Aspect Graph based Modeling and Recognition with an Active Sensor: A Robust Approach

Sumantra Dutta Roy
Department of CSE,
I. I. T., Hauz Khas,
New Delhi-110016,
INDIA.
sumantra@cse.iitd.ernet.in

Santanu Chaudhury *
Department of EE,
I. I. T., Hauz Khas,
New Delhi-110016,
INDIA.
santanuc@ee.iitd.ernet.in

Subhashis Banerjee
Department of CSE,
I. I. T., Hauz Khas,
New Delhi-110016,
INDIA.
suban@cse.iitd.ernet.in

Abstract

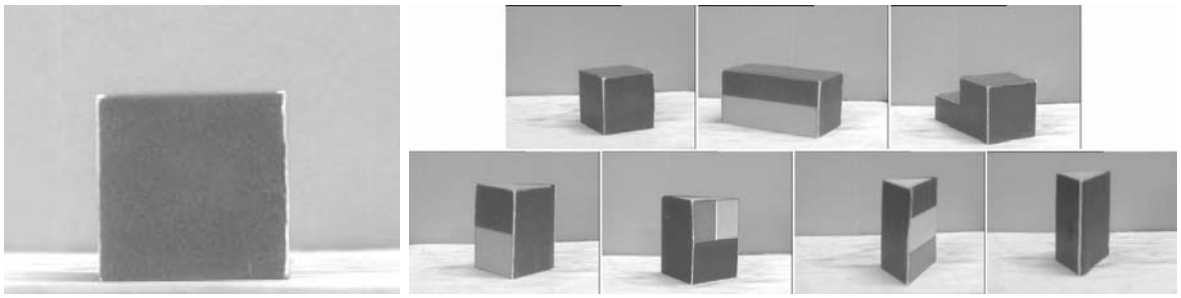
A single view of a 3-D object may not contain sufficient features to recognize it unambiguously. A further complication arises when noise and non-adaptive thresholds introduce errors in the feature detection process. This paper presents a robust aspect graph-based modeling and recognition system for isolated 3-D objects. This paper presents a characterization of errors in aspect data, and a robust aspect graph construction algorithm. The knowledge representation scheme encodes feature-based information about objects as well as uncertainty in the recognition process. The system uses a reactive next view planning scheme for isolated 3-D object recognition. We present extensive results of our strategies applied on a reasonably complex experimental set.

Keywords Aspect Graph, Feature Detection Errors, Aspects, Classes, Active Vision, Reactive Planning, 3-D Object Recognition

*Author for correspondence

1 Introduction

Model-based object recognition systems(e.g., [1, 2, 3]) typically extract features from an input image of the object and compare it with descriptions of the object stored in a model-base. This leads to the formation of different hypotheses about the identity of the object and other characteristics (e.g., its position and pose). Most model-based object recognition systems use information from a single view of an object (e.g., [1, 2, 3]). These approaches assume that the information extracted from a single image of a particular object is different from that of another object in the model base. For 3-D object recognition, the imaging process involves a projection of a 3-D entity to the 2-D image plane - this leads to loss of information about the object. One needs an effective representation of properties(geometric, photometric, etc.) of objects from images which are invariant to the view point, and should be computable from image information. Invariants may be colour-based (e.g., [4]), photometric (e.g., [5]) or geometric (e.g., [3]). For example, Burns, Weiss and Riseman prove a theorem in [6] that invariants cannot be computed for a set of 3-D points in general position, from a single image. Invariants can only be computed for a constrained set of 3-D points. The authors in [3], for example use the inherent symmetry present in an object, or a particular configuration of objects to compute invariants for recognition – this severely restricts the recognition system to only a specific class of objects that can be recognized (e.g., rotationally symmetric objects, translationally repeated objects, etc.) However, a single view of a 3-D object may not contain sufficient features to recognize the object unambiguously. In fact, two objects may have all views in common with respect to a given feature set, and may be distinguished only through a sequence of views. The authors in [7] cite a simple example of a sedan and a station wagon having indistinguishable front ends, but different side views. As another example, let us consider a setup with one degree of freedom (hereafter, DOF) between the object and the sensor - Figure 3(a) shows an example of such a setup. As feature detectors, let us consider horizontal and vertical lines, without making any metric measurements. Figure 1(a) shows the given view, which could have corresponded to any of the objects in Figure 1(b). It is not possible to determine which object the given view corresponds to, given only the single view in Figure 1(a).



(a) (b)

Figure 1: (a) The given view of an object, and (b) the objects which this view could have corresponded to (see text for details)

While invariants may be important for recognizing some views of an object, they cannot characterize all its views – except in a few specific cases (e.g., rotationally symmetric objects). We often need to recognize 3-D objects which because of their inherent asymmetry, cannot be completely characterized by an invariant computed from a single view. For example, certain self-occluded features of an object can become visible if we change the viewpoint. In order to use multiple views for an object recognition task, one needs to maintain a relationship between different views of an object, possibly described by image-computable invariants.

Aspect graphs are a convenient tool for multi-view representation of a 3-D object. Koenderink and van Doorn [8] define **aspects** as topologically equivalent classes of object appearances. Since sensors may be of different types(geometric, photometric), Ikeuchi and co-workers generalize this definition – object appearances may be grouped into equivalence classes with respect to a feature set. These equivalence classes are aspects [9]. A **Class (or, Aspect-Class)** is a set of aspects, equivalent with respect to a feature set. Aspect graphs partition the space of viewpoints around an object into maximal regions. Every viewpoint in each region (an aspect) gives the same view of the object, with respect to a given feature set. Figure 2 shows an example of aspects of a cube. In an aspect graph, nodes represent different aspects, and arcs link adjacent aspects (Arcs have been omitted in Figure 2 for clarity). The advantage of using aspect graphs lies in its representation of all views of an object with respect to a given feature set. An image represents a view of an object – the recognition process, therefore is simplified in matching view-based

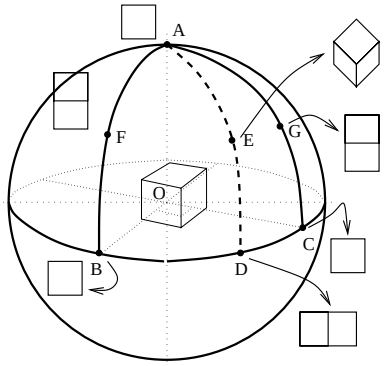


Figure 2: Aspects of an object

features in the given image with those in the model base.

Aspect graphs may be classified as Exact aspect graphs, and Approximate aspect graphs (hereafter, AAGs) Analytical approaches are used to construct exact aspect graphs – directly from object shapes and surface characteristics. A limitation of such an approach is its applicability to only a specific class of objects. Further, existing approaches usually employ geometric features obtained from CAD models (e.g., algebraic curves, triple points). (e.g., [10], [11], [12],[13], and [14]). Uniform partitioning approaches(e.g., [15], [16], [17]) tessellate the viewing space around an object into uniform partitions. Adjacent viewpoints which give the same appearance of the object with respect to a feature set, are grouped together to form an aspect. The granularity of the tessellation determines how close an AAG is to the exact aspect graph. Thus, the uniform partitioning approach for AAG construction is independent of the object shape and structure, the sensor, or the feature set. *The representation method in exact aspects graphs and AAGs is the same* – nodes representing aspects and arcs joining adjacent aspects. They differ only in the method of construction.

Figure 3(a) shows an example of the 1-DOF case – The sensor can move around the object in a circle, at a fixed height. Figure 3(b) shows different aspects and classes of an object (orthographic projection assumed). In this example, we illustrate the difference between an exact aspect graph and an AAG. Let us consider ASPECT 8 in Figure 3(b). In an exact aspect graph, ASPECT 8 would be visible at just one point, and not a region. For an AAG, however one of the following situations could occur. If the tessellation of the viewing circle is not fine enough, one may

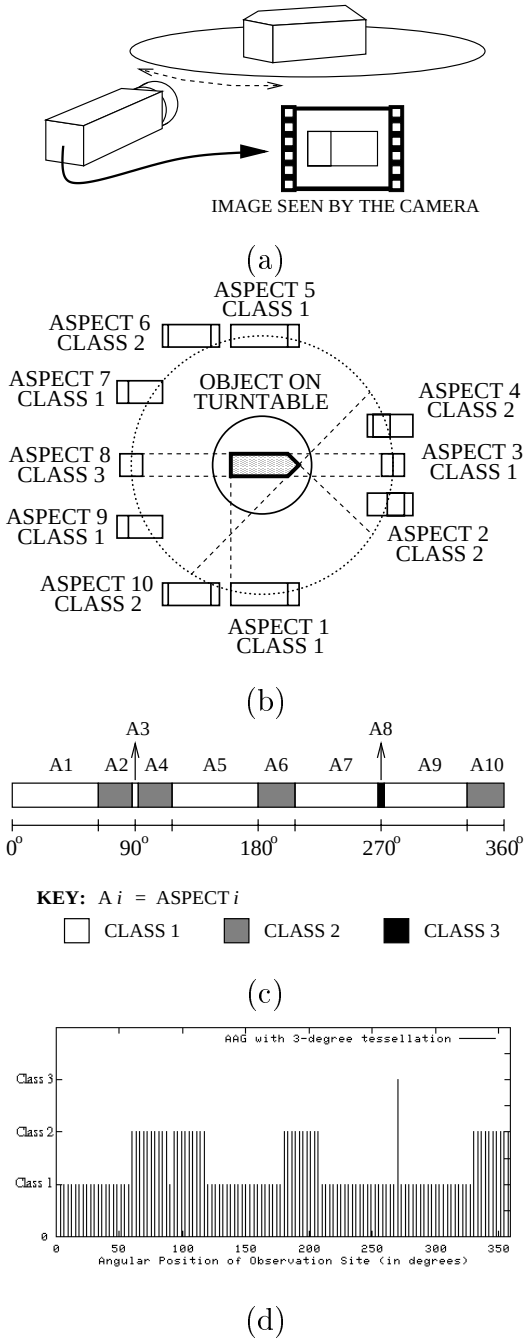


Figure 3: (a) An object and a sensor in a 1-DOF setting, (b) aspects and classes of the object, (c) Gantt chart representation of the aspect graph, and (d) an AAG of the object, shown as a bar plot.

not obtain this aspect at all, and ASPECT 7 and ASPECT 9 would appear as a merged new aspect. Alternatively, for a fine tessellation, ASPECT 8 may appear in a region rather than as a point – this is due to the limitations of the feature detection process, which might interpret two very closely spaced vertical lines in an image as one. Figure 3(c) shows the AAG of the above object represented as a Gantt chart - each aspect of the object is represented by a shaded rectangle, proportional to its angular extent on the perimeter of the viewing circle. Different shading patterns represent different classes. Different aspects belonging to the same class share the same shading pattern, but have different locations in the aspect graph. In this example, we illustrate the case when a point in an exact aspect graph appears as a region in an AAG (e.g., ASPECT 8) Figure 3(d) shows a bar plot presentation of an AAG. Here, the AAG has ASPECT 8 (CLASS 3) represented as one point itself, as in the corresponding exact aspect graph. While the exact aspect graph and the AAG may differ as described above, their representation is the same – nodes representing aspects and links connecting adjacent aspects.

We define an **Active Sensor** as one which can be purposively controlled. In the context of a vision task, an active sensor is one where vision guided feedback can be used to position the sensor. In other words, the information from one sensor reading can be used to position the sensor at the next point. The 1-DOF camera and turntable example of Figure 3(a) is such a sensor. Such a sensor may be used both for AAG construction, as well as for recognition. We elaborate on this in the next section.

1.1 Problem Formulation

Given an active sensor and a set of feature detectors, the fundamental problems involved in a multiple view-based recognition system are

- the design of a suitable modeling scheme, and
- an identification mechanism which can exploit properties of the sensing process and the modeling scheme.

These two problems have been addressed in this paper.

An active sensor has a critical role to play in both the above cases. For the AAG construction, the sensor is moved around the object – it is positioned at various viewpoints, and the features extracted from the sensor information form the input to an AAG construction algorithm. For the recognition task, the active sensor takes input from the given pose of the object. The sensor input is processed – this information decides the next viewpoint where the sensor has to be placed to get more information about the object. Here, we address the problem of planning the next view for unambiguous recognition of the object.

The input to the system comes from the output of feature detectors. While constructing the model base, the information from all views of an object comes from the output of feature detectors. The same feature detectors are used in the recognition phase - when the system derives information from a given view of an object, processes it and if required, decides on further views. Factors such as noise and non-adaptive thresholds may corrupt the output of a feature detector. Thus in either of the two cases mentioned above, one has to account for feature detection errors. Here, we consider the problem of formulating an AAG-based modeling scheme which can take care of feature detection errors. We also address the problem of handling feature detection errors in the recognition phase using probabilistic reasoning, based on the models thus constructed.

1.2 Relation with Other Work

Existing aspect graph construction algorithms do not account for errors in the feature detection process. For analytical approaches, one would need very precise models of not only the noise process, but also the sensors and detectors, and the imaging process. Even in the uniform partitioning approach, no related work accounts for feature detection errors in AAG construction. The authors in [7] mention only one type of error – errors on aspect boundaries. However, they do not model such an error in the AAG construction phase, nor do they account for it in the recognition phase.

Many 3-D object recognition strategies use aspect graphs (for example, [18], [19], [7], [21]). With an active sensor (i.e., a sensor used for active vision tasks, where one has purposive control over the movement of the sensor with respect to the object of interest), object recognition involves

identification of a view of an object and if necessary, planning further views. We compare different 3-D object recognition systems which use active sensors, on basis of the following properties:

1. *Nature of the next view planning strategy*

   The planning scheme should ensure adequate discriminatory ability between views common to more than one object in the model base. While the scheme of Maver and Bajcsy [20] is on-line, that of Gremban and Ikeuchi [7] is not. An off-line approach may not always be feasible, due to the combinatorial nature of the problem.

2. *Uncertainty handling capability of the hypothesis generation mechanism.*

   Approaches such as that of Maver and Bajcsy [20], and that of Gremban and Ikeuchi [7] are essentially deterministic. An uncertainty-handling mechanism makes the system more robust and resistant to errors compared to a deterministic one. Dickinson *et al.* [21] use Bayesian methods to handle uncertainty, while Hutchinson and Kak [19] use the Dempster-Shafer theory.

3. *Efficient representation of domain knowledge.*

   Dickinson *et al.* [21] use a hierarchical representation scheme based on volumetric primitives, which are associated with a high feature extraction cost. Due to the non-hierarchical nature of Hutchinson and Kak's system [19], many redundant hypotheses are proposed, which have to be later removed through consistency checks.

4. *Speed and efficiency of algorithms for both hypothesis generation and next view planning.*

   Hypothesis generation should be fast, and incur minimal error. In Hutchinson and Kak's system [19], although the polynomial-time formulation overcomes the exponential time complexity associated with assigning beliefs to all possible hypotheses, their system still has the overhead of intersection computation in creating common frames of discernment. Though Dickinson *et al.* [21] use Bayes nets for hypothesis generation, their system incurs the overhead of tracking the region of interest through successive frames.

In this paper, we present a novel aspect graph-based modeling and recognition system for isolated 3-D objects. This is based not on CAD data, but on image-based properties. Our system

works with noisy sensor data - handling and accounting for errors both at the aspect graph construction level, as well as at the object recognition level. This has not been done before. We present a classification of errors in aspect graphs. The paper gives an algorithm to construct an AAG, given noisy sensor data. We also present a new function to evaluate different AAG construction algorithms. Our hierarchical knowledge representation scheme facilitates recognition and the planning process. The hierarchy itself enforces different constraints to prune the set of possible hypotheses, unlike that in [19]. The planning process is reactive - it utilizes the current observation and past history for identifying a sequence of moves to disambiguate between similar objects. We handle feature detection errors through a probabilistic hypothesis generation mechanism. The hypothesis generation mechanism has low-order polynomial-time complexity. Our system is independent of the type of features used, unlike that of [21]. For this work, we consider one DOF between the object and the sensor. We present results of over 100 experiments with our recognition scheme on two sets of models. In recognizing 3-D objects from a single view, recognition systems often use complex feature sets [2]. Complex feature sets are often model base-specific and complex, Our experimentation shows the use of simple features and suitably planned multiple views to recognize fairly complex 3-D objects.

The organization of the rest of the paper is as follows: Section 2 presents our knowledge representation scheme. We discuss AAG generation from noisy sensor data in Section 3.2. Section 4 discusses our proposed scheme for recognizing a given 3-D object (whose AAG was constructed as above), using the same noisy feature detectors. In Section 5, we demonstrate the working of our system on two sets of objects. We summarize the salient features of our scheme and discuss areas for further work in the concluding section.

2 The Knowledge Representation Scheme

The input to the object recognition system comes from the feature detectors. In this context, we define the following term:

Feature-Class A Feature-Class is a set of equivalent aspects defined for one particular feature.

We propose a new knowledge representation scheme encoding domain knowledge about the

object, relations between different aspects, and the correspondence of these aspects with feature detectors. Figure 4 illustrates an example of this scheme. We use this knowledge representation

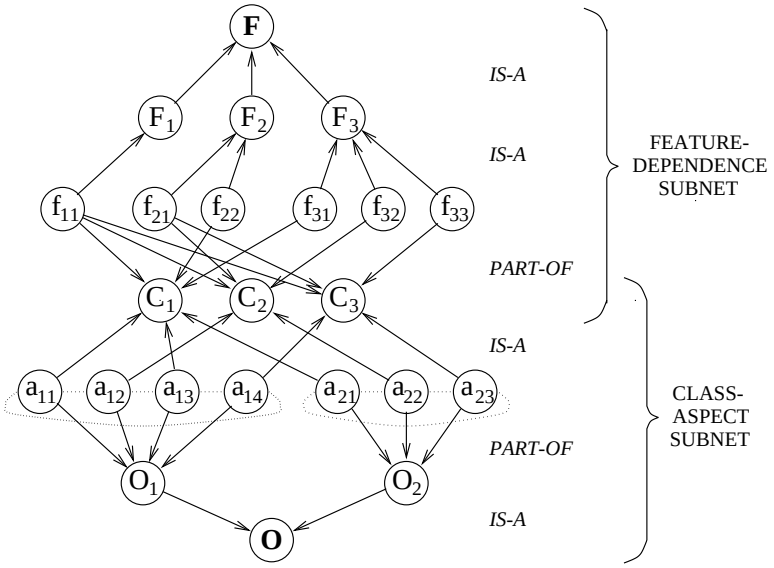


Figure 4: The knowledge representation scheme: an example

scheme both in belief updating(Section 4.2) as well as in next view planning(Section 4.3.2). Our knowledge representation is not just an aspect graph – it encodes the hierarchical relation between feature detectors, feature-classes, their relation to different aspect-classes, the grouping of different aspects across objects into classes, and the relation of aspects with their corresponding objects. The representation scheme consists of two parts:

1. **The Feature-Dependence Subnet**

   In the feature-dependence subnet

   - **F** represents the complete set of features $\{F_j\}$ used for characterizing views
   - A Feature node F_j is associated with feature-classes f_{jk}.

     Factors such as noise and non-adaptive thresholds can introduce errors in the feature detection process. Let p_{jlk} represent the probability that the feature-class present is f_{jl}, given that the detector for feature F_j detects it to be f_{jk}. We define p_{jlk} as the ratio of the number of times the detector for feature F_j interprets feature-class f_{jl} as

f_{jk}, and the number of times the feature detector reports the feature-class as f_{jk}. The F_j node stores a table of these values for its corresponding feature detector.

- A class node C_i stores its *a priori* probability, $P(C_i)$. A link between class C_i and feature-class f_{jk} indicates that f_{jk} forms a subset of features observed in C_i. This accounts for a *PART-OF* relation between the two. Thus, a class represents an n-vector $[f_{1j_1}\ f_{2j_2}\ \ldots\ f_{nj_n}]$. Since a class cannot be independent of any feature, each class has n input edges corresponding to the n features.

2. **The Class-Aspect Subnet**

   The class-aspect subnet encodes the relationships between classes, aspects and objects.

   - **O** represents the set of all objects $\{O_i\}$
   - An object node O_i stores its probability, $P(O_i)$
   - An aspect node a_{ij} stores its angular extent θ_{ij} (in degrees), its probability $P(a_{ij})$, its parent class C_j, and its neighbouring aspects
   - Aspect a_{ij} has a *PART-OF* relationship with its parent object O_i. Thus, the 3-tuple $\langle\ O_i,\ C_j,\ \theta_{ij}\ \rangle$ represents an aspect. Aspect node a_{ij} has exactly one link to any object (O_i) and exactly one link to its parent class C_j.

3 Construction of an AAG

An aspect Graph is a representation of an object in terms of its aspects – vertices represent aspects, each vertex has a link with its adjacent aspects. In this section, we present a characterization of different errors in raw aspect data. We present a new AAG construction algorithm using image-based features. The algorithm accounts for feature detection errors during AAG construction, and also helps in robust class recognition(Section 4.2). An AAG is constructed for each object in the model base. The output of the AAG construction algorithm is as follows:

- a list of aspects with adjacency information
- a list of classes corresponding to these aspects

- feature classes, corresponding to the output of each feature detector on all views of each object. Feature classes may be spurious (not having a corresponding aspect-class).
- ASSOC_TABLE estimates (described below in Section 3.2), which are used for computation of p_{jlk} values

The above information enables automated construction of the knowledge representation scheme for the given model base, using an active sensor.

3.1 Errors in Raw Aspect Data

The tessellation of the viewing space determines the positions in the viewing space around the object. The first step in the construction of an AAG involves applying our set of feature detectors on an image of the object taken from every such viewing position. In an error-free situation, AAG construction involves grouping adjacent viewpoints having the same feature vector, into an aspect. However, factors such as noise and non-adaptive thresholds introduce errors in these feature vectors. Let the term 'raw aspect data' denote the collection of feature vectors obtained at the set of sites in the tessellated viewing space. We refer to aspects and classes obtained from raw aspect data as **aspect-candidates** and **class-candidates**, respectively. Thus, we can have erroneous aspect-candidates and class-candidates, but no erroneous aspects and classes.

We may characterize an error in an aspect graph by the position (site) at which the error is introduced, and the 'value' or feature vector label at that site. One can also characterize an error in terms of the raw aspect data available, in terms of aspect-candidates, class-candidates

and their parameters. We define the following terms:

$\mathcal{A}$: The set of all aspect-candidates

$\mathcal{C}$: The set of all class-candidates

$CLASS\_CAND : \mathcal{A} \rightarrow \mathcal{C}$

$\mathcal{A}_c(\mathcal{A}_c \subseteq \mathcal{A}) \triangleq \{\ \alpha \mid CLASS\_CAND(\alpha) = c, \qquad \text{where } \alpha \in \mathcal{A}, c \in \mathcal{C}$

N_{min} : the minimum total number of sites at which a class-candidate should be present in the entire model-base to be called a 'Valid class/class-candidate'

$\mathcal{G}_a$: set of aspect-candidates in the neighbourhood of aspect-candidate a

$P(c' \mid c)$: the probability of the class-candidate actually being c', given that class-candidate c has been observed, for the given model base

θ_α : angular width of aspect-candidate α, in terms of the number of sites it occupies

Θ_{min} : the minimum extent which an aspect-candidate must have in order for it to be called a 'Valid aspect-candidate'

Θ_p : the minimum extent which an aspect-candidate must have in order for it to be called a 'Prominent aspect-candidate'$(\Theta_p \geq \Theta_{min})$

$\mathcal{A}_c^g \triangleq \{\ \alpha \mid \theta_\alpha \geq \Theta_{min}, \text{where } \alpha \in \mathcal{A}_c\}$

we classify errors in aspect data into the following five categories:

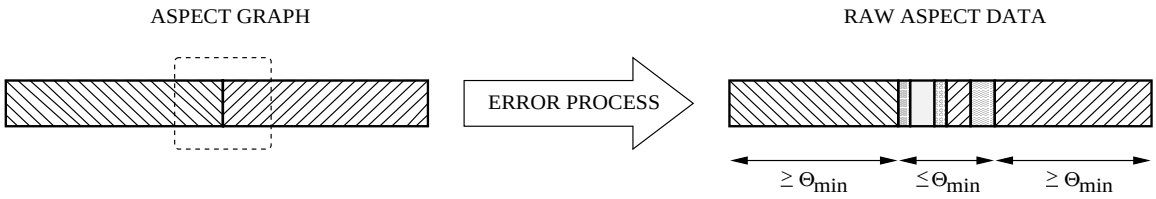
Type I Error A Type I error is present as a small transition region between two aspects of a correct AAG. (This corresponds to the "border effect" in [7]). In the raw aspect data, a Type I error can be described as follows:

$$\theta_{a_i}, \theta_{a_j} \geq \Theta_{min}$$

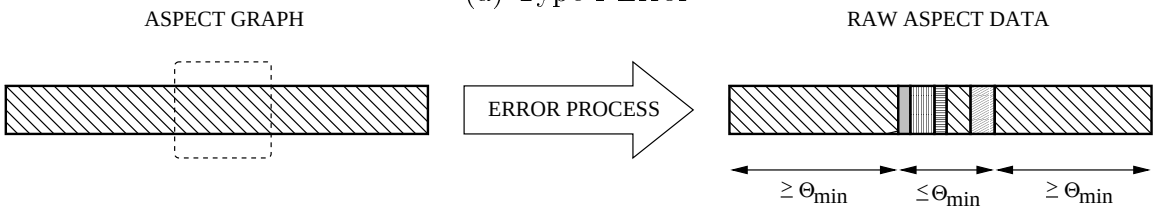
$$CLASS\_CAND(a_i) \neq CLASS\_CAND(a_j), \text{ and}$$

$$\sum_{a_k \ between \ a_i \ and \ a_j} \theta_{a_k} < \Theta_{min}$$

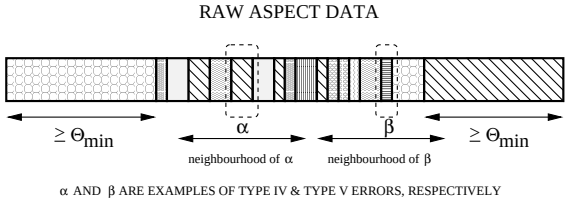
Here, a_i and a_j are two valid aspect-candidates belonging to different class-candidates such that there is a small region of width $\leq \Theta_{min}$ between them. Figure 5(a) illustrates an example of this type of error.



(a) Type I Error



(b) A Region having Type II and III Errors (see text)



(c) Types IV and V Errors

Figure 5: Errors in raw aspect data: a pictorial representation

Type II Error A Type II error can be present in a small isolated error region inside an aspect of a correct AAG. In the raw aspect data, we observe a small region of width $\leq \Theta_{min}$ in between two valid aspect-candidates a_i and a_j (Figure 5(b)):

1. $\theta_{a_i}, \theta_{a_j} \geq \Theta_{min}$
2. $\sum_{a_k} \theta_{a_k} < \Theta_{min}$ $\forall$ a_k between a_i and a_j
3. $CLASS\_CAND(a_i) = CLASS\_CAND(a_j) = c'$ (say)

For the small enclosed error region, those aspect-candidates constitute a Type II error, whose class-candidates observed are associated with the class-candidates of the enclosing aspect-candidates, for the particular model or the entire model base. Such 'association errors' characterize the response of a feature detector to a particular feature, or a combination of features - for the given model base. We may describe this as follows:

1. $\sum_{\alpha \in \mathcal{A}_c} \theta_\alpha < N_{min}$, where $CLASS\_CAND(a_k) = c$
2. $\frac{\sum_{\beta \in \mathcal{A}_c^g} \theta_\beta}{\sum_{\alpha \in \mathcal{A}_c} \theta_\alpha} <$ a threshold $T_1,\ 0 < T_1 < 1,$ and
3. $P(c' \mid c) \geq$ a threshold $T_2,\ 0 < T_2 < 1$

The probability estimates are characteristics of the given raw aspect data and are obtained from it.

Type III Error A small isolated error region inside an aspect of a correct aspect graph (as defined above) may contain either Type II or Type III errors. In a Type III error, there is no association of an error with a particular class-candidate.

Type IV Error Type IV errors are scattered in an aspect of a correct AAG. They can be present in a 'large' region between two valid aspect-candidates:

1. $\theta_{a_k} < \Theta_{min},\ \forall\ a_k$ between a_i and a_j
2. $\sum_{a_k} \theta_{a_k} \geq \Theta_{min}$

An aspect-candidate α constitutes a Type IV error if the following conditions hold:

1. $\max_{\beta \in \mathcal{A}_c} \theta_\beta \ \geq \Theta_{min}$, where $CLASS\_CAND(\alpha) = c$
2. $\exists \gamma \in \mathcal{G}_\alpha$ for which $CLASS\_CAND(\gamma) = CLASS\_CAND(\alpha)$

A Type IV error indicates the possibility of aspect-candidates with the same class-candidate being part of a single aspect. This considers those aspect-candidates which themselves do not have enough extent to be considered valid aspect-candidates. However, there are other valid aspect-candidates associated with this class-candidate. Figure 5(c) illustrates an example of this type of error.

Type V Error Like Type IV errors, Type V errors, too can be present in a 'large' region between two valid aspect-candidates (as defined above). An aspect-candidate α constitutes a Type V error if either

- $\forall \gamma \in \mathcal{G}_\alpha,\ CLASS\_CAND(\gamma) \neq CLASS\_CAND(\alpha)$, or
- $\exists \gamma \in \mathcal{G}_\alpha$ for which $CLASS\_CAND(\gamma) = CLASS\_CAND(\alpha)$, but $\max_{\beta \in \mathcal{A}_c} \theta_\beta \ < \Theta_{min}$, where $CLASS\_CAND(\alpha) = c$

Figure 5(c) illustrates an example of a Type V error. Type V errors are very difficult to correct since they give very little indication as to which aspect(in the correct AAG) they might have come from.

3.2 AAG Construction from Erroneous Raw Aspect Data

We are given an instance of noisy raw aspect data for each object in the model base. Our aim is to construct AAGs for each object, for robust class-recognition.

Let us define the terms 'smoothness' of model base data($\mathcal{S}(A)$), and the total model base error($\mathcal{E}(A)$) as follows:

$$\mathcal{S}(A) \stackrel{\triangle}{=} (1/M) \cdot \sum_{i=1}^{M} \sum_{j=1}^{G} d(c_{ij}, c_{ij+1})$$
$$\mathcal{E}(A) \stackrel{\triangle}{=} (1/M) \cdot \sum_{i=1}^{M} \sum_{j=1}^{G} d(c_{ij}, D_{ij})$$

where M is the number of objects in the model base and G is the number of tessellated viewpoints for the aspect data. Here, D_{ij} refers to the original raw aspect data at the jth site in model

number i. c_{ij} is the corresponding class-candidate label assigned to it by the AAG construction algorithm.

For the problem of AAG generation from noisy aspect data, one has to consider the following factors:

- the algorithm should be reasonably fast (i.e., have polynomial time complexity)
- the algorithm should give similar results for two 'similar' instances of raw aspect data
- the algorithm should give rise to prominent aspects, not too 'large' in number. The resulting AAG plot should be piecewise smooth
- the AAG should be close to the original raw aspect data

Many of these requirements are conflicting – optimality in terms of one compromises optimality in terms of the other.

We propose a new coefficient to evaluate the output of AAG construction algorithms. We define the 'Demerit Coefficient' for the AAG of model i in model base $\mathcal{M}$ as follows:

$$\eta(\mathcal{M}, i, \tau) \stackrel{\triangle}{=} \mu \sum_j (1 - \rho_{ij})\ d(c_{ij}, c_{ij+1}) + \nu \sum_j d(c_{ij}, D_{ij}) + \sigma \sum_j \rho_{ij}$$

where D_{ij} is the original raw aspect data item at site j of model i, c_{ij} is the class-candidate assigned to site j by the AAG construction algorithm, $d(\ ,\ )$ denotes the Euclidean distance operator for two normalized feature vectors, and μ, ν and σ are constants. (For our experimentation, we have chosen the constants μ, ν and σ such that all the three terms have the same order of magnitude.) ρ_{ij} is defined to be 1 if $d(c_{ij}, c_{ij+1}) > \text{threshold } \tau$, and 0 otherwise. The first term takes the piecewise smoothness criterion into account. The second term considers the fidelity between the original class-candidate at a site and the one assigned to the site by the algorithm. The last term considers the number of prominent discontinuities in the aspect data.

In case we have more than one instance of raw aspect data for a particular model, we consider the Demerit Coefficient for the AAG of the model as the average of the Demerit Coefficients computed for each given instance. We define the Demerit Coefficient for the set of AAGs for the

entire model base as the average of the Demerit Coefficients of the M individual models' AAGs, using the same constants and threshold for each model:

$$\eta_{model\ base}(\mathcal{M}, \tau) \stackrel{\triangle}{=} (1/M) \sum_{i=1}^{M} \eta(\mathcal{M}, i, \tau)$$

We evaluate an AAG construction algorithm by computing the values of $\eta_{model\ base}(\mathcal{M}, \tau)$ before (with $c_{ij} = D_{ij}$), and after the algorithm. In the former case, the second term is zero, while the first and third terms are large due to errors. The output of a good AAG construction algorithm is smooth, piecewise continuous, and is close to the original data. Hence, all three components of $\eta(\mathcal{M}, i, \tau)$ are expected to be low.

3.2.1 Algorithm for AAG Generation

This section proposes a low order polynomial time-complexity algorithm for building an AAG from noisy aspect data. We use clustering heuristics for modifying the raw aspect data in order to reduce its Demerit Coefficient. The algorithm maintains estimates of the probability with which one class-candidate is observed as another. The algorithm uses an $N_{\mathcal{C}} \times N_{\mathcal{C}}$ matrix, the ASSOC_TABLE to keep estimates of these association values.

Our algorithm is divided into three phases:

Algorithm Phase I

Phase I of our algorithm is primarily concerned with *identification of valid class-candidates*. The algorithm uses a 1-D version of Horn's sequential labeling algorithm ([22]) to cluster the raw aspect data into aspect-candidates This phase creates the aspect-candidate list for each object, and the class-candidate list for the model base. For an AAG which is not heavily corrupted with errors, one expects the class-candidates corresponding to prominent aspect-candidates to occupy more than N_{min} sites. Hence for such an AAG, this phase is expected to identify most of the valid class-candidates for the given model base. Phase I does not remove any errors from the raw aspect data. At the end of the entire algorithm, all valid class-candidates constitute the list of all classes for the given model base.

Algorithm Phase II

Phase II is primarily concerned with identification of prominent aspect-candidates after removing interspersed errors. This phase considers pairs of **proximal valid aspect-candidates**, with the same class-candidate, say c. We define a pair of valid aspect-candidates (a_i, a_j) as proximal valid aspect-candidates if

$$\theta_{a_k} \in (0, \Theta_{min}) \; \forall \; a_k \text{ lying in between } a_i \text{ and } a_j \text{ in the direction of traversal of the aspect-candidate list.}$$

For each pair of proximal valid aspect-candidates(a_i and a_j) with the same class-candidate(c, say) separated by a gap of width $\leq \Theta_{min}$, we integrate both the valid aspect-candidates and those in between them, into one. The correct class-candidate for the aspect-candidates in between the valid aspect-candidates a_i and a_j, is considered to be c. The algorithm updates the ASSOC_TABLE with this information.

Phase II of our algorithm removes Type II and Type III errors. (Whether the isolated error removed is a Type II or Type III error will be clear from the ASSOC_TABLE conditional probabilities). In Phase III, the algorithm uses the ASSOC_TABLE to account for association errors. If the value of $P(c \mid c')$ is above a particular threshold (c and c' are not necessarily different), the algorithm interprets an instance of c' to be c. Results of over 100 experiments with two model bases show the utility of having this association information calculated in Phase II (Section 5).

Algorithm Phase III

The third phase of our algorithm handles the rest of the raw aspect data. There are two passes through Phase III. The first is a logical pass, done in order to get further estimates for ASSOC_TABLE entries. The actual pass follows, using ASSOC_TABLE estimates made both during Phase II as well as the logical pass.

In Phase III again, we consider pairs of proximal valid aspect-candidates a_i and a_j. Depending on the gap between a_i and a_j in the direction of traversal, we consider two cases:

Case 1: gap(a_i,a_j) $< \Theta_{min}$

Here, we cannot have any valid aspect-candidate in between a_i and a_j. Hence, we obtain the minimum square-error decision boundary for the region of gap δ ($\delta < \Theta_{min}$). To get the minimum square-error decision boundary, we place the decision boundary at each position between the end point of a_i and the start point of a_j – an $O(\delta^2)$ operation. We consider the part of the gap till the decision boundary assigned to the class-candidate label $CLASS\_CAND(a_i)$ and the rest, $CLASS\_CAND(a_j)$. The error for a decision boundary is the sum of Euclidean distances between the original class-candidate label and the one just assigned for the entire region.

This is one point where we use the association of one class-candidate with another. We now consider the minimum-error decision boundary considering the association information collected thus far. For each class-candidate in the region of width δ, if its probability of being some other class-candidate is above a particular threshold, we replace it with this class-candidate for the purpose of getting an alternative decision boundary. The algorithm takes the one with the minimum error. We update the raw aspect data, the class-candidate list and the ASSOC_TABLE with this information. We can thus remove Type I errors.

Case 2: gap(a_i,a_j) $\geq \Theta_{min}$

We construct a normalized histogram for the class-candidates in the region between valid aspect-candidates a_i and a_j. We try the following three heuristic rules, in order of importance. If the maximum histogram value exceeds a threshold, and a_i and a_j have the same class-candidate, then we integrate all aspect-candidates from a_i to a_j into a_i, and return. Otherwise, for all histogram entries above a threshold, if a_i and a_j have the same class-candidate, we repeat the above procedure. If the above test fails, we try to grow aspect-candidates a_i whose histogram entry exceeds a threshold, if its distance to the next aspect-candidate a_j (of the same class-candidate) is less than the current extent of a_i. For regions still unaccounted for, we get a single minimum-error decision boundary.

Thus, our algorithm generates an AAG from noisy raw aspect data. In the first phase, it identifies prominent class-candidates. In the second, it gets information about the association of a class-candidate with another. This phase removes Type II and Type III errors. The third phase aims to remove other errors, with clustering heuristics. The entire algorithm runs in low-order polynomial time. The strategy presented here is an improvement upon our earlier work [23].

4 The Object Recognition Scheme

The previous section deals with the construction of AAGs from noisy aspect data. This section describes the use of AAGs thus constructed in the recognition of isolated 3-D objects, using the same noisy sensors. The section is structured as follows: we first give an overview of the entire recognition scheme. Then we describe our algorithm to recognize the class corresponding to a view of an object. The next section describes our algorithm which uses class and movement information to identify the given object.

4.1 An Overview of the Object Recognition Scheme

We are given an arbitrary view of an object in our model base (the AAGs of each object in the model base is constructed using the algorithm described in the previous section). The first step involves recognition of the class corresponding to the given view. This requires the use of different feature detectors to obtain feature-classes corresponding to each feature. (Section 2 describes the relation between objects and their aspects, classes, and feature-classes in our knowledge representation scheme.) We now probabilistically map the feature-classes onto the class corresponding to this view. However, this information may not be sufficient to identify the object uniquely – the given view could have come from more than one object in the model base. (Section 1 shows such an example: Figure 1). In other words, the class thus obtained could correspond to more than one aspect of different objects in the model base. Our probabilistic reasoning scheme uses the class information to generate hypotheses about the likely identity of the object. Based on the probabilities of the hypotheses generated, our planning algorithm plans the best move to obtain the next view, which would uniquely identify the object. The planning process is subject to memory and processing constraints, if any. The sensor is moved accordingly. If this does not resolve the ambiguity in the object's identity, the planning process is invoked again – the hypotheses are refined at each stage. The system repeats this process till the object is identified uniquely.

The following sections present the two important components of our system, namely

1. Class recognition from a given view of the object, and

2. Object recognition from the identified class

4.2 Class Recognition from a Given View of the Object

The input to this phase is an arbitrary view of any object in the model base. The recognition system uses the set of feature-classes from the view to generate hypotheses about the likely identity of the class. This step use our knowledge representation scheme (Section 2), and probabilistic reasoning. Figure 6 outlines the class recognition algorithm, which we describe below in detail.

ALGORITHM identify_class

```
1. compute_a_priori_class_probabilities();
          (* Eq. 1; Section 4.2  *)
2. fd := identify_feature_detector_to_use();
          (* Section 4.2 *)
3. fcl := get_feature_class(image,fd);
          (* Use fd on the image, identify feature class *)
4. compute_a_posteriori_class_probabilities(fcl);
          (* Eqs. 2,3; Section 4.2 Part 2 *)
5. IF the probability of some class is above a
   predetermined threshold THEN
          pass this class as evidence to the
          object recognition phase, EXIT
6. IF all feature detectors have been used
   AND the probability of no class is above
   the threshold THEN EXIT
7. GO TO Step 2
```

Figure 6: The Class Recognition Algorithm

We first calculate the *a priori* probability of each class. The probability of each class depends upon the probability of the aspects corresponding to it. Let aspects a_{pq} belong to class C_i. $P(a_{pq}|O_p)$ is $\theta_{pq}/360$. Initially, we assume each object to be equally likely to be present i.e., the *a priori* probability of object O_p is $1/n$, where N is the number of objects in the model base. We obtain the *a priori* probability of class C_i as:

$$P(C_i) = \sum_p [P(O_p) \cdot \sum_q P(a_{pq}|O_p)] \tag{1}$$

Let N_{F_j}, N_C and N_a denote the number of feature-classes associated with feature detector F_j, the number of classes, and the number of aspects, respectively. We can compute $P(C_i)$ from our knowledge representation scheme by considering each aspect node belonging to an object and testing if it has a link to node C_i – this takes $O(N_C + N_a)$ time.

The next step in the algorithm is to select a suitable feature detector. At any stage, we choose the hitherto unused feature detector for which the feature-class corresponding to the most probable class has the least number of outgoing arcs i.e., the least out-degree. This is done in order to obtain that feature-class which has the largest discriminatory power in terms of the number of classes it could correspond to.

Let the detector for feature F_j report the feature-class obtained to be f_{jk}. Given the *a priori* class probabilities computed in the first step and the feature class f_{jk}, the algorithm now computes *a posteriori* probabilities of all classes C_i.

$$P(C_i|f_{jk}) = \frac{P(C_i) \cdot P(f_{jk}|C_i)}{\sum_m [P(C_m) \cdot P(f_{jk}|C_m)]} \tag{2}$$

$P(f_{jk}|C_i)$ is 1 for those classes which have a link from feature-class f_{jk}. It is 0 for the rest. The computation of Equation 2 takes $O(N_C)$ time – this is done for each feature-class. Thus, it takes $O(N_{F_j} \cdot \ N_C)$ to compute $P(f_{jk}|C_i)$ value for all feature-classes f_{jk}.

To handle cases of feature detection errors, we compute the *a posteriori* probability of class C_i as follows:

$$P'(C_i) = \sum_l P(C_i|f_{jl}) \cdot p_{jlk} \tag{3}$$

where f_{jl}s are feature-classes associated with feature F_j. This summation reduces to one term, $P(C_i|f_{jr}) \cdot p_{jrk}$, since there is only one feature-class under feature F_j. The output of the AAG construction algorithm for each object in the model base gives us ASSOC_TABLE values(Section 3.2), from which we calculate p_{jlk} values(Section 2).

Thus, our algorithm can handle cases of feature detection errors, and recover from them. If the probability of any class is above a predetermined threshold, the system passes this information as evidence to the object recognition phase. Otherwise, it schedules the next feature detector as above, and repeats the process.

4.3 Object Identification Given the Identified Class

The input to the object identification phase is an arbitrary view of an object in the model base. The object identification phase uses the class recognition algorithm as a module. Figure 7 presents our overall object identification algorithm. In what follows, we describe the various steps of the algorithm in detail.

4.3.1 Object Probability Calculations in the First Phase

Before the system takes an image of the given view of the object, we calculate the *a priori* probabilities of each object in the model base.

$$P(a_{j_p k_p}) = P(O_{j_p}) \cdot P(a_{j_p k_p} | O_{j_p}) \tag{4}$$

The probability of each of the N objects is initialized to $1/N$ before the first observation. $P(a_{j_p k_p} | O j_p)$ is $\theta_{j_p k_p}/360$. *a priori* aspect probability calculations take $O(N_a)$ time.

The system now takes an image of the object. The class recognition algorithm identifies the class corresponding to a given view of the object. The class observed could have come from more than one aspect, each with its own range of positions within the aspect. The system now computes *a posteriori* probabilities as follows.

Let the class recognition phase report the observed class to be C_i. Let us assume that C_i could have come from aspects $a_{j_1 k_1}$, $a_{j_2 k_2}$, $\ldots a_{j_m k_m}$, where all j_1, j_2, $\ldots j_m$ are not necessarily different. We obtain the *a posteriori* probability of aspect $a_{j_l k_l}$ given this evidence using the Bayes rule:

$$P(a_{j_l k_l} | C_i) = \frac{P(a_{j_l k_l}) \cdot P(C_i | a_{j_l k_l})}{\sum_{p=1}^{m} [P(a_{j_p k_p}) \cdot P(C_i | a_{j_p k_p})]} \tag{5}$$

$P(C_i | a_{j_l k_l})$ is 1 for aspects with a link to class C_i, 0 otherwise. Finally, we obtain the *a posteriori* probability

$$P(O_{j_p}) = \sum_{l} P(a_{j_p k_l} | C_i) \tag{6}$$

where aspects $a_{j_p k_l}$ belong to class C_i. If the probability of some object is above a predetermined threshold(experimentally determined e.g., 0.87 for the model base consisting of polyhedral objects), the algorithm reports a success, and stops. *However, if the probability of no object is*

ALGORITHM identify_object

```
            (* ------ FIRST PHASE ------ *)
1.  initialize_object_probabilities();
            (* Initialize to 1/N *)
2.  image:=get_image_of_object();
3.  class:=identify_class(image);
            (* Section 4.2 *)
    IF class=UNKNOWN THEN exit;
4.  search_tree_root:=construct_search_tree_node(class,0);
5.  compute_object_probabilities(search_tree_root);
            (* Eqs. 5,6 *)
6.  IF the probability of some object is above a
    predetermined thresh.  THEN exit & declare success;
7.  expand_search_tree_node(search_tree_root,0,class);
            (* Section 4.3.2 *)
    best_leaf:=get_best_leaf_node(search_tree_root);
            (* ------ SECOND PHASE ------ *)
        previous:=search_tree_root;
        expected:=best_leaf;
8.  angle:=compute_angle_to_move_by(expected,previous);
    make_movement(angle);
    image:=get_image_of_object();
9.  class:=identify_class(image);
    IF class=UNKNOWN THEN exit;
10. new_node:=construct_search_tree_node(class,angle);
11. compute_object_probabilities(new_node);
12. IF the probability of some object is above a
    predetermined thresh.  THEN exit & declare success;
13. expand_search_tree_node(new_node);
    best_leaf:=get_best_leaf_node(new_node);
        previous:=new_node;
        expected:=best_leaf;
14. GO TO step 8
```

Figure 7: The Object Recognition Algorithm

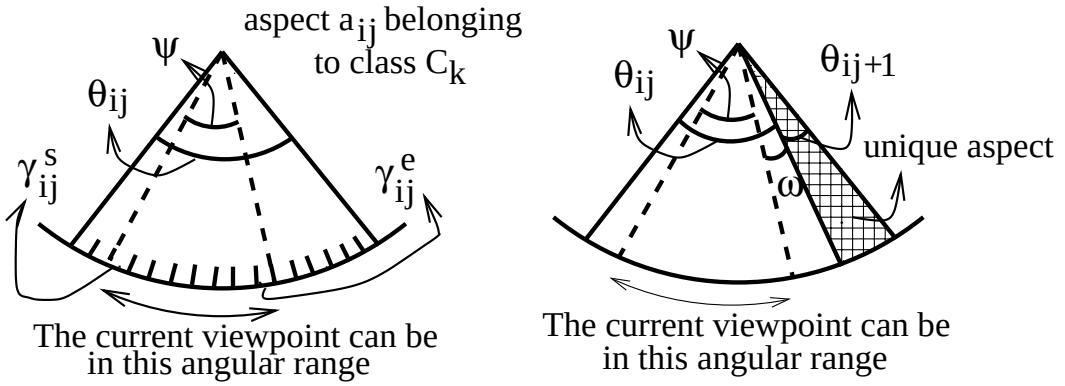


Figure 8: (a) The notation used (Section 4.3.2) (b) A case when our algorithm is not guaranteed to succeed (Section 4.3.4)

above the threshold, this implies that the information from the given view of the object is not sufficient to recognize it unambiguously. We have to take another view of the object, which will best disambiguate between the competing objects. The next section describes our next view planning scheme in detail.

4.3.2 Next View Planning

Next View Planning is required when the probability of no object is above a particular threshold. This means that the class observed could have come from more than one aspect, each with its own range of positions within an aspect. We use a search tree to find a state in the search space corresponding to a unique aspect. The parameters describing the state (a search tree node) are as follows: the unique class observed for the angular movement made so far, the aspects possible for this angle-class pair, and for each aspect, the range of positions possible within it ($\gamma_{ij}^s - \gamma_{ij}^e$). γ_{ij}^s and γ_{ij}^e denote the two positions within aspect a_{ij} where the current viewpoint can be, as a result of the movement made thus far. Here, $\gamma_{ij}^s \leq \gamma_{ij}^e$; and γ_{ij}^s, $\gamma_{ij}^e \in [0, \theta_{ij}]$, where θ_{ij} is the angular extent of aspect a_{ij}(Figure 8(a)). A leaf node is one which has a unique aspect associated with it, or corresponds to a total movement of 360 degrees from the root node, or more. The aim is to find the move that will best disambiguate between the competing objects, subject to memory and processing limitations, if any. An important parameter is the step size of movement. If it

is too small, then we may remain in the same aspect - incurring image processing cost. A large step size, on the other hand may cause us to miss a unique aspect corresponding to an object.

In view of these facts, we categorize moves from a particular viewpoint, as follows:

Primary Move A primary move represents a move from an aspect by α, the minimum angle needed to move out of it.

Auxiliary Move An auxiliary move represents a move from an aspect by an angle corresponding to the primary move of another competing aspect.

Let α_{ij}^c and α_{ij}^a represent the minimum angles necessary to move out of the current assumed aspect in the clockwise and anti-clockwise directions, respectively. Three cases are possible:

1. **Type I move**: α_{ij}^c and α_{ij}^a both take us out of the current aspect to a single aspect in each of the two directions – a_{ip} and a_{iq}, respectively. We construct search tree nodes corresponding to both moves.

2. **Type II move**: Exactly one out of α_{ij}^c and α_{ij}^a takes us to a single aspect a_{ip}. For the other direction, the aspect we would reach depends upon the initial position ($\in [\gamma_{ij}^s, \gamma_{ij}^e]$) in the current aspect. We construct a search tree node corresponding to the former move.

3. **Type III move**: Whether we move in the clockwise or the anti-clockwise direction, the aspect reached depends on the initial position in the current aspect. We choose the move which leads us to the side with the largest angular range possible in any reachable aspect.

We expand a non-leaf node by generating child nodes corresponding to primary moves for all competing aspects in its aspect list. We can also generate additional child nodes by considering auxiliary moves. We assign a code 0 to Type I and II primary moves and 1 to Type II auxiliary moves. Type III primary moves moves get a code of 2, and Type III auxiliary moves, 3. The weight associated with a node is $4^i \cdot Code$, where i is the depth of the node in the search tree. We determine the best leaf node using three levels of filtering: We use three levels of filtering to determine the best leaf node. First, we consider those on a path from the most probable aspect(s) corresponding to the previously observed node. Among these, we consider those having paths of

least weight. From these, we finally select one with the minimum total movement. Our earlier work [24] gives further details of the planning process.

4.3.3 The Second Phase of the Object Recognition Algorithm

The search process finds the best move to resolve the ambiguity associated with this view. The system takes the move corresponding to the best leaf node in the search tree, and the class identification module is called again. We construct a new search tree node corresponding to this move.

We again need to calculate the *a priori* probability of each class C_i. This process is similar to Equation 1 of Section 4.2, except for the following two differences. The *a posteriori* object probabilities $P(O_{j_p}$ of Equation 6 of Section 4.3.1 serve as the *a priori* probabilities for the class identification part here. Further, we compute the value of $P(a_{j_pk_p}|O_{j_p})$ as follows:

$$P(a_{j_pk_p}|O_{j_p}) = \phi_{j_pk_p}/360 \tag{7}$$

where $\phi_{j_pk_p}$ ($\phi_{j_pk_p} \in [0, \theta_{j_pk_p}]$) represents the angular range possible within aspect $a_{j_pk_p}$ for the move(s) taken to reach this position. Thus, unlike Hutchinson and Kak's system [19], the link conditional probabilities themselves enforce consistency checks at each level of evidence.

Depending on the observed class C_i, we calculate the *a posteriori* probability of aspect $a_{j_lk_l}$, and finally the *a posteriori* probability of each object O_{j_p} using Equation 5 and Equation 6 of Section 4.3.1. If the probability of some object is above the threshold, then we are done. Otherwise, the search process is repeated. This illustrates the reactive nature of our strategy. We discuss this further, along with issues relating to the finiteness of the search procedure and scalability, in the next section.

4.3.4 The Object Identification Algorithm: A Discussion

Given the combinatorial nature of the problem, taking primary and auxiliary moves helps us to prune the search space by selecting an appropriate step size of movement. This eliminate redundant moves(which would have incurred large image processing cost). Further, our planning strategy is scalable – one can plan with primary moves alone(greater pruning of the search space),

or with a combination of primary and auxiliary moves(which have greater discriminatory power), depending on memory and processing limitations. Our robust class recognition algorithm can recover from many feature detection errors at the class recognition phase itself(Section 4.2). If the view indeed corresponds to the most probable aspect at a particular stage, then our search process is guaranteed to perform aspect resolution and uniquely identify the object in the following step, assuming no feature detection errors. Even if the view does not correspond to the most probable aspect, the list of possible aspects a given view could correspond to, is refined at each observation stage(avoiding the exponential time complexity of having to expand out the entire search tree at one go). The reactive nature of our strategy incorporates all previous movements and observations both in the probability calculations(Sections 4.2, 4.3.1, and 4.3.3), as well as in the planning process.

Search tree node expansion is always finite due to the following reasons: The number of aspects is finite, and no aspect is repeated along a search tree path. Further, even if competing objects have the same aspects, search tree expansion stops when the total movement along a path is 360^o.

Assuming no feature detection errors, our algorithm is guaranteed to succeed except in three cases. The first is for objects with the same aspect structure(i.e. the layout of classes in the aspect graph) but different aspect angles. Further, our strategy does not handle the case when the aspect angles are greater than or equal to 180 degrees. Figure 8(b) shows an example of the third case. Let us suppose that we have to move anti-clockwise. Let ψ denote the angular extent of the smallest aspect observed so far. The current viewpoint lies in this angular range. Let a_{ij+1} be a unique aspect for the assumed object. The anti-clockwise movement will be by an angle $\psi + \omega$. If $\psi + \omega > \theta_{ij+1}$, we may miss this unique aspect altogether.

5 Experimental Results and Discussion

Our experimental setup has a camera connected to a MATROX Image Processing Card and a stepper motor-controlled turntable. The turntable moves by 200 steps to complete a 360 degree movement. We have experimented extensively with two object sets as model bases. Some details

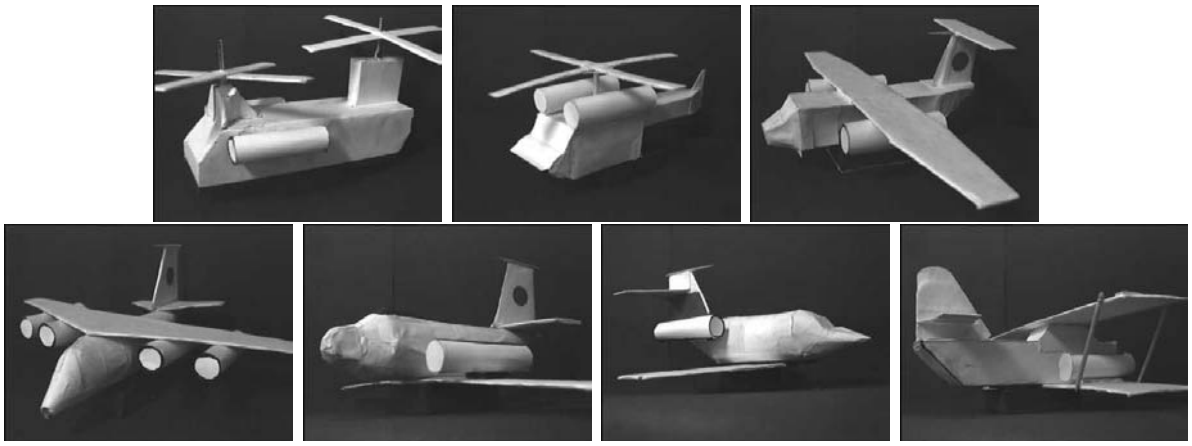


Figure 9: Model Base I: The objects (in row major order) are heli_1, heli_2, plane_1, plane_2, plane_3, plane_4, and biplane.

of the object sets are as follows:

1. **Model Base I: 7 Aircraft Models**

   We use as features, the number of horizontal lines ($\langle h \rangle$), the number of vertical lines ($\langle v \rangle$), and the number of circles ($\langle c \rangle$). We represent a class-candidate as $\langle hvc \rangle$. We have chosen this relatively feature-rich model base for two reasons. First, we wish to demonstrate the effectiveness of our AAG construction algorithm on raw aspect data with very low smoothness. The second reason is to show the effectiveness of using simple and robust features with multiple views for recognizing complex 3-D objects. Figure 9 shows the objects in this model base.

2. **Model Base II: 8 Polyhedral Objects**

   We use as features, the number of horizontal lines ($\langle h \rangle$), the number of vertical lines ($\langle v \rangle$), and the number of non-background segmented regions in an image($\langle r \rangle$). We represent a class-candidate as $\langle hvr \rangle$. The raw aspect data for this model base has higher smoothness compared to the aircraft models. The list of possible aspects associated with one initial view is quite large here(18, compared with 10 for the first set). Figure 10 shows the objects in this model base.

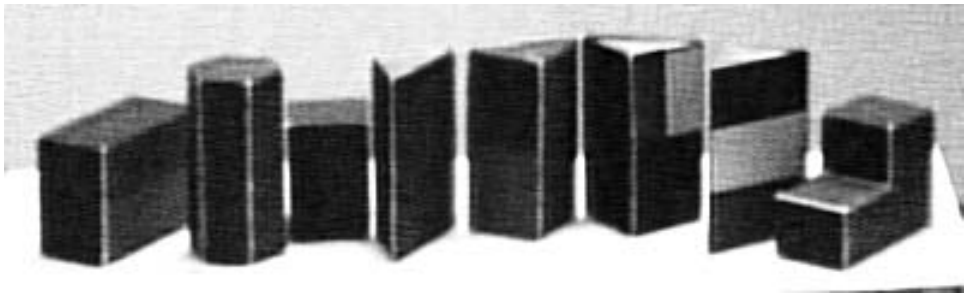


Figure 10: Model Base II: The objects (from left) are O_1, O_2, O_3, O_4, O_5, O_6, O_7 and O_8, respectively.

We use hough transform-based line and circle detectors. For getting the number of regions in the object, we use sequential labeling on a thresholded gradient image.

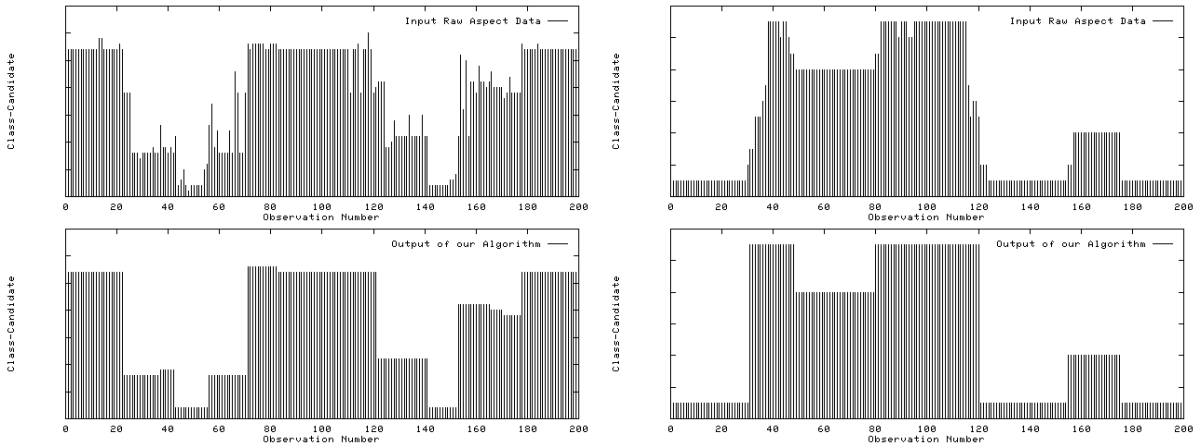
AAG construction

Let the term 'Input Smoothness' ($\mathcal{S}(I)$) refer to the smoothness expression for the raw aspect data. Thus, c_{ij} is D_{ij} here i.e., the raw aspect data item for the ith model at site number j. Similarly, we use the term 'Output Smoothness' ($\mathcal{S}(O)$) to refer to the smoothness expression for the output of the aspect graph construction algorithm.

The aspect data for Model Base I (aircraft models) has very high values of the Demerit Coefficient $\eta_{model\ base}$ and $\mathcal{S}(I)$ as compared to the aspect data for the other. Hence, we first present results of 100 experiments with the first model base. Then, we compare some figures with those of Model Base II (polyhedral objects).

Output of the AAG Construction Algorithm: Figure 11 shows a comparison of the raw aspect data and the output of our algorithm, for one instance of the aspect data for object plane_2 in Model Base I; and O_6 in Model Base I. A visual inspection of the lower graph shows that the aspects produced are prominent and not too large in number, the graph is piecewise smooth and at the same time, fidelity to the original data is high.

Input and Output Smoothness: Figure 12(a) shows a comparison of the input and output smoothness for 100 sets of aspect data for the aircraft model base. Even though the raw aspect data has a large variation in $\mathcal{S}$ values, the variation in $\mathcal{S}$ values for the output data is very small.

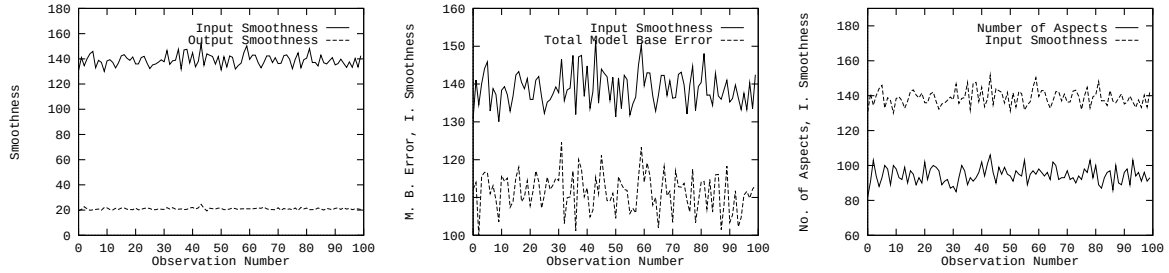


(a) (b)

Figure 11: Raw aspect data and the output of our algorithm:(a)plane_2, Model Base I; and (b) O_6, Model Base II. On the y-axis, each class-candidate is represented by an index. Different heights represent different class-candidates.

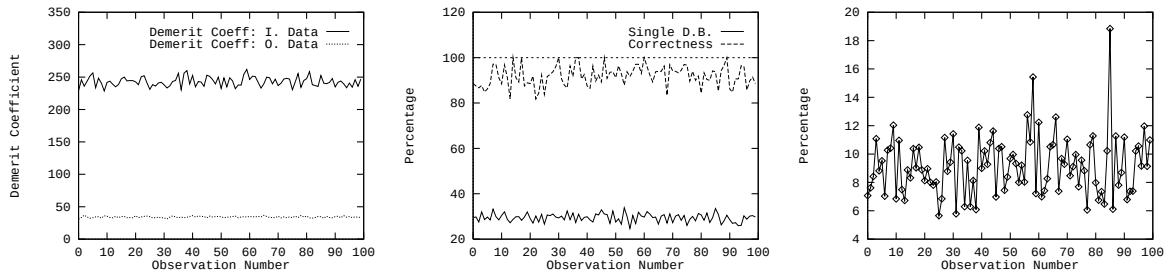
Total Model Base Error and Number of Aspects: $\mathcal{E}$ is a measure of fidelity of the output data to the input raw aspect data. Figures 12(b) and (c) show the variation in $\mathcal{E}$ and the number of aspects, respectively with the input smoothness for the 100 data sets.

Demerit Coefficients: Figure 13(a) shows the variation of the Demerit Coefficient for the input aspect data, with the Demerit Coefficient for the output of the AAG construction algorithm for the 100 data sets. Our AAG construction algorithm greatly reduces the Demerit Coefficient.



(a) (b) (c)

Figure 12: Variation in (a) input 'smoothness' with the output 'smoothness', (b) total model base error with input 'smoothness', and (c) the number of aspects with input 'smoothness',



(a) (b) (c)

Figure 13: (a) Variation in the Demerit Coefficient for the input raw aspect data with the Demerit Coefficient for the output of the AAG construction algorithm, (b) Percentage of sites where a single decision boundary had to be taken, and correctness in Phase II estimates in the ASSOC_TABLE, and (c) Percentage reduction in total model base error with ASSOC_TABLE data

Further, the variation in the Demerit Coefficient for the output data is quite less compared that for the input raw aspect data.

Percentage of sites where a single decision boundary had to be taken: The only part of our algorithm which has quadratic time complexity is where a single decision boundary has to be taken over a set of adjacent sites. The rest of it runs in linear time. Figure 13(b) shows the percentage of sites where a single decision boundary had to be taken, for 100 sets of aspect data.

Correctness of Phase II ASSOC_TABLE estimates: We compare the probability values in the ASSOC_TABLE at the end of Phase III, to what they were at the end of Phase II. We refer to this as 'correctness'. Figure 13(b) shows the variation in percentage correctness of Phase II estimates, with the input 'smoothness' for 100 data sets.

Percentage reduction in model base error with ASSOC_TABLE estimates:
The model base error $\mathcal{E}(A)$ is reduced if one uses the association data from the ASSOC_TABLE. Figure 13(c) shows the percentage reduction in error for 100 instances of model base data.

Comparison of performance factors of our AAG construction algorithm on the two model bases: Table 1 shows the comparison between the two model bases. The figures for Model Base I are for 100 experiments, whereas those for Model Base II are for 4. Though the feature detectors used for the two model bases are different, the range of values taken by the feature-classes for

	Model Base I		Model Base II	
PARAMETERS	*Mean*	*Variance*	*Mean*	*Variance*
Input Smoothness	138.85	4.51	43.99	0.47
Output Smoothness	20.99	0.74	18.93	0.34
Total Model Base Error	111.49	5.01	27.97	1.35
No. of Aspects	94.51	4.55	80.5	1.12
Demerit Coefficient (input data)	242.84	7.32	67.69	2.59
Demerit Coefficient (output data)	34.13	1.17	28.99	0.48
Quadratic Complexity Regions	29.47%	1.85	7.47%	0.41
Correctness of Phase II Estimates	91.91%	4.46	94.74%	3.72
Error reduction with ASSOC_TABLE	9.18%	2.08	4.85%	0.37

Table 1: Summary of AAG construction algorithm performance parameters for the two model bases

the two model bases are comparable.

3-D Object Recognition

We have experimented with both strategies - planning with primary moves alone, and both primary and auxiliary moves. Figure 14 and 15 show some experiments with objects in the two model bases.

We make the following observations:

Primary and Auxiliary Moves: In most cases, the number of image processing steps required is less in the latter case compared to the former. The exceptions can be accounted for by the ability of the system to handle cases opportunistically – when the sequence of moves turns out to be unique for a particular object. When memory and search time are limited, the planning process may use primary moves alone.

Ordering of feature detectors: The third image in Figure 15(a) shows an advantage of our scheduling of feature detectors. The line detector reports the feature-class present to be $\langle 23 \rangle$. For the objects in our model base, this could correspond to classes $\langle 232 \rangle$ and $\langle 233 \rangle$. Our probability calculations account for the movement taken around the object. The probability of class $\langle 232 \rangle$ for the movement made so far exceeds the class probability threshold(0.87). Hence, the system

does not need to use the other feature detector.

Recovering from feature detection errors: In the first image in Figure 14(b), due to the shadow of the wing on the fuselage of the aircraft, the feature detector detects 4 vertical lines instead of 3, the correct number. Our recovery mechanism (Section 4.2) corrects this error. The second image in Figure 15(a) shows another such situation. Due to the thresholds we use, the correct class is $\langle 221 \rangle$. The line detector, however reports the probabilities of classes $\langle 221 \rangle$ and $\langle 231 \rangle$ as 0.004 and 0.856, respectively. The probability of no class is above the threshold. The other feature detector is now scheduled, which reports the number of regions to be 1. Probability calculations of Equation 3 result in the probabilities of the two as 0.997 and 0.002, respectively.

Some sample search tree details: For each row in Figure 14, the initial view could have come from 10 aspects belonging to objects in the first model base. We consider the case of Figure 15 in detail, since the initial view in each row could have come from 18 aspects. For the strategy involving primary moves alone, the total number of search tree nodes generated for Figures 15(a) and 15(b) are 53 and 48, respectively. Using both primary and auxiliary moves(Figures 15(c) and 15(d)), the corresponding numbers are 324 and 279, respectively.

Average number of observations for a given number of competing aspects: Table 2 gives an idea of the average number of observations for a given number of competing aspects in our experiments. The average for the first model base is computed over 46 experiments, while the corresponding number for the second model base is 58, respectively.

6 Conclusion and Scope for Future Work

This paper presents an integrated approach for handling feature detection errors for use in robust active 3-D object recognition. First, we present an algorithm for AAG construction with noisy feature detectors – No other aspect graph construction algorithm is tolerant to the effect of noise for determination of aspects. We propose an evaluation function for comparing the output of different AAG construction algorithms. Our hierarchical knowledge representation scheme facilitates planning by exploiting relationships between features, aspects and object models. We demonstrate that our recognition strategy works even under processing and memory constraints

Table 2: The average number of moves for a given number of competing aspects

Model Base I: Polyhedral Objects		
Number of	*Average number of observations*	
Competing Aspects	*Primary Moves*	*Pri. & Aux. Moves*
5	2.00	2.50
17	3.09	3.07
18	4.00	3.38
Model Base II: Aircraft Models		
Number of	*Average number of observations*	
Competing Aspects	*Primary Moves*	*Pri. & Aux. Moves*
4	2.00	2.00
5	2.00	2.09
7	2.00	2.00
9	2.00	2.00
10	2.67	2.67

due to the incremental reactive planning method. No related work has addressed these issues. We present the results of extensive experimentation on a reasonably complex experimental set, in support of our system.

Major areas for further work include an extension to the 3-DOF case, handling cases of multiple objects, and searching for an object in a cluttered environment. An extension of this work would take movement errors into account.

Acknowledgements

We are thankful to the anonymous reviewers for helping us in improving the clarity of the paper.

References

[1] P. J. Besl and R. C. Jain *ACM Computing Surveys* **17** (1985) 76–145

[2] R. T. Chin and C. R. Dyer *ACM Computing Surveys* **18** (1986) 67–108

[3] A. Zisserman, D. Forsyth, J. Mundy, C. Rothwell, J. Liu, and N. Pillow *Artificial Intelligence* **78** (1995) 239–288

[4] B. V. Funt and G. D. Finlayson *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17** (1995) 522–529

[5] S. K. Nayar and R. M. Bolle *International Journal of Computer Vision* **17** (1996) 219–240

[6] J. B. Burns, R. S. Weiss, and E. M. Riseman in *Geometric Invariance in Computer Vision* (Eds.) A. Zisserman and J. Mundy, MIT Press (1992)

[7] K. D. Gremban and K. Ikeuchi *International Journal of Computer Vision* **12** (1994) 137–172

[8] J. J. Koenderink and A. J. van Doorn *Biological Cybernetics* **32** (1979) 211–216

[9] K. D. Gremban and K. Ikeuchi *Three-Dimensional Object Recognition Sytems*, (Eds.) A. K. Jain and P. J. Flynn, Elsevier-Science Publishers (1993) 229–258

[10] J. J. Koenderink and A. J. van Doorn *Biological Cybernetics* **24** (1976) 51–59

[11] J. H. Rieger *Phil. Trans. R. Soc. London. A* **354** (1996) 899–1940

[12] D. J. Kriegman and J. Ponce *International Journal of Computer Vision*, **5** (1990) 119–135

[13] Z. Gigus and J. Malik *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12** (1990) 113–122

[14] J. Stewman and K. Bowyer in *Proc. Int. Conf. Computer Vision* (1988) 494–500

[15] M. R. Korn and C. R. Dyer *Pattern Recognition* **20** (1987) 91–103

[16] K. Ikeuchi and T. Kanade *Proceedings of the IEEE* **76** (1988) 1016–1035

[17] C. H. Chen and A. C. Kak *IEEE Transactions on Systems, Man and Cybernetics* **19** (1989) 1135–1563

[18] I. Chakravarty and H. Freeman *Proc. SPIE Conference on Robot Vision* **336** (1982) 37–45

[19] S. A. Hutchinson and A. C. Kak *IEEE Transactions on Robotics and Automation* **5** (1989) 765–783

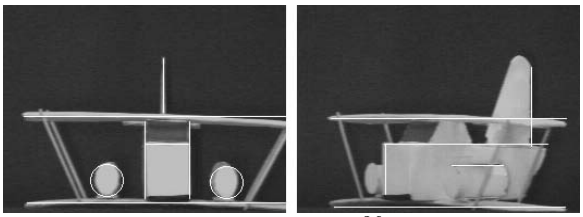
[20] J. Maver and R. Bajcsy *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15** (1993) 76–145

[21] S. J. Dickinson, H. I. Christensen, J. Tsotsos, and G. Olofsson *Computer Vision and Image Understanding* **67** (1997) 239–260

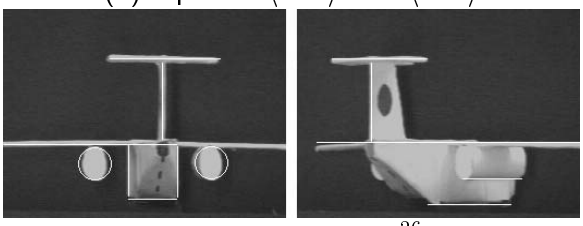
[22] B. K. P. Horn *Robot Vision* The MIT Press and McGraw-Hill Book Company (1986)

[23] S. Dutta Roy, S. Chaudhury, and S. Banerjee in *Computer Vision, Graphics and Image Processing: Recent Advances*, (Eds.) S. Chaudhury and S. K. Nayar, Viva Books Private Limited, New Delhi (1999) 166 – 172

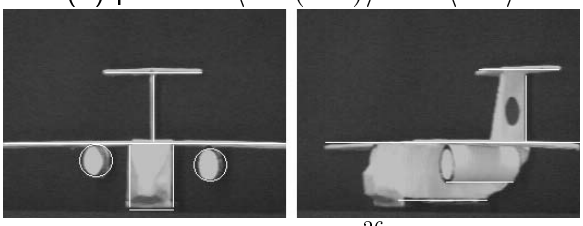
[24] S. Dutta Roy, S. Chaudhury, and S. Banerjee *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans* **30** (2000) 67–76



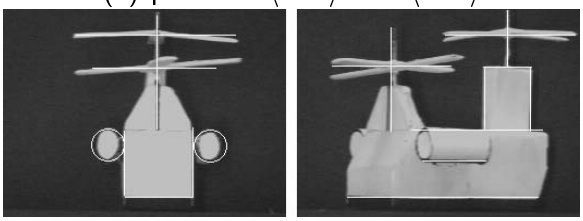
(a) biplane: $\langle 332 \rangle \xrightarrow{26} \langle 420 \rangle$



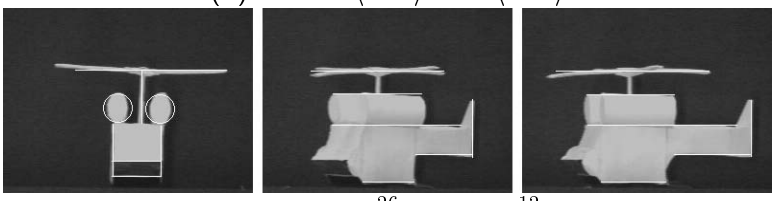
(b) plane_1: $\langle 342(332) \rangle \xrightarrow{26} \langle 410 \rangle$



(c) plane_1: $\langle 332 \rangle \xrightarrow{26} \langle 410 \rangle$



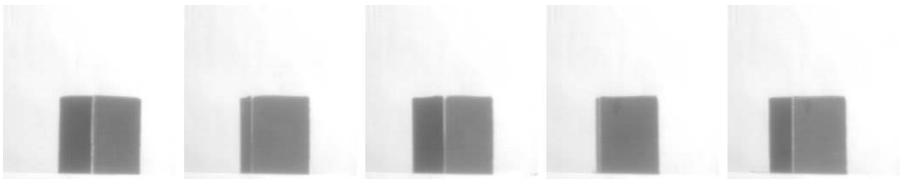
(d) heli_1: $\langle 332 \rangle \xrightarrow{26} \langle 540 \rangle$



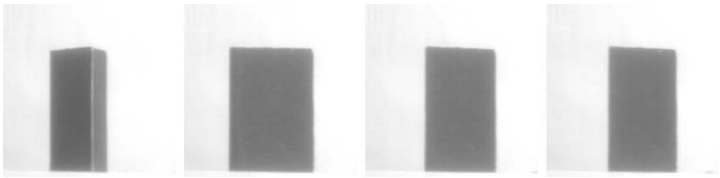
(e) heli_2: $\langle 332 \rangle \xrightarrow{26} \langle 510 \rangle \xrightarrow{12} \langle 510 \rangle$

Figure 14: Some experiments with the Model Base I: initial class $\langle 332 \rangle$. (The figure in parentheses shows an example of recovery from feature detection errors). In each of these cases, the results for planning with primary moves alone, and those for both primary and auxiliary moves are identical

PRIMARY MOVES ALONE

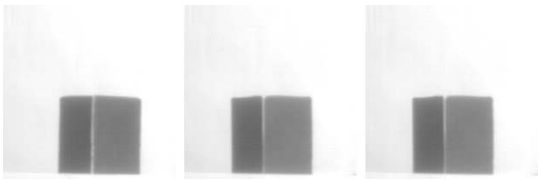


(a) $\langle 232\rangle \xrightarrow{-40} \langle 231(221)\rangle \xrightarrow{-12} \langle 232\rangle \xrightarrow{-34} \langle 221\rangle \xrightarrow{-11} \langle 232\rangle$

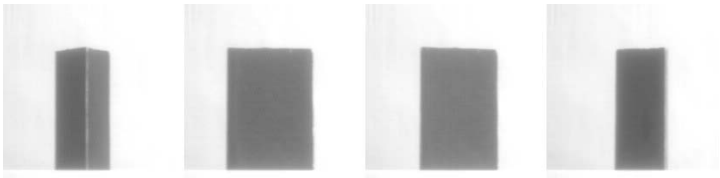


(b) $\langle 232\rangle \xrightarrow{-40} \langle 221\rangle \xrightarrow{-12} \langle 221\rangle \xrightarrow{-2} \langle 221\rangle$

PRIMARY AND AUXILIARY MOVES



(c) $\langle 232\rangle \xrightarrow{-40} \langle 232\rangle \xrightarrow{-77} \langle 221\rangle$



(d) $\langle 232\rangle \xrightarrow{-40} \langle 221\rangle \xrightarrow{-12} \langle 221\rangle \xrightarrow{-33} \langle 221\rangle$

Figure 15: Some experiments with Model Base II: initial class $\langle 232\rangle$. The objects are O_3((a), (c)) and O_4((b), (d)), respectively. The numbers above the arrows denote the number of turntable steps. A negative sign indicates a clockwise movement. (The figure in parenthesis shows an example of recovery from feature detection errors)