT-COMP10* frame in Fig. 10(b). The sum of the ranges are stored in slot *AREA-VARIANCE*, *MOMENT-VARIANCE*, and *MOMENT-RATIO-VARIANCE* of *ASPECT-COMP10* frame. Similarly, feature ranges of aspect component relations, such as *DISTANCE-VARIANCE*, *MOMENT-ANGLE-P-TO-N-VARIANCE*, *SURFACE-ORIENTATION-ANGLE-VARIANCE*, are obtained and stored. These ranges of features will be retrieved by generation rules at compile time to generate an interpretation tree and by the execution process at run time in recognizing a scene.

```
{{ ASPECT-COMP10
 ....
  (AREA-VARIANCE (13.94 14.85 15.75))
  (MOMENT-VARIANCE (22.77 25.06 27.34))
  (MOMENT-RATIO-VARIANCE (0.53 0.65 0.76))
  (VISIBLE-EDGE-LIST ASPECT-COMP10-VISIBLE-EDGE-LIST)
 ....
  }}

{{ ASPECT-COMP-RELATION-11-10
 ....
  (DISTANCE-VARIANCE (5.04 5.38 5.69))
  (MOMENT-ANGLE-P-TO-N-VARIANCE (1.29 1.53 1.8))
  (MOMENT-ANGLE-N-TO-P-VARIANCE NIL)
  (SURFACE-ORIENTATION-ANGLE-VARIANCE (0.04 0.21 0.40))
 ....
  }}
```

Fig. 14. Slots for uncertainty ranges of features.

VII. Generating Programs

In this section, we will consider the final step of compilation of a recognition program: rule-based generation of a recognition strategy and conversion of the strategy into an executable program. As was in Section II, the recognition strategy is represented by an interpretation tree which is made of two parts: the first part for classifying the input scene into one of the aspects and the second part for calculating the exact attitude.

A. Recognition Strategy: Classification

Strategy generation for aspect determination can be regarded as a process which classifies a group of aspect components into subgroups of aspect components by applying classification rules recursively. At the beginning of the classification, a starting node is prepared, which contains all aspect components. We represent each classification stage as a node.

The following sixteen rules have been prepared. Each rule tries to classify a group of aspect components at a node into smaller subgroups of aspect components by using the designated feature. For example, rule A1 will classify a group

[11]This probability is defined as a function over the configuration space. While the constraints on feature detectability in Section IV-B specify only whether or not a feature can be detected, this probability of feature detection specifies how likely the feature can be detected. See [20] for more details.

of aspect components comparing area sizes of their subaspect components.

- **A1** face area
- **A2** face inertia
- **A3** face inertia ratio
- **A4** number of surrounding faces
- **A5** distances between surrounding faces and the face
- **A6** angles between inertia direction of surrounding face and the face
- **A7** surface orientation differences between surrounding faces and the face
- **A8** face area of surrounding faces
- **A9** face inertia of surrounding faces
- **A10** face inertia ratio of surrounding faces
- **A11** surface characteristics of the face
- **A12** surface characteristics of surrounding faces
- **A13** surface characteristics distribution of the face
- **A14** surface characteristics distribution of surrounding faces
- **A15** edge distribution of the face
- **A16** edge distribution of surrounding faces

The cost of calculations increases in order from A1 to A16; templates are required to calculate the features for the rules after A12. The order of preference in application is A1 to A16. Application of a rule proceeds in the following steps:

1) A rule selects a node which contains a group of aspect components.
2) It computes the threshold values of the feature to be used for classification from ranges of the feature values over the group of aspect components.
3) The rule classifies the group of aspect components into subgroups by using the determined threshold values.
4) It generates new nodes for the newly generated subgroups of aspect components.

Since the preference of rules has been set in order of A1 to A16, a node will be kept divided by the applicable and the most preferable rules.

If no more rule is applicable (i.e., no more nodes are dividable), application of rules A1 to A16 stops. Those nodes which contain only one aspect component are ready for the next stage of generating strategy for its attitude determination. At the termination, if there is a node which contains more than one aspect component and yet no rule is applicable to it, the parallel verification rule will be applied to the aspect components contained in the node. Since no further classification is possible, all possible aspect components in the node must be examined to see if any particular attitude is recognizable.

Once a tree is obtained by these rules, unnecessary branches are pruned. A rule may have generated a single child node from a parent node because the rule could not divide aspect components in the parent node. This rule-based generation of a strategy for classification has been implemented in OPS-5.

In applying this method in practice, we require a principle to choose a particular object and thus a particular region in an input image from which to start a recognition process. For a bin-picking task, we assume that the highest object is the best object to recognize. Under this assumption, there are two alternatives for a starting region:

1) The largest region of the highest object (conservative principle).
2) Any region of the highest object (aggressive principle).

Since the conservative principle begins with a set of the largest visible aspect components, one from each aspect, the interpretation tree will have a smaller number of nodes than the aggressive principle which will begin with a set of all aspect components. Therefore it will be more efficient in search than that for the aggressive principle, while it may be less reliable because the system may fail to find the largest region in the image.

B. Recognition Strategy: Attitude Determination

Once the aspect classification part of the interpretation is completed, the part for attitude determination is to be constructed next. This part is constructed for each aspect component of a leaf node to determine the precise attitude using the linear feature calculations. First the z axis direction of the object coordinate system is determined and then rotation angle around it is determined.

The following two rules are prepared for the determination of the z axis direction:

- **D1** mass center of EGI distribution
- **D2** extended Gaussian image.

If there is no partial occlusion of visible faces over all possible attitudes within the aspect and all visible faces are planar surfaces, the EGI mass center by rule D1 is used to determine the viewer direction. In other cases, matching of EGI by rule D2 is used.

Once the viewer direction is determined, the rotation around the axis is obtained next. One of the following six rules will be adopted by examining one by one to see if it constrains the freedom of rotation:

- **R1** position of detectable region distribution
- **R2** position of EGI distribution
- **R3** inertia direction
- **R4** EGI inertia direction
- **R5** position of the surface characteristics distribution
- **R6** position of the edges.

C. Executable Program

Once recognition strategy has been obtained with the necessary rules to be used at each stage, we have to convert the recognition strategy into an executable program. We are using the technique of object-oriented programming, because it simplifies to combine various elementary modules into a complete program.

Each node of the tree is converted into an "object" in object-oriented programming. We prepare a library of prototypical objects which will be used to execute matching operations between image regions and models according to rules [21]. Each rule has one corresponding prototype in the library. A necessary instance of a prototypical object to be adopted at a node is instantiated from the corresponding rule of the node. The descendant nodes which will receive a message from this node are also inserted in slot *EXECUTION-DESTINATIONS* of the object. Slot *EXECUTION-ARGUMENTS* contains the threshold value and other matching templates.

Our system uses photometric stereo to obtain region information, and uses a line extractor to obtain edge information. To represent these pieces of information, we also prepare two prototypical data objects in the object library: region object and edge object. We can make instance objects corresponding image regions and image edges. When instance objects are generated, the image properties of each region or edge are extracted from an image and stored in the corresponding slots.

Actual operations are executed as message passing between objects (nodes). The operation begins by sending an execution message and a target region to the starting node object. After that event, a chain of operations takes place by passing execution messages from object to object. When an object receives an execution message, the object executes a matching method which had been particularly adopted to the node. Since regions in the image are also implemented as objects, a message is sent to the target region to receive a necessary feature value from it.[12] Then, the matching method compares the value which is returned from the target region with the values in *EXECUTION-ARGUMENTS* slot. Based on the comparison result, the object determines to which object in *EXECUTION-DESTINATION* slot it should send an execution message next. This event is repeated until an execution message reaches one of the leaf objects of the tree. At this point, the tree determines the object attitude exactly.

Rule-based automatic generation of an interpretation tree has been applied to an object shown in Fig. 15(a), which has fourteen aspects as shown in Fig. 15(d).[13] The aggressive principle was chosen to select the starting region. The generation process generated the interpretation tree shown in Fig. 15(b). After the pruning operation, the result was an interpretation tree shown in Fig. 15(e). This pruning operation reduces the depth of the interpretation tree from 14 levels to 4. The obtained recognition strategy is converted into a recognition program by using the object library.

The generated program is applied to the scene as shown in Fig. 16(a). Fig. 16(b) shows the needle map, Fig. 16(c) shows the segmented regions based on surface orientation distribution, and Fig. 16(d) shows edge distributions superimposed on the region distributions. The highest region, determined by the dual photometric stereo (indicated by an arrow in Fig. 16(c)), is given to the program. The black nodes in Fig. 16(e) indicates the nodes which receive the execution messages in the real run. The program classifies the region to the corresponding aspect successfully.

VIII. Future Direction

This paper has discussed issues and techniques to automatically compile object and sensor models into a visual recognition program. This automatic generation requires several key components: object modeling, sensor modeling, strategy generation, and program generation. Especially we have argued for the importance of sensor modeling, as it has been studied very little in the past. We have presented our effort toward a systematic way to modeling sensors: representation of geometrical relationships between a sensor and an object feature and calculation of a feature's detectability and reliability. Actual creation and execution of interpretation trees by our method has been demonstrated.

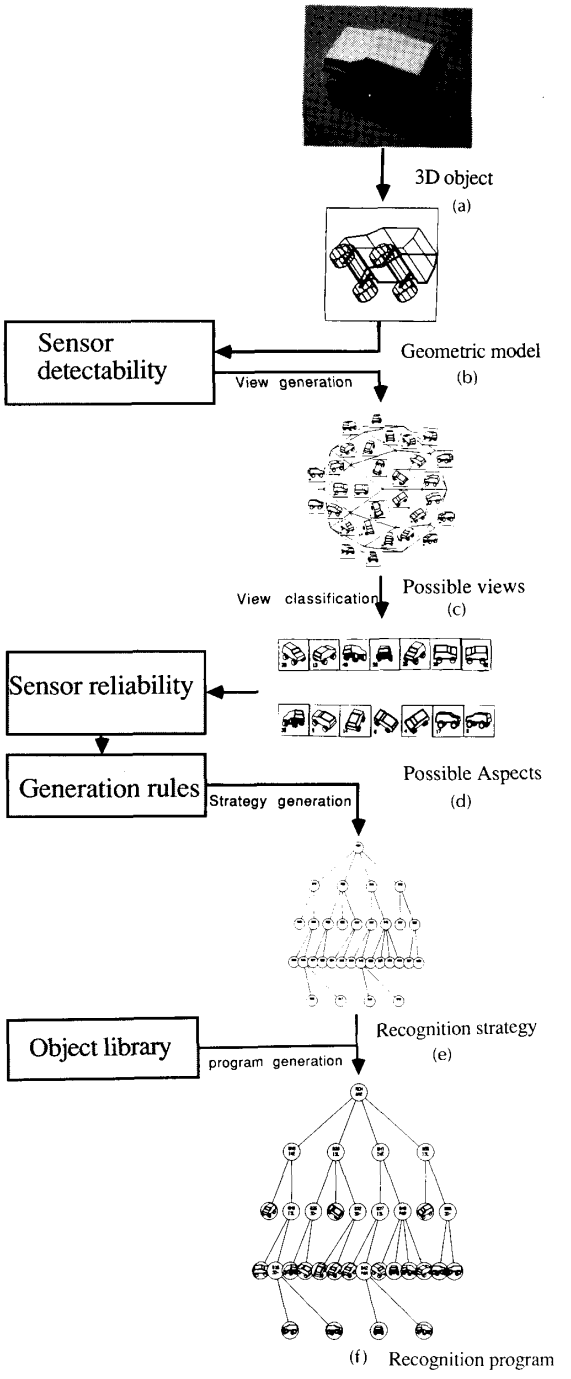


Fig. 15. Generation of an interpretation tree for a toy wagon.

[12]This mechanism is particularly useful when calculation of a feature is expensive, such as region relation. The system also converts an image value into a model value by using this mechanism. See [21] for more details.

[13]In this experiment, we only consider the northern hemisphere of the Gaussian sphere as viewer directions for the sake of simplicity. See Fig. 15(c).

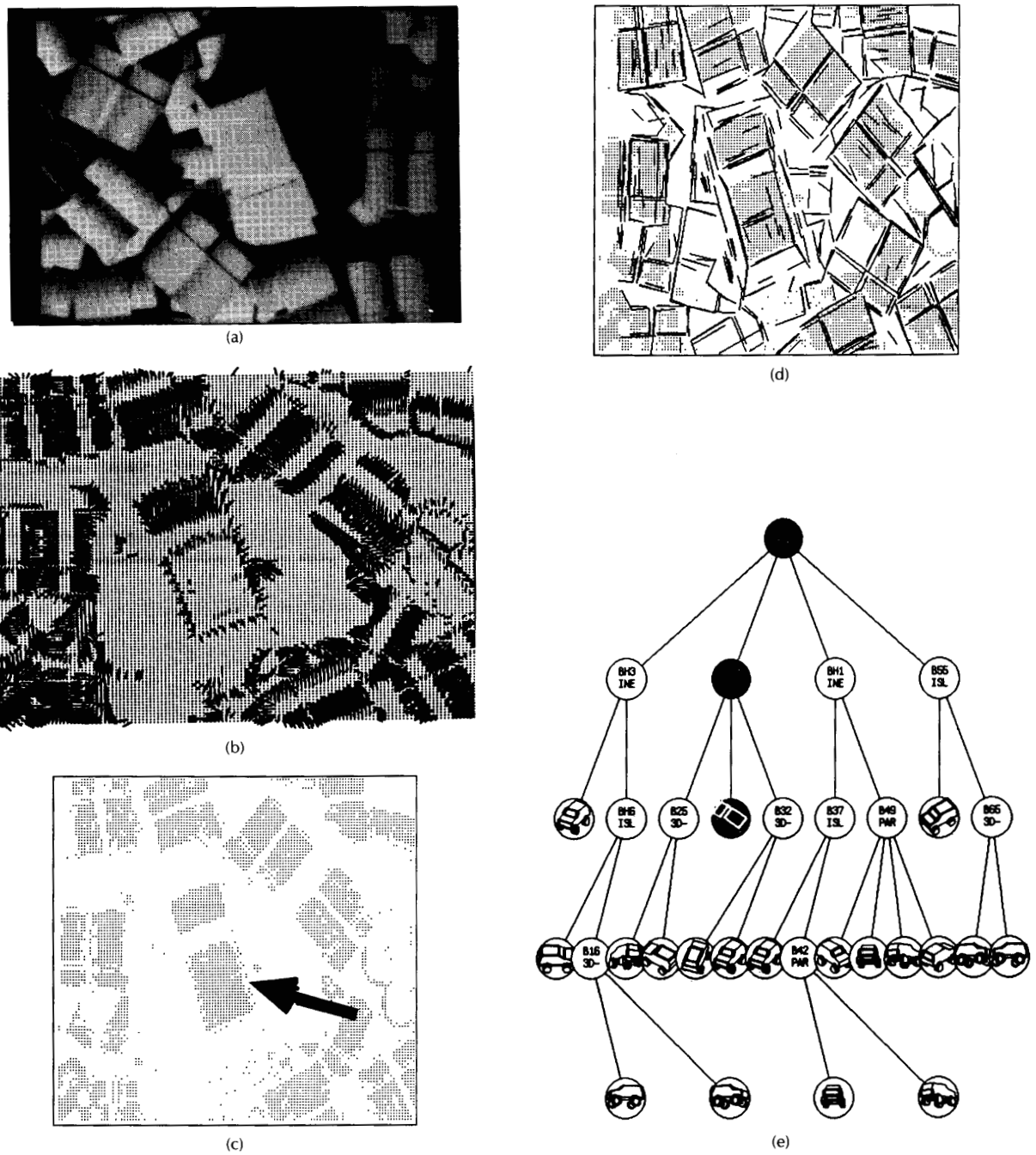


Fig. 16. Tree execution. (a) Input scene. (b) Surface orientation distribution of the scene. (c) Segmented regions using shadows and surface orientation discontinuities. The arrow indicates the target region. (d) Edge distributions superimposed on the region map. (e) Execution result. The target region is classified into the corresponding aspect successfully.

Vision has been recognized as an important, versatile sense for industrial applications. Yet, the number of successful applications seems to be far below the expectation. Apart from the large computational requirement and the cost, one of the serious factors which hinder wider application of vision is the time and expertise required to program a vision system. The automatic generation of recognition programs by compiling object and sensor models will mend the situation.

Moreover, automatic program generation may open a new dimension of capability in model-based vision when it comes to special sensors such as synthetic aperture radars (SAR) or FLIR. In those cases, since their sensor characteristics are not very intuitive, even capable vision imple-

mentors may not be doing the best job and an automatic and mechanical method of generating programs may be more advantageous.

Another area of investigation is learning from real scenes. For example, the range of a feature value is currently obtained solely from the analysis of sensor reliability and detectability. This information can be learned and modified by running the interpretation tree first generated from automatic analysis. The parameters used at branches are improved iteratively through real execution. Furthermore, branching structures themselves can be modified slightly. A critical difference of this approach from usual learning of recognition algorithms from scratch is that we start with the "skeleton" strategy which is more or less valid. Therefore there is a good chance that the final algorithm is truly competent.

Acknowledgment

K. Gremban proofread earlier drafts of this paper and provided useful comments which have improved the readability of this paper. The authors also thank H. Chang, S. Nayar, P. Balakumar, J.-C. Robert, Y. Kuno and the members of VASC (Vision and Autonomous System Center) of Carnegie Mellon University for their valuable comments and discussions.

References

[1] T. O. Binford, "Survey of model-based image analysis systems," *Int. J. Robotics Research*, vol. 1, no. 1, pp. 18–64, 1981.

[2] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Computing Surveys*, vol. 18, no. 1, pp. 67–108, Mar. 1986.

[3] G. Falk, "Interpretation of imperfect line data as a three-dimensional scene," *Artificial Intelligence*, vol. 3, no. 2, pp. 101–144, 1972.

[4] M. Oshima and Y. Shirai, "A model based vision for scenes with stacked polyhedra using 3D data," in *Proc. Intern. Conf. on Advanced Robot (ICAR85)*, Robotics Society of Japan, pp. 191–198, 1985.

[5] O. D. Faugeras and M. Hebert, "The representation, recognition, and locating of 3-D objects," *Int. J. Robotics Research*, vol. 5, no. 3, pp. 27–52, 1986.

[6] M. Nagao and T. Matsuyama, *A Structural Analysis of Complex Aerial Photograph*. New York, NY: Plenum, 1980.

[7] D. M. McKeown, W. A. Harvey, and J. McDermott, "Rule based interpretation of aerial imagery," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, no. 5, pp. 570–585, 1985.

[8] W. A. Perkins, "Model-based vision system for scene containing multiple parts," in *Proc. 5th International Joint Conf. on Artificial Intelligence*, pp. 678–684, 1977.

[9] K. Ikeuchi, H. K. Nishihara, B. K. P. Horn, P. Sobalvarro, and S. Nagata, "Determining grasp points using photometric stereo and the PRISM binocular stereo system," *Int. J. Robotics Research*, vol. 5, no. 1, pp. 46–65, 1986.

[10] T. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the Carnegie-Mellon NAVLAB," in *Annual Reviews of Computer Science*, J. Traub, Ed. Palo Alto, CA: Annual Reviews Inc, vol. 2, 1987, pp. 521–556.

[11] L. G. Roberts, "Machine perception of the three-dimensional solids," in *Optical and Electro-Optical Information Processing*, J. T. Tipplett, Ed. Cambridge, MA: MIT Press, 1965, pp. 159–197.

[12] W. E. L. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data," *Int. J. Robotics Research*, vol. 3, no. 3, 1984.

[13] R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 285–348, 1981.

[14] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[15] H. Yoda, J. Motoike, and M. Ejiri, "Direction coding method and its application to scene analysis," in *Proc. 4th Joint Conf. on Artificial Intelligence*, International Joint Conf. on Artificial Intelligence, 1975.

[16] R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," in *Three-Dimensional Machine Vision*, T. Kanade, Ed. Boston, MA: Kluwer, 1987, pp. 399–450.

[17] C. Goad, "Special purpose automatic programming for 3D model-based vision," in *Proc. of DARPA Image Understanding Workshop*, DARPA, pp. 94–104, 1983.

[18] T. Koezuka and T. Kanade, "A technique of pre-comiling relationship between lines for 3D object recognition," in *Proc. Intern. Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, IEEE Industrial Electronics Society, pp. 144–149, Feb. 1987.

[19] K. Ikeuchi, "Generating an interpretation tree from a CAD model for 3-D object recognition in bin-picking tasks," *Int. J. Computer Vision*, vol. 1, no. 2, pp. 145–165, 1987.

[20] K. Ikeuchi and T. Kanade, "Modeling sensors and applying sensor model to automatic generation of object recognition program," in *Proc. of DARPA Image Understanding Workshop*, Apr. 1988.

[21] H. Chang, K. Ikeuchi, and T. Kanade, "Model-based vision system by object-oriented programming," Tech. Rep. CMU-RI-TR-88-3, Carnegie Mellon University, The Robotics Institute, Mar. 1988.

[22] J. J. Koenderink and A. J. Van Doorn, "Internal representation of solid shape with respect to vision," *Biological Cybernetics*, vol. 32, no. 4, pp. 211–216, 1979.

[23] I. Chakravarty and H. Freeman, "Characteristic views as a basis for three-dimensional object recognition," in *Proc. The Society for Photo-Optical Instrumentation Engineers Conf. on Robot Vision*. Bellingham, WA: SPIE, vol. 336, 1982, pp. 37–45.

[24] K. Sugihara, "Automatic construction of junction dictionaries and their exploitation for analysis for range data." in *Proc. of 6th Intern. Joint Conf. on Artificial Intelligence*, pp. 859–864, 1979.

[25] M. Hebert and T. Kanade, "The 3D profile method for object recognition," in *CVPR*. IEEE Computer Society (San Francisco, CA), pp. 458–463, June 1985.

[26] R. J. Woodham, "Reflectance map techniques for analyzing surface defects in metal castings," Tech. Rep. AI-TR-457, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1978.

[27] K. Ikeuchi, "Determining a depth map using a dual photometric stereo," *Int. J. Robotics Research*, vol. 6, no. 1, 1987.

[28] C. Brown, "Fast display of well-tesselated surfaces," *Computer and Graphics*, vol. 4, no. 4, pp. 77–85, Apr. 1979.

[29] K. Ikeuchi, "Recognition of 3-D objects using the extended Gaussian image," in *International Joint Conf. on Artificial Intelligence*, pp. 595–600, 1981.

[30] ——, "Determining attitude of object from needle map using extended Gaussian image," Tech. Rep. AI memo 714, MIT Artificial Intelligence Laboratory, 1983.

[31] B. K. P. Horn, "Extended Gaussian images," *Proc. IEEE*, vol. 72, no. 12, pp. 1671–1686, Dec. 1984.

[32] P. J. Besl and R. J. Jain, "Intrinsic and extrinsic surface characteristics," in *CVPR*, IEEE Computer Society (San Francisco, CA), pp. 226–233, 1985.

[33] J. F. Canny, "Finding edges and lines in images," Tech. Rep. AI-TR-720, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1983.

[34] K. Ikeuchi and B. K. P. Horn, "Numerical shape from shading and occluding boundaries," in *Computer Vision*, M. J. Brady, Ed. Amsterdam, The Netherlands: North-Holland, 1981, pp. 141–184.

[35] K. Tomiyasu, "Tutorial review of Synthetic-Aperture Radar (SAR) with applications to imaging of the ocean surface," *Proc. IEEE*, vol. 66, no. 5, pp. 563–583, May 1978.

[36] R. A. Jarvis, "A laser time-of-flight range scanner for robotic vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, no. 5, pp. 505–512, 1983.

[37] W. E. L. Grimson, *From Images to Surfaces: A Computational*

Study of the Human Early Visual System. Cambridge, MA: MIT Press, 1981.
[38] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, no. 2, pp. 139–154, 1985.
[39] V. J. Milenkovic and T. Kanade, "Trinocular vision: using photometric and edge orientation constraints," in *Proc. of DARPA Image Understanding Workshop*, DARPA (Miami Beach, FL), pp. 163–175, Dec. 1985.
[40] K. Koshikawa and Y. Shirai, "A model-based recognition of glossy objects using their polarizational properties," *J. Robotics Soc. Japan*, vol. 3, No. 1, pp. 4–9, 1985 (in Japanese, brief English version is available as Y. Shirai, *Three-Dimensional Computer Vision.* Berlin, West Germany: Springer-Verlag, 1987, pp. 153–156, 274–276.

Katsushi Ikeuchi received the B.Eng. degree in mechanical engineering from Kyoto University, Kyoto, Japan, in 1973, and the M.Eng. and D.Eng. degrees in information engineering from the University of Tokyo, Tokyo, Japan, in 1975 and 1978, respectively.

He has held several research positions at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, and the Electrotechnical Laboratory, Ministry of International Trade and Industry, Tsukuba, Japan. He is a Senior Research Computer Scientist in the Computer Science Department and the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA. His research interests include low level vision problems, techniques for converting $2\frac{1}{2}$-D representations to 3-D representations, and geometric modeling and its application to vision. In particular, he has developed low level vision algorithms for shape-from-shading, shape-from-texture, and photometric stereo, conversion techniques based on Extended Gaussian images and aspect models, and 3-D object recognition algorithms based on the VANTAGE geometric/sensor modeler.

Takeo Kanade (Senior Member, IEEE) received the Ph.D. degree in information science from Kyoto University, Kyoto, Japan, in 1974.

He has served as Associate Professor of Information Science at Kyoto University. Currently he is Professor of Computer Science and Acting Director of the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA. He has worked on various problems in vision, sensors, manipulators, and mobile robots: including development and control of new direct-drive arms which were first conceived and prototyped at CMU (DD Arm I and DD Arm II), and development of a vision navigation system for the Navlab (a van with onboard sensors and computers). He has authored and edited three books, and authored over sixty papers and technical reports in these areas. Currently he is the Principal Investigator of three robotics research programs at CMU: Image Understanding, Autonomous Land Vehicle Vision System, and NASA Mars Rover.

Dr. Kanade served as a General Chairman of IEEE International Conference on Computer Vision and Pattern Recognition in 1983, and a Vice Chairman of IEEE International Conference on Robotics and Automation in 1986. He is the editor of *International Journal of Computer Vision.*