Three-Dimensional Object Recognition Systems
AK Jain and P.J. Flynn (Editors)

Appearance-Based Vision and the Automatic Generation of Object Recognition Programs'

Keith D. Gremban and Katsushi Ikeuchi'

' School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213-3890

Abstract

The generation of recognition programs by hand is a time-consuming, labor-intensive task that typically results in a special purpose program for the recognition of a single object or a small set of objects. Recent work in automatic code generation has demonstrated the feasibility of automatically generating object recognition programs from CAD-based descriptions of objects. Many of the programs which perform automatic code generation employ a common paradigm of utilizing explicit object and sensor models to predict object appearances; we refer to the paradigm as appearance-based vision, and refer to the programs as vision algorithm compilers (VACs). In this paper, we discuss the paradigm of appearance-based vision and present in detail two specific VACs: one that computes feature values analytically, and a second that utilizes an appearance simulator to synthesize sample images.

L Introduction

The generation of object recognition programs by hand is a time-consuming, labor-intensive process that typically results in a special purpose program for the recognition of a single object' or a small set of objects. The reason for this lies in the design methodology: on the basis of a representative set of sample images, the designer experiments on the images, selects a set of image features, and specifies the procedure to be used in matching image features to object features. The entire process requires a highly skilled and motivated designer.

Recently, research has been conducted towards the goal of automatically generating recognition code from a CAD-based description of an object. Most current industrial parts are designed and manufactured using computer-aided tools, so CAD descriptions exist for most parts; automatic generation of recognition code from the same model information used for design and manufacture would be an efficient, cost-effective approach. A number of programs for automatic generation of object recognition code have been written, and many of these programs employ

'This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597.

a common paradigm in which explicit object and sensor models are used to predict object appearances. We refer to the paradigm as *appearance-based vision,* and programs which generate object recognition programs are called *vision algorithm* compilers, or VACs.

Appearance-based vision represents an extension to the familiar paradigm of model-based vision. Model-based vision defines an execution-time strategy of matching observed image features to model features, but does not address the issue of defining a strategy for selecting either features or matching procedures. Appearance-based vision systems employ model-based matching during the execution-time recognition phase, but also employ a characteristic methodology during the off-line compilation phase, during which features are selected and processing strategies determined.

In principle, a CAD-like object model, augmented with sensor-specific information like surface color, roughness, and reflectance, can be used in conjunction with a sensor model to predict the appearance of the object under any specified set of viewing conditions. VACs can predict object appearances in two different ways: analytically, or synthetically.

In relatively simple domains, feature values *can* be analytically determined from model information. As objects and their properties grow in complexity, however, effects such as self-shadowing and inter-reflection become more important, but are difficult to incorporate into an analysis. Analytic prediction of appearances is therefore impractical for some domains. An alternative to analytic prediction of appearances is the use of an appearance simulator. An appearance simulator generates synthetic images of objects under specific viewing conditions, with respect to a given sensor. An appearance simulator *can* be used to generate a representative collection of sample images, which can then be automatically processed and analyzed to extract the feature values that characterize object appearances.

In this paper, we discuss the paradigm of appearance-based vision. In the next section, we review the state-of-the-art in appearance-based vision and the automatic generation of object recognition programs. In the course of the review, the defining characteristics of appearance-based vision systems will be noted. Following the review, we will present in detail two VACs which typify appearance-based systems. In section 3, we present a VAC that employs analytic prediction of appearances, and the advantages and limitations of this approach are discussed. Then, in section 4, we present a VAC that utilizes an appearance simulator. A brief summary concludes the paper.

2. The Paradigm of Appearance-Based Vision

The history of computer vision research has largely been a study in making vision systems work. Little attention has been paid to the study of how to design and build application systems. For example, the dominant paradigm in computer vision is that of *model-based* vision. Briefly, the model-based paradigm can be characterized as *hypothesize-predict-verify:* given a collection of image features, hypothesize a match of an image feature to a model feature; use the hypothesized match to *predict*

the image locations of other model features; *verify* the predictions and update the hypothesis. The paradigm does not specify how to select the features to use, or how to perform the matching. The paradigm defines an approach to execution-time processing, rather than an approach to system building.

Because every computer vision system is essentially a custom solution to a specific problem, a typical computer vision system is expensive to develop and install, and is capable of recognizing only a single part or a small number of parts under very special conditions. Modifications to existing systems are difficult to make, and the cost to develop a new system is as high as that of the first system. Clearly this is an unacceptable situation.

Appearance-based vision addresses the problem of building cost-effective computer vision systems; it specifies a methodology for the automatic generation of object recognition programs. Appearance-based vision is an extension of the model-based paradigm that formalizes and automates the design process. Appearance-based vision can be characterized as an automated process of analyzing the appearances of objects under specified observation conditions, followed by the automatic generation of model-based object recognition programs based on the preceding analysis, and ending with with repeated execution of the generated program.

One characteristic of appearance-based vision is that both objects and sensors are explicitly modeled, and therefore exchangeable. Hence, a given VAC can generate object recognition code for many different objects using the same sensor model, or the set of objects can be fixed and the sensor models varied.

A VAC incorporates a two phase approach to object recognition. The first phase is performed off-line and consists of analysis of predicted object appearances and the generation of object recognition code. The second phase takes place on-line, and consists of applying the previously generated code to input images. The first phase consists ideally of a single program execution for a given object recognition task, and can be relatively expensive. The second phase consists of many executions of the output object recognition program, which must be both fast and cost-effective. The high cost of the first phase is amortized over a large number of executions of the second phase.

In the next subsection, we present an historical overview of research on appearance-based vision. Then, building on the historical perspective, we enumerate and elaborate on the commonalities between the systems; it is this set of common characteristics that define the paradigm of appearance-based vision.

2.1. Historical Perspective

Goad [10] presented an early version of a VAC. In Goad's system, an object is described by a list of edges and a set of visibility conditions for each edge. Visibility is determined by checking visibility at a representative number of viewpoints obtained by tessellating the viewing sphere. Object recognition is performed by a process of iteratively matching object and image edges until either a satisfactory match is found, or the algorithm fails. The sequence of matchings is compiled during the off-line analysis phase. Goad's system was not completely automatic, however. Goad selected edges as the features to be used for recognition, and the

order of edge matching was specified by hand.

The 3DPO system of Bolles and Horaud [4] was built with the intended goal of using off-line analysis to produce the fastest, most efficient on-line object recognition program possible. 3DPO utilized the local-feature-focus method, in which a prominent focus feature is initially identified, and then secondary features predicted from the focus feature are used to fine-tune the localization result. The system was not fully automatic, as the feature matching strategies were determined interactively.

Ikeuchi and Kanade [15] [17] first pointed out the importance of modeling sensors as well as objects in order to predict appearances, and noted that the features that are useful for recognition depend on the sensor being used. Their system predicts object appearances at a representative set of viewpoints obtained by tessellating the viewing sphere. The appearances are grouped into equivalence classes with respect to the visible features; the equivalence classes are called aspects. A recognition strategy is generated from the aspects and their predicted feature values, and is represented as an interpretation tree. Each interpretation tree specifies the sequence of operations required to precisely localize an object.

Hansen and Henderson [11] demonstrated a system that analyzed 3D geometric properties of objects and generated a recognition strategy. The system was developed to make use of a range sensor for recognition. The system examines object appearances at a representative set of viewpoints obtained by tessellating the viewing sphere. Geometric features at each viewpoint are examined, and the properties of robustness, completeness, consistency, cost, and uniqueness are evaluated in order to select a complete and consistent set of features. For each model, a strategy tree is constructed, which describes the search strategy used to recognize and localize objects in a scene.

The system of Arman and Aggarwal [1] was designed to be capable of selecting the proper sensor for a given task. Starting with a CAD model of an object, the system builds up a tree in which the root node represents the object, and the leaves represent features (where features are dependent upon the sensor selected), and a path from the root to a leaf passes through nodes representing increasing specificity. Each arc in the tree is weighted by a "reward potential" that represents the likely gain from traversing that link. At run time, the system traverses the tree from the root to the leaves, choosing the branch with the highest weight at each level, and backtracking when necessary.

The PREMIO system of Camps et al [5] predicts object appearances under various conditions of lighting, viewpoint, sensor, and image processing operators. Unlike other systems, PREMIO also evaluates the utility of each feature by analyzing the detectability, reliability, and accuracy. The predictions are then used by a probabilistic matching algorithm that performs the on-line process of identification and localization.

The BONSAI system of Flynn and Jain [8] identifies and localizes 3D objects in range images by comparing relational graphs extracted from CAD models to relational graphs constructed from range image segmentation. The system constructs the relational graphs off-line using two techniques: first, view-independent features are calculated directly from a CAD model; second, synthetic images are constructed

for a representative set of viewpoints obtained by tessellating the viewing sphere, and the predicted areas of patches are determined and stored as an attribute of the appropriate relational graph node. During the on-line recognition phase, an interpretation tree is constructed which represents all possible matchings of the graph constructed from a range image, and the stored model graph. Recognition is performed by heuristic search of the interpretation tree.

Sato, et al [21] demonstrated a system for recognition of specular objects. During an off-line phase, the system generates synthetic images from a representative set of viewpoints. Specularities are extracted from each image, and the images are grouped into aspects according to shared specularities, and each specularity is evaluated in terms of its detectability and reliability. At execution time, an input image is classified into a few possible aspects using a continuous classification procedure based on Dempster-Shafer theory. Final verification and localization is performed using deformable template matching.

2.2. The Common Threads

After reviewing the different appearance-based systems that have been constructed, it is useful to go back and point out the common processing steps.

2.2.1. Two Phases of Processing

Each of the systems discussed employ two distinct phases of processing. The first phase, variously called off-line, compilation, or analysis, consists of analyzing object appearances and constructing recognition strategies. The second phase, called on-line, run-time, or execution, consists of applying the strategies generated in the first phase to an actual recognition task. In general, computational efficiency is not a concern in the first phase, since it does not directly affect the actual time or effort required to perform object recognition, and the cost is only incurred once. In contrast, the second phase is expected to execute many times as part of an application, and consequently must be efficient. In effect, the time spent during strategy generation can be amortized over the number of executions of the resultant strategy.

2.2.2. Explicit Object and Sensor Models

Any object recognition system must match the appearance of an object with respect to some sensor, to a model of the object. Consequently, to automatically generate a program for object recognition, it is necessary to predict and analyze object appearances. Objects appear differently to different sensors, so in order to predict object appearances, sensors must be modeled as well. An appearance-based system therefore includes both object and sensor models.

The early appearance-based systems only made use of explicit object models and utilized implicit sensor models, although the need for different types of models to represent different types of detectable features was acknowledged. All recent systems have emphasized the fact that appearance depends upon the sensor and include explicit models of both objects and sensors. Models may be exchanged so that the same VAC can generate object recognition programs for a variety of objects and sensors.

2.2.3. Appearance Prediction and Analysis

In general, there are two approaches to predicting object appearances: analytic and synthetic. The analytic approach uses the information stored in object and sensor models to analytically predict the appearance of an object from various viewpoints. Alternatively, it is possible to generate images of objects under specific sensor conditions and analyze the synthetic images. Both techniques have advantages and disadvantages that are discussed more completely in the next two sections.

The appearance of an object with respect to a sensor is characterized by means of the features that can be extracted from the sensor image. Hence, each sensor model includes a feature set, and a collection of image processing operators that are used to extract the features.

The appearance of an object also varies with respect to the relative geometry between sensor and object, which can be referred to as the viewpoint. Potentially, there are six degrees of freedom in viewpoint, each of which spans an infinite number of parameter values. Clearly, exhaustive computation of all possible appearances is impossible. To make the set of possible appearances manageable, similar appearances are grouped into sets called aspects. Formally, an aspect is a class of topologically equivalent views of an object [18]. However, since different sensors detect different features, the formal definition of an aspect is usually relaxed to be a class of appearances that are equivalent with respect to a feature set.

A substantial amount of work has been performed on deriving methods for analytically determining the collection of aspects of an object. Representative examples of work in this area include Plantinga and Dyer [20], Kriegman and Ponce [19], and Chen and Freeman [6].

An alternative to the exact analytic computation of aspects is the exhaustive approach, in which viewpoints are sampled uniformly throughout the space of possible viewpoints, and then similar viewpoints are grouped together. An approach of this sort was used by Ikeuchi and Kanade [17], Hansen and Henderson [11], Flynn and Jain [8], and Sato, et al [21]. In each of these systems, the space of possible viewpoints is uniformly tessellated, and the object appearance is predicted from each viewpoint corresponding to the center of a tessel. The fidelity of the sampling can be increased by subdividing each tessel. This approach is more general than the analytic approach, since the same procedure can be used independently of the sensor or feature set.

2.2.4. Generation of a Recognition Strategy

The result of the off-line, compilation phase of an appearance-based system is a strategy for object recognition. The strategy is often represented in the form of a tree that represents the sequence of operations to perform at each step of the recognition process. Since the generation of a strategy is performed off-line, it is possible to perform relatively expensive optimization.

There are many different computational approaches that can be employed for object recognition. Suetens, et al [23] is a recent survey of the range of approaches.

Other surveys of object recognition include Besl and Jain [3], and Chin and Dyer [7]. A VAC *can* be constructed for any given approach. Consequently, there is no standard form for the recognition strategy output by a VAC; all that can be said is that the strategy consists of executable code.

3. A VAC Utilizing Analytic Feature Prediction

The VAC discussed in this section was designed to generate an object localization (pose determination) program for a bin-picking task, and was initially presented in [14], at which time the system was not fully automatic. Further research has led to complete automation of program generation [16], as well as optimization of the resulting code [13]. This section presents the complete system. The inputs to the VAC consist of an object model, specifying geometric and photometric characteristics of the object, and a sensor model, specifying the sensor characteristics necessary for predicting object appearances and feature variations. The output consists of a recognition strategy in the form of an interpretation tree.

The VAC presented below utilizes a two-stage approach to object localization. The first stage, aspect classification, classifies an input image into an instance of an aspect. This stage is equivalent to rough localization, since the range of possible object poses is constrained to be consistent with the observed aspect. The second stage, *linear* shape change determination performs more accurate localization by analyzing the shape change within an aspect.

3.1. Explicit Object and Sensor Models

Computer vision systems *can* use many different types of sensors. A sensor *can* be considered to be a transducer that transforms object features into image features, and different sensors yield different image features. For example, a laser range finder detects the range and orientation of object surfaces, while edge-based binocular stereo yields the range computed by triangulation on detected edges. A sensor model must specify the features detectable by the sensor.

The list of features describes the qualitative characteristics of a sensor. The quantitative characteristics are given by the detectability and reliability of each feature. Detectability specifies the conditions under which a given feature can be detected. Reliability specifies the expected error in the feature value. Both characteristics depend on the configuration of the object feature and the sensor.

The detectability of a feature by a given sensor depends on factors such as range, relative attitude, reflectivity and so on. In many applications, such as industrial workstations, many of the factors can be fixed, and relative attitude becomes the dominant factor. To consider relative attitude, fix the sensor coordinate system, and consider the relationship of a feature coordinate system with respect to it. The feature coordinate system is defined such that the z-axis is aligned with the surface normal; x and y axes are assumed to be defined arbitrarily. There are three degrees of freedom in relative attitude. The feature configuration space, described below, provides a convenient representation for relative attitude that considerably simplifies many of the operations involved in predicting both detectability and

reliability.

Consider a solid unit sphere, called the ***orientation sphere,*** or ***o-sphere,*** in which each relative orientation of the feature coordinate system corresponds to a point. The direction from the center of the sphere to the point defines the orientation of the feature z-axis. Each point on the surface of the o-sphere represents the unique orientation of feature coordinates obtained by rotation about an axis perpendicular to the plane formed by the north pole, the center of the sphere, and the surface point itself. Interior points represent rotation about the (new) z-axis defined by the ray from the center of the sphere to the point. The north pole is taken to be the case in which feature coordinates and sensor coordinates are aligned. One o-sphere can be defined for each object feature, and is referred to as the feature configuration space.

As an example, consider the case of a feature coordinate system aligned with sensor coordinates; this is represented as the point (0.0.1) on the o-sphere. If the feature system in now rotated by +90 deg about the y-axis, the new orientation is represented by the point (1, 0, 0) on the 0-sphere. *An* additional rotation of +90 deg about the *new* z-axis moves the point inward to (0.75.0.0). Figure 1 further illustrates the concept.

A sensor system consists of two components:. an illuminant, and a detector. For a feature to be detected, it must be visible to both components. For a given feature, a separate configuration space *can* be defined for each sensor component. Within each configuration space, the configurations for which the feature is detectable *can* be easily defined by geometric constraints. The detectability *of* a feature with respect to a given sensor is then the intersection of the detectable regions *of* the configuration spaces of each of the sensor components. An example *of* the detectability computation for a light-stripe range finder is shown in Figure 2.

For a given viewpoint, the appearance of an object with respect to a particular sensor can be defined by a list of the detectable object features, along with the values of parameters extracted from those features. A viewpoint corresponds to a single point in the configuration space of each feature *of* the object.' A feature is detectable if and only if 'the point representing the sensor viewpoint lies in the detectable region of the feature configuration space, and if no other part of the object occludes the feature. These conditions can easily be checked using a geometric modeler.

3.2. Predict and Analyze Appearances

The techniques presented above make it possible to analytically determine the detectability of any feature from any viewpoint. The combination of features, along with predicted feature values, defines the appearance of the object. Next, the capability to predict appearances must be used to determine the aspects of the object.

The system employs an exhaustive approach to predicting appearances. The exhaustive approach is relatively easy to implement, especially for cases in whch the distance between sensor and object is assumed fixed. Then, all possible viewpoints *can* be represented as points on the surface of a sphere centered on the object.

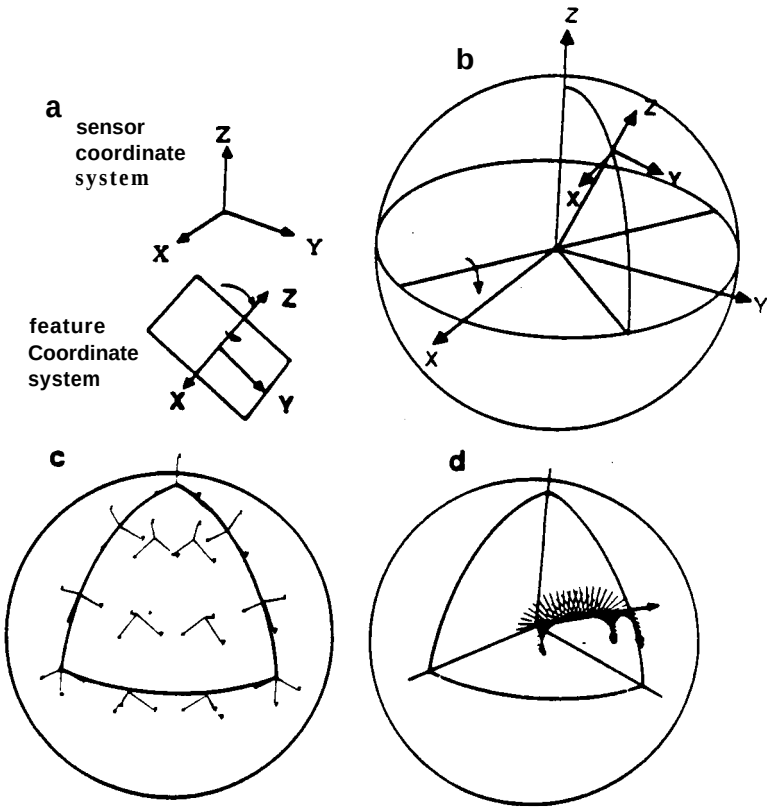


Figure 1. Feature Configuration Space. (a) **Relationship between sensor coordinates and feature coordinates.** (b) **Feature coordinates as points on the o-sphere. The bottom left drawing depicts the coordinates corresponding to points on the surface of the 0-sphere, while the bottom right drawing depicts the coordinates along one axis of the 0-sphere.**

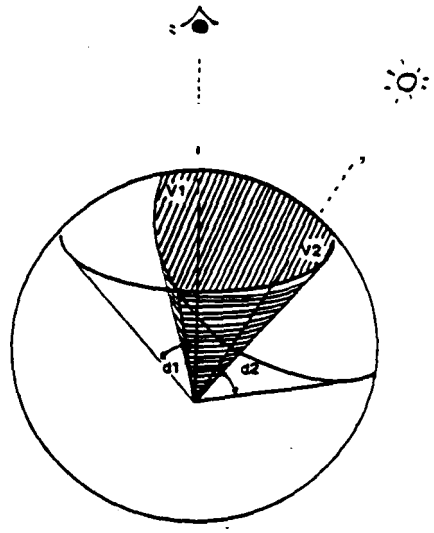


Figure 2. Detectability of a Face for a Light-stripe Range Finder. The detectable region is the intersection of the detectability of the illuminant and the detectability of the sensor.

A tessellation of the sphere using a geodesic dome which divides the sphere into many small spherical triangles yields a nearly uniform sampling of viewpoints. The triangles can be subdivided repeatedly to yield any desired level of sampling resolution. At the center of each spherical triangle, the detectability of each feature can be computed as outlined above.

Aspects can be selected in many different ways, depending upon the features being considered. For example, aspects can be defined as collections of viewpoints for which the same set of features are visible. Alternatively, as used here, aspects can be defined on the basis of detectable faces. Consider an object with N faces (planar or curved) $S_1, S_2, \ldots, S_N$ and define the face-label $X = (X_1, X_2, \ldots, X_N)$, where $X_i = 1$ or 0 according to whether or not face S_i is detectable. Viewpoints with identical face labels are grouped together into aspects. For each aspect, a representative attitude is selected and used to calculate representative feature values. Each aspect can be characterized by the feature values of the representative attitude. Further, the ranges of feature values can be obtained by examining the range of values of the features of each of the viewpoints constituting an aspect. Figure 3 illustrates the process of view generation and aspect selection.

3.3. Generation of a Recognition Strategy

The generation of a recognition strategy depends to some extent upon the sensor used, or at least upon the features used. In this section, results are presented for

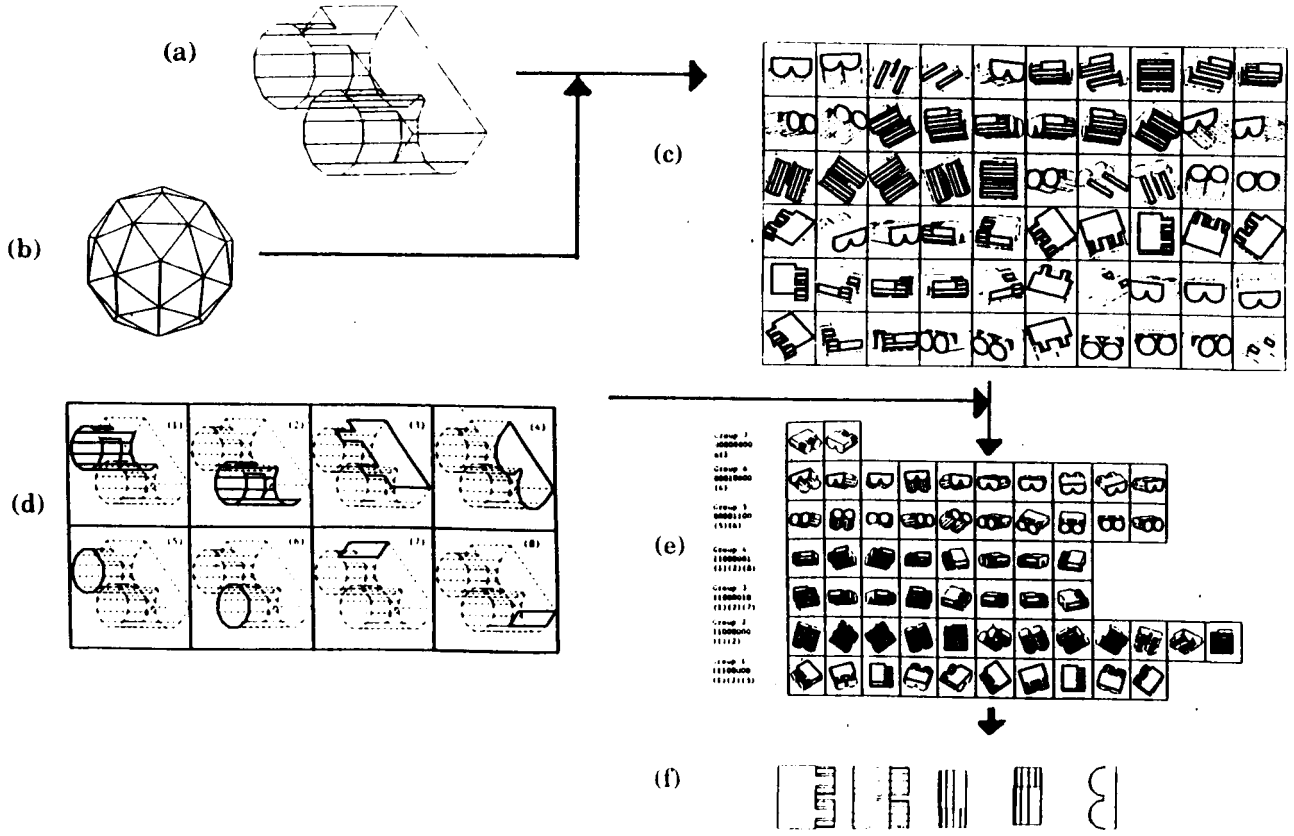


Figure 3. Extraction of Aspects. (a) Geometric model of an object. (b) The Gaussian sphere is tessellated into sixty triangles to represent viewpoints sampled. (c) Sixty computed appearances. Faces surrounded with bold lines are detectable by photometric stereo. (d) Eight component faces to be used for shape labeling. (e) The five aspects obtained through classification by shape label. Regions smaller than a certain threshold are regarded as undetectable. (f) Representative attitudes, one for each detectable aspect.

sensors which produce dense range maps.

3.3.1. Aspect Classification

Aspect classification is the process of classifying an input image into an instance of an *aspect*. Since an aspect represents a contiguous set of viewpoints, aspect classification is equivalent to rough localization. The parameters of object pose determined through aspect classification also provide good starting parameters for the stage of linear shape change determination that follows.

One way of performing aspect classification is to extract Feature values from the input image and compare this set of values to the stored value ranges that characterize aspects. This approach may be very inefficient, however, since only a few of the features may be needed to perform classification, yet all are computed. A more cost-effective approach is to determine the computational cost of each feature, and then determine a discriminating set of features that minimizes the expected cost of classification.

A ***classification tree,*** or ***decision tree,*** is a tree in which each node represents a collection of classes and an associated test; and arcs represent the possible results of a test. Leafnodes represent the final results of classification. Using a classification tree, a classification is performed by traversing the tree from the root to a leaf. A classification tree provides a convenient framework for optimizing the aspect classification process, since using a classification tree permits tests (and corresponding computations of feature values) to be performed sequentially.

Aspect classification trees can be used to optimize the classification process in the following way. In the off-line stage of processing, the entire set of possible aspect classification trees is examined systematically, and the minimum cost classification tree is identified and saved; this tree stores feature identifiers and test values at each node.

A path from the root of a classification tree to a leaf represents a complete classification operation. Computing the cost of a single path is straightforward. Each test requires a feature to be computed, and each such computation incurs a computational cost. Each node in the classification tree is assigned the cost of computing the feature needed for the corresponding test. The cost of a path is the sum of the costs of the intermediate nodes.

A classification tree contains many paths from the root to the leaf nodes, and different paths may be taken with different frequencies. Therefore. the cost of a classification tree is defined to be the expected cost of a classification; that is, the average cost taken over all possible inputs. The expected cost can be computed by weighting the cost at each node by the proportion of the sample population that will pass through the node. The cost of every node in the tree is summed and divided by the population size to yield the overall cost.

Finding a minimum cost classification tree can be formalized as a search problem over another kind of tree, called a ***strategy tree.*** A strategy tree for aspect classification is a tree in which each path from the root to a leaf represents a complete strategy for classification; that is, each path in a strategy tree can be expanded into a classification tree. Therefore, finding the least cost classification tree is

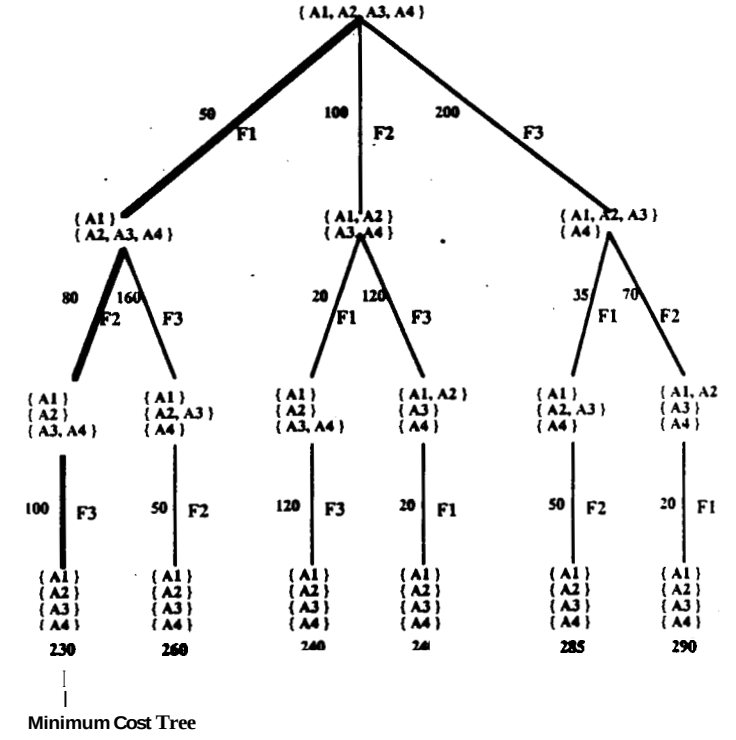


Figure **4.** Strategy Tree. Each path from the root to a leaf represents a complete strategy for aspect classification. Finding the minimum cost classification strategy is equivalent to finding the minimum cost path from the root to a leaf.

equivalent to finding the least cost path from root to a leaf in the strategy tree.

In a strategy tree, each node contains the results of applying a test using a given feature, while arcs represent the tests. Each arc is labeled with the product of the cost of the feature and the expected number of samples to which the test will be applied. *An* arc is present when a feature can be used to break up a set into smaller sets. At the leaf nodes, all the constituent sets should be singletons, unless the feature set is incapable of distinguishing some of the input classes.

Figure **4** illustrates a strategy tree for a simple case consisting of **4** classes, and 3 features. At the root, all the classes are grouped into a single set. An arc is present for every computable feature which can reduce the set size, so there are three arcs at the root. The darkened path in the tree is the minimum cost path.

There are cases for which is not possible, given the set of features, to distinguish two classes. Indistinguishable classes are referred to as ***congruent classes,*** and

the corresponding nodes as congruent nodes. Since the classes in an aspect classification tree represent aspects of some object, the existence of congruent classes means that the corresponding aspects cannot be distinguished with the available features; such aspects are referred to as *congruent aspects.* Congruent aspects do not represent a failure of the search procedure, but rather indicate a fundamental limitation of the feature set. In many cases, the linear shape change determination step corrects for ambiguous aspect classification and determines the correct object pose.

3.3.2. Linear Shape Change Determination

The aspect classification step results in the classification of an input image as an instance of an aspect. Since an aspect consists of a contiguous collection of mews of an object, aspect classification is equivalent to rough localization; the possible collection of object poses are limited to those consistent with the observed aspect.

The next step in the localization process determines the exact pose, given the initial estimate obtained from aspect classification. The same set of features is visible throughout an aspect, so no non-linear events such as the appearance or disappearance of a feature occur. Therefore, the second step consists of determining the exact pose within an aspect, subject only to linear changes (rotation and translation) of features; this step in the localization process is known as *Linear shape* change *determination* (LSCD).

One way to perform LSCD is to utilize a model-based approach in whch image features are matched to model features, a pose is hypothesized and used to predict locations of features in the image, and the pose is refined by computing the error in predictions and observations and updating the pose appropriately. In most model-based systems, the matching stage is very difficult, since all possible matches between image and model features must be investigated. Since aspect classification has been performed, however, the correspondence between some set of image and model features has already been established. In particular, the assumptions underlying the LSCD method presented here are:

- the correspondences between model and image faces are known;
- the correspondences between model and image edges are unknown.

The aspect classification strategy presented in the previous subsection was encoded in the form of a tree, the classification tree. Each leaf node of the tree represents an aspect or collection of congruent aspects, and for each leaf node a different LSCD strategy may be appropriate. The VAC presented here computes a separate LSCD strategy for each leaf node of the aspect classification tree. The steps in each LSCD strategy are encoded as nodes that are attached to the leaf nodes of the aspect classification tree. Although the particular computational procedures vary between aspects, the same steps are followed in the same order for each aspect:

1. determine the coordinate system of the primal face (the visible face with the largest 3D area):

(a) determine the origin of the primal face;

(b) determine the z-axis orientation of the primal face;

(c) determine the x-axis orientation of the primal face;

2. estimate the body coordinate system;

3. establish correspondences between image and model edges;

4. recover exact body coordinates by numerical minimization.

The LSCD strategy is determined off-line through the following steps:

1. For each aspect, the visible face with the largest 3D area is selected as the primal face;

2. Each primal face is analyzed, and a method for defining the face coordinate system is determined. Separate nodes are attached to the classification tree which define the exact procedures used in each individual step of determining the origin, z-axis, and x-axis of the primal face.

3. Given the estimated coordinate system of the primal face and the transformation between primal face coordinates and model coordinates, the body coordinate system can be estimated with respect to the sensor Coordinates. This is encoded as a separate node of the tree.

4. Knowing the object aspect and a rough estimate of the body coordinate system enables the prediction of the location of model edges in the image. The predicted edges can then be matched to observed edges. This process is encoded as a separate node of the tree.

5. A fine-tuning procedure is used to determine the exact body coordinates by adjusting the estimated body coordinates so that image edges exactly match predicted model edges. An exact match is not possible, so the procedure finds the body Coordinates that minimizes the error between predicted and observed edge locations. This procedure is encoded as a separate node of the tree.

3.3.3. The Interpretation Tree

The overall strategy for object localization is encoded in the form of a tree, the interpretation tree. The top part of the interpretation tree consists of the aspect classification tree, and consists of directions for a series of feature value computations and tests that result in the classification of an input image into an instance of an aspect. The bottom part of the interpretation tree is six nodes deep, and consists of the steps in the LSCD strategy that are appropriate for each aspect. Figure 5 illustrates the interpretation trees for a sample object: a toy car.

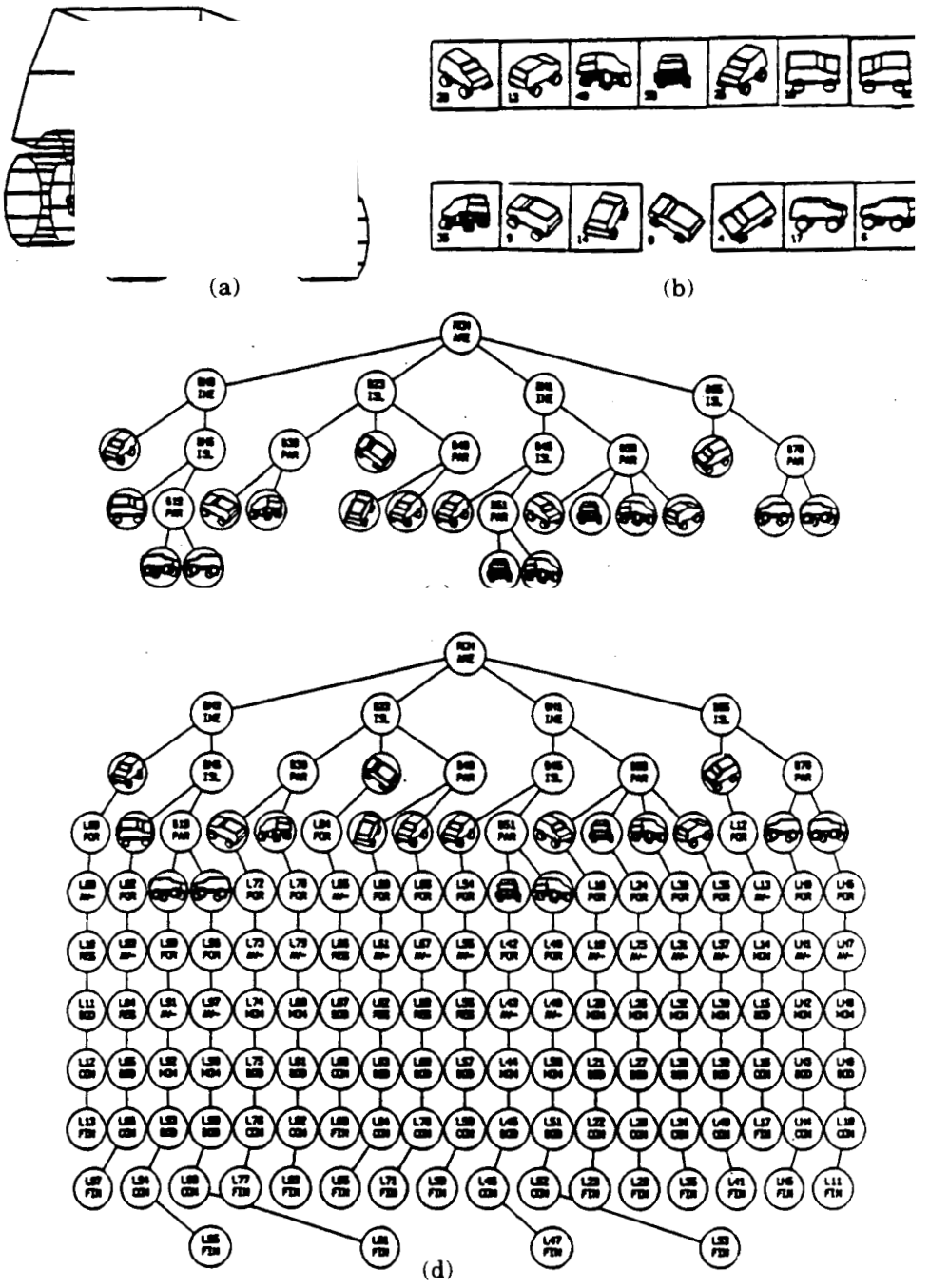


Figure 5. Generation of a Recognition Strategy for a Toy Car. (a) **Object** model. (b) **Object aspects.** (c) **Aspect classification tree.** (d) **Complete interpretation** tree.

3.4. Run-time Execution

Each of the procedures represented by a node of an interpretation tree corresponds to an executable object stored in a program library. During the off-line phase, the creation of a node of the interpretation tree is accompanied by the instantiation of an executable object from the program library, and the insertion of the object at the node. During the on-line phase, these instantiated objects are executed in order to perform object recognition. Message-passing is used for communication between objects.

For each execution of the object recognition strategy during the on-line phase, a preprocessed image is passed to the root of the interpretation tree and the object stored at the root is invoked. The execution of the root object results in the computation of some feature value and the application of a test. The outcome of the test results in the preprocessed image being passed to the appropriate node at the next level of the tree, and the corresponding stored object is invoked. The sequence of message-passing and object invocation proceeds until a leaf node is reached, indicating that processing is complete. The results of processing are then passed back up the tree and returned from the root.

At congruent nodes of the classification part of the tree, no unique aspect is identified. Instead, several aspects are determined to be possible. For each congruent aspect, the LSCD part of the tree is executed, and the results returned are compared by the congruent node; the aspect yielding the minimum error is selected as the correct interpretation and this result is passed back up to the root of the tree.

3.5. Application

In this section, we illustrate the execution of the compiled strategy. As stated previously, the VAC presented here was constructed to utilize sensors that produce dense range images; that is, the features used are those that can be determined given an input range image. In order to show the sensor independence of the VAC, the results have been demonstrated using two different range sensors: dual photometric stereo [14], and an ERIM laser range finder [12]. The results for dual photometric stereo are reported below.

The complete interpretation tree for the toy *car* is presented in Figure 5. The toy car was placed in a scene on top of a pile of other objects. The input car scene is shown in Figure 6. Preprocessing stages results in the computation of:

- A needle map containing the gradient space values at each pixel
- An edge map.
- A label map indicating the region to whch each pixel belongs.

The aspect classification steps performed are illustrated by the black nodes shown in the interpretation tree of Figure 7. Starting below the node at which the aspect is identified, the LSCD processing begins. The first node determines the mass center of the target region and declares that position to be the origin of the face coordinate system. The next node determines the average surface orientation of

Figure 6. Input Scene for **Dual** Photometric Stereo Experiment.

the target region and declares that to be the orientation of the z-axis of the primal face coordinate system.

The next two nodes complete the determination of the face coordinate system and estimate the body coordinate system. Figure 8 illustrates the estimated body coordinates overlaid **on** the input image.

The next step in the process is the determination of the edge pairs and fine-tuning of the object pose. **Figure** 9 illustrates the results of these processes.

4. A VAC Utilizing Image Synthesis for Prediction

The appearance-based system of Sato et al [21] was built for the **purpose** of recognizing specular objects. Specular objects pose a special problem for computer vision. Specularities are often the most prominent image features, and yet contain no brightness variations which can be used for edge detection or 3D shape analysis. Moreover, specular features may appear, disappear, or change shape abruptly with small variations in viewpoint. The presence of a specularity requires a precise configuration of illuminant, surface normal, and sensor, and therefore provides a powerful constraint on the underlying surface geometry. However, the constraint is purely local, and does little to constrain the object pose.

Specular reflections are found in nearly every imaging scenario. Metal, glass, plastic, and many other materials are highly specular. In addition to optical images, there are other imaging systems that are based on specular reflection. For example, radar is based on specular reflection, as are ultrasonic underwater imaging and

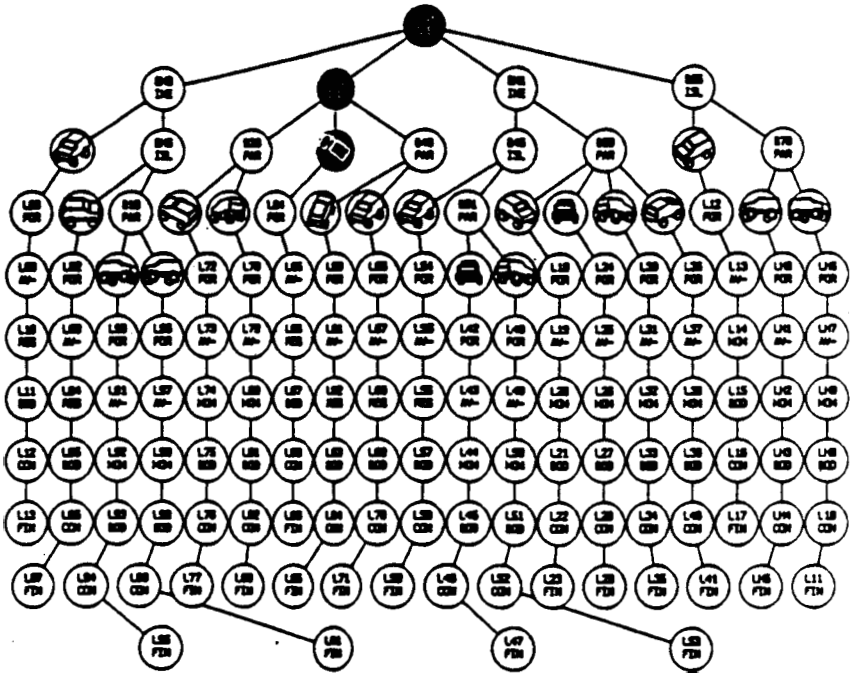
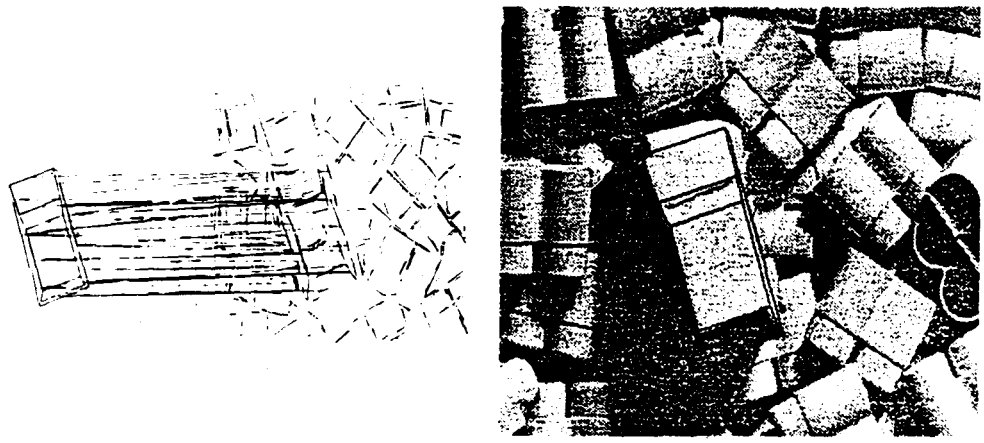


Figure 7. Execution of Aspect Classification Tree. Black nodes indicate the path followed during execution.

Figure 8. Estimated Body Coordinates Overlaid on Input Scene.



(a) (b)

Figure 9. Final Determination of Object Pose. (a) Correspondences between model and image edges. (b) Final pose overlaid on input scene.

medical sonography. Therefore, it is important to establish techniques to recognize objects using specular images.

4.L Explicit Object and Sensor Models

An analytic approach to appearance-based vision is impractical in the domain of specular objects. Because of interreflections between shiny surfaces, analytic prediction of specularities is extremely difficult. An alternative is the use of a sensor simulator, which generates object appearances based on both object and sensor models.

Sensor simulators are very similar to ray tracers of computer graphics. A 3D scene is described by a geometric modeler. A sensor simulator traces the path of light rays from pixels into the scene. Every time a ray hits an object surface, the ratios of reflected and transmitted energy are computed on the basis of the surface reflectance function and coefficient of refraction, respectively. The sensor simulator then traces both the reflected and refracted rays. When a ray reaches a light source, the energy emitted by the source is specified by the sensor model. The incident energies of all the rays towards a pixel are summed to determine the brightness value at the pixel and therefore predict the object appearance at the pixel.

One critical difference between a sensor simulator and a ray tracer is that sensor simulators maintain the symbolic correspondences between regions of the image and the object surfaces underlying the regions. In the case of specular objects, this

correspondence permits the analysis of which object surfaces produce strong, stable specularities, and from which directions specularities are visible on each surface.

Specular features are characterized by strong, distinct, and saturated brightness. In many cases, specularities *can* be extracted by the simple procedure of binary thresholding. Some specularities are easier to detect than others, however. In general, the size of a feature determines the ease with which it *can* be detected. For example, elongated specularities on a cylindrical surface are easy to detect, while a specular spot on a small sphere is difficult to detect. Thus, the detectability of a specular feature is related to the 3D shape of the surface underlying the feature.

While a specular feature might be easily detected, it could be quite unstable; that is, the specularity might be visible only over a small range of viewpoints, and a slight change in viewpoint could cause it to disappear. Such specularities are termed unstable, and are poor choices for use in recognition. The stability of a specular feature is related to the size of the collection of viewpoints from whch the feature can be detected.

To make the concept of stability clearer, consider a co-located camera and light source. Specular reflections are detectable when the object surfaces are nearly perpendicular to the camera line-of-sight. Consider the motion of the camera/light system around the surface of a sphere centered on an object, with the line of sight always toward the center of the sphere. If a small specular sphere is being imaged, a specular spot will be observed, and will continue to be observed for all viewpoints; hence the specularity arising from a spherical surface is extremely stable. Now consider a cube being imaged. Each planar surface only yields a speculanty when the line of sight is perpendicular to the surface, and the specularity disappears for small changes in viewpoint; hence, specularities from planar surfaces are unstable. The area on the viewing sphere corresponding to detectable viewpoints is a measure of stability of a specular feature.

Figure 10 illustrates the detectability and stability for specular features over four different surface types: planar, cylindrical, conical, and elliptical. As can be seen in the figure, planar surfaces have easily detectable specularities that are low in stability, while spherical surfaces have low detectability but high stability. Cylindrical and conical surfaces fall somewhere in between.

4.2. Predict and Analyze Appearances

The system employs an exhaustive method for appearance analysis. Assuming that range to the object is constant, all possible viewing directions can be represented as points on the unit sphere. A geodesic partition of the viewing sphere uniformly tessellates the sphere into small triangles. The center of each triangle is chosen to represent a viewing direction. Triangles can be further subdivided into smaller triangles to make the sampling as fine as desired. The sensor simulator is then used to generate synthetic images at each representative viewpoint.

Each image is processed to extract specularities, and the data structure created by the sensor simulator is used to determine the primitive component underlying each specularity. For N primitive components $P_1, P_2, \ldots, P_N$, a cell label can be defined as an N-tuple $(X_1, X_2, \ldots, X_N)$ such that $X_i = 1$ or 0 according to whether or not

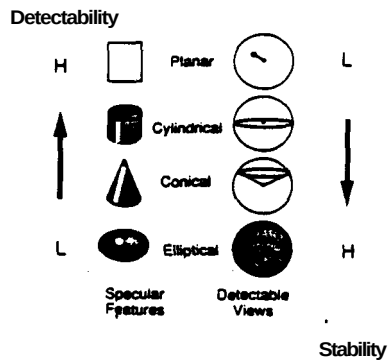


Figure 10. Detectability and Stability of Specular Features

component P_i gives rise to a detectable specularity at the viewpoint represented by the cell. Cells with identical cell labels are grouped together to form aspects. Thus, in the case of specular objects, aspects are defined with respect to detectable specularities.

4.3. Generate Recognition Strategy

As discussed above, specular features can vary in their characteristics of detectability and stability. For the purposes of object recognition, the system sorts specular features on the basis of detectability and stability in order to select the most effective features for aspect classification.

Detectability was defined above as the measure of the ease with which a specularity can be detected, and was related to the area of the specularity. The system uses as a measure of detectability the number of pixels of the largest appearance of a specularity, normalized by dividing by the area of the largest detected specularity.

Stability was defined as the area of the viewing sphere over which a given specularity is detected. In the case of a tessellated viewing sphere, this measure can be approximated by counting the number of cells within whch the specularity is detected, normalized by the total number of cells.

An evaluation function is required to combine the measures of detectability and stability into a measure of overall feature utility. For each aspect, the features are ordered by decreasing utility. At run-time, matches are made in order of decreasing utility.

Aspect classification is equivalent to rough localization. Finer localization is difficult in the case of specularimages because specular features change their shape drastically with small changes in viewpoint. Moreover, the exhaustive approach used in aspect determination may miss an unstable specularity that is only visible between two cells. Consequently, deformable template matching was selected as the procedure for fine localization.

Deformable template matching permits the template to deform according to certain constraints. An appearance is described as a combination of templates, each of which describes a specularity. The templates are interconnected conceptually by springs. The quality of match is measured by the sum of the internal deformation energy of the springs, and the external energy needed to fit each template to a real specularity. Thus, a deformable template *can* deform to find a match, even when a specularity changes shape or position. Moreover, matches *can* still be made even in the presence of accidental appearances or missing features.

A deformable template is prepared for each aspect using the appearance which is located at the center of the aspect. Specularities appear as spots or line segments, so each template consists of spots and line segments. Specular features are extracted from the central appearance. For an elongated feature, a line is fit to the feature and used to represent it in the template. For a spot feature, a point located at the center of the feature, is used to represent the feature in the template. A conceptual spring is located at each endpoint of a line feature, and at the point representing a spot feature. The spring energy is calculated from the displacement between the original and current location of the spring. Thus, the energy of a spot feature is a function of the displacement between the current and original position. For a line feature, the energy is a function of the displacement energy of the two endpoints.

4.4. Run-time Execution

Run-time execution is broken into two distinct stages: aspect classification and verification. In contrast to the system discussed in section 3, aspect classification does not uniquely classify an input image as an instance of an aspect. Rather, aspect classification is used to eliminate impossible aspects. Remaining aspects are input into the next stage, in which deformable template matching is used for verification.

4.4.1. Aspect Classification

Specular features can be very unstable. Small changes in viewpoint may cause a given specularity to appear, disappear, or change shape. Consequently, it is difficult to identify a single specularity, or the set of specularities that define an aspect, with complete confidence. Therefore, rather than employ a binary classification of an input image as an instance of an aspect, the run-time aspect classification system employs a continuous classification method based on the Dempster-Shafer methodology [22]. Figure 11 illustrates the classification method. For simplicity, the illustration is limited to aspects lying on a single great circle of the viewing sphere. Each match with a template from a single feature generates a likelihood distribution, in which a value close to 1 means that the corresponding aspect is very likely. Likelihood distributions from separate features are merged using Dempster-Shafer theory. As shown in the figure, each additional feature reduces the number of likely aspects and sharpens the peaks of the remaining ones. The likelihood values for impossible aspects decrease with each additional feature, while the likelihood values of possible aspects increase. After the evidence from every available feature has been applied, the overall likelihood distribution may still contain several peaks, each of which represents a possible aspect classification for the input image.

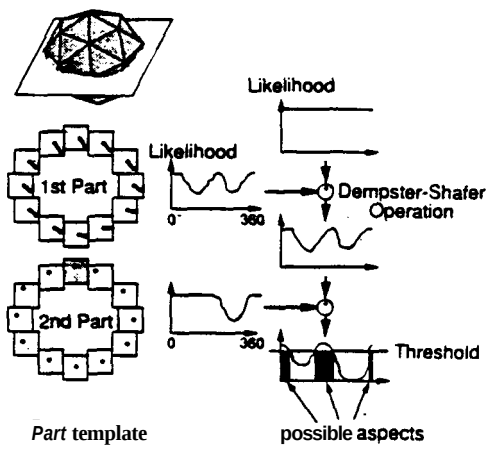


Figure 11. Aspect Classification Based on Evidential Reasoning.

4.4.2. Verification

The verification process determines the correct aspect by matching the input image to the complete template for each of the possible aspects. Each template can move over the entire image to minimize the total energy. The total energy is comprised of a weighted sum of constraint energy, and potential energy:

$$E_{total} = W_{constraints} E_{constraints} + W_{potential} E_{potential}$$

Potential energy represents the energy of the position of the template, while constraint energy represents the energy of the relations between template components. Potential energy is readily visualized as the height of the template in a potential field defined by the detected specularities in the image. Constraint energy is modeled by springs connecting the template points to image feature points; as the template deforms, the springs stretch and the constraint energy increases. Figure 12 illustrates template matching.

An optimization procedure is used to find the energy minimum. To avoid getting trapped in local minima, some noise is added to the total energy. The global minimum energy for each template specifies the quality of fit of the input image to the template. The best match is chosen by comparing the minimum energies for each of the candidate aspects.

4.5. Application

The VAC for specular object recognition has been applied to two different kinds of specular images: real optical specular images, and synthesized synthetic aperture radar (SAR) images. In this section, the results for real optical specular images are reported. Results for SAR images are reported in Sato et al [21].

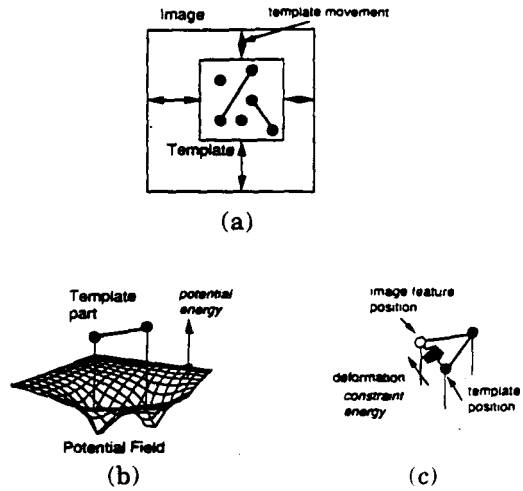


Figure 12. Deformable Template Matching. (a) Template. (b) Potential energy. (c) Constraint energy.

For this experiment, a real toy airplane was constructed. The sensor used was a tv camera with a co-located light source. The sensor parameters were obtained by calibration. Figure 13 shows a real specular image of the toy airplane.

The object was modeled using the Vantage geometric modeler [2]. Object aspects were determined using the exhaustive method. Since man-made objects such as the airplane have only a few stable poses, an aspect map over the entire viewing sphere was not generated. Instead, only the appearances of the airplane from the equator of the viewing sphere were considered. Sample images were generated on the equator at 5 deg increments for a total of 72 samples. Appearances were generated using a sensor simulator [9]. Figure 14 illustrates the object model and some sample appearances.

The concentric arcs in Figure 15 correspond to the visibility maps for each primitive surface. The outermost arc corresponds to the detectable directions of the rear fuselage. The arc is unbroken - the part can be observed from all viewing directions. The top-left image in shows the set of possible appearances of specular features arising from the rear fuselage as a function of viewing direction The other images correspond to other arcs of the visibility map. Some of the arcs in the map are broken; the missing arc regions correspond to viewpoints from which the primitive surface is not visible. The figure shows the features in their computed order of significance.

To test the resulting recognition program, a real specular image of the toy airplane was obtained and input to the system. The first step in the recognition process is aspect classification, in which possible aspects are searched by matching with partial templates. Figure 16 shows the input image and the results of matching to the first

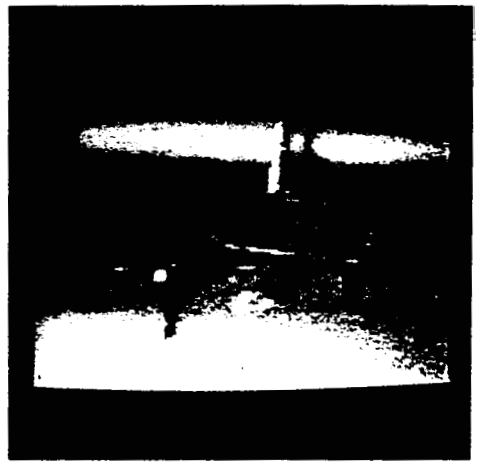
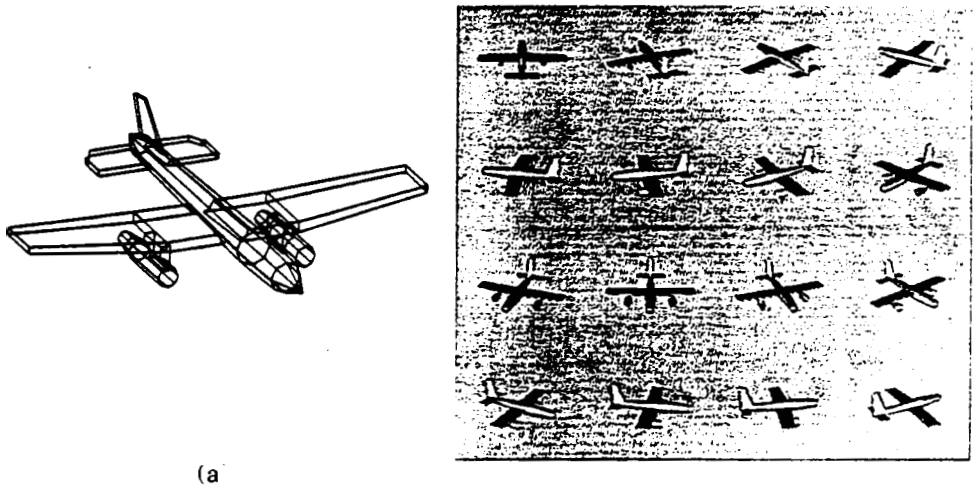


Figure 13. Real Specular Image of Toy Airplane.



(a)

(b)

Figure 14. Model Airplane and Predicted Specular Appearances. (a) **Airplane model.** (b) **Sample appearances.**

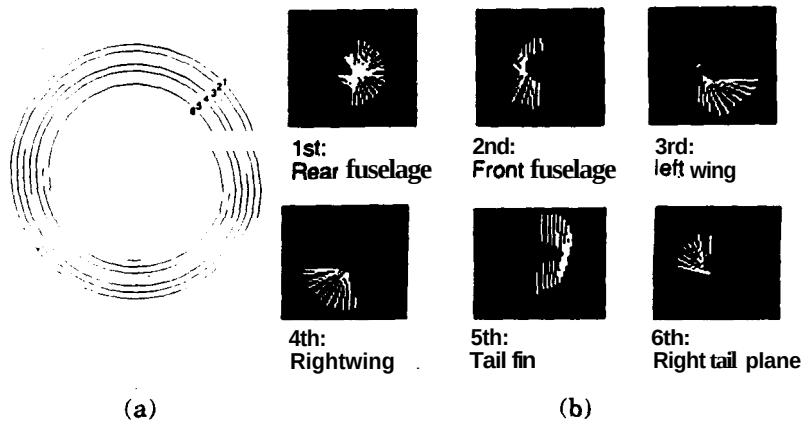


Figure 15. Visibility and Significance of Features in Optical Experiment. (a) Visibility map. (b) Specular features in order of significance.

three partial templates. The figure clearly shows the narrowing of the likelihood distribution as additional matching is performed. The result of the aspect classification step was the selection of aspects at 45 deg, 115 deg, 125 deg, 165 deg, and 170 deg.

Following aspect classification, verification was performed using deformable template matching. Figure 17 illustrates the verification step. One template was used for each of the aspects selected. In each case the template changed its shape to match the specularities in the input image and converged to the shapes shown by the white lines superimposed on the copies of the real image in the figure. The minimum energy, and hence the best match, was obtained for the aspect at 170 deg.

5. Summary

In this paper, we presented the paradigm of appearance-based vision, which is a paradigm for building object recognition systems. The paradigm is called appearance-based, since an integral step is the prediction and analysis of object appearances. An appearance-based system is called a vision algorithm compiler, or VAC. The input to a VAC is a set of object and sensor models, and the output is an executable object recognition program.

Appearance-based systems share four principle defining characteristics:

- *Two-stage* **Process** A VAC operates in two distinct stages. The first stage is performed off-line, and consists of the analysis of appearances and the generation of an object recognition program. The second stage is performed on-line, and consists of the execution of the previously generated program.

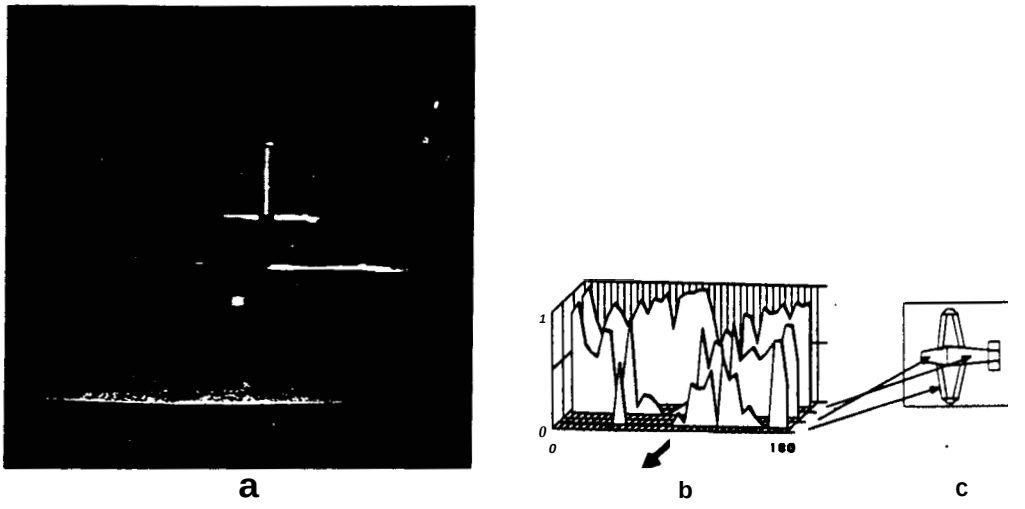


Figure 16. Aspect Classification Stage for Optical Experiment. (a) Input optical image. (b) Accumulation of evidence. (c) Airplane parts.

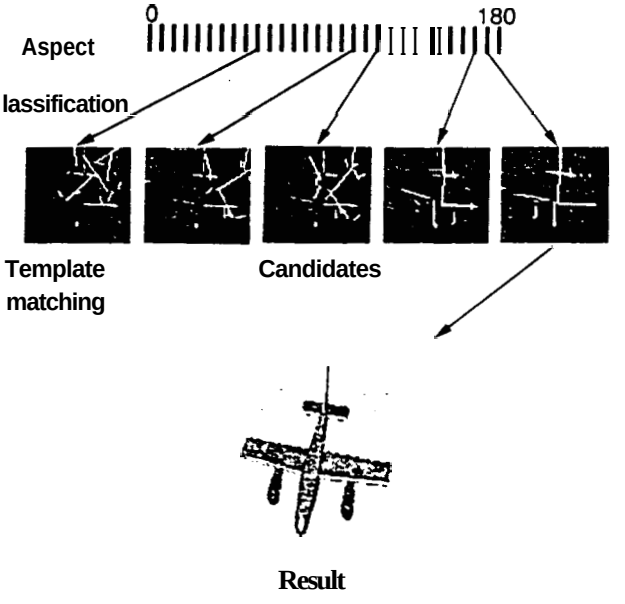


Figure 17. Verification Stage for Optical Image.

- ***Explicit Object and Sensor Models*** A VAC embodies an overall approach to object recognition that *can* be applied to a variety of objects and sensors. Therefore, explicit and exchangeable models *are* utilized. Sensor models specifies the features detectable by the sensor, along with procedures to compute the detectability and reliability of each feature. Object models include geometric and photometric properties of the objects.

- ***Appearance Prediction and Analysis*** Objects are recognized based on their appearances in images. Therefore, the prediction and analysis of appearance is fundamental to generating competent object recognition programs. A VAC predicts appearances based on the information in object and sensor models. The range of appearances may be determined analytically or exhaustively, and the appearances may be predicted analytically or through image synthesis.

Acknowledgements

The authors thank Takeo Kanade and the members of VASC (Vision and Autonomous System Center) of Carnegie Mellon University for their valuable comments and discussions.

REFERENCES

1 F. Arman and J. K. Aggarwal. Automatic generation of recognition strategies using cad models. In *Proc. of the IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 124–133. IEEE, 1991.

2 P. Balakumar, J.C. Robert, R. Hoffman, K. Ikeuchi, and T. Kanade. Vantage: A frame-based geometric/sensor modeling system – programmer/user's manual v1.0. Technical Report CMU-RI-TR-91-31, Carnegie Mellon University, Robotics Institute, 1991.

3 P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, March 1985.

4 R. C. Bolles and P. Horaud. 3DPO: A three-dimensional part orientation system. In T. Kanade, editor, *Three-Dimensional Machine Vision*, pages 399–450. Kluwer, Boston, MA, 1987.

5 O. I. Camps, Shapiro. L. G., and R. M. Haralick. PREMIO: an overview. In *Proc.* of *the IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 11–21. IEEE, 1991.

6 S. Chen and H. Freeman. On the characteristic views of quadric-surfaced solids. In *Proc. of the IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 34–43. IEEE, 1991.

7 R.T. Chin and C.R. Dyer. Model-based recognition in robot vision. ACM *Computing Surveys*, 18(1):67–108, March 1986.

8 P. J. Flynn and A K. Jain. BONSAI: 3D object recognition using constrained search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(10):1066–1075, 1991.

9 Y. Fujiwara, S. Nayar, and K. Ikeuchi. Appearance simulator for computer vision research. Technical Report CMU-RI-TR-91-16, Carnegie Mellon University, The Robotics Institute, 1991.

10 C. Goad. Special purpose automatic programming for 3D model-based vision. In *Proc. of DARPA Image Understanding Workshop*, pages 94–104. DARPA, 1983.

11 C. Hansen and T. C. Henderson. CAGD-based computer vision. *ZEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(11): 1181–1193, 1989.

12 M. Hebert and T. Kanade. Outdoor scene analysis using range data. In *Proc. of Znt. Conf. on Robotics and Automation*, pages 1426–1432, San Francisco, April 1986. IEEE Computer Society.

13 K. S. Hong, K. Ikeuchi, and K. D. Gremban. Minimum cost aspect classification: a module of a vision algorithm compiler. In *Proc. 20th Znt. Conf. on Pattern Recognition*, pages 65–69, 1990. A longer version is available as CMU-CS-90-124, School of Computer Science, Carnegie Mellon University (1990).

14 K. Ikeuchi. Determining a depth map using a dual photometric stereo. *The International Journal of Robotics Research*, 6(1):15–31, 1987.

15 K. Ikeuchi. Generating an interpretation tree from a CAD model for 3-D object recognition in bin-picking tasks. *International Journal of Computer Vision*, 1(2):145–165, 1987.

16 K. Ikeuchi and K. S. Hong. Determining linear shape change: toward automatic generation of object recognition programs. *CVGIP:Image Understanding*, 53(2):154–170, 1991.

17 K. Ikeuchi and T. Kanade. Towards automatic generation of object recognition programs. *Proc. of IEEE*, 76(8):1016–1035, August 1988. A slightly longer version is avaiable as CMU-CS-88-138, School of Computer Science, Carnegie Mellon University (1988).

18 J. J. Koenderink and A. J. Van Doom. Internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32(4):211–216, 1979.

19 D. J. Kriegman and J. Ponce. Computing exact aspect graphs of curved objects: solids of revolution. *Int. Journal of Computer Vision*, 5(2): 119–135, 1990.

20 H. Platinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *Int. Journal of Computer Vision*, 5(2): 137–160, 1990.

21 K. Sato, K. Ikeuchi, and T. Kanade. Model based recognition of specular objects using sensor models. *CVGIP: Image Understanding*, 55(2):155–169, 1992.

22 G. A. Shafer. *A Mathematical Theory of Evidence*. Princeton Univ. Press, Princeton, NJ, 1976.

23 P. Suetens, P. Fua, and A. J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–61, 1992.