# Learning to Recognise 3D Objects from 2D Intensity Images

Thesis submitted by

Brendan James M$^c$CANE BSc(Hons)

in March 1996

for the degree of Doctor of Philosophy

in the Department of Computer Science at

James Cook University of North Queensland

I, the undersigned, the author of this thesis, understand that James Cook University of North Queensland will make it available for use within the University Library and, by microfilm or other photographic means, allow access to users in other approved libraries. All users consulting this thesis will have to sign the following statement:

> In consulting this thesis I agree not to copy or closely paraphrase it in whole or in part without the written consent of the author; and to make proper written acknowledgement for any assistance which I have obtained from it.

Beyond this, I do not wish to place any restriction on access to this thesis.

................ ............

(signature) (date)

This thesis is dedicated to Mum and Dad

**Shirley Patricia M^c^Cane**

and

**James Owen Crofton M^c^Cane**

*Of all the things I have learnt, what you have taught me is still the most important.*

# Acknowledgements

Firstly, I would like to thank my supervisors, Olivier de Vel and Terry Caelli, for providing support, ideas and inspiration. I would also like to thank Walter Bischof whose implementation of the CRG classifier formed the heart of FCRG. The developers of the IPRS image processing and pattern recognition system also deserve thanks, they are: Erhardt Barth, James Cooper, Craig Dillon, Ashley Dreier, Jonathan Gifford, Mark Ollila, David Squire, Tom Wild, and John Krivitsky. Without IPRS, my thesis would probably have taken twice as long to complete.

Finally, I would like to make a special thanks to Simone, without whom I may not have started a PhD, and almost certainly wouldn't have completed it.

# Abstract

The problem of three dimensional object recognition is one which has been studied extensively by computer vision researchers over the previous 30 years. Recently, significant research effort has concentrated on solving the 3D object recognition problem using dense range data. However, range data is of limited usefulness over large distances and in outdoor settings, and the human visual system does not make use of this type of data. Alternatively, some work has been done on recognising 3D objects from 2D intensity images, but of the reported systems, many use only a limited database of one or two objects, others utilise "perfect" or synthetic input data (eg line drawings obtained from CAD packages), few systems attempt to learn object models, and the problem of recognising objects in complex scenes is often neglected. Also, considerable research has been reported on techniques which extract sparse 3D depth data from 2D images (depth-from-X), but no system incorporates this data into a working object recognition system.

This dissertation describes a 3D object recognition system that learns to recognise 3D objects from 2D intensity images. Input images are segmented using a novel segmentation algorithm, and depth data is extracted using a novel stereo algorithm. Attributes (including depth attributes) are extracted from each of the parts produced by the segmentation algorithm, and these are used by the Fuzzy Conditional Rule Generation (FCRG) machine learning classifier to either learn object models in the training samples, or to classify object models in test scenes containing one or more objects. A final hypothesis verification routine is used to verify the classifications produced by FCRG.

The system has been demonstrated using a database of 18 non-trivial objects, for which a classification rate of up to 78% was achieved. This dissertation demonstrates that learning to recognise 3D objects from 2D intensity images is a viable alternative to recognising objects from dense range data, that sparse range data obtained from depth-from-X techniques (in this case, depth-from-stereo) can be used to aid the recognition process, and that it is possible to recognise objects in complex scenes without prior object partitioning.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Three dimensional (3D) object recognition is a problem which continues to challenge the computer vision research community. The problem is an extremely difficult one, yet it is obviously possible since there is an existence proof for such a working system (the human visual system). However, to date, no system approaches the performance of the human visual system - even in simplified domains.

The object recognition problem can be defined as recognising an object in isolation, or embedded in a more complex scene, from a pre-compiled database of objects. In the case of 3D object recognition, these objects are three-dimensional, and the scenes are either two-dimensional (eg intensity images), or three-dimensional (eg dense range data). The predominant approach in the literature is to develop object recognition systems (ORS) which utilise 3D scenes as the form of input (eg [111], [48], [31], [66], [68], [45], etc), as it is generally believed that "recognising objects by their geometric shape should be much easier using range images instead of intensity images" (Besl and Jain [11]). Even so, such systems do not approach the performance of the human visual system which uses predominantly 2D intensity images. More importantly, range finders typically work over only a limited range, and have difficulty (adequately) representing the external "real" world (eg foliage) as opposed to highly constrained environments (eg industrial inspection environments). This is not to say that range data is not useful for many applications, yet it does have limitations when dealing with more complex environments. Having said that, this dissertation does not attempt to address many of the problems inherent in dealing with the real world, rather it focuses on furthering the state-of-the-art in performing 3D object recognition from 2D intensity images, with the belief that such an approach has more promise with reference to complex environments than systems relying on dense range data obtained from active range finders.

Regardless of whether range or intensity data is used as input to the system, the object recognition problem can be subdivided into a number of sub-parts:

- Image acquisition.
- Image segmentation.
- Attribute extraction.
- Recognition.

Since the object recognition problem is such a difficult one, and since it is generally believed it can be conveniently divided into the above sub-problems, the majority of researchers have concentrated only on one aspect of the problem, assuming that other aspects have been adequately solved, or will be adequately solved in the future. Significant research has been performed in the area of 2D intensity image segmentation, typically in the form of edge extraction or region extraction algorithms. Yet very little of this work has been successfully applied in the recognition phase of the problem, making it difficult to empirically evaluate the goodness of these algorithms (what is an edge in the real world anyway?). Similarly in the attribute extraction phase, significant research efforts have centred on extracting depth data from 2D intensity images - the so called shape-from-X problem (eg shape-from-stereo, shape-from-focus, shape-from-shading etc). Yet, here again, little effort has been made to utilise these techniques in a complete recognition system. Partly, this is because the suggested solutions to these problems are not perfect nor applicable over a wide range of problem domains and certainly there is still room for improvement for many of the solutions. But it is becoming increasingly apparent that solutions to such problems are never likely to be perfect or widely applicable, but should still be extremely useful if some effort were made to incorporate the techniques into working object recognition systems (ORS). This deficiency has also been noted by other authors ([42], [26]). The work described here is a step towards producing systems which integrate all aspects of the vision problem into a single system, and report on the usefulness or otherwise of these sub-systems to the overall object recognition goal.

Many current recognition systems do not attempt to solve the problem of model acquisition - it is assumed that the models are fully determined, typically in the form of CAD models ([74], [6], [9], [48], [68], etc). Such an approach takes significant human resources to build the object models. A more general and extendible approach is to present an ORS with examples of the objects to be recognised and let the ORS learn the representations. This is the *modus operandi* for the human visual system and would seem necessary for the development of complex applications such as autonomous mobile robots. Such an approach is especially important where the input data to the system is complex and imperfect as no translation between scene representation and object model representation is required. This dissertation presents a system which *learns* to recognise 3D objects from 2D intensity images - a subject which has been addressed only sporadically in the literature (eg [92]).

## 1.1 The FCRG Object Recognition System

In this dissertation, a 3D object recognition system (FCRG - fuzzy conditional rule generation) is presented which begins to address many of the problems which have been mostly ignored in the literature. This involved the development of a complete object recognition system which can learn and recognise 3D objects from 2D intensity images. Although, due to the complexity of the task and the time required for implementation, one of the modules is not well developed (the hypothesis verification), and one issue was largely not addressed (figure-ground separation - the objects are placed on a black background for the experiments). Nevertheless, aspects of most other issues are addressed here.

Figure 1.1 displays an outline of the system which has been implemented. Stereo pairs of intensity images are input into the system, some stereo processing and segmentation is performed, attributes are extracted from the resulting segments (including some depth attributes), the results are passed to the FCRG system for learning or recognition, and, finally, hypothesis verification is performed resulting in class labels for all image regions.

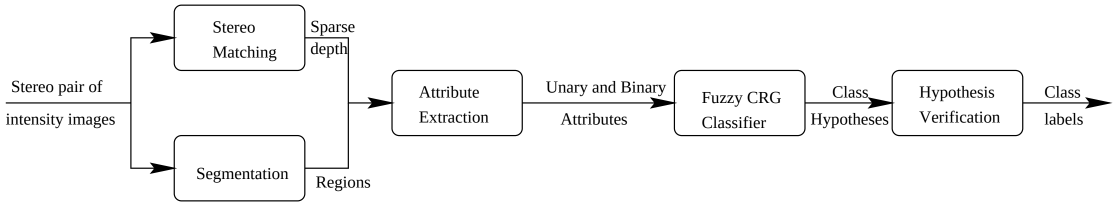



Figure 1.1: Outline of the FCRG object recognition system.

The initial processing of the input is split into two parts - the stereo algorithm and the segmentation algorithm. The stereo algorithm is a novel edge-based approach which works by clustering edge contours in an attribute space and forming stereo match pairs based upon nearest neighbour proximity in attribute space. These attribute-space matches are then matched in image space to verify if a true match has been found. If the match is rejected in image space, then the edges are split into smaller edges for matching at the next iteration - a type of coarse-to-fine matching approach. The segmentation algorithm is an adaptive multi-scale algorithm using edge and region information for segmenting intensity images into closed regions. The need for segmentation is determined by region statistics (intensity variance) but is actually performed using edge based information. This procedure produces reasonably robust regions which are used as the basic parts for the learning and recognition strategy. Attributes are then extracted using the output of the segmentation algorithm as the basic object part and using the output of the stereo algorithm for extracting depth information.

The heart of the FCRG object recognition system is the fuzzy conditional rule generation (FCRG) machine learning algorithm. This is an extension of the conditional rule generation (CRG) algorithm described by Bischof and Caelli ([17]). The CRG (and FCRG) technique works by grouping unary attributes from all parts, of all views of all objects into a unary feature space $U$. Each point in the unary feature space is representative of a single part. The feature space $U$, is then partitioned into a number of clusters $U_i$. Some clusters may be unique with respect to class membership, and these clusters uniquely define which class the corresponding part belongs to. Other clusters will contain multiple classes and these need to be examined further. For the non-unique clusters, the binary features between a part and other parts in the view, are calculated to form a binary feature space, $UB_i$. The binary feature space, $UB_i$, is clustered into a number of clusters $UB_{ij}$. Again, some clusters will be unique with respect to class membership while others will not. The unary features of the second part, $p_s$, are used to form another unary feature space $UBU_{ij}$, which is again clustered to form clusters $UBU_{ijk}$. This process can continue until all clusters are unique, resulting in clusters at the $UBUB, UBUBU, UBUBUB, ...$ levels, but, in some cases, clusters may be unresolvable at any level. In this case, the clusters need to be refined at some level, either by reclustering a particular feature space into more clusters, or by splitting clusters. An example of the clustering process is shown in Figure 1.2 ([17]). The main criticism of the CRG classifier is that the rule boundaries are hard (or crisp) boundaries - a part is either inside or outside of the rule, there is no concept of partial membership. This problem is remedied by the FCRG classifier which employs fuzzy rules to allow for more flexible classification of objects.

The outputs of the FCRG classifier are what the classifier believes to be the most likely labels for each of the regions. These are then used as input to the hypothesis verification routine, which uses the original image data to choose the best hypothesis. This is done using a correlation based template matching scheme.

## 1.2 Contributions

The primary contribution of this work is the integration of numerous processing elements into a complete recognition system. Unlike many other systems which utilise explicit depth data, this system learns and recognises 3D objects from 2D intensity images using a non-trivial database of 18 objects. The system is designed specifically for recognising objects in multiple-object scenes, a problem which has not been addressed previously for 2D intensity images using an object database of more than one or two objects. As far as the author is aware, such a system has not

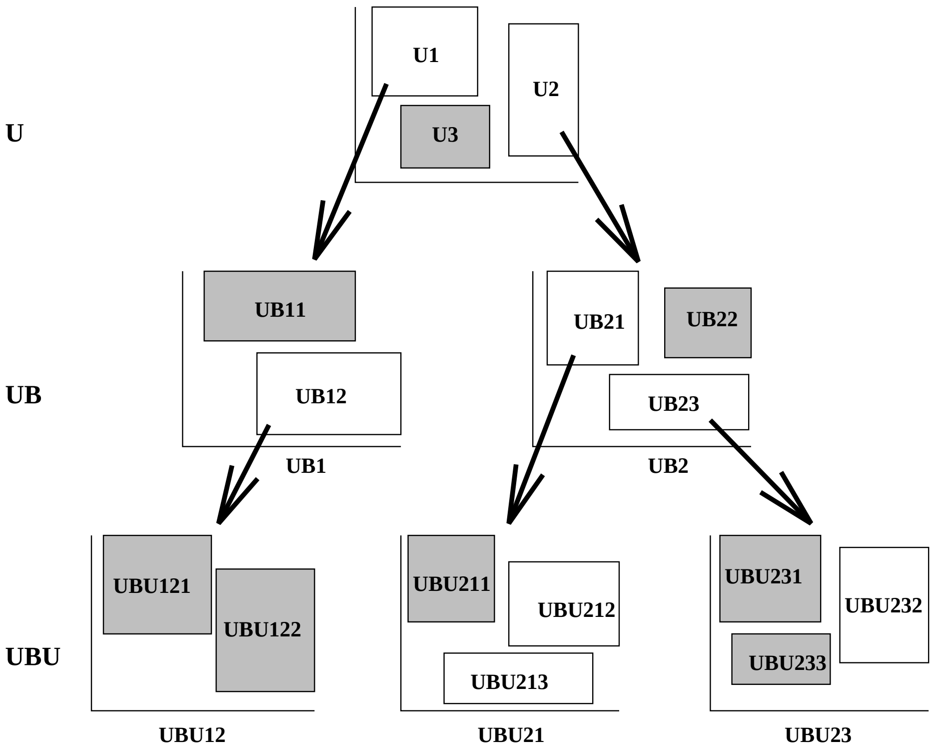



Figure 1.2: An example CRG tree: Displays the tree-like structure of the CRG classifier (taken from [17]). In this figure, grey clusters have zero entropy and are therefore not expanded. White clusters are expanded to the next level.

been previously implemented.

Secondary contributions of the work include: the development of novel stereo and segmentation algorithms with the segmentation algorithm being particularly important; the extension of the CRG classifier to a fuzzy machine learning system and the application of the FCRG system to the problem of learning and recognising 3D objects from 2D intensity images; the use of sparse depth data obtained from the above mentioned stereo algorithm to aid recognition; and the analysis of the performance, with respect to recognition, of a number of unary and binary part attributes.

## 1.3 Outline of the Dissertation

This dissertation is organised in a similar fashion to the workings of the FCRG object recognition system. Chapter 2 gives an overview of the current approaches in 3D object recognition systems. An implementation of a theory of perceptual organisation reported by Mohan and Nevatia ([82]) is described in Chapter 3. Ultimately, it was decided that such techniques were not suitable for the current system, yet it is still important to describe the attempt, and to illustrate the weaknesses of the techniques. Chapter 4 describes the segmentation algorithm which was used in preference to the perceptual grouping techniques of Chapter 3. Chapter 5 describes the novel stereo algorithm. Chapter 6 describes the sets of attributes which were used in the experiments. Chapter 7 describes the CRG classifier and its extension to a fuzzy machine learning system. The implementation of an hypothesis verification routine is described in Chapter 8. The experimental design is described in Chapter 9 and results are presented in Chapter 10. Finally, Chapter 11 discusses the conclusions which can be drawn from the work, some possible future work, and some general comments about the object recognition problem.

# Chapter 2

# Overview of 3D Object Recognition Systems

## 2.1 Object Recognition as a Pattern Recognition Problem

Traditional pattern recognition techniques encode objects or patterns as vectors of characteristic attributes extracted from the whole shape. An attempt is then made to group these vectors so that vectors from the same pattern class are placed into the same group. However, as the patterns become more complex, such techniques become less successful as it becomes increasingly difficult to adequately characterise the whole pattern with a limited number of global attribute vectors. An alternate approach is to segment the pattern into a number of constituent parts (features) each of which can be represented as a vector of characteristic attributes. This is the preferred method of pattern representation in most object recognition systems and other complex pattern recognition problems. In fact, Biederman ([15]) postulated that humans conceptually model objects as a limited number of volumetric components, and different objects are recognised as different combinations of these volumetric primitives (Recognition-By-Components - RBC). However, representing complex patterns in this manner introduces the label compatibility problem: two patterns can be identical in terms of their unary and binary features, and yet be structurally different ([17]). Figure 2.1 displays an example of the label compatibility problem ([17]). Here, the unary features are the colour of the circles and the binary features are the distances between the circles. In this example the unary and binary features are identical for both objects, yet the objects are clearly distinct.

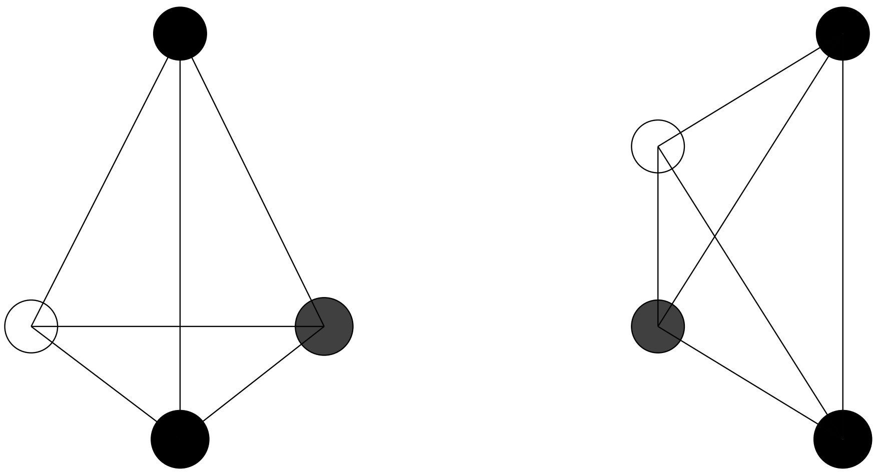


Figure 2.1: The label compatibility problem: Two different objects with identical unary and binary features (from [17]).

The object recognition problem involves recognising objects in an image (view) where each object consists of a number of parts, $p_i, (i = 1, 2, ..., N)$. Each part consists of a number of unary attributes $\vec{u}(p_i)$, and binary attributes $\vec{b}(p_i, p_j), (i, j = 1, 2, ..., N;\ i \neq j)$ which can be calculated between pairs of parts (not necessarily all pairs). In the recognition-by-example problem, we want to learn how to classify each of the samples $S(p_i)$ correctly into each of the classes $C(p_i)$, and also be able to recognise unseen samples.

## 2.2 A Taxonomy of Object Recognition Systems

Numerous approaches to the 3D object recognition problem have been reported in the literature. Although there are many different approaches, an ORS can generally be classified according to the decisions made for each of the following aspects of the object recognition system:

1. Model / Scene acquisition.

2. Model representation.

3. Scene representation.

4. Matching approach.

5. Types of test scenes.

6. Types of object databases.

In the following sections, each of the above categories will be described and the possible choices will be enumerated.

### 2.2.1 Model / Scene Acquisition

There are numerous methods of model acquisition, the most popular being; full 3D models acquired by human input (CAD models), dense 3D views acquired from active sensing techniques (range finders), sparse 3D views obtained from passive sensing techniques (depth-from-stereo), photometric based views (intensity, infra-red).

The most comprehensive and the predominant form of acquisition is the 3D CAD-like approach because it fully describes the model without any sensor errors. This type of approach is most applicable to industrial part recognition, or environments where the objects are well defined and easily modelled via CAD. However, such an approach is not general and requires significant human effort to create the models (unless of course the models are required for other purposes anyway as would be the case in an industrial part setting). Also, it is often difficult to model many classes of objects using CAD models (eg many natural objects).

More recently, range images have become a popular method for model acquisition. This method of acquisition is more general than the CAD approach as it can be used to represent a range of objects that are not conducive to CAD representations, and in general does not require significant human resources to obtain the models. However, it does introduce more problems and uncertainties into the acquisition phase of an ORS. Firstly, there is the problem of "how many views". That is, how many are required to adequately capture all the necessary information about the object (see Section 2.2.2). Secondly, using range images as input, introduces the segmentation problem - models are usually represented using volumetric or surface based descriptions (see Section 2.2.2), but it is a difficult problem (and an area of considerable research) to reliably segment range images into surfaces or volume elements.

Acquiring 3D models using photometric input is probably the most difficult method of model acquisition, although the human visual system appears to use this mode extensively. The same problems exist here as with range image acquisition (ie how many views, segmentation), but a further problem of loss of information in the projection from 3D data to 2D data exists. Many attempts to solve this problem have been reported in the literature (the shape-from-X problem, eg [61], [50], [53], [90], [41] etc), but rarely have these techniques been applied in actual recognition systems. Still, using such data is a valid avenue of research, since photometric images are easier and cheaper to obtain than range images, and are more generally applicable (range finders generally do not work over large distances). Also, the human visual system makes extensive use of such data which tends to indicate that such an approach is particularly useful (although not necessarily optimal) and it is worthwhile to attempt to emulate the human visual system.

The methods for scene acquisition are the same as those for model acquisition, except that acquisition from CAD models is not a valid form of scene acquisition.

### 2.2.2 Model Representation

Current model representation schemes are typically part-and-relations based schemes. In these schemes, objects are represented by a number of primitive object parts and the relationships (usually geometric) between the parts. Another representation scheme is a global based scheme where an object is represented by attributes taken from the whole object itself. This has a specific disadvantage in that it is difficult to perform recognition when parts of the object are absent (eg due to occlusion). Hence, the current paradigm in representation is a parts-and-relations one. However, this paradigm can also be split into two opposing methods of representation - object centred and viewer centred.

Object centred representations are typically CAD-like and are generally an intuitive form of representation. Such representations typically fully characterise an object and may even include information that is not interesting or necessary for recognition purposes (may contain information about non-visible interior volumes or surfaces for example). However, object centred representations are non-commensurate with the input scenes which an ORS may wish to classify. This is because input scenes are typically viewer centred and a form of translation - either from object centred to viewer centred, or vice versa - is generally necessary for recognition purposes. Obviously, this introduces the problem of representation translation, which is not a simple problem, and is particularly evident if a full 3D object centred model must be acquired from viewer centred training data. Even so, such representations are useful, especially in such areas as industrial part inspection, where full CAD models of objects are available. Typically, these representations are volumetric in nature - object parts are a set of volumetric primitives (eg geons [15], generalised cylinders [16]) - but surface based representations are also possible (although not as common for object centred representations).

The other major paradigm for object representation is viewer centred. Here, each view of an object is represented separately to other views (with possible information characterising the relationships between adjacent views). The advantage of this is that translation between input data and model data is generally not required, simplifying the matching process (though not necessarily improving performance). Problems are created however. Predominantly, the problem of how many views and at what spatial position to measure input is particularly troublesome. In general, more views are better in terms of information content, but obviously, the fewer views, the more compact the representation. Some work has been done on extracting a small set of topologically different views (characteristic views) from a large set of “training” views and this is a promising area of research ([103], [91], [44], [104], [22], [34]). Such representations are mostly restricted to surface (or region) based information. This representation may be 2D or 3D, depending on the type of input and the pre-processing performed. Also, because of the need for multiple views of each object, the representation may be less compact than object centred representations.

### 2.2.3 Scene Representations

Input scenes are always viewer centred by definition of the object recognition problem. As discussed in Section 2.2.1, the two most common types of sensing input data are active sensing (range) and passive sensing (intensity). Although it has been stated that range data should be more appropriate for 3D object recognition, the major problem of using input data obtained via passive means also exist using active sensors anyway - that is the problem of data segmentation. Nevertheless, fairly reliable forms of dense 3D data segmentation have been devised (eg [11]), although the form of these methods is generally determined by the type of representation used. In general, 3D representations are either surface based or volumetric based - objects are represented by either surface or volumetric primitives and the relationships between adjacent primitives (an object "graph"). Volumetric primitives (eg generalised cylinders [27], geons [15]) are generally more complex than surface based primitives, and as such can be used to represent and index complex objects more easily. However, due to their complexity, volumetric primitives are often difficult to reliably extract from range data. Alternatively, surface primitives ([11], [110], [48]), are simpler and easier to extract than volumetric primitives, but object models tend to be more complex and hence recognition is often more expensive (though not necessarily less accurate).

An alternative to active sensing, and one which the human visual system makes extensive use of, is passive sensing techniques (typically intensity). Use of intensity images also introduces the segmentation problem, one which in many cases has not been solved adequately. The two major approaches for segmenting intensity images have been edge based and region based approaches, although the predominant approach has been edge based ([32]). The number of representations which have been proposed for 2D imagery is large, including; 2D contour groupings ([74]), region based representations; and geons ([10]). However, since the segmentation of intensity images is so difficult, many researchers investigating recognising 3D objects from 2D images ([10], [38], [80]), have conveniently ignored the problem to focus upon the learning and recognition issues, by using "perfect" line drawings instead. Also, data extracted from 2D intensity images is not confined to 2D data. A significant amount of work has been performed on shape-from-X techniques, but there has been little application of this data to 3D ORS systems, rather researchers have attempted to solve the shape-from-X problem in isolation to other problems in computer vision (this tendency of attempting to solve sub-problems in isolation has in fact been the predominant approach of vision researchers thus far). Recently, some researchers have attempted to use 3D data extracted from 2D line drawings ([10], [38]) using geometric reasoning techniques, but there seems to have been no work done using other forms of shape-from-X (eg stereo, shading, focus etc), although many researchers refer to these techniques as possible extensions.

### 2.2.4 Matching/Recognition Approaches

In recent years their have been 3 predominant matching approaches in 3D ORSs. These are:

- Graph matching approaches.
- Feature indexing / hash tables.
- Evidence / rule based approaches.

In the graph matching approach ([27], [6], [9], [110], [68], [45], [119]), graph nodes represent object parts and graph edges represent the relationships between those parts. The problem then becomes a sub-graph isomorphism problem, where a given scene graph is generally a sub-graph of a full model graph, and the best scene-model graph match must be found from the database of model graphs. Unfortunately, the sub-graph isomorphism problem is NP-complete and hence computationally expensive. Typically, coarse matching schemes are initially employed across all object models, to eliminate many of the possible models from contention. Computationally expensive full graph matching algorithms are then applied to the few remaining object models to find the most likely scene to model match. Nevertheless, the initial coarse matches need to be performed on all object models, and this may become computationally infeasible as the number of objects in the model database becomes large (eg $> 1000$). Still, the graph-matching solution is one of the most comprehensive approaches to recognition, being probably the best method for distinguishing between objects. Although, because the approach can be fine tuned, it often suffers from problems in the presence of noise (due to segmentation errors, sensor errors, etc).

An alternative approach which attempts to solve the slow matching problem evident with the graph matching approach, is the feature indexing or hash table approach ([48], [103], [39]). In this approach, scene features are first extracted from the scene (eg lines, regions, groups of lines, groups of regions), attributes of these features are computed, and these attributes are used to index into a hash table, with each entry in the table containing a list of possible model hypotheses. All model hypotheses are then used in a verification step - usually a graph matching approach. This approach has the advantage of quickly reducing the search space of possible models, without performing any actual searching - a big advantage especially when large object databases are considered. However, the advantage does come at a price. As the number of objects increase, it is necessary to increase the size of the hash table such that the number of hypotheses in a given table entry does not become too large. Increasing the size of the hash table generally requires the use of more complex features which are more difficult to reliably extract.

Another matching approach which has been successfully employed in 3D ORS ([66], [31]), is the evidence/rule based approach (EBS). In contrast to the other two techniques described above, this technique is a machine learning technique, that learns to recognise objects from the training data - essentially a learning extension to the feature indexing techniques. As with other techniques, features (eg surface patches) are first extracted from an input image, attributes are then computed from these patches (eg surface area), and from the relationships between the patches (eg distance, normal angle, etc). These two sets of attributes are grouped into separate attribute spaces, and rules are formed in these spaces which evidence a particular object from the object database. This evidence is typically in the form of the probability that a given attribute vector (part or relationship) would give rise to a given object model, and is calculated as the relative frequencies of the training objects which mapped into that particular rule. The evidence from all the instantiated rules is then used to decide which objects are in the scene (neural networks were used in [31] for this stage). Such an approach is general and efficient and only suffers from one major problem - the label compatibility problem. Since the unary and binary attribute spaces are considered separately, it is possible that two objects can be identical in terms of their unary and binary attributes, but still be structurally different.

### 2.2.5 Object Databases and Test Scenes

In the literature reporting on the success or otherwise of a particular ORS has been extremely ad hoc. Many claims have been made about how good particular systems are, or what their potentials are (mostly by the developers of the system) - and many of these are unsubstantiated. No systems, to the author's knowledge, have come close to addressing all the problems inherent in such a complex domain. Some systems have been evaluated using only single object databases ([25, 27], [74] [6]), many have been evaluated using relatively few objects - many systems with $< 10$ ([66], [111], [103], [45], [72], [39]) and all systems (as far as the author is aware) with less than 100 objects. This is many orders of magnitude below what the human visual system can deal with. Also, many systems are tested using synthetic data with only one object per scene ([6], [92], [66], [31], [9], [48], [119] [72]). Few systems use real data with possibly multiple objects in a scene, and those that do have typically only one object to recognise ([25], [74]). In light of all these observations, it is clear that there is still much to achieve before ORSs approach anything near that of human performance.

## 2.3 Literature Review

A number of ORSs will be described in the following sections, with an indication of how these systems relate to the above taxonomy. Also, limitations of the ORSs will be described.

### 2.3.1 ACRONYM

Earlier intensity based object recognition systems include ACRONYM, which was developed by Brooks ([25, 27]). Models are acquired in ACRONYM from full 3D CAD models, and are represented as relational graphs of volumetric primitives called generalised cones ([16]). The recognition approach used by ACRONYM is a complicated constraint manipulation system that performs a type of graph matching. The scenes which ACRONYM recognises are real 2D intensity images which possibly contain multiple objects per scene and from which edges are extracted. The number of objects in the object database reported is one or two. The system was quite successful at finding aeroplanes in aerial photographs of an airport, but it is difficult to quantify the successfulness of the approach, and impossible to guess its applicability to other domains.

Generic models were also modelled using geometric and relational constraint bounds of the form $lowerbound < expression < upperbound$ (eg $length < 5 \times width$). Geometric reasoning techniques are utilised to predict observable features that are likely to be invariant, and these are typically formed by means of ribbons and ellipses. Observed ribbons and ellipses are then used to constrain models based on the predicted image features. Recognition commences at a local level, and constraints formed from these initial interpretations are utilised by a constraint manipulation system to form globally consistent interpretations. Although ACRONYM is a general and complex system, it does suffer from a number of limitations. Firstly, the system is model based, and no thought is given to acquiring object models, it is assumed that the object models are fully specified to the system. Secondly, the performance of ACRONYM was only demonstrated on very few object models (a model of an aircraft in [25, 27]). If a large number of objects were present in the object database, then the method for predicting and constraining possible interpretations of the image would need to be improved.

### 2.3.2 SCERPO

Lowe ([74]) describes SCERPO, a system for recognising 3D objects from single 2D images. Object models were acquired and represented using a 3D CAD-like wireframe representation. Scenes are in the form of 2D intensity images from which edges are extracted. The recognition approach can be considered as a matching approach, where the most perceptually salient features from the image (features include collinear edge segments, parallel edge segments, trapezoidal shapes etc) are matched with the object models first. Test scenes consist of complex scenes which contain multiple objects - although for the tests reported, only scenes containing multiple instances of a single object were used. The object database contained only 1 object, and most instances of this object were found in a test scene.

No depth reconstruction was attempted in the recognition stage, and hence the system relied on purely 2D intensity features. The recognition process involved the following stages; edge detection, line segmentation, perceptual grouping, matching and search, and model verification. The first two stages involve detecting zero-crossings of the Laplacian-of-Gaussian convolved image using a Sobel gradient operator, linking edge points into connected edge curves, and segmenting these edge curves into linear edge segments. Techniques were then employed to detect perceptually salient features such as collinearity, endpoint proximity and parallelism. These groupings were then grouped into higher order perceptual groupings such as trapezoidal shapes and skewed symmetries (representing projections of rectangular surfaces in 3D space). These perceptual groupings were ranked according to the probability of such groupings occurring in a non-accidental way. The groupings were pre-computed off-line for all object models and a list of all possible groupings was maintained for each object model. Scene perceptual groupings then had to be matched to each possible model grouping of the same type for all object models. The most probable scene grouping is matched first, and for its most likely model match, the pose of the object is estimated using a technique developed by Lowe ([74]). The object model was then projected onto the image data to verify the match and possibly restrict the search space of other matches in the image (model projection may verify other groupings not involved in the grouping used to predict the object and its pose).

There are limitations to the approach used in SCERPO. Primarily, a search and match must be made between groupings in an image and all possible model groupings of the same type. Lowe ([74]) mentions that as the number of models increase (he uses only one in [74]), the groupings would need to be more complex to reduce the potentially large search space. However, as features become more complex, it typically becomes more difficult to extract them from images that are prone to noise (which is certainly the case in 2D intensity images). Also, it is not clear that such an approach would work when the object database became large as the search space for matching perceptual groupings is likely to be large regardless of the more complex groupings in use.

### 2.3.3 Bamieh and Figueiredo

Bamieh and Figueiredo ([6]) describe a graph based method for recognising 3D objects from a single image. Object models are acquired using a CAD based modelling system and are represented using a surface based attribute graph. Test scenes are input as "perfect" 2D line drawings and an attributed graph of the scene is extracted. The matching approach used here is graph matching. The test objects were fairly simple 3D polyhedrons, and were presented only one per scene, with an object database of only one object. Hence, it is not sensible to try and evaluate how good the system is based on such a small example set.

Objects are represented as an attributed graph describing the geometrical shape and the structure of the object's surface. Each surface is represented using a number of general moment invariants. During recognition, it is assumed a wireframe of the scene to be recognised can be extracted. A graph of the scene is then extracted and the problem is reduced to a sub-graph isomorphism problem. Unfortunately, sub-graph isomorphism problems tend to be computationally expensive and although the recognition time reported is reasonable, the system was only demonstrated on scenes containing a single object. The problem of recognising scenes containing multiple objects was not addressed, nor was an object database with more than one object used. Also, the system was tested on mainly noiseless images of objects with polygonal faces, and thus did not address the problem of generality. Finally, once again, the system (as reported in [6]) was model based and the problem of model acquisition or model learning was not considered.

### 2.3.4 Poggio and Edelman

Poggio and Edelman ([92]) describe a neural type network that utilises generalised radial basis functions for learning object models from examples. Models are acquired via synthetic 2D images of 3D wireframe objects which are represented simply as the vertices of the object. The test scenes were of the same type as the learning scenes (though not necessarily identical) - synthetic images of single wireframe objects, and the number of objects used in the database is one object per neural network (only one network is presented in the paper). The recognition approach used is a neural network based on radial basis functions which can be viewed as an evidence based technique (the activation of a particular neuron is similar to the evidence associated with a given evidence rule).

However, their system suffers from a number of disadvantages. Firstly, the objects used for testing the system are wireframe objects from which all features are visible from each view. Hence, the problem is simplified because little self-occlusion is evident, and the object parts are simple and can be represented by the vertices of the wireframe. Secondly, for each object a new network is required for the learning and recognition of the object, and this would be a prohibitive approach if the object database contained many objects. Thirdly, only one object is presented to each network at a time, so presumably the object segmentation problem would have to be solved prior to recognition. Edelman and Weinshall ([43]) describe a similar approach that utilises a two layer network of thresholded summation units and an unsupervised Hebbian learning strategy. However, they also use only wireframe objects and only present one object at a time to the system.

### 2.3.5 Jain and Hoffman

Jain and Hoffman ([66]) describe a system based upon evidential reasoning and Caelli and Dreier ([31]), describe an extension to the system. Models are acquired from either synthetic or real range based images and are represented using a surface based graph approach. The recognition approach used by both systems is an evidence based approach, although the systems use different methods of rule generation and evidential reasoning. The test scenes are also in the form of 3D range images of a single object. The number of objects in the database for Jain and Hoffman is 10 and Caelli and Dreier use between 5 and 25 objects to test their system. The test scenes in both systems are single object images. Jain and Hoffman test their system by training on 10 views of each object and then testing on another 5 views of the object, for which they report 100% recognition rate. The reported recognition rate is impressive, however, many of the attributes used in the system are not size invariant and thus restrict the applicability of the system, and all of the objects used are quite distinct in shape from each other. In contrast, Caelli and Dreier test their system in a domain that contains many similar objects. In general, they achieve a recognition rate of around 80%, which is quite good considering the objects used in the experiments.

In the original system developed by Jain and Hoffman ([66]), attributes of segmented (range) view parts were used to determine feature bounds which evidenced different objects and these rules were then evaluated over individual object samples to result in an overall estimate of a given sample view's classification. Segmentation of a range image is performed by clustering the points according to $x, y, z$ coordinates and the three values associated with the surface normal at each point $(n_x, n_y, n_z)$. Each surface patch was then classified as either planar, convex or concave, and boundaries between surface patches were classified as either jump or crease boundaries. In the case of over-segmentation of the input range image, merging of adjacent surface patches was performed based on patch and boundary classifications. Three types of representations were used for each of the objects; morphological information which characterised the range image as a whole, surface patch information describing a particular patch, and boundary information describing relationships between patches. These attributes were then used to gain evidence for each of the possible object classes from a rule-base which was either hand-coded or automatically generated. The object class is determined by which class has the maximal evidence given the range image. This evidential system was then extended by Caelli and Dreier ([31]) to generate more discriminating rules in automatic rule-base generation, by using different methods for generating attribute bounds (on parts and relations) and evidence values.

These systems have been quite successful with recognising isolated objects where many objects have been used (in excess of 20). However, it is unclear how such systems would handle multiple-object scenes, and this point needs to be investigated in relation to EBS systems. Also, such systems typically do not explicitly handle the label compatibility problem where two objects are identical in terms of unary and binary attributes, but are structurally different ([17]).

### 2.3.6 PARVO

More recently, Bergevin and Levine ([9]) describe PARVO, which is a system that performs generic object recognition from 2D line drawings. Models are acquired using a CAD modelling system and are represented using coarse volumetric primitives call geons. The recognition strategy used by the PARVO system is basically a graph-matching type of approach. The test scenes are in the form of 2D line drawings and are input one object per scene during recognition. The object database consists of 25 generic (but somewhat complex) objects. Experiments were performed using each of the objects as the test object in a single object scene. Of these, 15 of the 23 objects were unambiguously discriminated, and 2 others produced identical scores for the correct model and one other model - resulting in 17 out of 23 correct classifications (74%). Only one view of each of the objects was presented however, and comprehensive testing over a range of views was not reported. Nevertheless, the performance of the system is quite good.

This system is an attempt to implement the "recognition-by-components" theory developed by Biederman ([15]). PARVO recognises objects from a line drawing of a single object by first generating a coarse part-based volumetric description of the object which is achieved by first segmenting the line drawing into parts using perceptual organisation techniques (cotermination, parallelism, symmetry). Each part is then labelled as one of a number of volumetric primitives ("geons") using an hypothesis and test procedure. The object models to be recognised are specified to the PARVO system prior to object recognition via a generic graph representation (the geon type is represented as a graph node, and relations between the graphs are represented as the graph arcs). Since model and scene representations are equivalent, the problem becomes a sub-graph isomorphism problem. However, due to the computational complexity of such an approach, only objects that contain all geons found in the image are considered as possible interpretations of the image. From the remaining possible object models, a similarity measure is computed between the object model and the scene. The model which scores the highest similarity is considered the most likely object in the scene - a type of "generalisation" procedure.

PARVO, as presented by [9], suffers from a number of fundamental problems. Firstly, the system can only handle one object at a time. This means that the segmentation of an image into separate objects must be performed prior to recognition. Currently, there is no evidence to suggest that such a segmentation is possible, and given the nature of complex scenes, it seems unlikely. Secondly, there is no evidence to suggest that volumetric features are better than surface features for the recognition task, especially since the extraction of volumetric features from single 2D images is an under-constrained problem and some volumetric features are simply not visible. Thirdly, PARVO requires that the input line drawings be "perfect", or near perfect, for successful extraction of the volumetric primitives (geons) and it is unclear how the system would cope with imperfect input and it would certainly not be able to cope with interior surface markings. Finally, PARVO is a model based object recognition system that assumes the input of CAD-like models and does not address the model acquisition problem.

### 2.3.7 Flynn and Jain

Flynn and Jain ([48]) describe a recognition system for recognising 3D object from 3D range images, although presumably, the techniques described are applicable to other environments. The models are acquired using a CAD modelling system and are represented by a surface based graph model. The test scenes are in the form of dense 3D range images which are pre-processed into viewer centred surface based graph representations. The basic recognition strategy is a feature indexing / hash table type approach. The test scenes are presented with one object per scene and the number of objects used in the object database is 23 objects. This paper is one of the few that report comprehensive testing of the system. Here, 5 views are generated randomly from each of the objects over the entire viewing sphere, and each view is then recognised by the system. All new views were correctly recognised by the system - an impressive result. Nevertheless, many of the objects are simplistic, and it would be interesting to see how the system would perform on more complicated objects. Also, for many of the objects, a large number of hypotheses were generated ($> 20$), each requiring verification, which was a slow process.

The system utilises hash tables using invariant features extracted from 3D images. The 3D objects are modelled using 3 types of surfaces - plane, cylinder and sphere. For table building and recognition, groups of 3 surfaces are used to generate the features. This results in ten possible types of groupings (eg plane-plane-plane, plane-cylinder-sphere etc), and hence ten tables are constructed for each of the groupings. Each is a 2D table relating two of the three model surfaces in the triple (which two to use is decided prior to table building). In the case of triples that contain 2 or more types of the same surface, redundant entries into the table must be made since the ordering of surfaces is not known a priori in image scenes. The tables are generated automatically from CAD models of the objects. At recognition time, a range image is input to the system of a single object only and is segmented into surfaces using a clustering procedure described in [47]. Recognition proceeds by finding all triples of surfaces in a scene and hypothesising likely models from the model lists maintained in the feature tables. The correct model can be maintained by hypothesis verification or using geometric predicates ([47]).

There are a number of limitations to the system described by Flynn and Jain ([48]). Firstly, the system is model based and is only tested on single object scenes. It would probably be possible to apply the system to multiple object scenes, but this would most likely lead to an explosion of generated hypotheses and the authors have not addressed this problem. Secondly, the number of hypotheses which are generated (even in single object scenes) is potentially large, in some cases the system generated greater than 100 hypotheses. It was unclear if many of the objects which caused large numbers of hypotheses were simple or relatively complex since this fact was not mentioned by the authors and not all objects used were described. Nevertheless, it is apparent that the number of generated hypotheses would increase with the number of objects in the object database, and this would have adverse effects on system performance.

### 2.3.8 Stein and Medioni

Stein and Medioni ([111]) describe a system for recognising 3D objects from 3D imagery. Models are acquired from dense 3D range data and are represented using a combined surface and edge based graph representation. The basic recognition approach is a graph matching technique. The test scenes are input using dense range images and possibly consisted of multiple objects, and are drawn from a database of 9 objects. The results of four experiments are reported - two experiments with a single object per scene, one with two objects, and another with three objects. In all scenes, only the objects in the scene were considered part of the object database, so the problem was essentially reduced to an expected object recognition problem. All experiments reported successful results, but these experiments are obviously biased in favour of the recognition system.

Both models and scenes are represented in the same fashion using two types of primitive descriptors - super segments representing depth discontinuities and splashes which represent small surface patches. Super segments are basically a polygonal representation of a 3D curve. Splashes represent surface patches with a surface normal at the reference point, $p$, and surface normals at points along a circle of fixed radius, centred at $p$. Splashes are calculated only at points of high curvature to reduce the representation space required and to provide richer object descriptions. Codes are calculated for each image feature (super segment or splash) and these are used to index into a hash table. Recognition proceeds by finding all features in a scene, encoding them and retrieving hypotheses from the hash table. A final classification is made based on the results of a verification procedure. The verification procedure computes all possible matches between scene parts and hypothesised model parts by calculating the required 3D transformation. Non-rigid transformations are discarded immediately. All consistent hypotheses for each possible model are clustered together (according to transformation parameters) and the final transformation parameters are calculated for each of these clusters if more than three hypotheses are clustered together. The scene object is then classified depending on the best matching model transformation.

The major advantage of such a scheme is that both models and scenes are represented in the same way. That is, both are input into the system using dense range data. However, no mention was made of how multiple training views of the same object would be handled. Also, the reported testing of the system typically included only a few objects in the object database, and this raises important questions about the scalability of such an approach. As the database increases, the number of hypotheses generated may increase rapidly, possibly reducing the performance of the system.

### 2.3.9 Siebert and Waxman

Siebert and Waxman ([103]) describe a system for learning and recognising objects from multiple 2D views. Models are acquired from a temporally related set of 2D input images (object silhouettes) and are represented using an aspect network of 2D aspects of the object. The form of matching employed is a variant of feature indexing which has been called aspect indexing. The object database consists of 3 objects, and the objects are presented singly in a temporally related image sequence. The recognition results for two view sequences are presented with both sequences being recognised correctly. Comprehensive testing of the system was not reported.

The test objects are different types of aeroplanes and these are input into the system via a sequence of temporally related frames. The objects are painted black and are presented on a light background so the problem of figure ground separation is avoided. The silhouette of the input object is found and corners are detected along this boundary. These corner points are used to form a global object representation (in terms of an attribute vector) which is invariant to 2D affine transformations (position, orientation and scale). Aspect networks are constructed based upon training sequences of images and temporal data is also maintained between related aspects. Aspects are learned via a clustering type approach and if the distance between an input aspect and a previously formed cluster is too great, then a new aspect is formed. A graph is constructed from all learning aspects which represents aspect adjacencies and these are encoded in a view transition matrix for that object. Recognition of an input image proceeds by finding the most likely aspect (which may or may not be unique to a particular object - that is different objects can share the same aspect) and this evidence is fed through all view transition matrices resulting in evidence for each of the possible objects. Evidence is accumulated over time and at any time the object with the most evidence is considered to be the object in the scene. Also, the view transition matrix of the winning object is updated to reflect possibly new information from test scenes.

The scheme described by Siebert and Waxman appears to be reliable for recognising objects which can be discriminated based upon their silhouettes and when the input involves sequences of temporally related images. They demonstrate their system using 3 different types of model aeroplanes which are moved across the field of view of the camera - a domain for which their system is particularly applicable. Also, recognition time can be made constant due to the massively parallel nature of feeding the aspect through the view transition matrices. Unfortunately, the space requirements for the system will increase dramatically with each new object as a new view transition matrix is required. Also, it is unclear how the system would handle domains which required more complex object representations and how accurate the system would be using large object databases. Obviously, the current system would perform poorly in the presence of occlusion due to the global nature of the representation, although presumably, this would be less of a problem for moving objects. Finally, the problem of multiple object scenes has not been addressed.

### 2.3.10 Kim and Kak

Kim and Kak ([68]) describe a model based 3D recognition system using the ideas of discrete relaxation and bipartite matching. Models are acquired from a CAD modelling system and are represented using a surface based attributed graph representation. The main recognition approach utilised is a graph matching technique. Test scenes are presented as dense 3D range images with both single and multiple object scenes, with the object database consisting of 12 objects. Results are reported using two images - a single object scene and a multiple object scene. The single object scene was correctly classified, but no overall classification rate for all objects was reported. In the multiple object scene (a complex image with 5 objects and numerous occlusions), 3 out of the 5 objects were correctly recognised, and considering the complexity of the scene, this is a fair result. However, results over a wide range of scenes were not reported.

Objects are modelled using an object centred surface representation in the form of an attributed graph which is generated from a CAD representation, and test images are input to the system via dense range images. Graph nodes represent the object's surfaces and graph arcs represent edges between the surfaces. Surface attributes depend on the surface type which could be one of; planar, cylindrical, spherical, conical or other quadric. Similarly, edges are represented by edge type (convex, concave, inflected or a combination of concave and convex) and dihedral angle. Since object models are completely specified, no learning is required. Recognition proceeds by first extracting surface and edge features and their attributes from a scene. Hypotheses are formed by testing the scene objects against all model objects by applying a number of filters in increasing order of computational complexity. The filters reject inapplicable models by; comparing the order of model and scene graphs, checking for node consistencies, and finally checking for at least one complete match using a bipartite matching scheme. Further hypothesis pruning is achieved by combining bipartite matching and discrete relaxation labelling to check for relational consistencies. Verification is then performed via pose estimation and ambiguity resolution techniques.

The main advantages of such a system is that it handles complex objects and only performs expensive computations on a limited number of hypotheses. However, methods of early hypothesis rejection need to be applied to all object models in the model database and this could still prove to be computationally crippling if large object databases were used.

### 2.3.11 Fan, Medioni and Nevatia

Fan, Medioni and Nevatia ([45]) describe a surface based object recognition system that uses range images for both learning and recognition tasks. Models are acquired from dense 3D range images and are represented using a surface based graph approach. Test scenes are input and represented in a similar manner and may contain more than one object, although the recognition task is dependent on a preprocessing scheme which separates objects in a scene, thus reducing the generalisability of the approach. The object database consists of 10 objects and the major form of recognition is a graph based matching approach. Test results are displayed for only 3 scenes for which perfect recognition is attained, although a more extensive set exists ([46]), no indication of the recognition performance for all the scenes is reported. Also, the multiple object scenes which are used for testing are formed by taking range images of the objects separately (ie one per scene) and synthetically merging two or more scenes to create a multiple object scene. Such an approach for testing the system on multiple object scenes is quite suspect as many of the problems evident in such scenes (mainly shadows) were conveniently avoided.

From an input range image, jump boundaries and creases are first extracted from curvature zero-crossings and extrema. These boundaries are then extended to enforce them to be closed and the regions produced are assumed to correspond to elementary surface patches. These patches are then approximated using a second order polynomial in three dimensions. An attributed graph representation is then built up from the surfaces extracted from the input image where graph nodes refer to surfaces and graph arcs refer to the geometric relationship between adjacent surfaces. Attributes of the surfaces include; area, orientation, mean principal curvature, etc. Attributes of the arcs include; adjacency type (convex, concave or jump boundary) and connection possibility (which is dependent on the type of boundary).

Object segmentation is performed by reasoning about relationships between adjacent patches. The matching process is divided into three phases; screener, graph matcher, and analyser. The screener performs a coarse match (using attributes such as the number of nodes in each graph) between scene and all object models and views and outputs an ordered list of candidate matches (with at most 5 candidates). An extensive graph matching process is then performed to find the most likely match. Finally, the analyser analyses the output from the graph matcher and possibly provides critical feedback to correct for incorrect segmentation. The system is tested using 10 objects for which 32 training views were used.

The advantages of the approach described here is that multiple scenes can be handled since objects are separated prior to matching. Also, both models and scenes are acquired from input range images and thus removing the need for a transformation between the two representations. However, the authors do not address the problem of determining how many views per object should be used nor mention grouping training views into characteristic aspects. As with numerous other systems, as the number of objects becomes large, the simple hypothesis rejection methods will become computationally infeasible since these must be applied to all models in the model database. Also, although the system can handle multiple objects in the current domain, the recognition procedure still requires single objects to be input at a time. The problem is overcome here by segmenting objects by reasoning about adjacency relationships of surfaces, but this technique can only be applied to range data and is not applicable to 2D imagery.

### 2.3.12 Wong, Lu and Rioux

Wong, Lu and Rioux ([119]) introduce the concept of attributed hyper-graph representations (AHR) for recognising 3D objects from range images. Models are acquired from CAD based models but can also be extracted from multiple dense range images although it is unclear how effective the latter approach is. The models are represented using a surface-based hyper-graph representation and the recognition approach is a graph matching approach (although hierarchical due to the nature of the hyper-graph representation scheme). Test scenes are presented singly in an image and the number of objects used in the database is 44, although the objects are typically simplistic polyhedral shapes with some cylindrical and spherical shapes. Tests were performed on all objects and only one object was mis-classified, although it is unclear which views of each of the objects were used for the tests (or if multiple views were used). The recognition results reported for this system are quite good, but since only simplistic objects were used, it is difficult to make any generalisations about the usefulness of such an approach in more complex domains.

Nodes of an AHR are associated with elementary area attributed graphs which in turn represent basic surfaces using an attributed graph representation of the surface. The surface is represented by its boundary segments (graph nodes) and the geometrical relationships between these segments (graph edges). Edges of an AHR correspond to so called primitive blocks in the model which are again represented by an attributed graph with surfaces corresponding to graph nodes and relationships between the surfaces corresponding to graph edges. A primitive block is defined as a collection of surfaces for which no surfaces have a concave relationship with neighbouring surfaces in the primitive block. Model AHRs can be generated from CAD models or calculated from multiple range images of an object and then combined to form a single AHR. Recognition commences by eliminating possible models using a coarse matching approach. Initially, models are rejected if a scene AHR contains a surface type that the model AHR does not (surface types include; cylindrical, conical, planar, etc). Further eliminations are made by matching AHR nodes and edges separately and in isolation. Finally, a more complete graph match is performed on all remaining hypotheses to select the most likely object. Matching between AHR nodes and edges is performed based on finding mono-morphisms between the attributed graphs associated with those entities. This monomorphic matching is also extended so that it can be applied to AHRs as well in the final verification stage of recognition.

The good points of the above approach are that model AHRs can be built directly from range images of these models and different views of the model can be combined to form a complete model AHR. Also, the representation is hierarchical and elegant and the monomorphic matching algorithms are valid for both levels of the hierarchy. However, the system can not tolerate any errors in the range image segmentation process thus limiting its applicability for real problems. Also, the representation scheme is limited to objects which have well defined parts with easily identifiable surface types. Multiple object scenes are not handled and the computational complexity of the approach is liable to increase dramatically as the number of objects become large and the object complexity increases.

### 2.3.13 Lin, Liao, Tsao and Lingutla

The authors ([72]) describe a system for recognising 3D objects from 2D line drawings using a multi-view matching approach with a Hopfield network. Models are represented by a number of 2D aspects which are extracted from CAD-like model representations. Test scenes are presented to the system in the form of "perfect" 2D line drawings with a single object per scene. The major recognition approach used is based on a Hopfield neural network which is a type of associative memory and is somewhat related to graph matching. The number of objects used in the database is 3 and only the results of 2 experiments (2 objects presented to the system) were reported and both were successfully recognised. This is a very limited set of tests and it is nearly impossible to draw any general conclusions from them.

No learning takes place in the system, rather models are compared directly using Hopfield networks. Firstly, coarse surface based information is matched using a 2D Hopfield network with the scene surface attributes being input via column neurons and model surface attributes being input via row neurons. The weights in the network are adjusted via Hopfield dynamics until the network stabilises, at which stage the most likely hypotheses are indicated by neurons with high activations. These hypotheses are then used as input to the finer matching stage which matches vertices which are evident in the selected model views and the scene. Again, a 2D Hopfield network architecture is used to extract the most likely model in a similar way.

The advantages of such an approach is that a Hopfield network can be implemented on a parallel architecture making the matching process fast. Unfortunately, to gain maximum parallelism with the system described, the Hopfield networks must be large enough to accommodate all model views of all objects at the same time, otherwise the model must be matched in a linear fashion. As the object database becomes large, the total number of views required increases dramatically, requiring either an extremely large Hopfield network (which would probably be prone to numerous local minima) or a long recognition time. Also, the current system was tested only on a few objects (3) and the input to the system was perfect line drawings (unlikely in real applications). Finally, the problem of handling scenes with multiple objects was not addressed.

### 2.3.14 Dickinson, Pentland and Rosenfeld

The authors ([38, 39]) describe one of the most comprehensive systems for recognising 3D objects from 2D views. Objects are modelled using 3D volumetric primitives in a CAD-like model. Test scenes are presented to the system using “perfect” 2D line drawings and the system can apparently handle multiple object scenes. The main form of recognition is an indexing type approach. It is unclear from the paper just how extensive testing of the system was, making it difficult to determine how many objects were used in the test database, but it seems that few objects were used ($< 10$).

Their major contribution is the means by which 3D volumetric primitives are extracted from a 2D image. The described system used a set of 10 volumetric primitives similar to the geons described by Biederman ([15]), but the system is not dependent on the type or number of volumetric primitives which are used. The primitives are represented by a hierarchical representation of aspects, faces and boundary groups. At the top of the hierarchy are the primitives themselves. All possible aspects of the primitives make up the next lower level with aspects possibly being shared by a number of primitives. All the faces (rectangular, spherical, etc) of each of the aspects comprise the next level of the hierarchy which are again possibly shared by different aspects. At the lowest level of the hierarchy are the possible boundary groups involved in the faces (parallel lines, curved contours, symmetrical curved contours, etc). Each element in the aspect hierarchy is assigned a probability which indicates the likelihood of that element being visible in an image, and can be calculated from the object models.

The first step in the recognition phase is to recognise the primitives that are visible in the image. Firstly, regions need to be extracted from the image, possibly using one of the many segmentation algorithms reported in the literature. Contours bounding the region are then segmented based on curvature discontinuities along the boundary. Faces are represented by a graph where nodes of the graph represent boundary segments and graph edges represent relationships between adjacent segments. The extracted face graphs are then matched to all possible model face graphs (typically a small number) and for all possible matches, a face hypothesis with a probability is generated. If no matches are found, then the aspect hierarchy is descended and an attempt is made to match boundary groups of the face with model boundary groups, which again enumerate possible face hypotheses. This scheme results in the generation of a probabilistic face graph, where each node in the graph represents a given face and contains face hypotheses and their associated probabilities. Graph edges correspond to the geometric relationships between adjacent faces.

The face graph is then matched against the aspect graphs and a number of aspect hypotheses are generated and a verification technique is used to keep only the most likely aspects. This produces an aspect graph of the image and a similar process is performed to generate primitive hypotheses, resulting in a volumetric primitive graph description of the scene image. In the unexpected object recognition problem, a hash table is used to generate a number of object hypotheses. Indices into the hash table are computed from attributes of volumetric primitives which are strongly connected in the scene (the hash table is built via a similar mechanism from the object models). Hypothesis verification is then performed on the hypotheses generated from the hash table.

The advantages of such a scheme are many. Multiple object scenes are handled the same as single object scenes. Expensive graph matching is only performed on a small set of volumetric primitives, thus ensuring the extraction of primitives is computationally feasible. Also, the scheme is flexible and can be applied to either intensity or range images in a similar manner. Although the system is general, its testing has not been thorough as yet and hence, the usefulness of the system is still in question. Also, the possible problems inherent at lower levels of processing (image segmentation, contour segmentation) were not addressed in the paper except in the context of synthetic images. So, although the system appears promising, it is difficult to judge how successful it will be until more comprehensive tests have been performed.

### 2.3.15 TINA

Porrill et al ([94]) describe a 3D vision system (TINA) for locating objects in a scene and picking them up using a robot arm. This is achieved by building 3D wireframe object models of the scene. The system described is not an object recognition system per se, although it could be extended in a fairly straightforward manner. The most interesting point about this system is that stereo data is used to build a 3D model from a scene in an incremental manner. A stereo pair of intensity images are input to the system, the Canny ([32, 33]) edge detection algorithm is used to extract edges from the images and the well know PMF ([93]) stereo algorithm is used to obtain depth data. Straight lines, planes and space curves are then extracted from the depth edge data, and a 3D geometric wireframe description of the scene is produced. A match is then performed between the model and the scene according to the following procedure (from [94]):

1. A focus feature is chosen from the model.
2. The $S$ closest salient features are identified.
3. Potential matches for the focus feature are selected.
4. Consistent matches, in terms of a number of pairwise geometrical relationships, are located for each of the neighbouring features.
5. The set of matches is searched for maximally consistent cliques (each of these can be thought of as an implicit transformation).
6. Synonymous cliques (that represent the same implicit transformation) are merged.

7. Merged cliques are ranked on the basis of the number and length of their members.

8. The transformation implicitly described by the clique is recovered.

From the results of the matching process, and other image information, the model is updated from the scene using statistical combination of the model and scene. This can occur across several poses of the object to form a complete object model.

The test images reported in [94] were laboratory scenes of man made objects (with prominent edges and simple surface shading) in good lighting conditions, and the system works very well under these conditions. It would be interesting to see the performance of TINA under less ideal conditions.

Although not a complete object recognition system (in the sense of multiple object databases) in its present form, the TINA system could be successfully applied to two important aspects of the object recognition problem - model acquisition and hypothesis verification. It would be relatively simple to apply some of the other methods reported here for initial model matching in the TINA system (or even the method for matching reported in this thesis 7), and it would be interesting to compare the performance of such a system with the system reported in this thesis.

### 2.3.16 MORSE

Mundy et al [85, 84] describe a complex 3D object recognition system that uses geometric invariances for description and matching. Object models are acquired from multiple 2D intensity images and recognition is performed on single 2D intensity scenes. Models are represented by a number of affine 3D geometric invariants including: cross-ratios of points on rotationally symmetric image features; symmetrical feature correspondences on bilateral symmetries and more general symmetries; projective invariants of algebraic surfaces, etc. The system recognises object classes where each class is described geometrically (surfaces of revolution and polyhedrons are used in [85]).

MORSE uses a fairly standard control system involving segmentation and grouping, indexing and hypothesis generation, and verification. At the grouping stage, pixel level features are grouped into linked edges from which geometric features such as algebraic curves, smooth curves and concavities are extracted. Generic grouping is then performed on the geometric features to create a database of groupings including: near incidence; collinearity; marking bitangent points; affine or projective equivalence of curve segments. Finally, each class has an associated class based grouper which interrogates the generic groupings and attempts to build groupings appropriate for its class. Indexing and hypothesis verification is handled by a series of hash tables, one per class, that associate the invariants of a system of generalised features with models in the model-base. Verification is performed at many levels of the hypothesis formation process, including verification of classes, and lower level groupings (like polyhedral faces).

It is premature to evaluate the usefulness of MORSE, since it is a prototype system which is still under development. However, in it's current form ([85]), it would be more accurate to describe it as a "shape" recognition system rather than an object recognition system. This is because it recognises classes of shapes such as surfaces of revolution and polyhedral objects. In this sense, it is somewhat similar to the system being developed by Dickinson, Pentland and Rosenfeld ([38, 39]). To be considered a true "object" recognition system, MORSE must be able to handle complex objects which include conglomerations of different shape classes. Although this has not yet been demonstrated, presumably it could be achieved by modifying the indexing scheme to allow different shape class hash tables to index into different object models.

MORSE represents an extremely well thought out system for recognising 3D shape classes from single 2D intensity images which attempts to address many of the problems involved in such a domain. It represents a different philosophy for approaching the problem than the one described in this thesis (although not a less valid one), but further experimentation is needed to demonstrate MORSE's true value.

## 2.4 Limitations of Current Approaches

In summary, most object recognition systems presented to date display some limitations and do not fully address all the problems inherent in such a complex domain:

1. Very few systems attempt to solve the learning-from-example problem and those that do are typically demonstrated only in simplistic problem domains.

2. Many systems are demonstrated using only a very small number of objects (ACRONYM, SCERPO), although more recently, PARVO has been demonstrated using 23 objects. However, PARVO assumes that the input is perfect (or near perfect) and that the image has been successfully segmented into separate objects.

3. Very few systems are designed to handle multiple object scenes (ie they expect objects in a scene to be clearly partitioned). Those systems that do not assume object partitioning (ACRONYM, SCERPO) utilise only very few objects and hence the problem of searching for the most likely object model in the model database is greatly reduced.

4. No system, to the author's knowledge, has attempted to address all of the problems inherent in recognising 3D objects from 2D scenes (including part segmentation and multiple object scenes).

5. Very few systems have attempted to use 3D data extracted from 2D scenes for solving the 3D object recognition problem.

Clearly, there is scope for further work, especially in the area of 3D object recognition from 2D images. This dissertation is an attempt to address these issues.

# Chapter 3

# Perceptual Organisation

The first step in the object recognition problem is the successful segmentation of input images (in this case, intensity) into different parts. Hopefully, the segmented parts will correspond to sensible parts of the objects in the scene (eg different surfaces). One of the traditional approaches to solving this problem involves detecting edges in a scene. However, due to the noisiness and variability of the image formation process, low level segmentation procedures such as edge detectors, are unavoidably data sensitive. This means that small changes in the input data causes changes in the output data. These changes are not necessarily large, but are invariably large enough to make consistent description of similar images very difficult. For example, the two images displayed in Figures 3.1 and 3.2, are the original image and the edge image of an laboratory scene. Although the edges shown in Figure 3.2 are clearly representative of the image shown in Figure 3.1, the local errors which are evident in some parts of the edge image make recognition and analysis of the scene particularly difficult for any automatic process.

So how is it that the human visual system can so readily correctly interpret images formed with such unreliable data? One theory which addresses this problem is due to the Gestalt psychologists, and is known as *perceptual organisation*. Perceptual organisation is the process by which humans can *"detect relationships such as collinearity, parallelism, connectivity, and repetitive patterns among image elements"*[1]. It is theorised that such processes help to form more stable low level

[1] Mohan and Nevatia ([82]) and references contained therein

features, which can be grouped into more consistent higher level features such as surfaces and objects. The application context for performing perceptual organisation in this study is for recognition of objects in reasonably complex images. Hence the main aim of perceptual organisation is to produce sensible segmentation of scenes into surfaces and objects.

Figure 3.1: An intensity image.

Perceptual grouping can occur at a number of different levels using many different primitive features. Possible features include: points, lines, texture elements, surfaces, regions etc. For the sake of simplicity, and because other systems (eg. stereo systems) also make use of them, it has been decided to employ edge elements in an image as the basic grouping primitive. Mohan and Nevatia ([82]) describe a computer vision system (CANC2) that utilises perceptual organisation techniques on curves to perform surface and object segmentation.

The following sections contain the description of a perceptual grouping system which has been implemented, based largely on the work performed by Mohan and Nevatia ([82]). This chapter consists of the following sections:

1. The edge extraction and line formation stage.

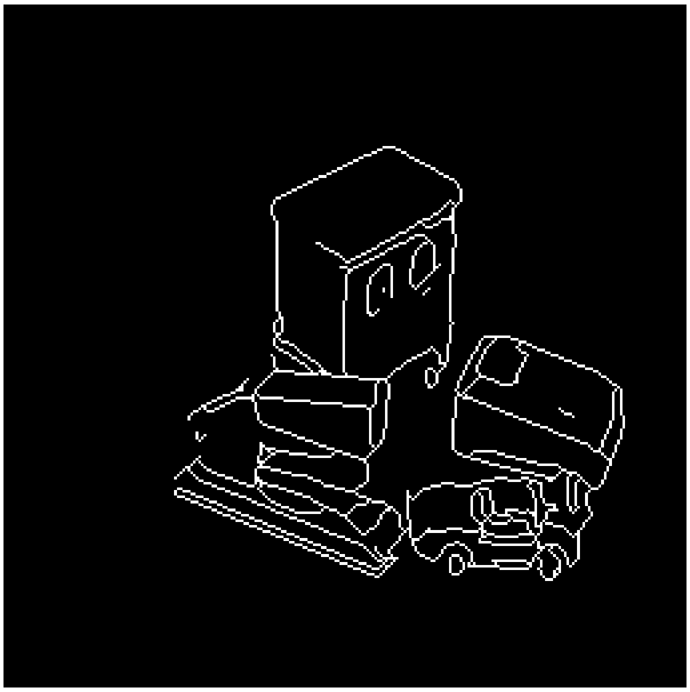


Figure 3.2: Extracted edges.

2. The grouping of lines based on the ideas of proximity and co-curvilinearity.

3. The grouping of lines to form ribbons (symmetric lines and their end closures), based on the idea of symmetry.

4. The grouping of ribbons to form objects.

## 3.1 Edge Extraction

Edges are extracted from the images using an implementation of Canny's edge detector ([32, 33]) with a scale of $\sigma = 1.0$ pixels with respect to a 256 by 256 pixel image. The edge contours are formed as the list of edge pixels between two junctions, where a junction is a meeting point of three or more edges. Any reasonable edge extraction algorithm may be used. It is interesting to note that the edge linking process can be considered a form of low level perceptual grouping joining points into the more abstract feature of edges.

The output from the edge segmentation stage is a list of free-form contours, each represented as a list of pixel coordinates. It is generally more useful to represent these edges in a more meaningful way. For example, depending on the application area, it might be useful to represent the edges as a series of line segments, or arc segments etc. Since most of the techniques which will be described here utilise line segments as the basic grouping primitive, it makes sense to segment the edge contours into straight line segments. This introduces a difficult segmentation problem, for which many solutions have been suggested (Shirai [108], Wuescher and Boyer [120], Haralick and Shapiro [58], Rosin [100], Rosin and West [101] to name a few). Many of these techniques require thresholds which are set to remove the estimated amount of noise present in the edges. With this approach, there is a possibility of setting the threshold too high resulting in under-segmentation, or too low, resulting in over-segmentation. An elegant solution proposed by Lowe ([74]), utilises the concept of perceptual significance to overcome the threshold setting problem.

According to Lowe ([74]), the significance of a straight line fit to a sequence of edge points can be estimated as the ratio of the length of the line segment divided by the maximum deviation of any point from the line. The maximum deviation is set to a minimum of two pixels to account for limitations on measurement accuracy. The segmentation algorithm proceeds in a recursive fashion. Firstly, a line segment fit is made between the two endpoints of the edge contour, and its significance is calculated. The point of maximum deviation is then used to split the edge into two smaller edges. The process is called recursively using each of the smaller edges as input, and continues until each segment is no more than four pixels in length. This results in a binary tree of subdivided edge segments. As the process unwinds back up the tree, the current line segment is replaced by the child subsegments, if the significance of any of the subsegments is greater than the significance of the current edge. Otherwise the current line segment is returned. Figure 3.3 illustrates the concepts of splitting an edge based on this procedure, and the algorithm is given in Algorithm 3.1. In the algorithm, the function *distance* calculates the perpendicular distance between a line segment and a point.

The appeal of this line segmentation procedure is its simplicity and more importantly, its scale independence. The significance measure is approximately constant over all scales. Thus the problem of supplying a threshold parameter is removed

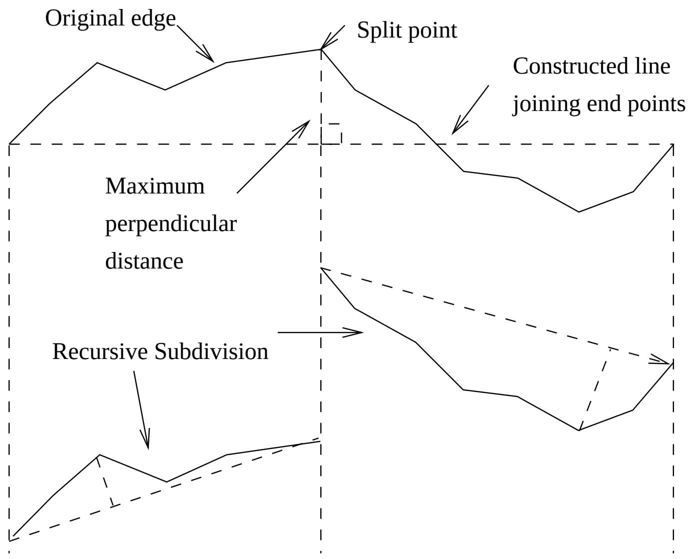



Figure 3.3: Illustration of the edge splitting technique.

whilst also providing scale invariance.

## 3.2 Line Grouping

To improve the output of the edge extraction routine, it is useful to group lines together to form a more reliable representation of the edges evident in a given image area. The process of line grouping can be broken down into a two sub-sections:

- proximate and parallel grouping, and
- continuity and co-curvilinearity grouping.

The input to these processes are the edges which are output from the edge detection phase. Following is a description of the two line based grouping processes.

### 3.2.1 Proximity Grouping

The idea of proximity grouping is to group proximate image artifacts into a single structure to form a more coherent image. The human visual system naturally groups proximate and near parallel curves into a single curve structure. This aids in the segmentation of images as the grouping is often simpler than the pure edges.

**Algorithm 3.1.1** *Edge Splitting*

**function** $split\_edge\_lowe(E, significance)$
**inputs:**
  $E$ /* edge to be split */
  $significance$ /* significance of the edge */
**endinputs**
**variables:**
  $N$ /* number of pixels in the edge */
  $E^i$ /* the $i^{th}$ coordinate in edge $E$ */
  $S$ /* line segment */
  $R_1, R_2$ /* subdivided edges */
  $sig1, sig2$ /* significance of the subdivided edges */
**endvariables**
**begin**
  **if** $(N \leq 4)$ **then**
    $significance \leftarrow N/2$ /* 2 being the minimum allowable deviation */
    **return**$(E)$
  **endif**
  $S \leftarrow$ line segment joining $E^0$ and $E^N$
  **for** $i \leftarrow 0$ **to** $N do$
    **if** $(distance(S, E^i) > distance(S, E^p))$ **then**
      $p \leftarrow i$
    **endif**
  **endfor**
  $significance \leftarrow N/distance(S, E^p)$
  $R_1 \leftarrow split\_edge\_lowe(E^0 \ldots E^{P-1}, sig1)$
  $R_2 \leftarrow split\_edge\_lowe(E^P \ldots E^N, sig2)$
  **if** $(sig1 > significance) \wedge (sig2 > significance)$ **then**
    $significance \leftarrow min(sig1, sig2)$ /* the minimum */
    **return**$(R_1 \cup R_2)$
  **else**
    **return**$(E)$
  **endif**
**end**

Figure 3.4 displays curve grouping based on proximity. Such a grouping scheme can be considered a scale space approach, where the edge on the right of Figure 3.4 represents the same edge as those on the left, but at a larger scale.

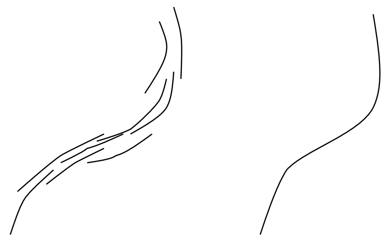


Figure 3.4: Proximity grouping.

This process is currently implemented by considering curves whose orientation of major axes differ by no more than 30°, and whose maximum separation is less than a certain threshold. The curves which obey these criteria are represented by the longest curve in the group. The choice of the threshold is crucial to the success of this procedure. If the threshold is too large, then an unacceptable loss of detail will result. If it is too small, the procedure will have little effect. Unfortunately, it is not a simple problem to determine the correct value for the threshold. Typically, an analysis of the application area and problem parameters (eg camera distance and resolution) would be required to determine an appropriate value for the threshold. Alternatively, an analysis of the edge distribution across the image may be useful in automatically setting the threshold. In the current implementation, the threshold was arbitrarily set to be 4 pixels.

The grouping process is performed by stepping along the shorter edge in reasonable increments (5 elements) and finding the shortest distance to the longer edge at each edge point considered. If this distance exceeds the threshold, then the curves are not proximate and parallel. In other words, all points along the edge must have a separation of no more than the threshold. A side effect of this operation is that small edges are often erroneously considered to be proximate and parallel. However, this is a minor problem as smaller edges typically do not contain a significant amount of information other than connectivity information (are two edges or points connected?). Many of the gaps produced by incorrectly removing small edges will be bridged by the method described in the following section. The result of grouping Figure 3.2 based on proximity and parallelism is shown in Figure 3.5.

As can be seen from Figure 3.5, some of the detail in the image is lost. A number of small edges were erroneously removed and some joined edges are no longer connected. A specific problem of the procedure is demonstrated at the top front edge of the wooden train in Figure 3.5. Here, two parallel edges are close to each other with some confusion towards the right end of the edges causing both edges to be segmented into two separate edges. The proximity grouping operation correctly recognised these edges as proximate and parallel, but decided to keep one edge segment from each of the different complete edges, resulting in two misaligned edges representing what should have been a single edge. This is a problem which is not guaranteed to be solved by the co-curvilinearity grouping. Hence, a more sophisticated routine is required to fully solve this problem. Nevertheless, the routine has performed tolerably well, resulting in a reduction of image complexity and leaving only the more significant edges in the image, which was the aim of the operation. To recover the erroneous loss of the smaller edges, and to bridge the gaps caused by segmentation errors, the image is augmented by grouping the remaining curves on continuity. This is discussed in the next section.

### 3.2.2 Continuity and Co-curvilinearity Grouping

The process of grouping based on continuity and co-curvilinearity can be described as the joining of edges which lie along lines of similar curvature. This idea is demonstrated more clearly in Figure 3.6. Although only short linear edge segments are evident in Figure 3.6, it is obvious that a number of longer curves appear in the figure. Apparently, the human visual system extends the short segments to join naturally with other segments which lie along lines of similar curvature.

If the human visual system performs grouping based on continuity and co-curvilinearity, it would appear reasonable to assume that such an approach may aid automatic object recognition. A description of the method used by Mohan and Nevatia ([82]) to implement continuity grouping (and which was used in this study) follows. Joins from each curve end to other curve ends within a specific range (10%

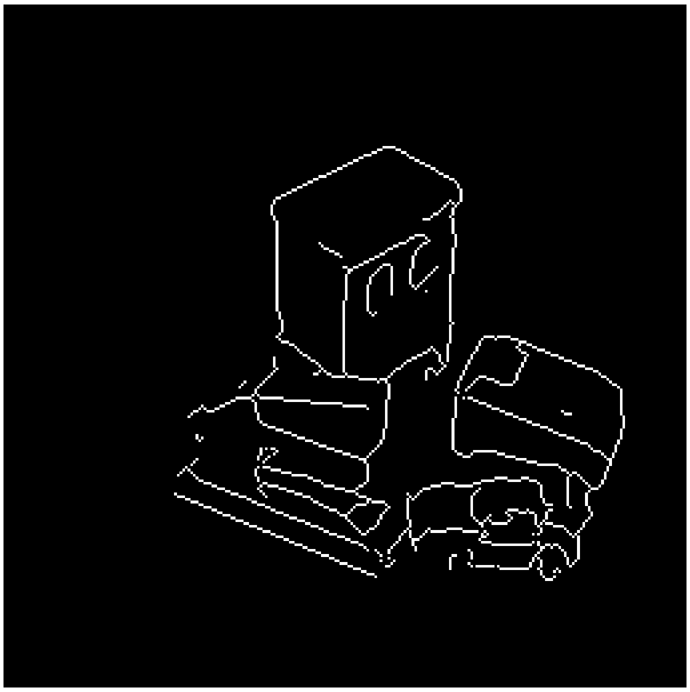


Figure 3.5: Proximate and parallel edges.

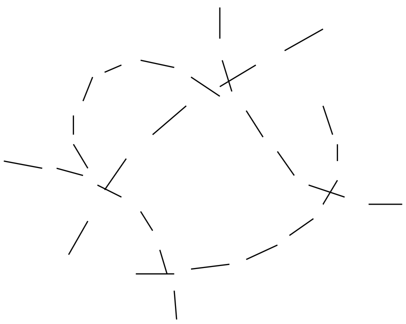


Figure 3.6: Curvature grouping.

of the image size) and whose tangents at the considered end, do not differ by more than 30°, are considered for possible joinings. For each of these joins a measure of bending is calculated according to the following formula:

$$B = (\alpha^2 + \beta^2)(a + bl) \tag{3.1}$$

where $\alpha$ and $\beta$ are the angles made between the tangents at the curve ends and a line joining the ends (displayed in Figure 3.7), $a$ and $b$ are constants, and $l$ is the separation of the two curves. The constant $a$ controls the departure from collinearity of the joined curves, and $b$ controls the sensitivity to the lengths of the gaps. Mohan and Nevatia ([82]) use $a = 1.0$ and $b = 0.1$.

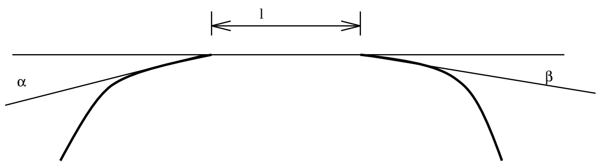



Figure 3.7: Calculation of bending.

The process for joining lines is given in Algorithm 3.2.2. The above algorithm is an iterative process. The joins with the least bending factor are joined in the first iteration. Subsequent iterations cause joins with higher bending factors to be considered. In this way, more global information is brought to bear as the algorithm proceeds. The finishing criteria of the algorithm can depend on a number of factors. For example, the algorithm can continue until a maximum bending factor is reached, or until a maximum number of joins have been performed. Currently, however, the number of iterations is arbitrarily fixed at 5.

For the class of images used here, it was found that the effects of the co-curvilinearity procedure are not significant. The main application for such a technique is to recover from occlusion effects, where the object being occluded is visible on both sides of the occluding object (see Figure 3.8). In the images studied here, this form of occlusion rarely occurred. With this in mind, and the fact that the co-curvilinearity operation is expensive ($O(n^2)$, where $n$ is the number of edges), it was decided to remove this operation from the overall system.

**Algorithm 3.2.1** *Join Edges based on Co-curvilinearity*

**function** $join\_edges(E)$
**inputs:**
    $E$ /* a list of edges */
**endinputs**
**variables:**
    $J$ /* a list of joins joining edges in E */
    $j$ /* a single join */
    $j_{e1}, j_{e2}$ /* the edges associated with join $j$ */
    $E1, E2$ /* lists of edges */
**endvariables**
**begin**
  $E_1 \leftarrow E$
  **while** $\neg$(finished) **do**
    $E_2 \leftarrow \phi$/* empty set */
    $J \leftarrow$ all possible joinings of edges in $E_1$
    calculate $B$ for all possible joinings, $J_1$
    calculate the minimum bending factor, $B_m$
    **foreach** $j \in J$ **do**
      **if** $(B_i <= 3 \times B_m)$ **then**
        $e \leftarrow j_{e1} \cup j_{e2}$ /* join the two edges together */
        $E_2 \leftarrow E_2 \cup e$
      **endif**
    **endforeach**
    **foreach** $(e \in E_1)$ **do**
      **if** $\neg(joined(e))$ **then** /* if $e$ has not just been joined */
        $E_2 \leftarrow E_2 \cup e$
      **endif**
    **endforeach**
    $E_1 \leftarrow E_2$
  **endwhile**
  **return**$(E_1)$
**end**

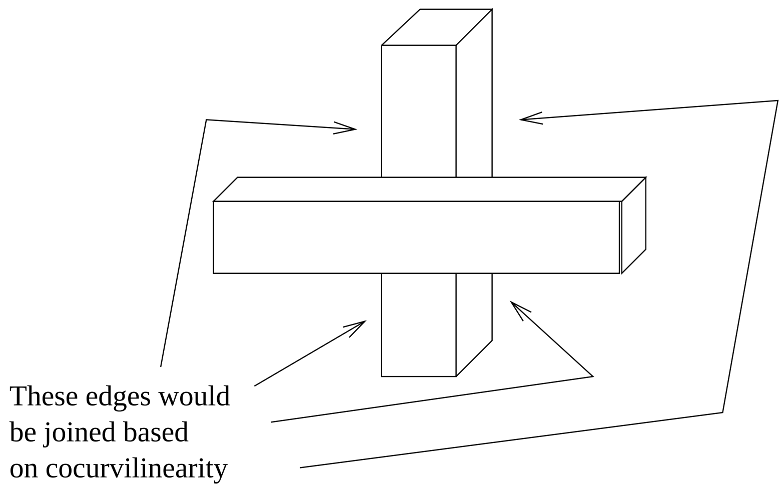



Figure 3.8: Occluding objects.

## 3.3 Ribbon Formation

The goal of the processes described in the previous sections, is to produce more reliable edges by reducing noise and correcting for minor errors in the edge formation process, without sacrificing any useful information. Object recognition typically requires features at a higher level of description than edges (eg surfaces). Mohan and Nevatia ([82]) postulate that curves which bound the same surface or object are likely to exhibit higher measures of symmetry than curves belonging to different objects. Such a belief seems reasonable, especially when man-made objects are considered. Hence, if symmetries can be reliably detected in an image, then the process of image segmentation - or more specifically, surface and object segmentation, becomes simplified. Importantly, many symmetries are also invariant to a large class of 3D linear transformations (Mohan and Nevatia ([82])). Hence, symmetrical lines in a 3D scene will appear symmetrical (or near symmetrical) in a 2D image of that scene.

The ribbon formation process can be defined as the process of extracting symmetric lines and their end closures from an edge image. This process involves techniques such as: symmetry extraction; forming end closures for the symmetries; and finally choosing the correct ribbons. The following sections describes each of these processes.

### 3.3.1 Evaluating Symmetries

Symmetry can be defined as (Mohan and Nevatia ([82])):

> *"A one to one mapping (if the curves are considered infinitesimally divisible) between the points of two curves. The symmetry axis is defined as the locus of the midpoint of the straight lines (or, for generality, some curve) joining a point on one curve to its image in the other."*

Mohan and Nevatia ([82]) developed a quantitative measure of the symmetrical relationship between two curves, and utilised this for selecting symmetries from an image. This measure makes use of the following quantities:

- The aspect ratio - defined as as the ratio of the length of the proposed axis to the distance between the two curves.
- The normalised length difference between the two axes - defined as the difference between the lengths of the two curves normalised by the length of the longer curve.
- Amount of skew between the two curves - defined as the deviation from normal of the angle between the proposed end-curves and the symmetric curves.
- The amount of parallelism between the two curves - defined as the angular difference between the two curves.
- The amount of parallelism between the endpoints of the two curves - defined as the angular difference between the two proposed end-curves.
- The length of the proposed symmetry axis.

The measure of goodness given by Mohan and Nevatia ([82]) for a symmetry axis is given in Equation 3.2:

$$G = 8.0a - 15.0c - 4.0n - 3.0e - 5.0s + 2.0l \tag{3.2}$$

where $G$ is the goodness function, $a$ is the aspect ratio, $c$ is the angular difference between the curves, $n$ is the normalised length difference between the curves, $e$ is the angular difference between lines joining the ends of the curve, $s$ is the skew angle between the curve and curve end, and $l$ is the length of the axis in pixels ([87]).

The weights on the different measures were defined by Mohan and Nevatia ([82]) according to subjective importance of each measure, in order to normalise the units of each measure and give a factor of relative importance. Using such subjective measures for calculating the relative importance of the possible symmetries is somewhat ad hoc. However, it is not clear how to obtain a more rigorous measure of importance. It may be possible to modify the classical descriptions of symmetry (such as the symmetric axis transform of Blum ([20]), the smoothed local symmetries of Brady ([23, 24]), and the generalised cylinders of Binford ([16])) to produce a measure of how symmetric two lines are, but these methods are typically computationally expensive, and often noise sensitive (Mohan and Nevatia ([82])). Zabrodsky, Peleg and Avnir ([124]) have defined a continuous symmetry measure for rotational and mirror symmetries which encode the distance a certain shape is from being symmetric. Their method is general and would be an interesting area for further research. However, their technique is not considered here.

It is important to note here that evaluating symmetries is only a part of the process of symmetry extraction. It has been postulated that curves bounding the same surface typically exhibit higher levels of symmetry than unrelated curves. This is not always the case, and hence other information must be used to determine if a given symmetry is representative of a surface in the scene. For this reason, it can be argued that the evaluation of symmetries is not critical. Provided that the symmetry evaluation results in a reasonable measure for a majority of cases, then this is probably sufficient for the needs of extracting salient features from an image. Efficiency considerations must also be taken into account and it is felt that the above measure of symmetry, whilst not perfect, is both easy and efficient to compute and is sufficient for our needs.

The coefficients used by Mohan and Nevatia ([82]), would appear to require some analysis, however. An analysis of the typical values for the possible symmetries across a number of images gave the following results:

$$a = 1.6, c = 0.16, e = 0.48, s = 0.37, n = 0.35, l = 36.6 \tag{3.3}$$

Multiplying the above averages with the coefficients given in Equation 3.2, gives the following *normalised* results:

$$a = 12.8, c = 2.4, e = 1.44, s = 1.11, n = 1.4, l = 73.2 \tag{3.4}$$

With these coefficients, the importance of the length of the proposed axis outweighs its nearest rival by a factor of 6, which may be useful for some images, but is unlikely to be widely applicable. In any case, the use of any length measure is likely to be problematic - the size of any object in a scene is inherently dependent on the parameters of the image formation process. Such parameters include; distance between the camera and the object, focal length of the camera, size of the image etc. An attempt can be made to normalise the length, but problems still remain. If the length is normalised based on the image size, then the larger the image, the less important the length measure. If the length is normalised with respect to the contents of the image, then the importance of the length measure is dependent on the contents of the scene. Indeed, the use of length in the goodness function appears to be the only measure which is not largely invariant. Hence, it should play only a minor role (if at all) in the evaluation of symmetries.

Through a process of trial and error, and taking into account the most reliable measures (and giving them more weighting), the following coefficients were decided upon:

$$G = 3.0a - 10.0c - 4.0n - 1.0e - 1.0s + 0.01l \tag{3.5}$$

The resulting normalised values are:

$$a = 4.8, c = 1.6, e = 0.48, s = 0.37, n = 1.4, l = 0.37 \tag{3.6}$$

The aspect ratio ($a$) is considered the most important measure, followed by the amount of curve parallelism ($c$) and the normalised length difference. The skew ($s$) and end-curve parallelism ($e$) are given less weighting as in some cases curves are segmented incorrectly resulting in shorter curves than in reality, thus adversely affecting the skew and end-curve parallelism measures. The length measure is considered less important due to the reasons given above.

One of the problems of equation 3.5 is that it relies solely on edge information. Other important information sources such as texture and intensity measures are ignored when evaluating the symmetries. If accidental symmetries occur between unrelated objects in an image, then it is likely that an erroneous symmetry will be extracted. Hence it may be useful to include other information sources into the goodness function. Intensity variation has been used in this implementation for reasons which will be discussed, but in a more general system, texture variation would be more applicable.

A property of the objects used in this experiment is that a single surface has little intensity variation across it. Although this is not true in general, it is true in some special cases (automated machine part recognition for example), and was deemed a necessary assumption in the current system. Hence, the smaller the intensity variation across a ribbon, the more likely it is to correspond to, or be part of, a surface. With this in mind, it would be useful to include a term in the goodness function which varies inversely with the intensity variance across a ribbon. The modified goodness function has been defined as follows:

$$G' = \frac{G}{1.0 + G} + \frac{K}{1.0 + \sigma}. \tag{3.7}$$

Where $K$ is a constant used to *normalise* the variance measure, and $\sigma$ is the standard deviation of the intensity values within the proposed ribbon. The boundaries of the ribbon are determined by the curves involved in the symmetry axis, and straight line end-curves are proposed to form a closed ribbon.

The first term in equation 3.7 is a non-linear response function with the old goodness function as input. This has the effect of transforming the previously unbounded goodness function to lie between 0 and 1, whilst maintaining the ordering of points (for $0 < G < \infty$). Similarly, the second term in equation 3.7 is used to map the standard deviation of the intensity values to lie between 0 and 1, where a low $\sigma$ results in a response close to 1, while a high $\sigma$ results in little response. In practice, across ribbons of any significant size, $\sigma$ is rarely less than 3, resulting in a maximum response of the second term of 0.25. Hence, the normalisation factor, $K$, was introduced to make the second term comparable to the first in magnitude. In the current implementation, $K$ was set to equal 3.

Figure 3.9 compares the results of ribbon formation with and without the intensity measure. Part (a) displays the original image, part (b) displays the extracted edges, part (c) displays the ribbons formed without the intensity measure, and part (d) displays the ribbons formed with the intensity measure. The improvement in ribbon formation is shown clearly between parts (c) and (d) where the top of the cup forms an incorrect ribbon with an edge of the book and obscures the ball. This does not occur in part (d). Other problems with the segmentation include the Rubik's cube area which is confusing to the procedure due to the relatively large number of possibly conflicting ribbons. Further error are also caused by incorrect line segmentation which is particularly noticeable in the area around the title of the book.

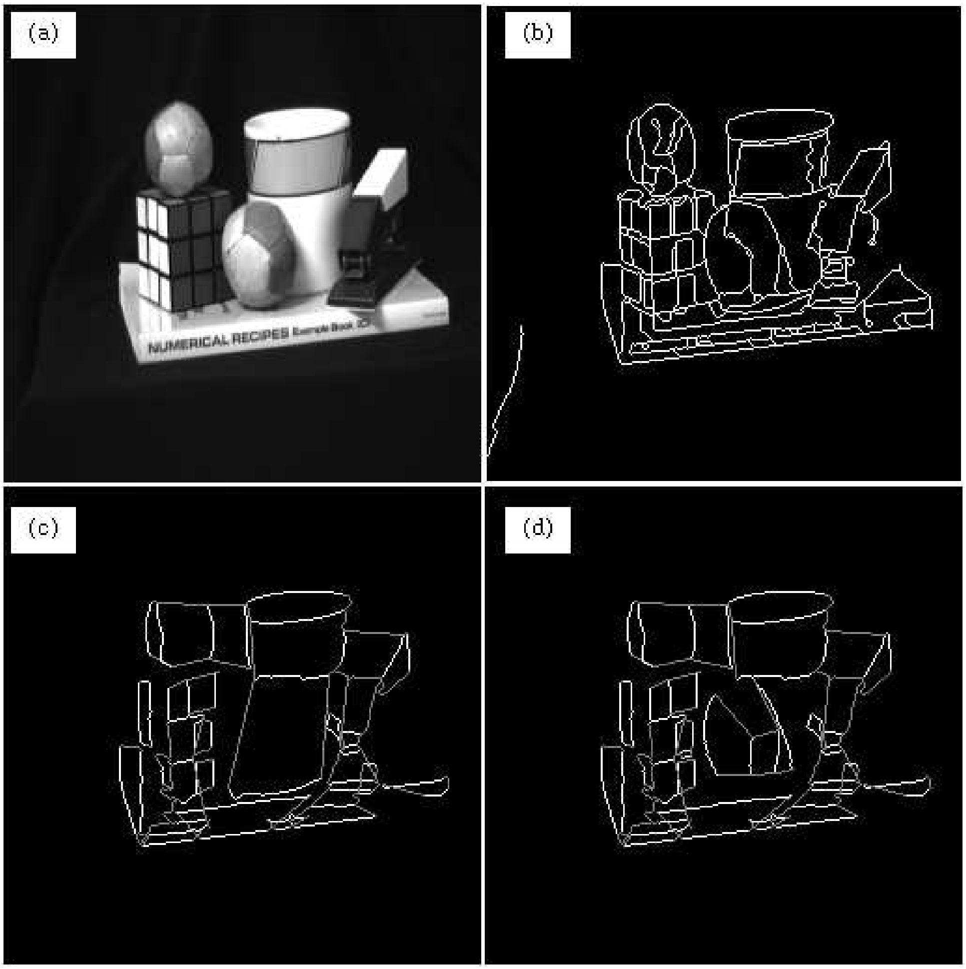



Figure 3.9: Ribbon formation using intensity.

### 3.3.2 Implementation

The implementation of the ribbon formation process consists of the following processes:

1. The symmetry extraction process.

2. Conflict resolution.

3. Finding end closures.

4. Choosing the correct ribbons.

5. Surface and object segmentation.

These processes are discussed in the following subsections.

#### 3.3.2.1 Symmetry Extraction

Symmetries are extracted from an image by first considering all possible pairings of edges in the image. The large number of postulated symmetries is reduced by considering an edge pairing as a possible symmetry only if it obeys the following conditions:

1. The shorter of the two curves should be no less than one third of the length of the longer curve.

2. The curves should overlap in some manner.

Mohan and Nevatia ([82]) defined overlap by proposing a linear symmetry axis based on the correspondence between the end points of the curves being considered. If both curves after orthographic projection onto the proposed axis, overlap with the axis by at least one third, then the symmetry is considered as possible, otherwise it is rejected. However, this is not really a valid selection criterion, as most pairings will be accepted. Consider the image in Figure 3.10. It is unreasonable to consider they might form a symmetry (at least in the context of the symmetry evaluation function), yet by the definition of overlap given by Mohan and Nevatia ([82]), the edges will at least be considered. This is clearly not a valid selection criterion as it is quite difficult for any pair of edges to fail.

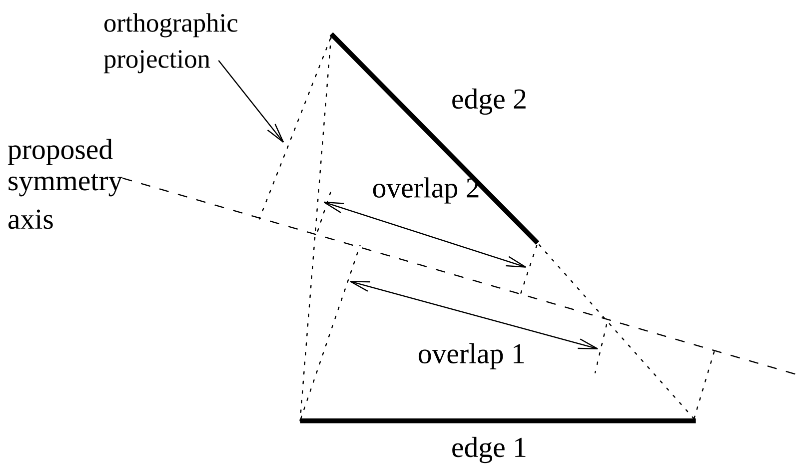



Figure 3.10: Invalid overlap.

The criterion has been modified to allowing a symmetry if each curve overlaps the other by more than one third, when orthographic projection is used. This criterion is displayed in Figure 3.11. This criterion is somewhat more strict than that used by Mohan and Nevatia([82]), and hence should reduce the symmetry search space by a considerable margin. The proposed overlap is also displayed using the same example displayed in Figure 3.10. Figure 3.11 displays the resulting failure of the new overlap criterion. Figure 3.12 displays all the possible symmetries extracted from Figure 3.5 using the new overlap criterion.

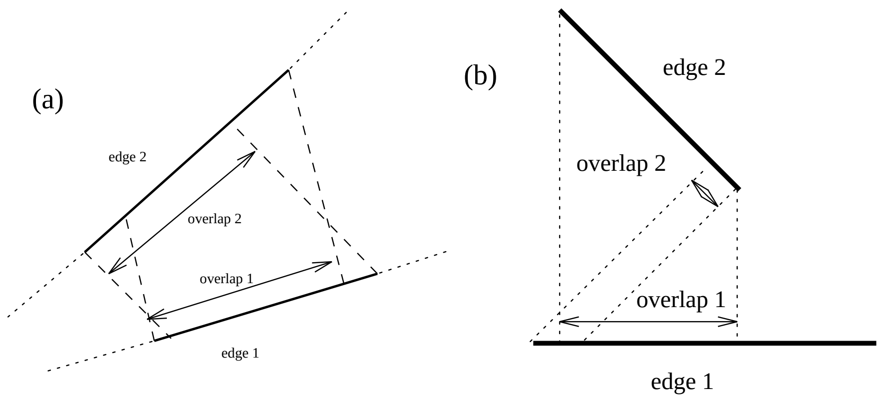



Figure 3.11: Better overlap.

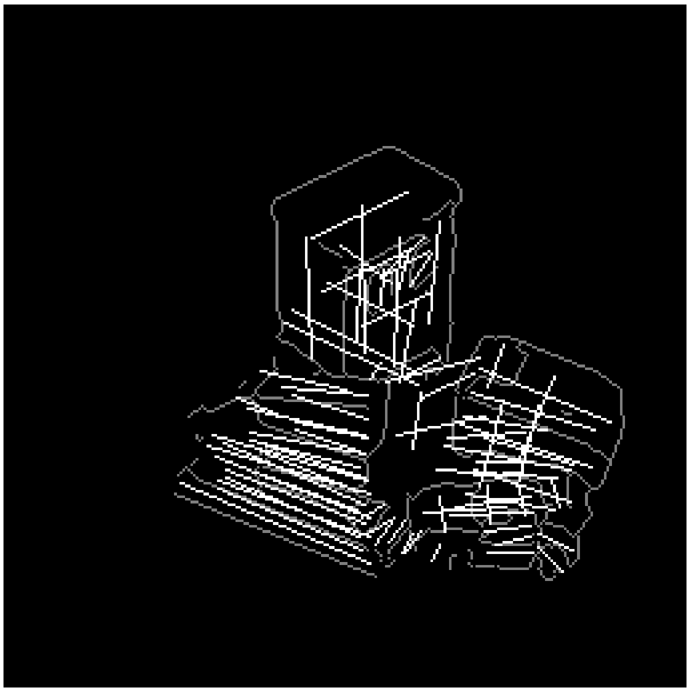


Figure 3.12: All possible symmetries.

#### 3.3.2.2 Conflict Resolution

At this stage, we have some measure of the relative goodness of each proposed symmetry axis in a local or isolated sense. Each axis has been treated independently to calculate its measure of goodness. A percentage of the best axes could be selected and used to form ribbons. However, conflicts between axes can arise, and these need to be dealt with. In most situations, an edge forms a boundary to at most two surfaces, each surface on opposite sides of the bounding edge. Conflicts between axes arise when an edge is hypothesised as belonging to two surfaces, both on the same side of the edge. An example of such a situation is given in Figure 3.13. Here, two axes have been proposed, $ab$ and $ac$, from the three curves $a$, $b$, and $c$. The axes $ab$ and $ac$ are in conflict as they both lie on the same side of the line $a$.

Many methods of solution exist for this type of constraint satisfaction problem including: simulated annealing, relaxation labelling, Boltzman machines and Hopfield Nets. Mohan and Nevatia ([82]) have used a Hopfield net like that introduced by Hopfield and Tank ([60]). A Hopfield net has also been used in this implementation.

The problem of selecting the best set of axes can be formulated as selecting the

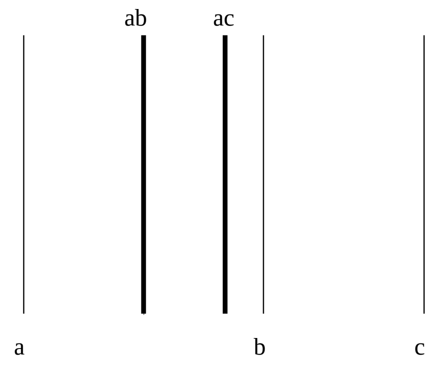



Figure 3.13: Axes in conflict.

best set of hypotheses, given support and conflict relations between them. In a similar manner to Mohan and Nevatia ([82]), the problem is formulated as follows:

- A set of hypothesis $h_i \in H$.
- An input function $I(h_i)$.
- A support function $S(h_i, h_j) > 0$.
- A conflict function $C(h_i, h_j) < 0$.

The best set of hypotheses can then be found by maximising the function:

$$\sum S \ + \ \sum C \ + \ \sum I \tag{3.8}$$

To achieve this, we can consider each hypothesis (axis) as a node of a constraint satisfaction network, with the arcs between each node representing the functions $S$ and $C$. The network is fully connected, with the weights on the arcs being positive for supporting hypotheses and negative for conflicting ones. The dynamic behaviour of the network can be described as follows:

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} T_{ij} V_i + I_i - b_i \tag{3.9}$$

$$V_i = g(u_i) = 0.5(1 + \tanh(u_i)). \tag{3.10}$$

Where $N$ is the number of nodes, $T_{ij}$ is the weight on the link from node $i$ to node $j$, $I_i$ is the input to node $i$, $V_i$ is the output from node $i$, $u_i$ is the membrane potential of node $i$, and $b_i$ is the resting potential of node $i$.

The decision on which nodes satisfy the constraints is made by applying equations 3.9 and 3.10 to the network until the network converges. If the weights are symmetric, (ie $T_{ij} = T_{ji}$), then the network is guaranteed to converge to a stable state (Hummel and Zucker [63]). A stable state is reached if:

$$\|V_{i_t} - V_{i_{t-1}}\| < \alpha, \forall i. \tag{3.11}$$

That is, the difference between the outputs of the last iteration and the outputs of this iteration are minimal. The nodes which satisfy the constraints are those with the highest outputs.

The Hopfield network is used to resolve the conflicts which arise between possible symmetric relationships. For each proposed symmetry axis, there exists a corresponding node in the Hopfield network. The values computed from Equation 3.7 are passed as inputs to the nodes. The weighted connections between the nodes are computed depending on support or conflict (or neither) between the two symmetry axes.

If an edge is involved in two symmetry axes, and the axes lie on opposite sides of the common edge, then the axes are said to support each other. The weight between the corresponding nodes in the Hopfield network is set to 0.2. Conflict between two axes only occurs when the two axes lie on the same side of the common edge. The weight of the arc between the two conflicting symmetries in this case is set to -0.2.

After all possible symmetries have been found, the Hopfield network iterates until it converges. In all the experiments performed so far, this was not more than 10 iterations. All symmetries with an output of greater than 0.7 were selected as possible symmetries. Those symmetries which had end closings at one end were also selected. This last condition was used to select possible ribbons that are not particularly symmetric, yet have a high probability of corresponding to a surface due to the closure property (edges are connected).

Figure 3.12 displays all the possible symmetries given as input to the Hopfield network. There are 81 symmetries in total. After 4 iterations, the network converged and the symmetries which were selected are displayed in Figure 3.14. The number of symmetries in this figure is 37, thus reducing the possible number of symmetries by more than half. Of these symmetries, 18 of them were included due to the end closure property.

Figure 3.14: Proposed symmetries.

#### 3.3.2.3 Finding End Closures for the Ribbons

After selecting all the viable symmetries as those with high outputs from the Hopfield network, it is necessary to compute possible end closures to form ribbons. Hopefully, these ribbons will constitute surfaces or sensible portions of those surfaces (eg if a shadow lies across a surface, then two ribbons will most likely be formed).

To find possible end closures, the edge image is considered to be a graph structure where the edges are the arcs of the graph, and the junctions between the edges are the nodes of the graph. Since we wish to find end closures, the problem is reduced to finding a path between two nodes - the end points of the edges involved in the symmetry axis. To find the shortest path between two nodes, a breadth-first searching strategy can be employed.

There are a number of problems which need to be addressed at this stage:

- What if no path exists between two nodes?

- What if the path which is found is not a sensible one?

The first problem can be solved by postulating a straight edge between the two endpoints. The second problem, however, is not so simple. Mohan and Nevatia ([82]) solve this problem by grouping the curves on co-curvilinearity between the two endpoints. That is, if the curves between the two endpoints are found to be co-curvilinear, then an end-curve is proposed involving those edges. If the edges are not co-curvilinear, then a straight line end-curve is proposed.

A problem with this technique arises when one of the edges in the symmetry axis is erroneously segmented into a number of smaller edges such as is the case in Figure 3.15. The symmetry axis is proposed between the edges $s1$ and $s2$. A path is found between the right hand endpoints of the two symmetry edges, namely the edges $e1, e2$. If a test for co-curvilinearity was made on the edges $e1$ and $e2$, then it would obviously fail. Yet the region presented appears to be a valid ribbon. To combat this problem, in this implementation, an end-curve is considered valid as long as the ribbon remains approximately convex.

Convexity is defined as follows (Preparata and Shamos [97]):

> A domain $D$ in $E^d$ is *convex* if, for any two points $q_1$ and $q_2$ in $D$, the segment, $\overline{q_1 q_2}$, is entirely contained in $D$.

This definition can be efficiently applied to ribbons by testing for inclusion of lines between the vertices of the ribbons only. The use of the convexity property has the effect of accepting ribbons which have a high probability of corresponding to surfaces, but would not normally have been accepted due to erroneous segmentation of the ribbons boundary, or because the boundary is irregularly shaped.

Figure 3.16 displays all the end closings computed from the set of possible symmetries. As can be seen from the figure, many of these end closings are incorrect. Hence, we need to determine which ribbons are correct, and which are erroneous. The process for achieving this is discussed in the next section.

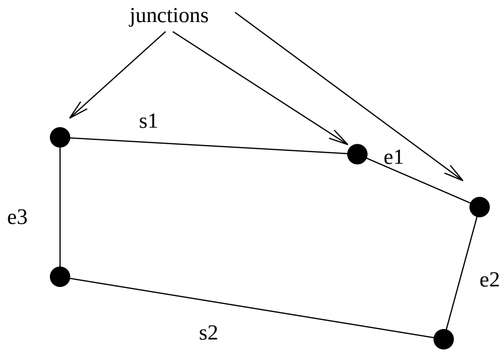



Figure 3.15: Convexity of a ribbon.

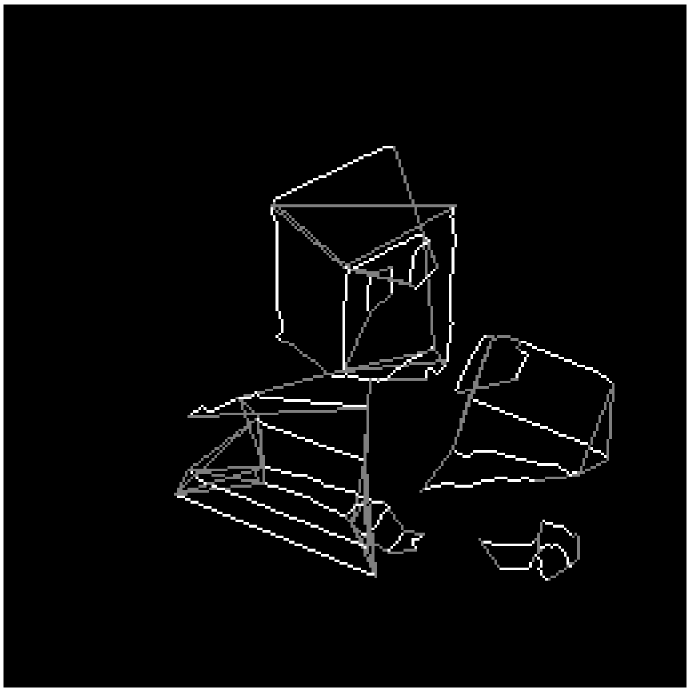


Figure 3.16: End closures: endcurves are less bright than the symmetry edges.

#### 3.3.2.4 Choosing the Correct Ribbons

At this stage of processing, we have a number of postulated ribbons that may or may not be reasonable. A decision still has to be made whether or not a given ribbon should be accepted or rejected. The heuristic used here is quite simple. If the proposed ribbon has both end-curves involved in the original image (ie not postulated as a straight line), then the ribbon is accepted as being correct. If the ribbon has one or two postulated end-curves, the ribbon is only accepted if the postulated end-curve does not significantly cross any other edge in the image. The reason for rejecting ribbons if they significantly cross another edge and not simply if they cross another edge is due to possible errors in the segmentation process. Often, small gaps between edges appear, and even the techniques described above (co-curvilinearity etc) do not always bridge these gaps. Hence, there is a possibility that a viable end-curve will not be found even though it does actually exist, and a straight line end-curve will be proposed in its place. It is postulated that if this is the case then the proposed end-curve will probably cross the correct end-curve but should not be too different from it. If a crossing does occur, then it should only occur at the ends of an edge, or if the angular difference between the two edges is minimal. In this implementation, the factor of significant crossedness is defined as an end-curve differing in angular difference from the crossed edge by more than $15°$, or by crossing an edge greater than the ratio of 0.1 from the ends of the crossed edge. Figure 3.17 displays the ribbons which were found to be correct.

#### 3.3.2.5 Surface Segmentation

Although it is not obvious in Figure 3.17, a number of ribbons are overlapping. Overlapping ribbons can occur for a number of reasons:

- A surface marking may cause one ribbon to be enclosed in another.
- Segmentation errors can cause a number of ribbons to be postulated which are only slightly different, and are more appropriately represented as a single ribbon.

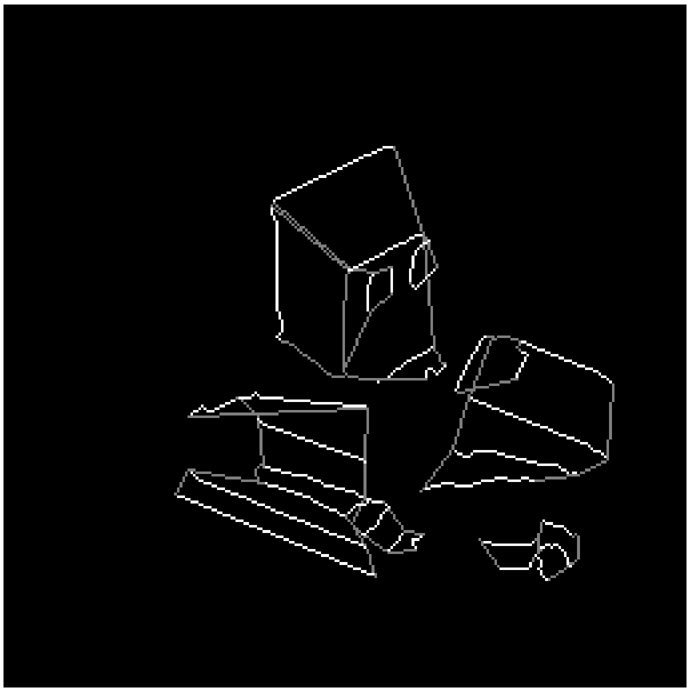


Figure 3.17: Correct ribbons: end curves are less bright than the symmetry axes.

- Due to the nature of the ribbon formation process, a ribbon may be extracted which completely overlaps a number of smaller ribbons.

The application will determine how best these ribbons should be dealt with. Two related applications which will be discussed here are object segmentation and object recognition. Although related, successful implementation of these applications is best served by different segmentation procedures. For object segmentation, interior surface markings are of little importance compared to the overall surface ribbon. In contrast, interior surface markings can be extremely important for object recognition. For object segmentation, any ribbon which shares greater than 90% of its area with a larger ribbon is rejected.

The basic algorithm for choosing the best ribbons follows:

**Algorithm 3.3.1** *Choosing the Best Ribbons*

**inputs:**
$\mathcal{S}$/* the set of all ribbons in ascending order of ribbon score. */
**endinputs**
**begin**
**foreach** $(S_i \in S)$ **do**
**if** $\left(\mathit{overlap}\left(\mathcal{S}_i, \bigcup_{j=1}^{i-1} \mathcal{S}_j\right) > 0.9\right)$ **then**
$S \leftarrow S - \{S_i\}$
**fi**
**endforeach**
**end**

Where the function *overlap* is defined as the geometric overlap of the two regions passed as input parameters. The result of the algorithm is shown in Figure 3.18.

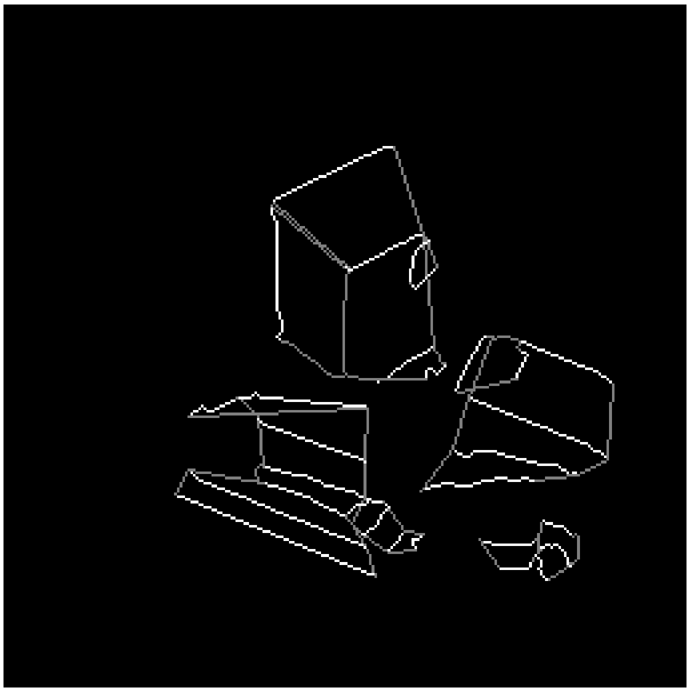


Figure 3.18: Surfaces: end curves are less bright than the symmetry axes.

#### 3.3.2.6 Object Segmentation

The last remaining task to perform is that of object segmentation. Assuming there is a low probability of accidental alignments (which is often a false assumption, especially in complex scenes), then it is reasonable to assume that the relationship between surfaces belonging to the same object is likely to be more regular than the relationship between surfaces on differing objects. Mohan and Nevatia ([82]) base their object segmentation procedure on the following two premises:

- The two ribbons share a complete side. A complete side of a ribbon is either one of the symmetric curves or one of the contours closing the gaps.
- One is a component of the other.

The second condition has been addressed with the surface segmentation procedure described above which removes ribbons which are enclosed by other ribbons and hence does not need to be explicitly dealt with here. The first condition is implemented basically as stated. In more detail this translates to two ribbons belonging to the same object if one of the following conditions are true:

1. The ribbons share a common edge.
2. An edge of one ribbon is the same as one of the edges involved in the end-curve of the other ribbon.
3. The complete end-curve of one ribbon corresponds to the complete end-curve of the other.

The second condition noted above also has an extra condition associated with it. If the junctions between the end-curve and the symmetry edge is a T-junction, then it is assumed that an occlusion has occurred and the objects are separate. The resulting segmented objects are displayed in Figure 3.19.

## 3.4 Results and Discussion

The results of performing all the above procedures on a number of images are displayed in Figures 3.20 to 3.23. As can be seen from these images and the images displayed in previous sections, the techniques appear to perform tolerably well in a large percentage of cases. However, in many cases the results are confusing or incomplete. Generally, this last case occurs where little or no symmetry is evident

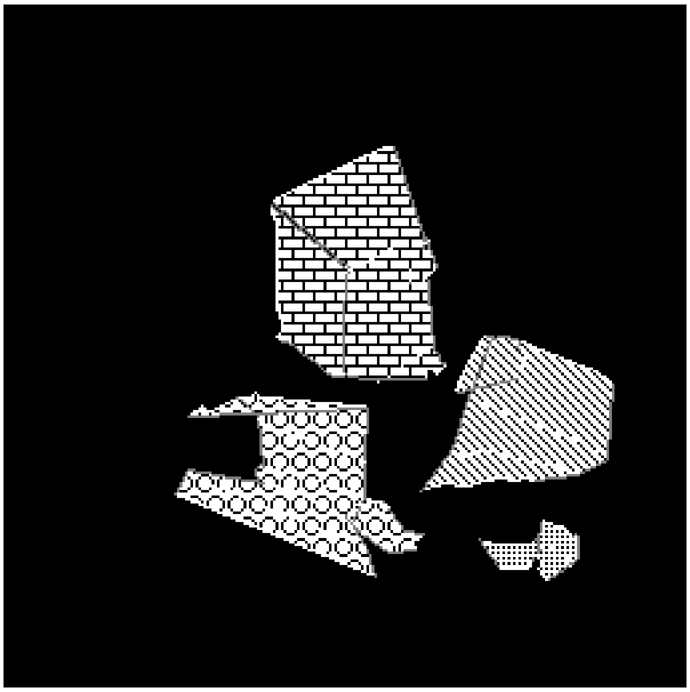


Figure 3.19: The segmented objects.

(eg. the teapot), yet important information is still contained in those image regions.

The major problem with the current perceptual organisation techniques are that they are limited to specific types of symmetries, thus limiting their application to relatively few objects and situations. As well as this, the techniques are ad hoc and suffer from a lack of robustness, often not being able to handle relatively small changes in the input, and are computationally expensive. Nevertheless, the techniques presented do have some merit, and may be useful for some applications. However, significant problems need to be addressed before such techniques become useful for anything other than simplistic or highly specialised scenes:

- a more general definition of symmetry,
- a more general and less ad hoc form of the goodness function,
- a more reliable and robust symmetry extraction algorithm,
- a method for attaching likelihoods to the extracted ribbons.

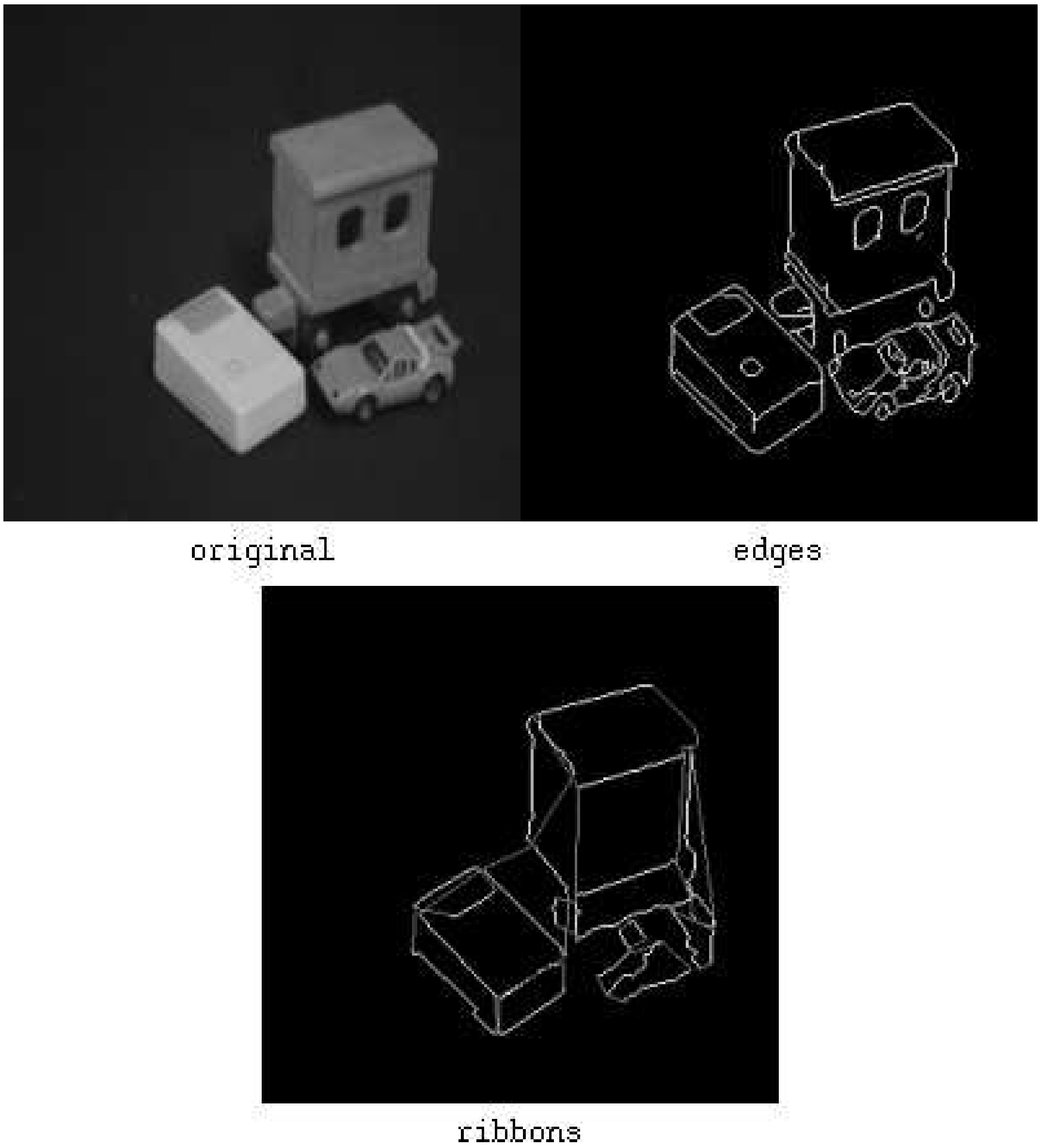



Figure 3.20: An examples of the perceptual organisation process. (a) The original image, (b) the extracted edges, and (c) the resultant ribbons

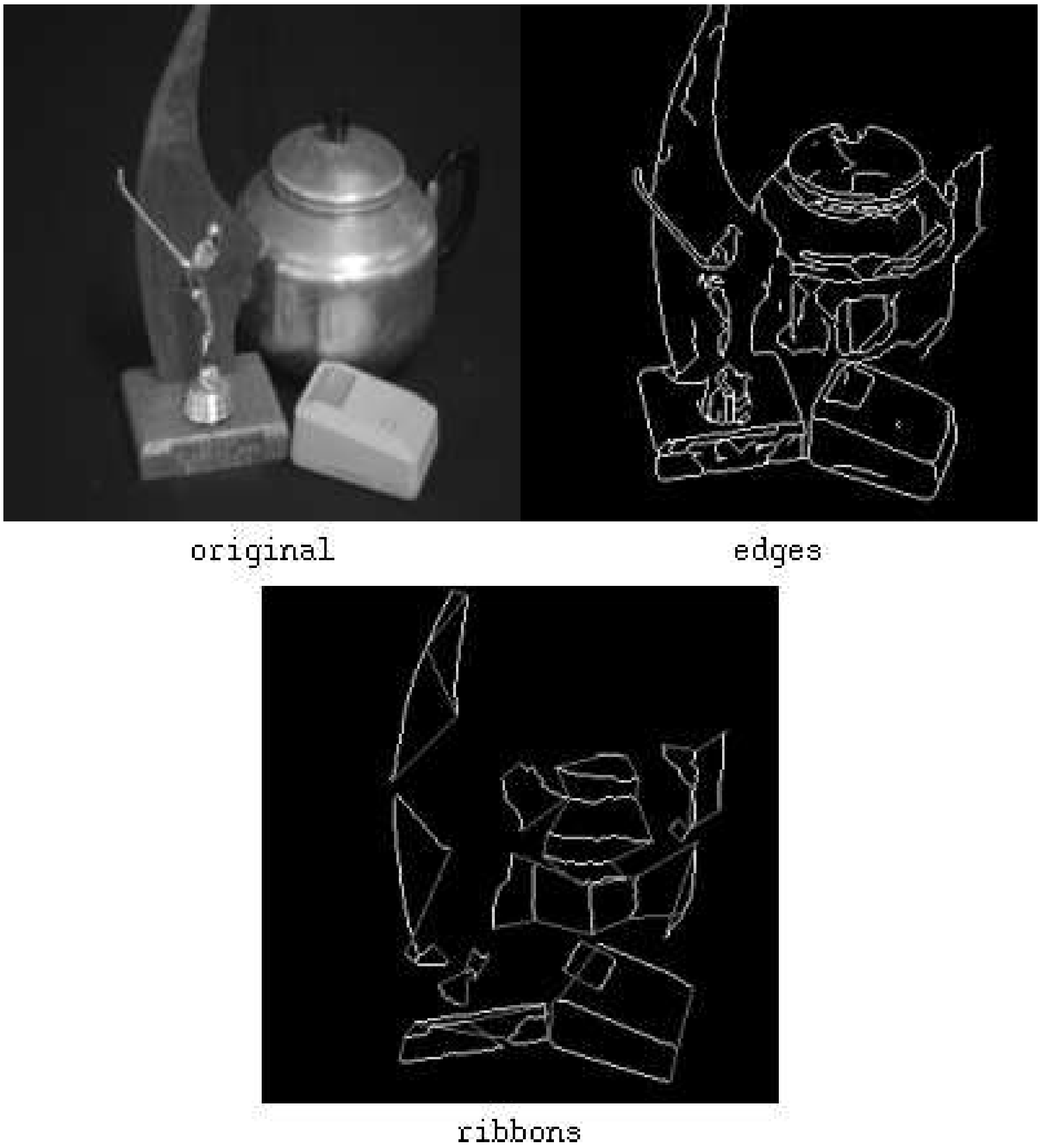



Figure 3.21: An example of the perceptual organisation process. (a) The original image, (b) the extracted edges, and (c) the resultant ribbons

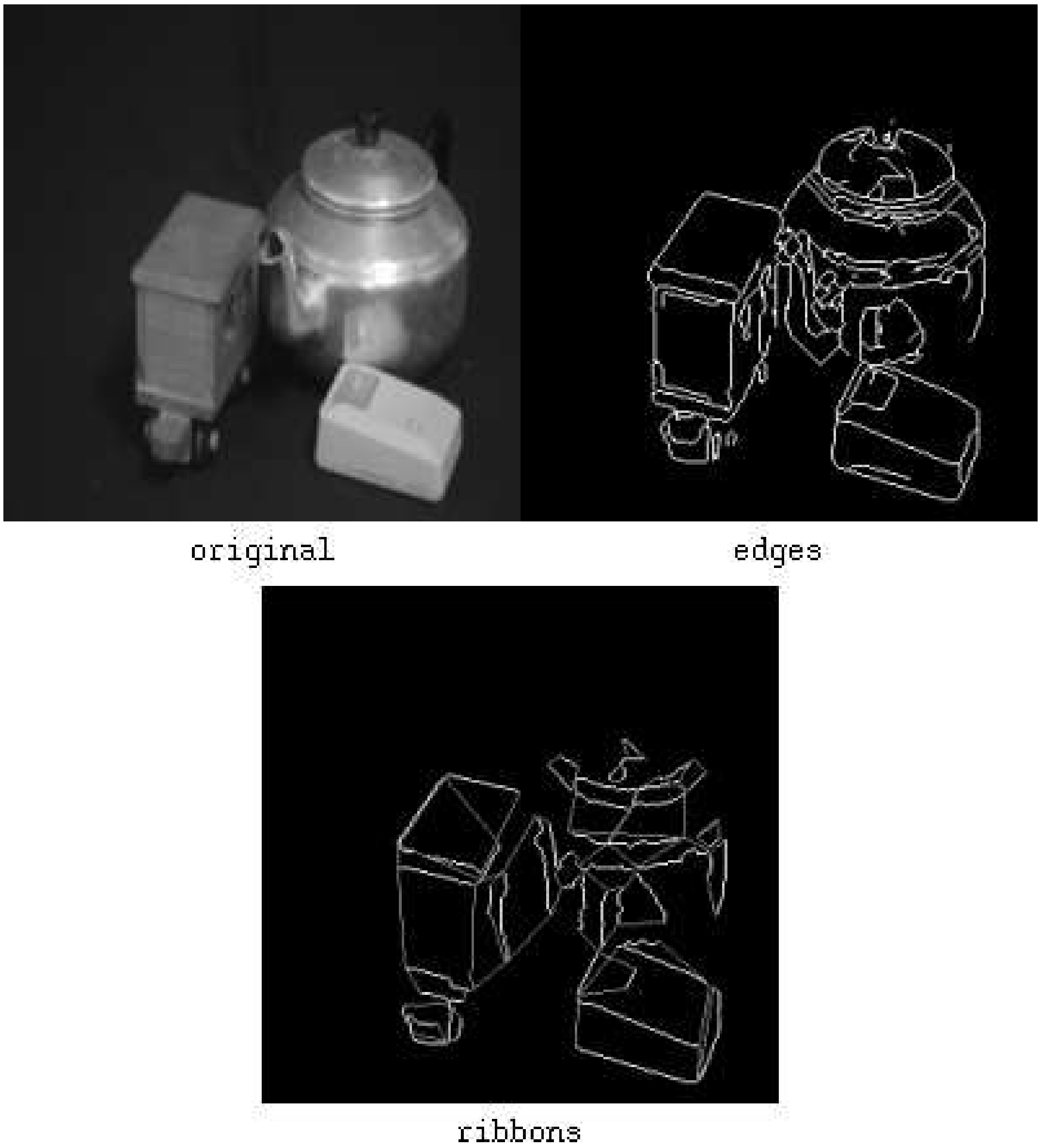



Figure 3.22: An example of the perceptual organisation process. (a) The original image, (b) the extracted edges, and (c) the resultant ribbons

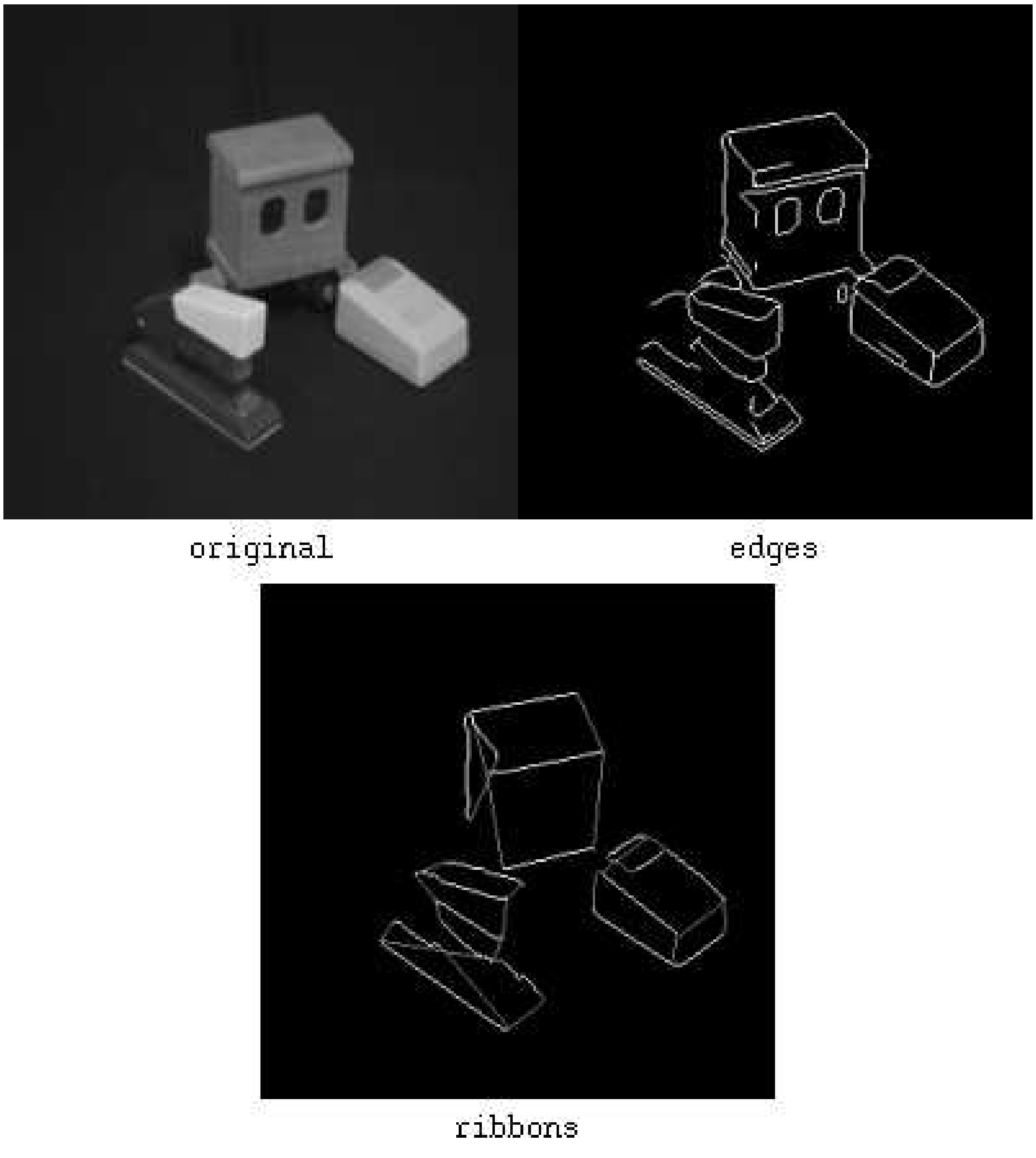



Figure 3.23: An example of the perceptual organisation process. (a) The original image, (b) the extracted edges, and (c) the resultant ribbons

If the techniques were extended to address these problems, then they could be useful in a complex object recognition system. However, such techniques are still likely to fall short of a generally applicable segmentation procedure, but could be used to aid such segmentation. Since we require a more general segmentation procedure for the current object recognition system, these techniques have not been used in the current system.

# Chapter 4

# Multi-scale Adaptive Segmentation using Edge and Region Based Attributes

The previous chapter on perceptual organisation was an attempt to solve the problems of image segmentation using higher order information (ie information about edge symmetries, rather than just edges). That is, edges are first extracted, and then specific image features (symmetries in this case) are searched for. This is a valid approach, but lacks generality. Still, the need for extracting reliable image regions is required, and common segmentation algorithms do not suffice.

Two major approaches to the segmentation problem thus far include edge-based and region-based methods. Edge-based approaches find edges in an image, but "edge elements often exist where no meaningful scene boundary does, and conversely are often absent where a boundary is" ([5], pg. 119) and hence do not always produce good regions (ie they may not produce closed regions). On the other hand, region-based methods, do produce closed regions, but typically do not make use of edge information. In fact, Haralick and Shapiro ([58]), state that "adjacent regions of a segmentation should have significantly different values with respect to the characteristic on which they are uniform", and this is a commonly used criteria for region-based techniques. However, such a criterion is not applicable to all images or all application domains. For example, consider a recognition system attempting to recognise an object which has uniform surface characteristics across the entire object. In such a case, there may exist a sharp edge between two surfaces of the object where the region characteristics are very similar on both sides of a rather prominent edge. Since the recognition-by-parts paradigm is used throughout this thesis, it would be useful to segment the two surfaces into different regions or parts, but these regions "may be merged because the change in pixel characteristics at the boundary is frequently swamped by the overall character of the regions" ([57]). Taking these problems into account, it is reasonable to suggest that "region-finding and line-finding techniques can cooperate to produce a more reliable segmentation" ([5], pg. 149). Other authors have also investigated such procedures, including Pavlidis and Liow ([89]), Haddon and Boyce ([57]), and Chu and Aggarwal ([36]).

The basis of the approach used here is to use region characteristics to decide if a given region should be segmented at a finer scale. Multi-scale approaches have also been utilised by other authors to improve segmentation ([77], [69], [83], [21], [73], [109], [59]). For example, Marr and Hildreth ([77]) accept edge elements only if they occur in at least two edge images created using two distinct scales. But typically, such approaches make use of multi-scale techniques using only one form of information (ie not both region information and edge information).

For these reasons, an algorithm was developed for segmenting images which uses both edge and region attributes at multiple scales to adaptively and automatically choose the best scale at which to segment different parts of an image. In this way, images with widely varying characteristics can be successfully segmented using one procedure. The regions produced by this algorithm can then be used as the basic parts of the objects in the images.

## 4.1 Overview of the Segmentation Process

As was stated above, the current segmenter utilises a multi-scale approach to segmentation. The basic segmentation algorithm is defined recursively as follows:

1. Edges are extracted from an input image at a given (largest) scale.

2. A parallel edge growing procedure is performed to close any small gaps in the edges (a primitive type of perceptual organisation).

3. Regions are then formed using a simple boundary fill algorithm on the edge image.

4. The intensity variance of the region is measured and the region is further segmented at the next scale down (recursively calling this procedure).

5. If the average variance of the child regions is less than the variance of the current region, then the current region is replaced by the child regions.

The remainder of this chapter is organised as follows. In Section 4.2 the edge extraction and extension phase will be described, Section 4.3 describes the adaptive segmentation procedure and results are given in Section 4.4.

## 4.2 Edge Extraction

Initially, edge points are found in an input image using Canny's edge detection algorithm ([32, 33]). These are then linked into edges using an hysteresis linking routine. For the current set of images, the low hysteresis threshold was set to 0 and the high set to 25 (the input images are greyscale with intensities ranging from 0 to 255). Junctions are then formed at points where two or more edges meet. Those edges with isolated endpoints are extended in a parallel fashion until they hit another edge, or until a maximum extension limit is reached. Although sometimes, it would be useful if small edges could be extrapolated over large gaps, more often such edges would be misleading. Hence, the maximum extension possible for a given edge is also limited by the length of that edge. Algorithm 4.2 describes the procedure in algorithmic form. The inputs to the algorithm are the intensity image from which edges are to be extracted, the extension limit, and the scale at which to apply the edge detector ($\sigma$). A list of edges are returned by the procedure. In a sense, this type of procedure can be considered a type of primitive perceptual organisation where previously unrelated edges are joined to form larger edges. The current procedure uses linear extrapolation, but extrapolation techniques using more

**Algorithm 4.2.1** *Extracting Edges*

```
function  extract_edges(intensity_image, extension_limit, σ)
inputs:
          intensity_image /* input intensity image */
          extension_limit /* maximum extension for an edge */
          σ /* scale at which to perform edge extraction */
endinputs
outputs:
            edges /* list of edges formed by the procedure */
endoutputs
begin
  edges ← canny(intensity_image, σ)
  I ← {set of isolated endpoints}
  for (time = 0 to extension_limit) do
       foreach (endpoint ∈ I) do
          linearly extend edge by one pixel
          if ¬(isolated(endpoint)) then
             I ← I − {endpoint}
          fi
       endforeach
  endfor
  foreach (endpoint ∈ I) do
     discard any extension on endpoint
  endforeach
  return(edges)
end
```

complex curves could be used. Such a procedure was found to be unnecessary for the current application, and, indeed, since the maximum extension of an edge is typically small, there would be little to gain from this enhancement, although, in some applications it may be useful to use more complex curves.

Figure 4.1 demonstrates the usefulness of such an approach. In this example, the extension limit was set to 12 pixels, and this appears to be a fair limit for the tests conducted here. As can be seen from the figure, most of the gaps in the edge image have been closed by the edge extension procedure in a satisfactory way, thus allowing for the more reliable extraction of closed regions based on the extended edge map.

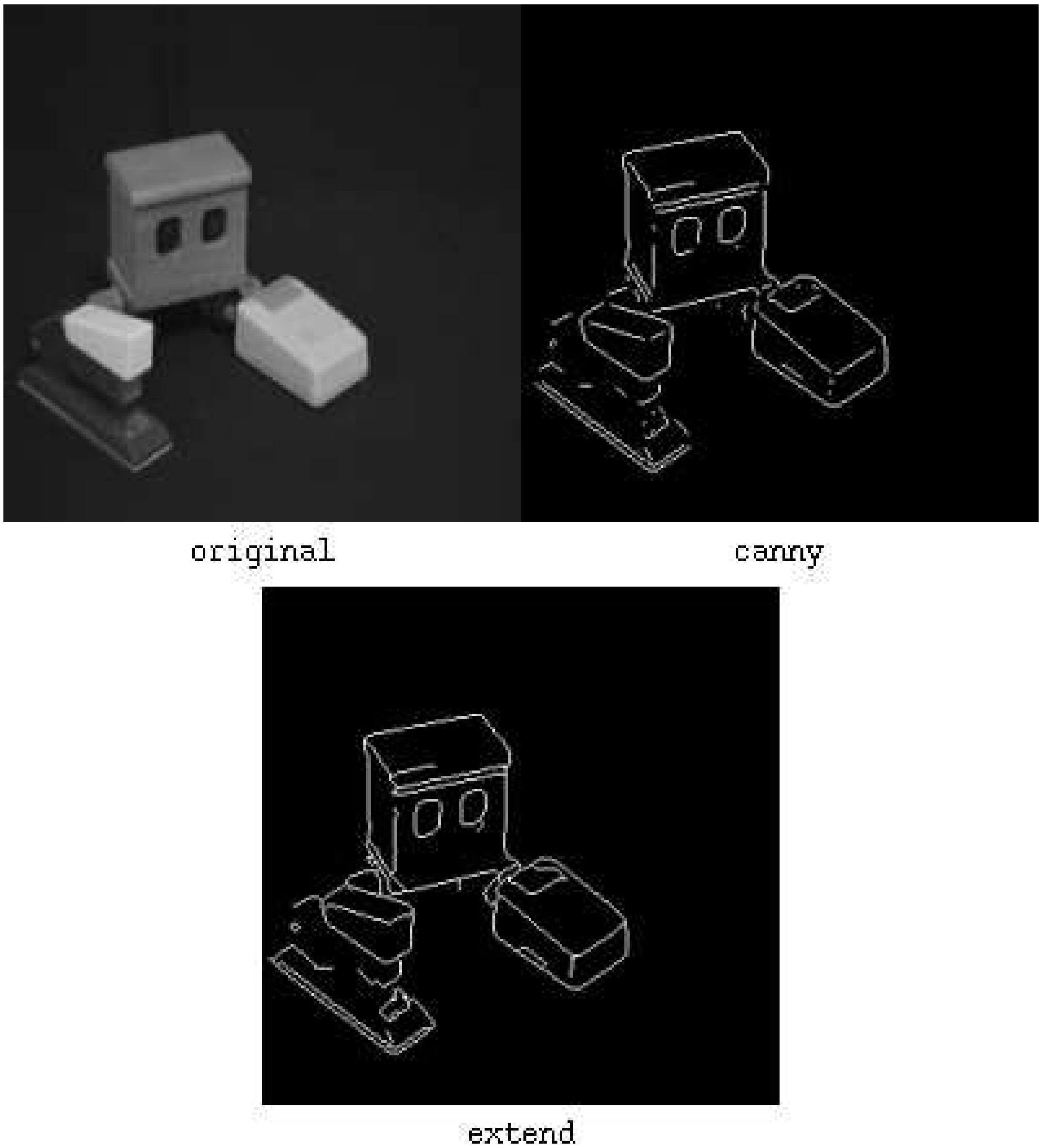



Figure 4.1: An example of the parallel edge growing procedure: the left top image is the original input, the right top image is the edges extracted using Canny's edge extraction procedure with $\sigma = 1.0$, and the bottom image is the extended edge map.

## 4.3 The Adaptive Segmentation Procedure

The adaptive segmentation procedure is an inherently recursive one that makes use of both edge and region information to perform the segmentation. The actual segmentation is performed using edge information, but the decision on whether to further segment a region is based upon region statistics. In this case the statistic used is the variance of the intensity inside the region, but other statistics are equally applicable (texture statistics, for example). The elegance of the method is that the decision to further segment a given region is not based on a thresholding of the region statistic. Instead, a decision is made by comparing the region statistics of the sub-regions at the next lowest scale to the region statistic at the current level. In this way, the segmentation process occurs automatically in a self-similar way, similar to the method introduced by Lowe ([74]) for segmenting edges into linear edge segments.

The region statistic which has been utilised is the *mean absolute deviation* or the *mean deviation* ([98]) of the intensity values within a region. This measure was chosen as it tends to be more robust than the variance in that it is more stable in the presence of outliers (salt and pepper noise for example), especially when dealing with small sample sizes (eg small regions). The *mean absolute deviation* is defined as:

$$ADev(x_1 ... x_N) = \frac{1}{N} \sum_{j=1}^{N} |x_j - \bar{x}| \qquad (4.1)$$

The recursive procedure for the adaptive segmentation algorithm is given in Algorithm 4.3. The procedure accepts as input a region (initially covering the whole image), an intensity image, an array of scales for performing the edge extraction (the array $\sigma$ should be in decreasing order of size, ie largest scale in $\sigma[1]$ etc), the current level of scale, the minimum size below which a region will not be segmented, and an output parameter called *variance* which contains the value of the region statistic on return. A list of regions is returned by the procedure. As can be seen from Algorithm 4.3, if the input region is less than the size threshold, or the minimum scale has been reached, the input region is returned, otherwise the input region is segmented into smaller regions using the current scale factor, and these regions are possibly further segmented using the next scale factor down. The size threshold parameter is necessary to avoid segmenting regions into small non-meaningful fragments and will be determined by the nature of the application.

The result of applying this algorithm to the entire image is a segmented image (represented as a list of regions) where each region corresponds to a segmentation of the input image at one of the input scales. Different regions may correspond to segmentations at different scales. Hence, the final segmentation is not limited to one scale, but may include information from a number of scales.

## 4.4 Some Examples

With the above segmentation procedure, the only parameters which need to be determined are the number and size of the scales with which to perform the adaptive segmentation. In general, this is going to largely depend on the class of images which are being input, and the type of information required which will vary widely from application to application. Nevertheless, it is useful to examine how these parameters affect the results of the procedure.

In general, it has been found that the use of three different scales is sufficient at least for the class of images being used here. Typically, these values have been with $\sigma = 4.0,\ 2.0,$ and $1.0$ pixels as the Gaussian parameter with respect to a $256 \times 256$ image format. Using $\sigma > 4.0$ produced edges which were too smooth and too badly localised to be of much use, and $\sigma = 1.0$ is generally as fine a resolution as needed. It was found that even using only two scales ($\sigma = 2.0,\ 1.0$) produced a marked improvement over using just one scale, and in many cases this produced results comparable to using three scales.

Figures 4.2 to 4.4 display the results from the procedure using a number of different images which are representative of the test images used for the object recognition system. The figures display the original image, the edge images at different scales and the final segmentation. The segmented image has different greyscales for separate regions and the region boundaries are emphasised by black edges. As can be seen from the figures, the procedure has worked well in finding closed regions in the

**Algorithm 4.3.1** *Adaptive Segmentation*

```
function  adaptive_segmentation(input_region, intensity_image,
                                σ[], level, size_threshold,
                                variance)
inputs:
        input_region /* initial region to segments */
        intensity_image
        σ[] /* array of scales for edge extraction */
        level /* index into σ[] */
        size_threshold /* minimum size of regions */
        variance /* intensity statistic */
endinputs
outputs:
          ret_regions /* list of regions */
          variance /* also an output variable */
endoutputs
begin
  if (input_region.size < size_threshold) ∨ (level > max_level) then
    variance ← region_statistic(input_region, intensity_image)
    return(input_region)
  endif
  edges ← extract_edges(intensity_image, extension_limit, σ[level])
  sub_regions ← boundary_fill(edges)
  ret_regions ← ϕ /* empty set */
  foreach sr ∈ sub_regions do  /* adaptively segment regions */
    child_regions ← adaptive_segmentation(sr, intensity_image,
                                          σ, level + 1, child_var,
                                          size_threshold)
    avg_var ← avg_var + child_var /* average variance */
    ret_regions ← (ret_regions ∪ child_regions)
  endforeach
  avg_var ← avg_var/sub_regions.number
  current_var ← region_statistic(input_region, intensity_image)
  if (current_var > avg_var) then  /* return child regions */
    variance ← avg_var
    return(ret_regions)
  else  /* return input region */
      variance ← current_var
      return(input_region)
  endif
end
```

image, and in general has found a better representation of the image than each of the three edge images would have found alone. The smallest region allowed in these images was set to 100 pixels. Figure 4.3 shows the results of the segmentation using just two scales instead of three ($\sigma = 2.0$ and $\sigma = 1.0$). As can be seen from the figure, the results are comparable to Figure 4.2 although marginally different.

The algorithm can also be successfully applied to different classes of images. For example, Figure 4.5 displays the segmentation results using the well known photograph of Lenna. The results are extremely good in all areas except for the feathered addition to Lenna's hat, which is as expected since that region is particularly complex and would be extremely difficult to segment successfully. Since more detail was required to segment Lenna's face, the minimum region size was set to 20 pixels in this case, and only two scales were used. The advantage of using multiple scales is clearly demonstrated with the segmentation of Lenna. For example, Lenna's chin, eyes and nose are not well extracted at $\sigma = 1.0$, but are extracted with $\sigma = 2.0$. Similarly, the top of the hat is not well extracted with $\sigma = 2.0$, but is extracted with $\sigma = 1.0$. In the final segmentation, however, all of these regions are successfully extracted, thus combining the best segmentations from both of the scales.

From the results displayed here, at least subjectively (it is difficult to define objective measures in such a domain), it appears that the segmentation algorithm works quite well, producing reliable regions which can be used as the basic parts for the recognition algorithm which will be described. The algorithm is more appropriate for the current recognition system since it always produces closed regions (as opposed to edge-based algorithms), yet still makes use of edge information (as opposed to region-based algorithms). Clearly, the results are an improvement over those obtained in the previous chapter, which often ignored significant proportions of the image. Also, it should be noted that although the algorithm does not perform perfectly on all images (image of the teapot for example), the results are still reasonable for those images, and it is not necessary to segment the image similar to how a human subject would anyway (who would probably ignore the specular regions of the teapot for example). As long as the regions can be robustly segmented in a number of images, and they provide useful discriminating information between different objects (eg the specular regions on the teapot are probably important clues for recognising the teapot), then the segmentation algorithm has performed successfully.

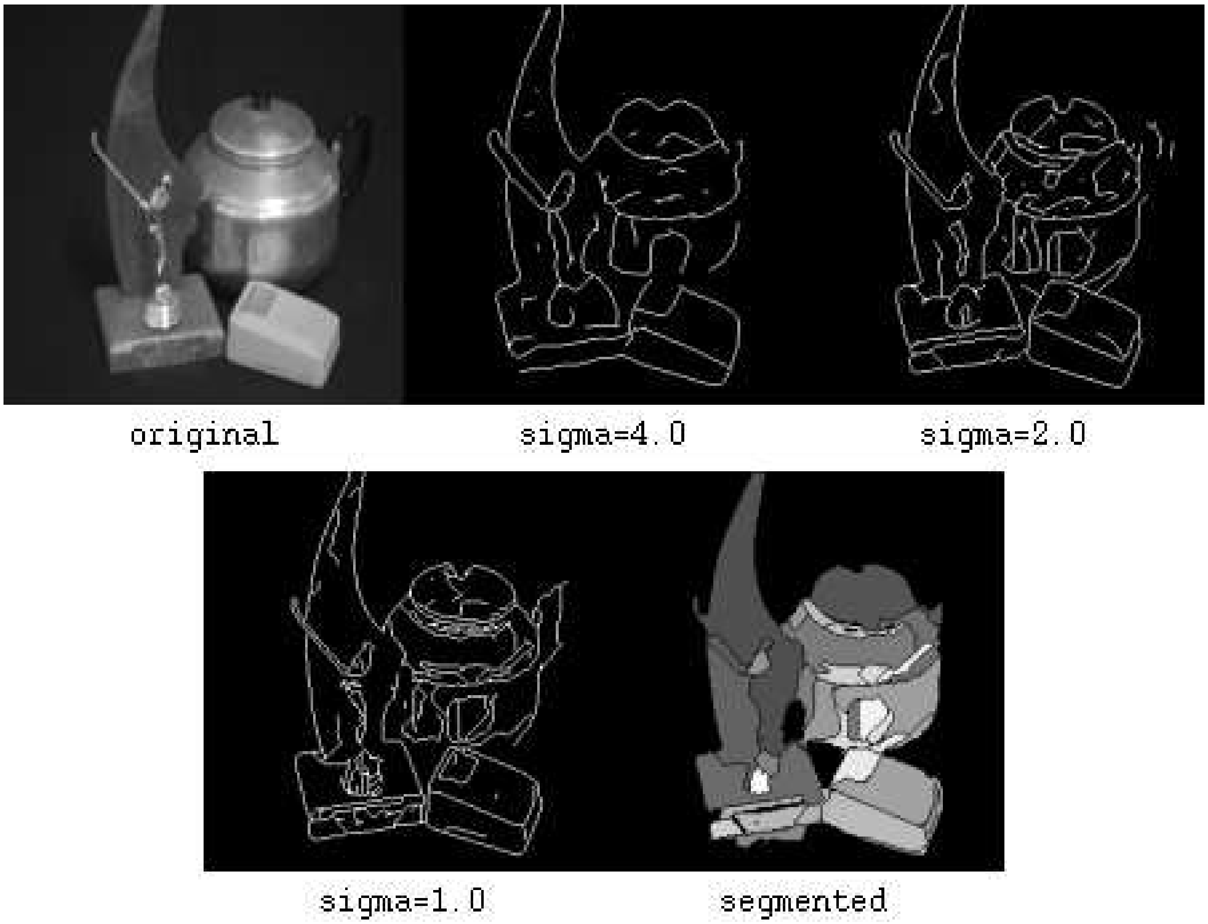



Figure 4.2: Three objects on a black background: The trophy and the mouse are segmented well. Segmentation of the teapot is somewhat more complex due to the nature of its specular surface. The figure displays the original image, the edge images at different scales and the final segmentation.

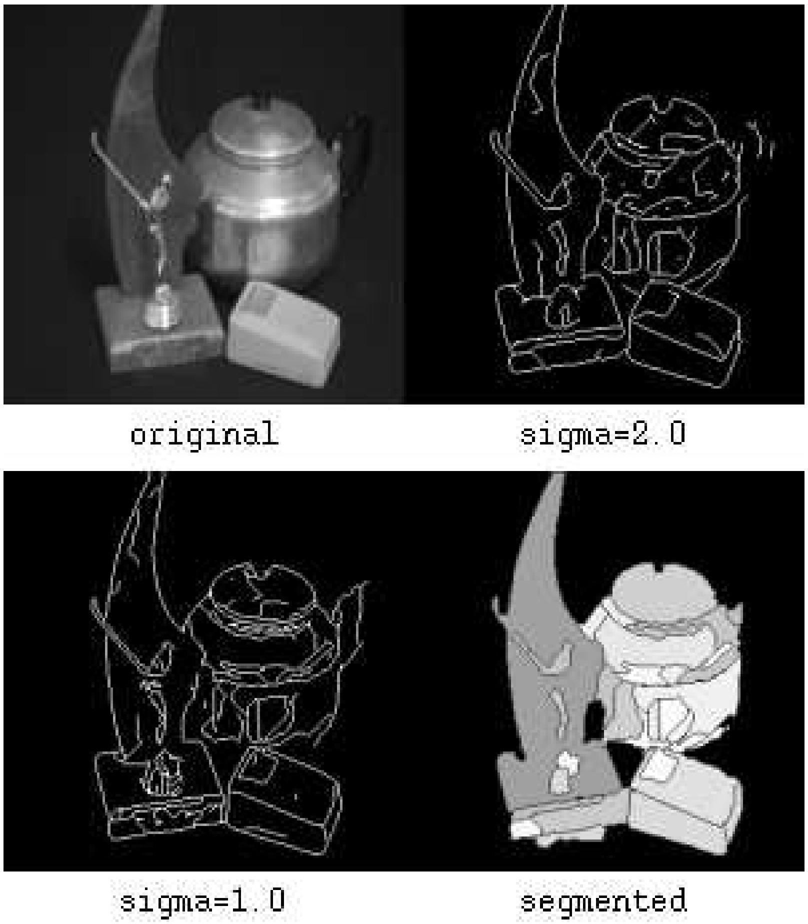



Figure 4.3: Three objects on a black background: The same objects as in the previous figure except segmented using fewer scales. The segmentation is somewhat different, but the algorithm has still performed well. The figure displays the original image, the edge images at different scales and the final segmentation.

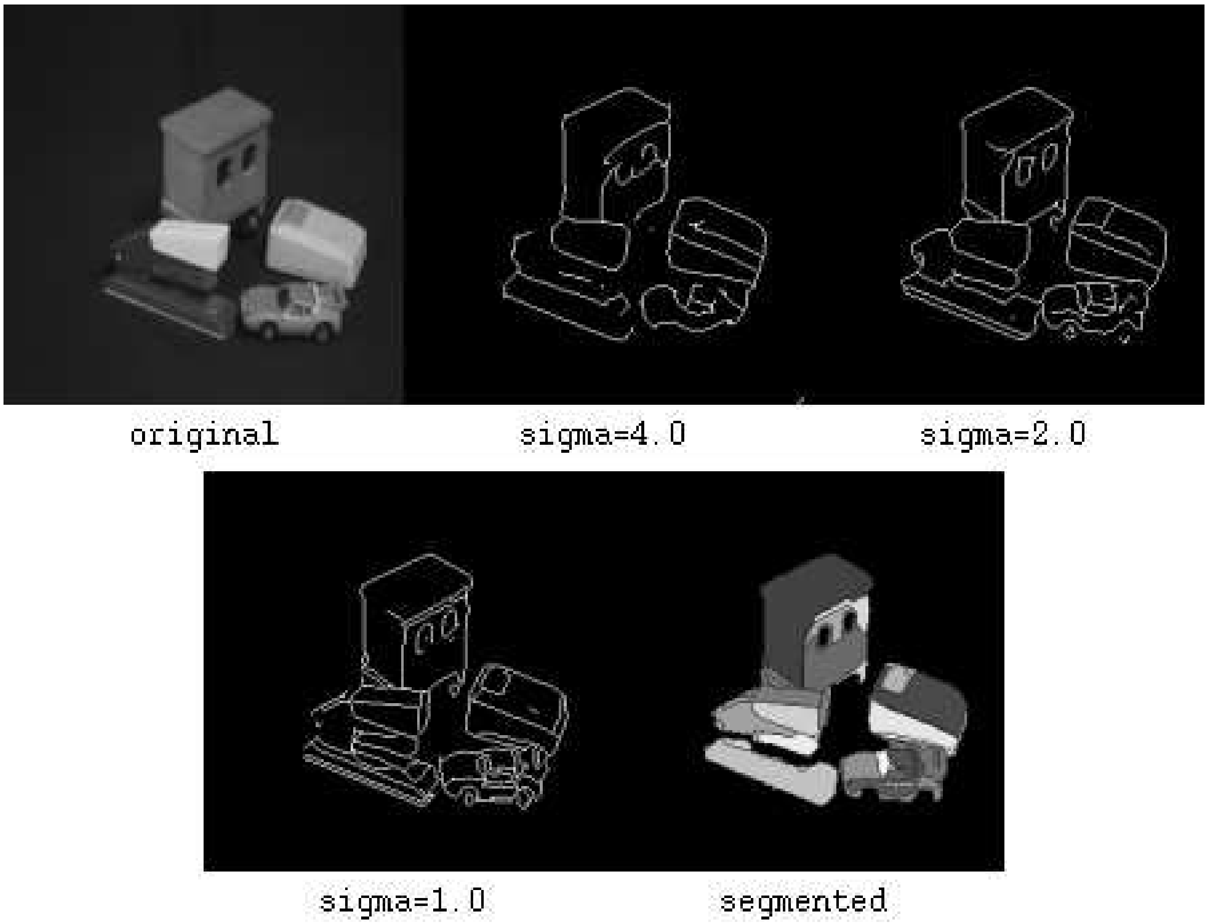



Figure 4.4: Four objects on a black background: Here the algorithm has performed well on all objects. The advantages of the algorithm are clearly demonstrated on the stapler, where the top part has been segmented at the smallest scale, while the bottom part has been segmented at a larger scale. The figure displays the original image, the edge images at different scales and the final segmentation.

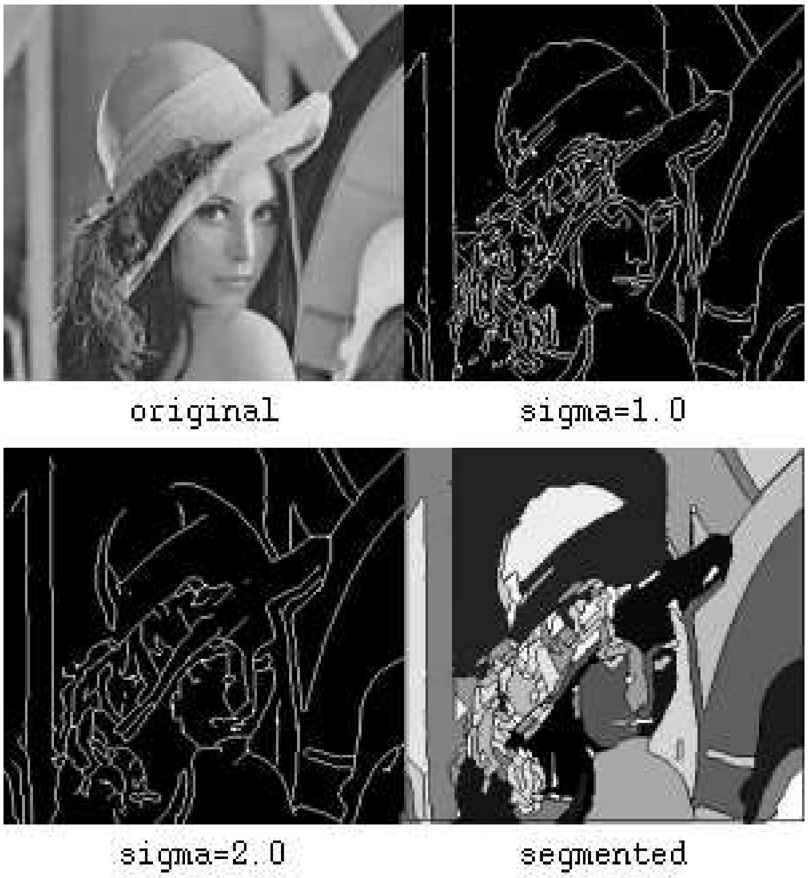



Figure 4.5: The segmentation of Lenna: Again, the advantages of the multi-scale approach are displayed especially around Lenna's eyes and nose. The figure displays the original image, the edge images at different scales and the final segmentation.

# Chapter 5

# A Stereo Algorithm Based on Cluster Analysis of Curve Segments

The techniques of the previous chapter are useful for extracting the parts of an object from a scene. However, it is possible to extract more than purely 2D information from 2D intensity images, and it seems likely that such information could be used to improve object recognition performance. There are numerous methods for calculating depth from intensity images including: depth-from-stereo, shape-from-shading, depth-from-focus etc (commonly referred to as shape-from-X or depth-from-X techniques). The human visual system apparently uses all of these techniques, and similarly, these techniques have been widely addressed in the computer vision literature. The depth-from-stereo problem has probably received the most attention in the computer vision literature as it is the most obvious way in which we infer depth. Nevertheless, very few object recognition systems attempt to use depth data calculated from a depth-from-stereo process. The TINA system ([94]), makes use of stereo data for extracting a wireframe model of an object and for locating an object for grasping with a robot arm, but does not explicitly use this data for object recognition. Depth data obtained from such a source is likely to improve object recognition performance, and this is the motivation for using a depth-from-stereo algorithm here. Still, it is difficult to extract useful depth data by performing a stereo match between entire regions (as are produced by the techniques discussed in the previous chapter). For this reason, the stereo algorithm described here operates independently of the segmentation algorithm described in Chapter 4.

The depth-from-stereo problem involves calculating the depth information of a scene from two or more images of that scene taken at different positions in space. Given the position of two imaging devices, and points in the two images which correspond to one scene point, it is relatively simple to uniquely determine the position of the scene point relative to the imaging devices from the coordinates of the two image points (see Figure 5.1). However, difficulty arises in obtaining correspondence between a point in one image with the equivalent projection of the scene point in the other image. This is the major problem with stereo vision for which Dhond and Aggarwal ([37]) give an excellent review.

Many of the techniques proposed thus far, make use of point based features as matching primitives (eg Marr and Poggio [78], Grimson [54]). Such techniques consider the matching of feature points in a local sense. That is, an attempt is made to match a feature point in one image to a feature point in the other image, using features which have been computed locally. A major problem with such approaches is that calculated depths of neighbouring pixels are not guaranteed to be consistent with each other. For example, two neighbouring pixels lying on the same surface may have widely varying depth values when the depth values should be similar. Marr and Poggio ([78]) attempt to solve this problem by only accepting matched points if a certain percentage of matched points in any given region lie within the allowable disparity range. However, this statistical constraint does not overcome the problem of discontinuities between neighbouring pixels ([54]). Therefore, a number of techniques have been proposed that impose constraints on the matching algorithm in order to obtain a more consistent solution. These techniques include the concepts of figural continuity (Mayhew and Frisby [79], [54]), the disparity gradient limit (Burt and Julesz [30] and Pollard, Mayhew and Frisby [93]) and the coherence principle (Prazdny [96]). Basically, these techniques assert that matter is cohesive, and hence the disparity difference between adjacent matched feature points should be reasonable (where reasonableness is defined by the particular technique being used).

Only relatively recently have efforts been made to develop stereo matching algorithms which take advantage of higher level non-local features such as edge contours and regions. Such techniques lessen the need for additional constraints such as those discussed above. Examples of such efforts include Medioni and Nevatia [81] and Ayache and Faverjon [4], who both utilise linear edge segments, which are approximations of the actual edges, for matching. More recently, Nasrabadi [86] has proposed a stereo matching algorithm based on the generalised Hough transform of small edge segments, and a graph matching approach. From a curve based approach, Ma, Si and Chen [75] utilise quadrics to perform matching, although their algorithm assumes that quadric based curves exist on the surfaces of the objects in the scene. This assumption restricts the generality of the approach. Robert and Faugeras [99] approximate edge contours as B-splines and use these as their matching primitives. Their algorithm, however, is a trinocular stereo based approach and also requires characterisation of the contours as B-spline curves.

Algorithms which utilise free form contours are less common. Kim and Bovik [67] match high information points along free form contours first, and propagate the disparities along the contours making use of the disparity continuity constraint. Sherman and Peleg [107] make use of a relaxation labelling scheme to match free form contours. A list of possible matches for each contour is first constructed based upon the following constraints: epipolar, matched contours must have at least 8 scan-lines in common; disparity range, determined by approximate distance to scene points and known imaging geometry; contrast sign; and local similarity of the orientation of points on the same scan-line. At each iteration, support or otherwise for each pair of matches is determined using similarity of contours and the disparity continuity constraint between this possible match and the most likely match of neighbouring contours.

The approach presented here is similar to that proposed by Sherman and Peleg [107], but is different for a number of important reasons. Firstly, this approach is not an iterative optimisation approach like relaxation labelling, and hence should prove to be more efficient. Secondly, the matching process is formulated as a nearest neighbour problem in a contour attribute space, and matches are then accepted or rejected in edge contour space. Thirdly, edges which are not matched are split into smaller edges for matching at the next stage.

The geometry of the image formation process and a solution to calculating the depth of a scene point given matching pixels is outlined in Section 5.1. Section 5.2 describes the novel matching process which has been developed. Finally, the performance of the algorithm is discussed in Section 5.3.

## 5.1 Imaging Geometry

The determination of the depth of a scene point from two images is dependent on the geometry of the imaging system used. The simplest geometry involves two vertically aligned cameras with parallel axes utilising lenses with the same focal length and image formation characteristics. A top view of such a system is shown in Figure 5.1. In this system the y-axis is projecting out of the page towards the reader, and the world coordinate system corresponds to $O_L$.

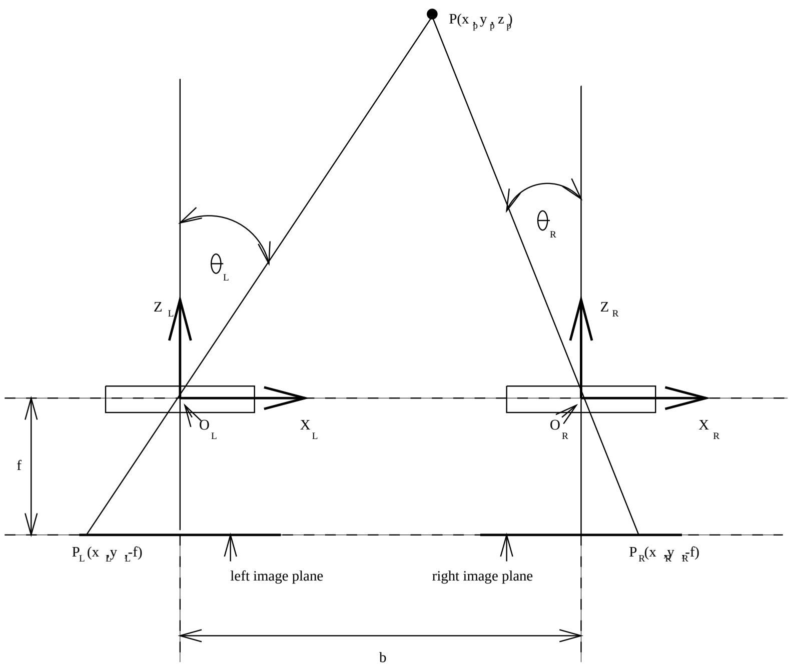



Figure 5.1: Geometry of the stereo imaging system.

The use of this geometry simplifies the point correspondence problem, since a given point in one image will have its corresponding point in the other image lying on a horizontal line. This line is called the epipolar line. It also exists in other imaging geometries, but is not horizontal. Due to the simplicity involved with this type of imaging system, it will be used to derive formulae for calculating the scene coordinate of a point given the coordinates of the corresponding image points. It is also possible to derive formulae with more complicated imaging systems, but the mathematics involved is more cumbersome.

The problem can now be formulated as the recovery of the scene point $P(x_P, y_P, z_P)$, from the two image points $P_L(x_L, y_L, z_L)$ *and* $P_R(x_L, y_L, z_L)$. To solve this, we must first derive the equations for projecting the scene point $P$ onto the image planes $L$ and $R$. These equations can then be solved to perform the reverse perspective projection and hence recovering the scene point $P$.

Firstly, for perspective projection, we wish to intersect the lines passing between $P$ and $O_L$, and $P$ and $O_R$ with the plane $z = -f$. Hence we need to solve the equations:

$$z = -f \tag{5.1}$$

and

$$\frac{x - x_{O_i}}{x_p - x_{O_i}} = \frac{y - y_{O_i}}{y_p - y_{O_i}} = \frac{z - z_{O_i}}{z_p - z_{O_i}} \tag{5.2}$$

where $i$ represents either $L$ or $R$. Solving these equations, we get points in world space:

$$x_L = \frac{-fx_p}{z_p} \quad y_L = \frac{-fy_p}{z_p} \tag{5.3}$$

$$x_R = \frac{f(b - x_p)}{z_p} + b \quad y_R = \frac{-fy_p}{z_p} \tag{5.4}$$

The only unknowns in these equations are the coordinates of the scene point $P(x_P, y_P, z_P)$. Hence, we have three equations and three unknowns and can solve for $(x_P, y_P, z_P)$. Solving for $z_P$ gives:

$$z_P = \frac{fb}{x_R - x_L - b} \tag{5.5}$$

The denominator in this expression refers to the disparity, $d$, between the two image points when one image is overlayed on the other. This quantity can be measured from the two images and hence can be used directly in the computation of $P$. Hence, we have:

$$z_P = \frac{fb}{d}, \tag{5.6}$$

$$x_p = \frac{bx_L}{d}, \tag{5.7}$$

and

$$y_p = \frac{by_L}{d}. \tag{5.8}$$

Although the imaging geometry employed above is simple, it is probably one of the most practical, as it is easy to implement, and the mathematics of the geometry is simple. The actual physical setup is described more fully in Chapter 9. It remains to develop techniques by which corresponding points in the two images can be found.

## 5.2 The Stereo Matching Algorithm

The stereo matching process as presented here can be divided into five reasonably independent steps. These are:

- the edge extraction stage,
- the vertical registration stage,
- the disparity estimation stage,
- the edge matching stage,
- the matching verification stage,
- the edge splitting algorithm.

Each of these stages are discussed in the following subsections. Although the disparity estimation stage occurs before the edge matching stage in an algorithmic sense, it will be discussed after the edge matching stage as an understanding of the edge matching stage is required before the disparity estimation stage can be explained.

### 5.2.1 Edge Extraction

Edges are extracted from the images using the Canny edge extraction algorithm ([32, 33]) at a scale of $\sigma = 2.0$ pixels with respect to a $256 \times 256$ pixel image. The choice of scale is important as too small a scale can cause confusion due to the large amount of edges, and too large a scale reduces the usefulness of the algorithm due to the lack of depth information. It is felt that choosing a value of $\sigma = 2.0$ is a good compromise between the two extremes. A non-maximal suppression routine is then used to thin the edge image, and edge pixels are linked into edges using an hysteresis thresholding routine. The hysteresis thresholds used are 25 and 0 for the high and low thresholds respectively. The techniques presented here are not dependent on the edge extraction algorithm, however, and any reasonable extraction technique may be used.

### 5.2.2 Vertical Registration

Due to the inaccuracies of the stereo setup (one camera on a sliding arm), it is likely that a stereo pair of images will not be in vertical registration. Since the stereo algorithm assumes that matching points are on the same horizontal scan-line, it is important that the images be brought into vertical registration before the stereo matching process proceeds. An approach based on scene knowledge is used in this case. Since the stereo images are of objects placed on a black background, it is reasonably simple to perform figure-ground separation on these images using a simple histogramming technique. This technique works by finding the largest peak in an image histogram (assumed to correspond to the background), and then searching for a minimum in the histogram on the right of the peak (greater intensity value) but reasonably close to the peak. The value of this minimum is the threshold value for the image - all intensity values lower than the threshold are background pixels and all intensities higher are foreground pixels. This process is performed on both the left and right images and the right image is moved vertically so that the $y$-coordinates of the centroids of the two foregrounds coincide. This technique has proved very successful for vertical registration and should not suffer from any problems unless the lighting conditions between the left and right images are significantly different.

### 5.2.3 Edge Matching

For each edge in the left and right images (denoted $E_L$ and $E_R$), attributes are extracted to form attribute spaces for the left and right images (denoted $F_L$ and $F_R$). The attributes used in the current implementation are:

1. the number of pixels in the edge contour,

2. the coordinates of the centroid of the edge contour (with the x-value shifted by a constant $\Delta x$. Edge contours in the left image are not shifted, edge contours in the right image are shifted by an estimate of the disparity. This helps to bring edge contours into alignment),

3. the first and second moments of the distance of each pixel from the centroid of the edge contour, $L$, defined by:

$$m_1 = \sum_{i=1}^{N_L} \frac{d(L^i, L^c)}{N_L} \tag{5.9}$$

   and

$$m_n = \sum_{i=1}^{N_L} \frac{(d(L^i, L^c) - m_1)^n}{(N_L - 1)} \tag{5.10}$$

   where $L^i$ refers to the coordinates of the $ith$ point on the $Lth$ edge, $L^c$ refers to the coordinates of the centroid, $N_L$ refers to the number of points on the edge contour, $L$, and $d(a, b)$ refers to the Euclidean distance between the two points $a$ and $b$, and

4. the orientation of the major axis of the edge contour.

These attributes should form a good characterisation of the edge contour and its position. An example edge contour displaying some of these attributes is given in Figure 5.2. Each of these attributes is normalised to have a mean of zero and a variance of one, so that the units are comparable.

Matches are determined by finding the nearest-neighbour (1-NN) in the right image, for each of the edges in the left image, in attribute space. A match in

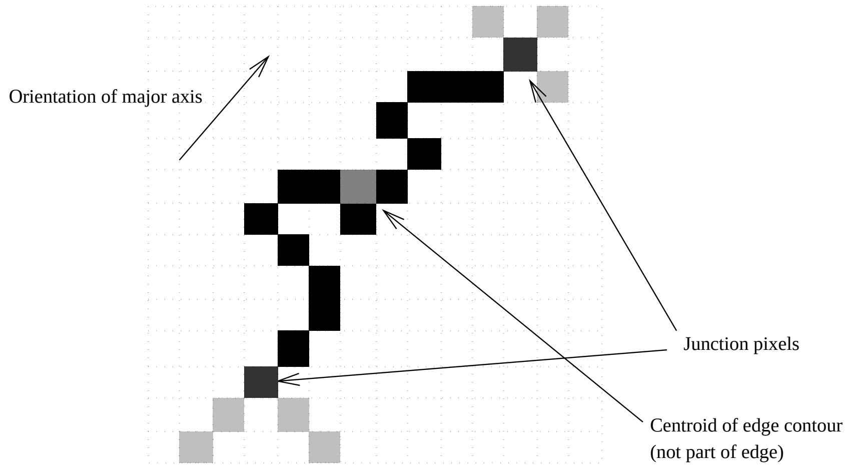



Figure 5.2: An example edge contour.

attribute space, however, does not guarantee a true match in the edge contour space. Problems which are addressed are:

- A nearest neighbour match does not guarantee that the edge contour points fall within a reasonable disparity range which can be determined by the approximate depth of the scene, and the known imaging geometry (Sherman and Peleg [90]). The approximate depth of the scene can generally be estimated through knowledge of the problem domain. This problem is overcome by not accepting matches for which the disparity of the centroids falls outside the allowable range.

- The epipolar constraint may not be satisfied. This is enforced by only accepting matches for which the edges overlap horizontally (ie they intersect at some point if slid over each other horizontally).

- Due to noise and differing lighting conditions, each image may have been segmented differently causing edge contours to be broken in different places. This problem is solved by splitting edges which do not match in edge contour space into smaller edges.

- The matches may not be unique. Uniqueness is ensured by allowing edges only one match and the best combination of matches is found globally by minimising the Euclidean distances of the matches in attribute space (see Algorithm 5.2.3).

- Ordering between pairs of matches may be inconsistent (see Figure 5.3). This problem is solved by defining a number of vertical *ordering* windows, within which matches must be ordered consistently (ie the same ordering from left to right).

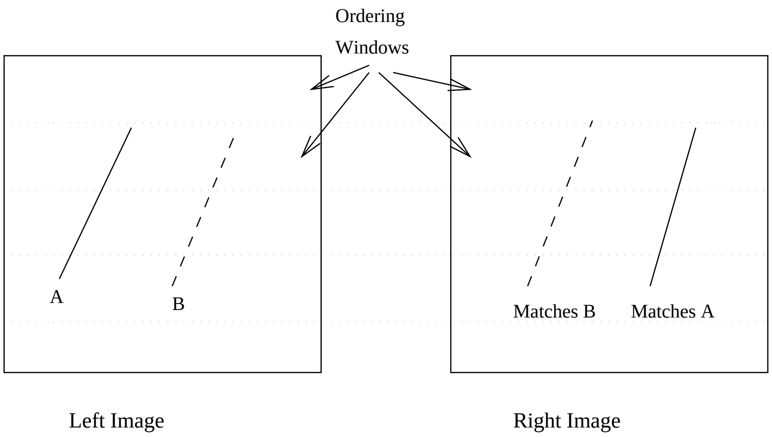



Figure 5.3: The ordering constraint: The matching shown violates the ordering constraint.

An extra constraint which is also enforced is to accept matches only if the orientation of the major axis of each edge differs by no more than 30°. The method is described more thoroughly in Algorithm 5.2.3.

In Algorithm 5.2.3, the function *extract_attributes* takes as input a number of edges, for each of the edges, the attributes described above (number of pixels, coordinates of centroid, first and second moments of the distance of each pixel from the centroid, the orientation of the major axis) are computed and are returned as a list of points in a multi-dimensional attribute space. The function *calculate_edge_match* is described in Algorithm 5.2.5, and the function *split_edge_lowe* is described in Algorithm 3.1. The function *calculate_disparities* accepts two matching edges as input and calculates the disparities between the two edges on a pixel-by-pixel basis, and returns a list of disparity values. The ratio of 0.7 in Algorithm 5.2.3 refers to the probability of a zero-crossing having at least one candidate match (Marr and Poggio [78]). Subsequently, Grimson ([53]) accepts a region of matches if the ratio of matched points is greater than 0.7. Thus, the ratio has been used as a reasonable measure of matching success along an edge contour.

### 5.2.4 Disparity Estimation

As discussed in the previous section, one of the attributes used for the matching of the edges is the coordinates of the centroid of the edge contour. It is also noted that the x-coordinate of the edge contours in the right image are shifted by a constant ($\Delta x$), to bring these edges into alignment with the edges in the left image. Obtaining the estimate of the disparity is the subject of this section.

Depending on the image, Algorithm 5.2.3 can be quite sensitive to the accuracy of the disparity estimate, which defeats the purpose of using the algorithm. An allowable disparity range is one of the input parameters to the algorithm, so the initial disparity is estimated as the average of the minimum and maximum disparities. In a practical situation, however, it may not be possible to accurately estimate the minimum and maximum disparity within a scene, and an average of these estimates may not reflect the average depth of the objects in the scene. Hence, it is important to be able to automatically estimate the disparity to be used by the algorithm.

To achieve this, a pre-match loop of Algorithm 5.2.3 is used with a percentage of the longest edges (25%) to more accurately estimate the approximate disparity. A more accurate disparity estimate is calculated as the average of the disparities found in this initial stage. This loop is continued until the disparity estimate converges, which required at most three passes with all the images tested. It is felt that the matching of large edges is likely to be more accurate and less ambiguous than matching smaller edges, and hence should lead to an accurate estimate of the average disparity in the images. An alternative approach is to estimate the disparity by performing a stereo match at a larger scale (which contains less edges), but since the current technique works well, this was not used here. Algorithm 5.2.3 is then re-executed using all the edges (including the edges involved in the initial matches, these matches are discarded) with the new disparity estimate. The disparity estimate is recomputed after each pass of the algorithm.

**Algorithm 5.2.1** *Matching Edges in Attribute Space*

**function** *stereo_match*($E_L$, $E_R$ : list of edges from left and right images)
**variables:**
  $F_L$, $F_R$ /* attribute spaces of edges */
  $P_L^i$, $P_R^j$ /* points in the attribute spaces $F_L$, and $F_R$ */
  $NN$ /* nearest neighbour mapping between $F_L$ and $F_R$ */
  $E_L^i$, $E_R^j$ /* $i^{th}$ and $j^{th}$ edge of edge sets $E_L$ and $E_R$ */
  $m(e)$ /* number of matched points in edge e */
  $n(e)$ /* number of points in edge e */
  $M_L$, $M_R$ /* set of matched edges in left and right images */
  $D$ /* set of disparities */
**endvariables**
**begin**
 **for** $i \leftarrow 1$ **to** *NUM_ITERATIONS* **do**
  $F_L \leftarrow$ *extract_attributes*($E_L$)
  $F_R \leftarrow$ *extract_attributes*($E_R$)
  /* find a mapping that minimizes the sum of the distances */
  Find $\{NN : F_L \leftrightarrow F_R | \min \sum_{P_L^i \in F_L,\ P_R^j \in F_R} d(P_L^i, P_R^j)\}$
  **foreach** $NN$ pair $(P_L^i, P_R^j)$ **do**
   *calculate_edge_match*($E_L^i, E_R^j, e_{L_{match}}, e_{L_{no-match}}, e_{R_{match}}, e_{R_{no-match}}$)
   **if** $(m(E_L^i)/n(E_L^i) > 0.7)$ *and* $(m(E_R^j)/n(E_R^j) > 0.7)$ /* both edges match */
    **then** /* add edges to matching sets and remove from non-matched sets */
    $M_L \leftarrow M_L \cup \{E_L^i\};\ E_L \leftarrow E_L - \{E_L^i\}$
    $M_R \leftarrow M_R \cup \{E_R^j\};\ E_R \leftarrow E_R - \{E_R^j\}$
   **elsif** $(m(E_L^i)/n(E_L^i) > 0.7)$/* left edge matches */
     **then** /* add left edge and part of right edge to matching sets*/
     $M_L \leftarrow M_L \cup \{E_L^i\};\ E_L \leftarrow E_L - \{E_L^i\}$
     $M_R \leftarrow M_R \cup \{e_{R_{match}}\};\ E_R \leftarrow E_R - \{E_R^j\} \cup \{e_{R_{no-match}}\}$
   **elsif** $(m(E_R^j)/n(E_R^j) > 0.7)$/* right edge matches */
     **then** /* add right edge and part of left edge to matching sets */
     $M_R \leftarrow M_R \cup \{E_R^j\};\ E_R \leftarrow E_R - \{E_R^j\}$
     $M_L \leftarrow M_L \cup \{e_{L_{match}}\};\ E_L \leftarrow E_L - \{E_L^i\} \cup \{e_{L_{no-match}}\}$
   **else** /* no edge matches */
     $E_L \leftarrow E_L - \{E_L^i\} \cup$ *split_edge_lowe*($E_L^i$)
     $E_R \leftarrow E_R - \{E_R^j\} \cup$ *split_edge_lowe*($E_R^j$)
   **endif**
  **endforeach**
 **endfor**
 **foreach** $e_L \in M_L$ and $e_R \in M_R$ **do**
  $D \leftarrow D \cup$ *calculate_disparities*($e_L, e_R$)
 **endforeach**
 **return**($D$)
**end**

### 5.2.5 Acceptance of Matches

Although the above procedure finds the nearest match in terms of the attributes used, this may not correspond to matches in the edge images due to different segmentation in each image or due to the lack of discrimination power of the edge attributes. Errors in segmentation occur for a number of reasons:

- Noise in the images.
- Errors in the segmentation procedure itself caused by discretisation of the images and the limited precision of floating point arithmetic.
- Differing occlusion of objects between differing views due to parallax error.
- Different lighting conditions between views causing specularities to appear in different locations.

Such errors can cause an edge in one image to appear as a number of edges in the other image, or not to show up at all. This is a major problem with using more complex primitives for stereo matching. Hence, a nearest neighbour match in attribute space, may not correspond to an actual match in edge space.

The problem is solved by performing a pixel-pixel match between each pair of nearest neighbour edges. Assuming a horizontal and parallel stereo camera setup, two pixels match if their y-coordinates are equal and their disparity is within the allowable range. Unfortunately, it is not always clear how to perform a pixel-pixel match between two edges. If figural continuity is to be maintained, then it would seem sensible if matching proceeded stepwise along each edge. However, due to segmentation differences, even in valid matches, there may exist points in either edge that have no corresponding matching point in the other edge.

To solve this problem it must be decided if one of the points doesn't have any corresponding point at all, or if both of the points have no corresponding point. Depending on the decision made, it may be necessary to step along only one of the edges (if only one point has no match) or to step along both of the edges (if neither points have a match). The decision is made here by moving along the edge for which the next matching point is closest to the current point, or if no matches are found then to step one pixel along both edges. Algorithm 5.2.5 describes the procedure in more detail.

### 5.2.6 Splitting the Edges

The edge splitting algorithm mentioned in Algorithm 5.2.3 utilises a variant of the edge segmentation algorithm (*split_edge_lowe* in the algorithm) developed by Lowe ([74]). This algorithm is described in Algorithm 3.1 in Section 3.1. In the original form of the algorithm, an edge is recursively split into linear edge segments. In this case, however, we only need to split an edge once, resulting in two edges being formed which are then used for matching at the next iteration of the edge matching algorithm.

## 5.3 Performance

The images used in testing the stereo algorithm are laboratory scenes of man made objects which is also the environment used for the entire object recognition system. A more detailed discussion of the experimental setup can be found in Chapter 9. If segmentation of the left and right images can be guaranteed to be very close, then stereo matching is greatly simplified and many algorithms will probably work equally well. However, as can be seen from the images shown in Section 5.4, even in a controlled environment, significant differences between the segmentation of the two images are evident. This is a major problem which this stereo algorithm attempts to overcome. Figures 5.4 to 5.7 illustrate the more complicated images on which the algorithm was tested.

Due to the nature of the application (for object recognition by examples, only relative depths are important as all objects are transformed in the same manner) and for the purposes of evaluating the algorithm, absolute depth values are deemed unimportant. It is sufficient and simpler to evaluate the accuracy of the algorithm in terms of disparity values. This removes the added inaccuracies of depth calculation and should give an unbiased performance criterion. Unfortunately, due to the excessively large number of possible pixel matches, it is not practical to evaluate each

**Algorithm 5.2.2** *Pixel-Pixel Edge Matching*

**function** *calculate_edge_match*$(L, R, e_{L_{match}},\ e_{L_{no-match}}, e_{R_{match}}, e_{R_{no-match}})$
**inputs:**
  $L, R$ /* edges from left and right images with N and M points */
  $e_{L_{match}}, e_{L_{no-match}}, e_{R_{match}}, e_{R_{no-match}}$
   /* return parameters: possible matching and non-matching sections of input edges */
**endinputs**
**variables:**
  $i, ii, j, jj$ /* counters for $L$ and $R$ */
  $L^i$ /* the $i^{th}$ point on edge L */
  $L^i_x, L^i_y$ /* the x and y coordinates of the $i^{th}$ point */
  *left_start*, *left_end*, *right_start*, *right_end*
   /* start and end indices of the matching edge sections */
**endvariables**
**begin**
 Find the disparity for which the maximum number of edge points match.
 Select starting points which satisfy the optimal disparity, $i, j$
 Scan in both directions from the starting point, along each edge.
 **while** $\neg$(all pixels visited) **do**
  **if** $(L^y_i = R^y_j)$ **then** /* Accept as a match */
   **if** $(i >$ *left_end*$)$ **then** *left_end* $\leftarrow i$ **endif**
   **if** $(i <$ *left_start*$)$ **then** *left_start* $\leftarrow i$ **endif**
   **if** $(j >$ *right_end*$)$ **then** *right_end* $\leftarrow j$ **endif**
   **if** $(j <$ *right_start*$)$ **then** *right_start* $\leftarrow j$ **endif**
   $i \leftarrow i + 1;\ \ j \leftarrow j + 1;$
  **elsif** $\exists\{ii | L^{ii}_y = R^j_y\} \vee \exists\{jj | R^{jj}_y = L^i_y\}$ **then**
   /* move to closest matching point */
   **if** $(abs(i - ii) < abs(j - jj))$ **then**
    $i \leftarrow ii$
   **else**
    $j \leftarrow jj$
   **endif**
  **else** /* no points match, move to next point */
   $i \leftarrow i + 1;\ j \leftarrow j + 1;$
  **endif**
 **endwhile**
 $e_{L_{match}} \leftarrow \{L^{left\_start} \ldots L^{left\_end}\}$
 $e_{L_{no-match}} \leftarrow \{L^0 \ldots L^{left\_start}\} \cup \{L^{left\_end} \ldots L^N\}$
 $e_{R_{match}} \leftarrow \{R^{right\_start} \ldots R^{right\_end}\}$
 $e_{R_{no-match}} \leftarrow \{R^0 \ldots R^{right\_start}\} \cup \{R^{right\_end} \ldots R^M\}$
**end**

match on a pixel by pixel basis. Instead, the success of a match is determined at the edge contour level and this is weighted by the number of pixels in the edge to give a good approximation of the accuracy of the pixel matches. The evaluation only deals with false positives, as it is more difficult to evaluate false negatives (points for which the algorithm hasn't found a match, but should have).

The results for matching the images displayed in Figures 5.4 to 5.7, are given in Table 5.1. As can be seen from Table 5.1, the algorithm's matching performance is good on all images ($> 85\%$). The degraded performance for Figure 5.6 is due to the number of similar and parallel edges which are matched incorrectly or not at all, in the Rubik's cube, and the noisiness in the extraction of the buttons on the telephone. The Rubik's cube problem is a difficult one as there exist many possible matches which lie within the allowable disparity range and are also close to the average disparity. One method of solution would be to employ a cooperative algorithm like that of Sherman and Peleg ([107], but this would be a computationally more expensive approach. Correctly matching the telephone buttons is extremely difficult due to the noisy edges of the buttons and in general would require an extremely good edge extractor, or a more global matching scheme. Despite these problems, the large scale structure evident in the images is generally extracted correctly especially in the less cluttered images (Figures 5.4 and 5.5).

It is also apparent from Table 5.1 that the time complexity of the algorithm is related to the complexity of the image (or more specifically, the number of edges in the images) as was expected. A thorough time complexity analysis has not been performed, and it is therefore difficult to compare the efficiency of different algorithms. In spite of this, the author believes that the timing results are very reasonable. It should be noted, however, that the timing results listed do not include the edge extraction phase.

There are several advantages of using the matching strategy described here:

- Matching is contour based and hence figural continuity along edge contours is assured.

- As there are fewer edge contours than edge pixels, the search space for the

| Image | Number matched edges | Number matched pixels | Number correct edges matched | Number correct pixels matched | % correct pixels matched | time (s) (DECstation 5000) |
|---|---|---|---|---|---|---|
| Figure 5.4 | 59 | 1225 | 49 | 1099 | 89.7 | 3.1 |
| Figure 5.5 | 78 | 1327 | 67 | 1197 | 90.2 | 3.2 |
| Figure 5.6 | 109 | 1269 | 87 | 1094 | 86.2 | 6.1 |
| Figure 5.7 | 132 | 1654 | 115 | 1522 | 92.0 | 5.4 |

Table 5.1: The recognition results for the test images

matching problem is greatly reduced and therefore the computation cost of the method is low.

- Matching is performed on the basis of contour shape as well as position, so complex curves are matched as easily (if not easier) than simple curves (eg. linear edge segments).

- Due to the curve-splitting strategy employed, occlusion differences between the two images is handled particularly well.

- Horizontal edges are treated the same as all other edges and are generally matched correctly. This provides a richer source of disparity information than many stereo algorithms.

Due to the simplicity of the algorithm, however, there are deficiencies in the matching process. Most notably:

- Although figural continuity along edge contours is assured, continuity between contours may be violated, as each contour is matched independently of other matches (aside from the uniqueness and ordering constraints).

- Although horizontal edges are matched, calculation of the disparity along each edge can cause errors, especially if the edge contours are of different lengths due to occlusion or segmentation differences between the two images. It is felt, however, that in most cases the calculated disparity values are approximately correct.

The first problem, although not solved, is somewhat minimised by enforcing the ordering constraint between matches. A full solution to this problem may require global optimisation techniques, which are typically computationally expensive. As far as the author is aware, a more reasonable solution to the second problem without higher level reasoning techniques may not exist.

It remains to be seen if the depth data extracted using the stereo algorithm aids the recognition of objects. The usefulness of the stereo data will be evaluated in Chapter 10.

## 5.4 Images

The following images were used to test the stereo algorithm described above. The images are labelled in the following manner:(a) left image; (b) right image; (c) left edges; (d) right edges; and (e) depth map. For the depth maps, the brighter the pixel, the closer it is to the viewer.

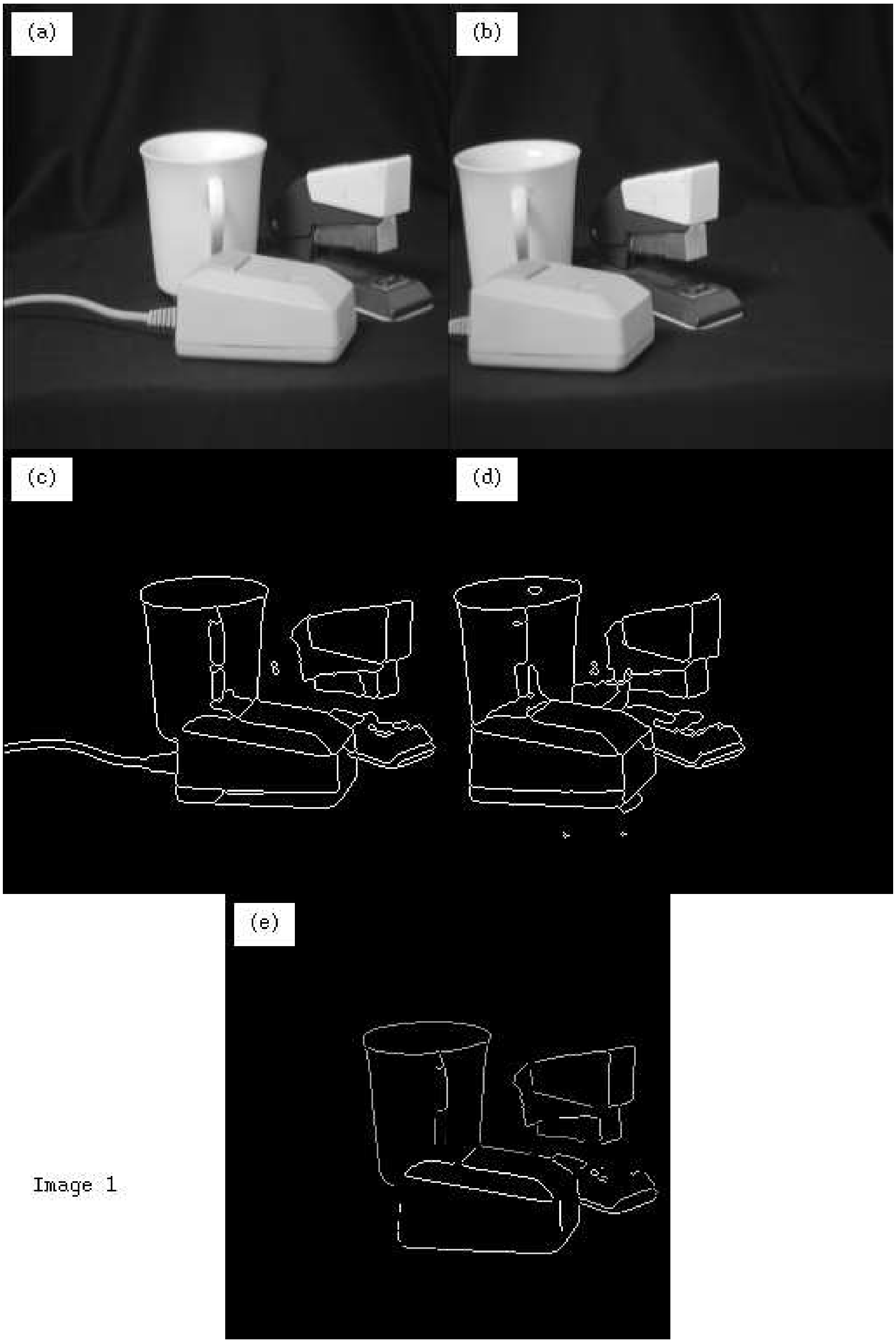



Figure 5.4: An example of the stereo algorithm.

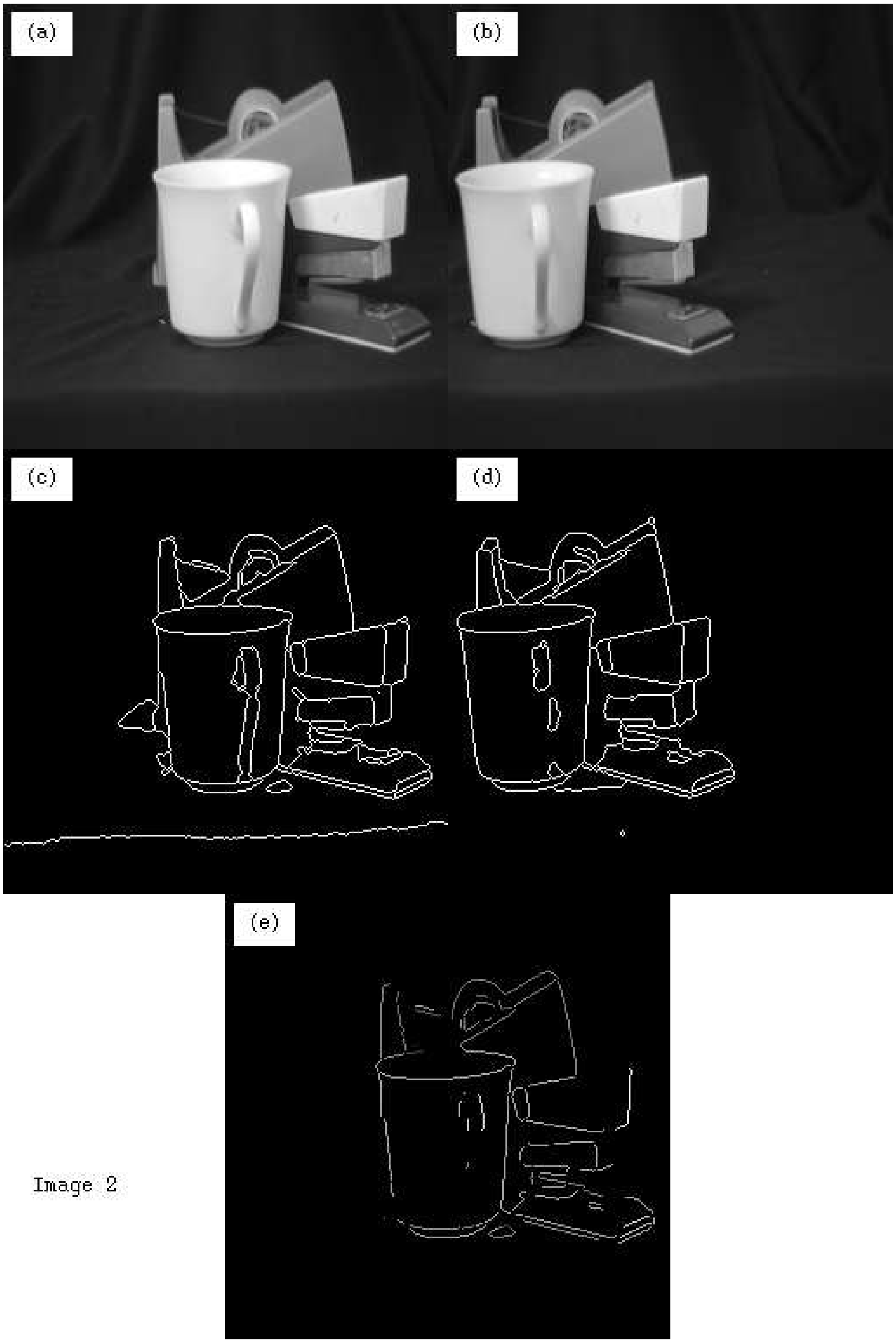



Figure 5.5: An example of the stereo algorithm.

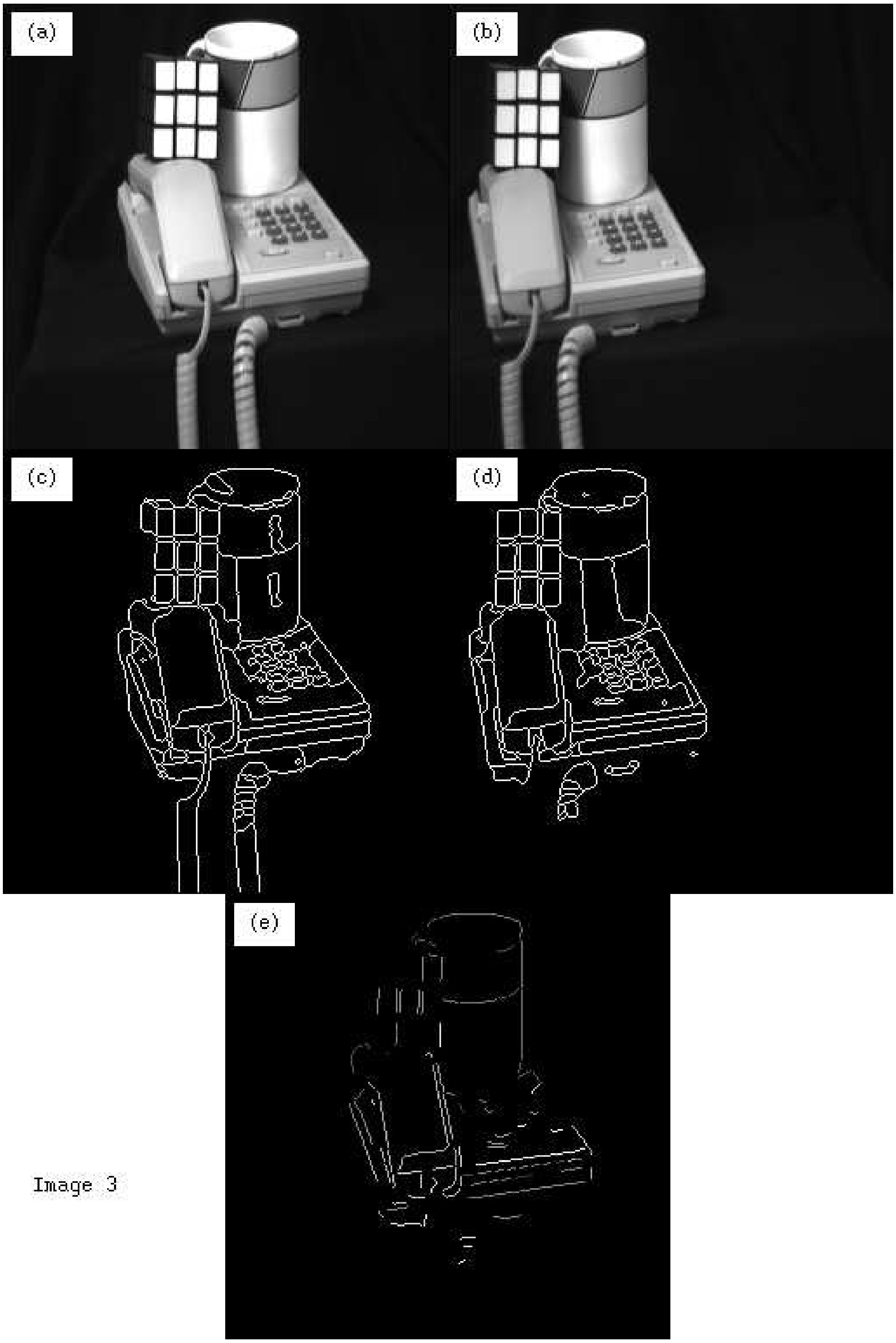



Figure 5.6: An example of the stereo algorithm.

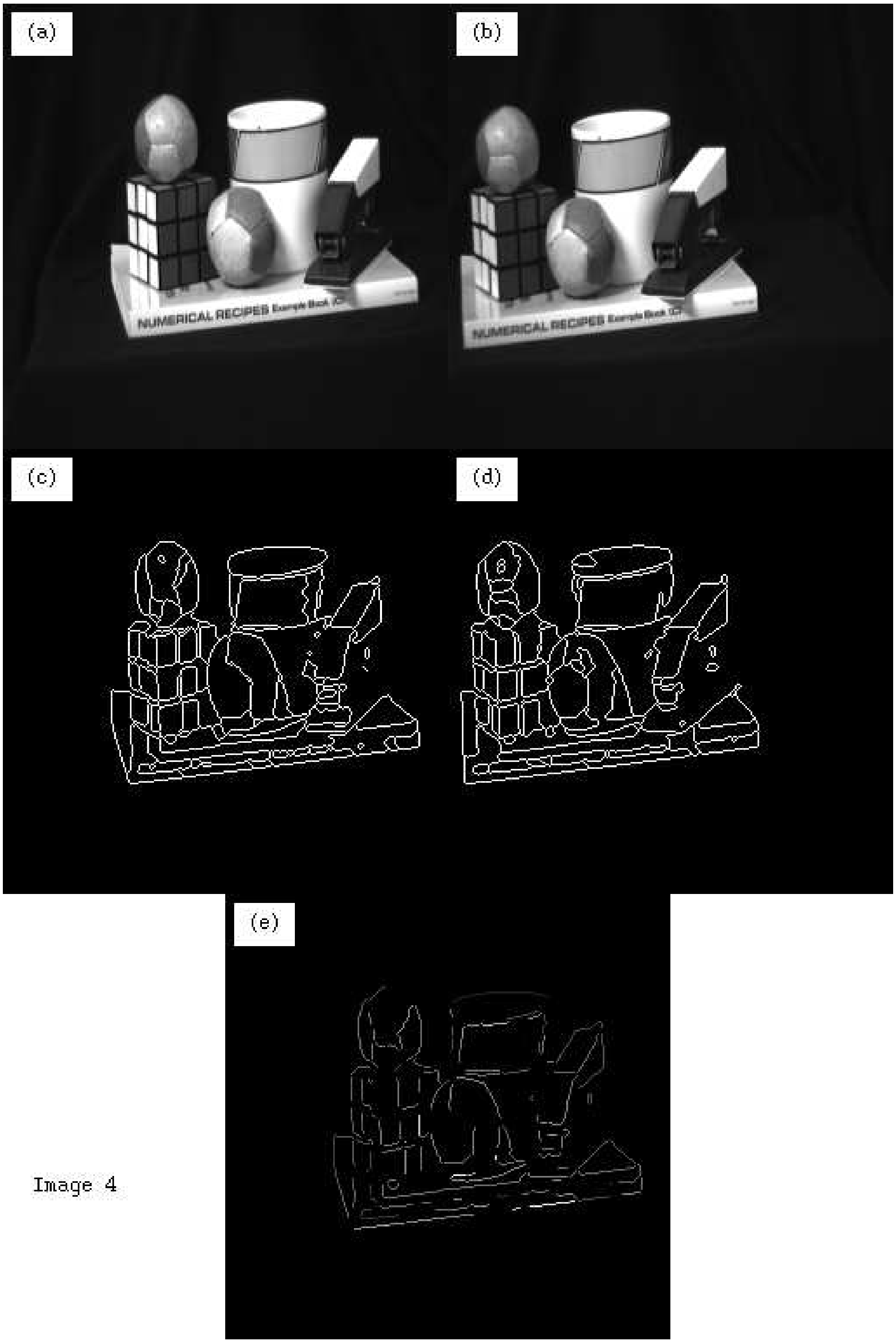



Figure 5.7: An example of the stereo algorithm.

# Chapter 6

# Part Attributes

The regions formed by the segmentation algorithm are treated as the basic parts of the objects to be learned/recognised. According to the parts-and-relations paradigm used in this system, attributes measured from these parts and between neighbouring parts are required to represent each of the objects. These attributes are often either unary (part measurements) or binary (relational measurements) attributes. After image segmentation, attributes are extracted from all parts found in the image, and this is done across all images in the image database. A discussion of the images and objects used in the experiments is given in Chapter 9. A discussion of a number of different sets of unary and binary attributes will be given in the following sections along with some general comments about these sets. The attributes have been evaluated for the current application, and the results of this evaluation are given in Chapter 10.

## 6.1 Unary attributes

In general, unary attributes can be grouped into two distinct types:

1. Reconstructive: the parts can be reconstructed (at least approximately), from the attributes.

2. Descriptive: the attributes are a general description of the parts, but the parts cannot be reconstructed from the attributes.

Here, both types of attributes are evaluated for use in an object recognition system. Attributes of the first type include: Fourier, sinusoidal and Hadamard descriptors. Attributes of the second type include: Hu invariants, arboricity measures, and general shape descriptors. The following subsections will describe each of these attributes.

### 6.1.1 Hu Invariants

There has been significant research effort into the development of moment invariants which were first introduced by Hu [62], and have been further studied by various researchers including: Maitra [76], Abu-Mostafa and Psaltis [1, 2], Teague [112], and Balkasim, Shridhar and Ahmadi [7]. The basic idea of moment invariants is to define a set of measures which are invariant to scale, rotation, translation, and contrast changes in a 2D plane. The justification for using them in the current context is that they are valid for recognising a part from a particular pose which may be rotated, translated or scaled from the original training pose. Obviously, the invariants will have different values from different poses, but hopefully they will be similar enough for similar poses to be grouped together. The following definitions have been taken from Maitra [76].

Given a 2D image intensity distribution $f(x, y)$, the moments of this function are defined as:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy, \quad \text{for } p, q = 0, 1, 2, ... \tag{6.1}$$

The above invariants can be modified to include translational invariance in the following way:

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y)dxdy, \tag{6.2}$$

where $\bar{x} = \frac{m_{10}}{m_{00}}$, and $\bar{y} = \frac{m_{01}}{m_{00}}$. Scale invariant moments can be derived from the above to give a set of normalised central moments:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \text{ where } \gamma = \frac{p+q}{2} + 1. \tag{6.3}$$

A set of 7 functions can then be defined which are invariant to translation, rotation, and scale changes in the image plane:

$$\begin{aligned}
\phi(1) &= \eta_{20} + \eta_{02} && (6.4)\\
\phi(2) &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 && (6.5)\\
\phi(3) &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 && (6.6)\\
\phi(4) &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 && (6.7)\\
\phi(5) &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta 12)^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (6.8)\\
\phi(6) &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) && (6.9)\\
\phi(7) &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. && (6.10)
\end{aligned}$$

Contrast invariance can also be included (see [76]), but a discussion of contrast invariance will not be included here.

Figures 6.1 and 6.2 display plots of each of the 7 moment invariants across all parts of all objects used in the experiment. The plots are scatter plots where the $y$-axis indicates the value of the attribute. The $x$-axis has no meaning, but points are spread out along the $x$-axis to give some indication of point density along the $y$-axis. As can be seen from the plots, only $\phi(1)$ and $\phi(2)$ show much variation across different parts, indicating that shapes which are visually very different, will probably have similar attribute states for attributes $\phi(3)$ to $\phi(7)$. Hence, these attributes will have little use for discriminating between different shapes.

An interesting point can be made about the moment invariants described above. Clearly they are not independent quantities since a number of the invariants contain the same combinations of the base moments. This presumably reduces the recognition power of the invariants. Figures 6.3 and 6.4 display two x-y plots which demonstrate the correlation between some of the invariants. In fact, subtracting $\phi(2)$ from $\phi(1)^2$ results in the following:

$$\phi(2) = \phi(1)^2 - 4\eta_{20} - 4\eta_{11}^2 \qquad (6.11)$$

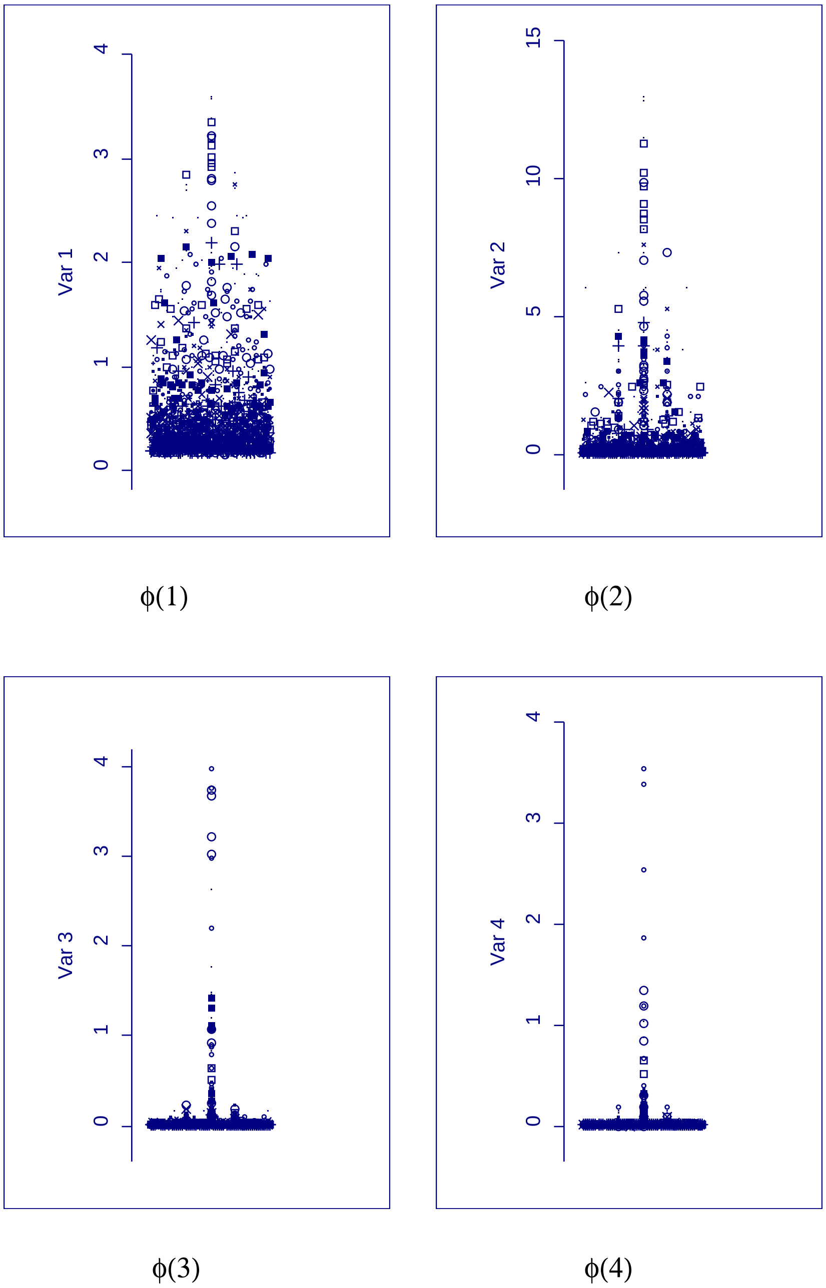



Figure 6.1: Scatterplots of $\phi(1)$ to $\phi(4)$.

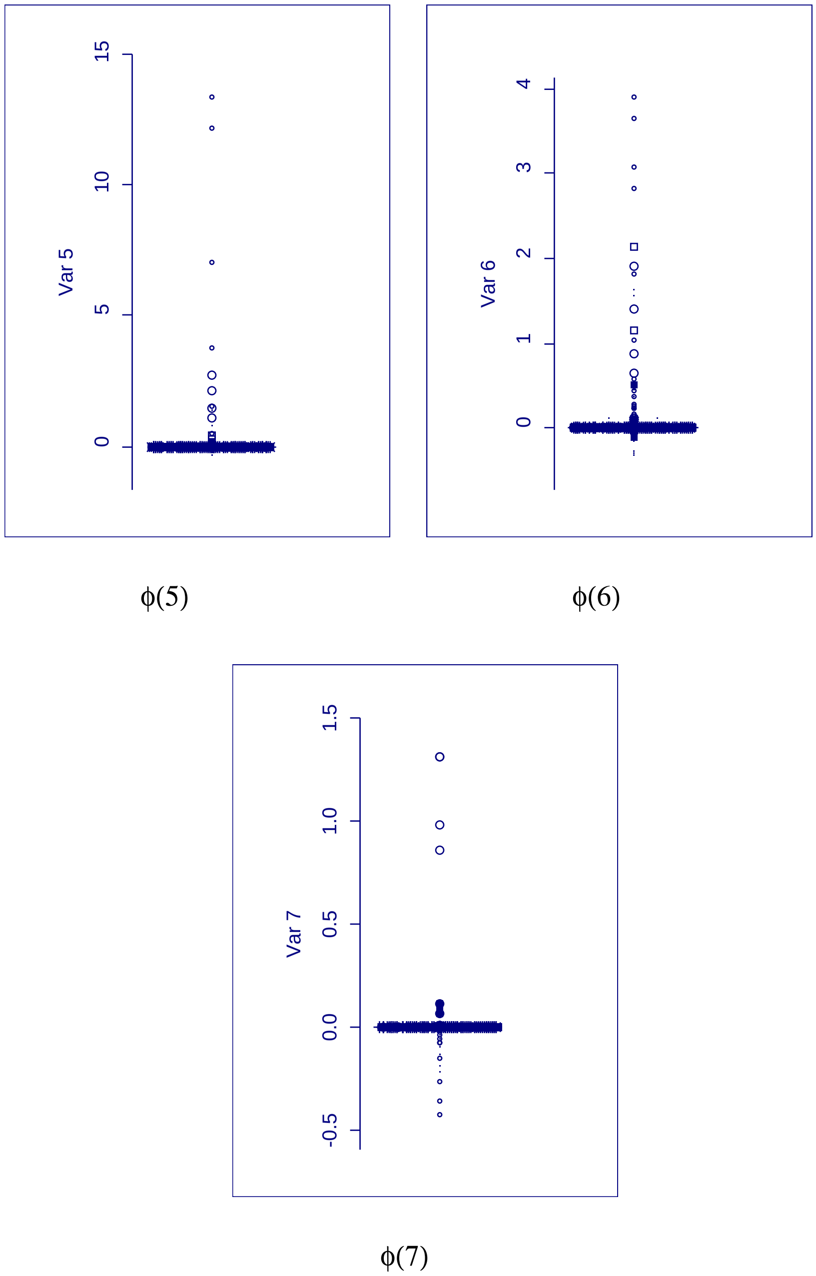



Figure 6.2: Scatterplots of $\phi(5)$ to $\phi(5)$.

indicating that a some type of squared relationship exists between $\phi(1)$ and $\phi(2)$. It would seem likely that some of these quantities are redundant, at least for the experiments performed here, and hence, only one of the correlated variables would be useful for recognition. A discussion of the recognition results using these invariants in given in Chapter 10.

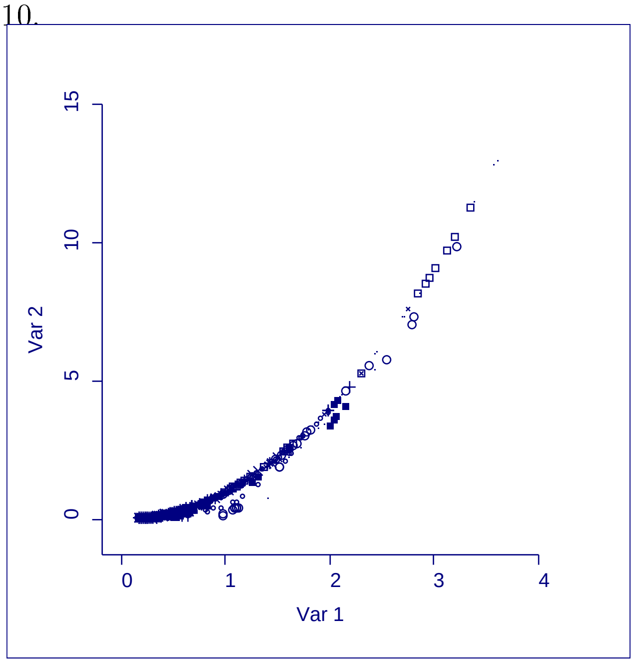



Figure 6.3: x-y plot of $\phi(1)$ versus $\phi(2)$.

### 6.1.2 Some General Shape Descriptors

An alternative approach to using Hu invariants is to use measures which are based on the general shape of specific parts. In this way, we should be able to discriminate between parts which exhibit different gross shapes (eg circle versus an elongated rectangle), but not necessarily between similar shapes (eg circle versus a regular hexagon). This is exactly the effect that is required, and should prove useful even under minor variations of input data (eg caused by small 3D rotations of objects).

Many of the attributes extracted rely on the extraction of the major and minor axes of a region. These are extracted by calculating the two principal components of the region $(\vec{v}_1, \vec{v}_2)$. This is achieved by treating each point in the region as a 2D random variable, and calculating the eigenvectors of the covariance matrix of the population ([51]). This technique is commonly referred to as the eigenvector,

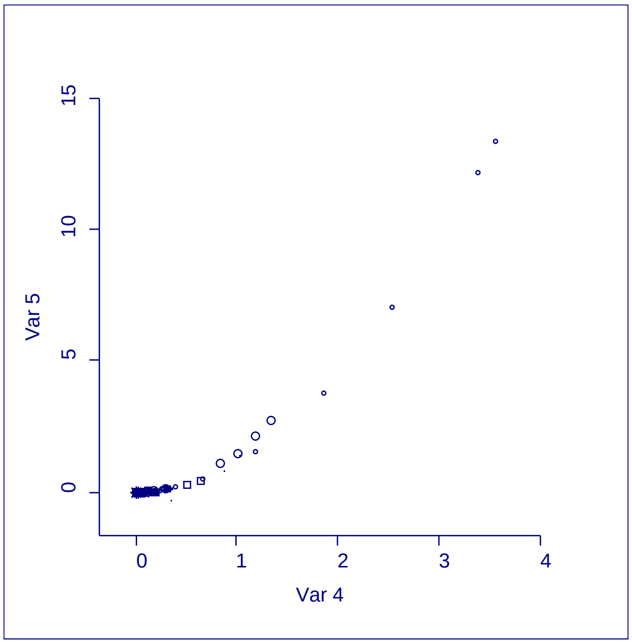



Figure 6.4: x-y plot of $\phi(4)$ versus $\phi(5)$.

principal component, or discrete Karhunen-Loeve transform. We have chosen the following region attributes derived from such analysis:

- Aspect ratio.
- Symmetry measures.
- Compactness ($perimeter^2/area$).
- Convex deficiency.

The aspect ratio is calculated as the ratio of the length of the minor axis to the length of the major axis. The symmetry measure is calculated as the mean and standard deviation of the difference between the distances from the major axis and the point pairs on the boundary perpendicular to the major axis:

$$s = \frac{1}{N}\sum_{k=1}^{N}(d(\vec{A}_i, \vec{P}_{i1}) - d(\vec{A}_i, \vec{P}_{i2})), \tag{6.12}$$

where $N$ is the number of pixels along the major axis $A$, $d(\vec{x}, \vec{y})$ is a Euclidean distance measure, $\vec{A}_i$ is the $i$th point along the major axis $A$, and $\vec{P}_{i1}$ and $\vec{P}_{i2}$ are the two points which lie along the perimeter of the region $P$ and lie on a line which passes through $\vec{A}_i$ and is perpendicular to $A$. Such a measure is only valid for regions for which the following is true: for each point on the major axis, there exist at most two points at which the line perpendicular to the major axis, and passing through the point, intersects with the region boundary. Fortunately, most regions in the current experiments satisfy this condition and for those that do not, the two points furthest away from the axis are chosen for the symmetry calculation. Figure 6.5 displays the meaning of the symmetry measure. The compactness of the region

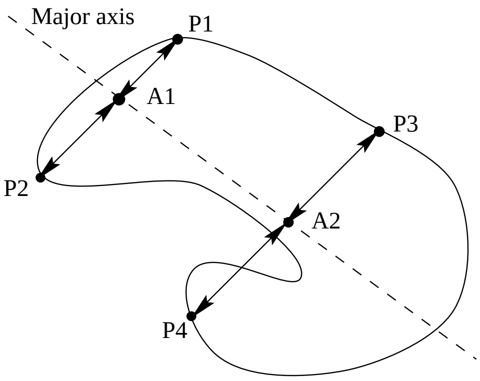


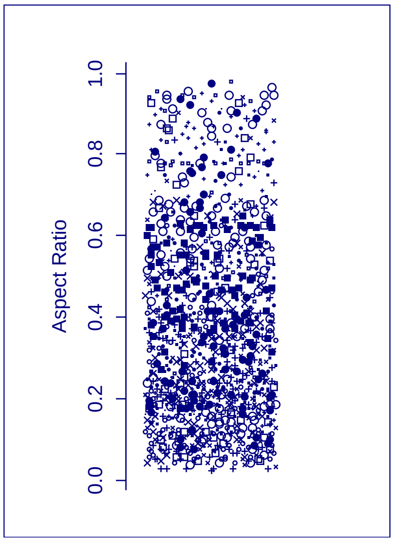

Figure 6.5: The symmetry measure: This figure displays the meaning of the symmetry measure. In the simple areas of the region the definition is obvious, in the more complex areas, the outside points are used for measuring the symmetry.

provides a measure of how close the region is to a circle. The convex deficiency of a region is calculated as:

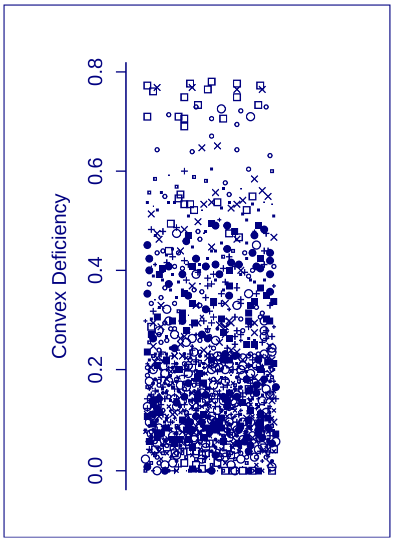
$$\frac{\mathcal{A}(C) - \mathcal{A}(R)}{\mathcal{A}(C)} \tag{6.13}$$

where $R$ and $C$ represent the region and its convex hull respectively, and the function $\mathcal{A}$ represents the area of a given region. The convex hull was calculated using an algorithm due to Graham ([52], [97]). This attribute provides a measure of the convexity of a region.

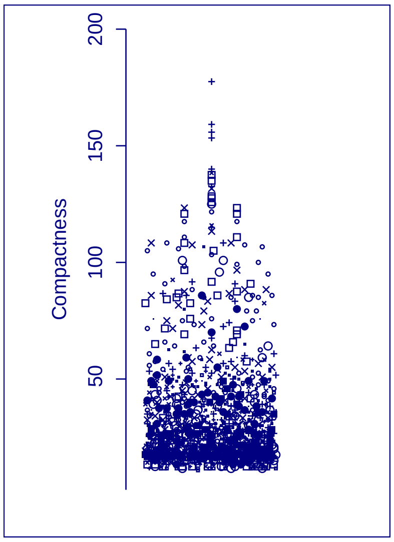
These unary attributes were chosen because they can be used to discriminate gross differences in part structure whilst being relatively invariant, at least over a number of different viewpoints. Figures 6.6 to 6.7 display the scatterplots for each of the attributes across the entire data set. As can be seen from the plots, the values are well spread across the data, and although nothing can be directly said about their discriminating power, a good distribution of data is generally a necessary condition for discriminating attributes.




Aspect Ratio

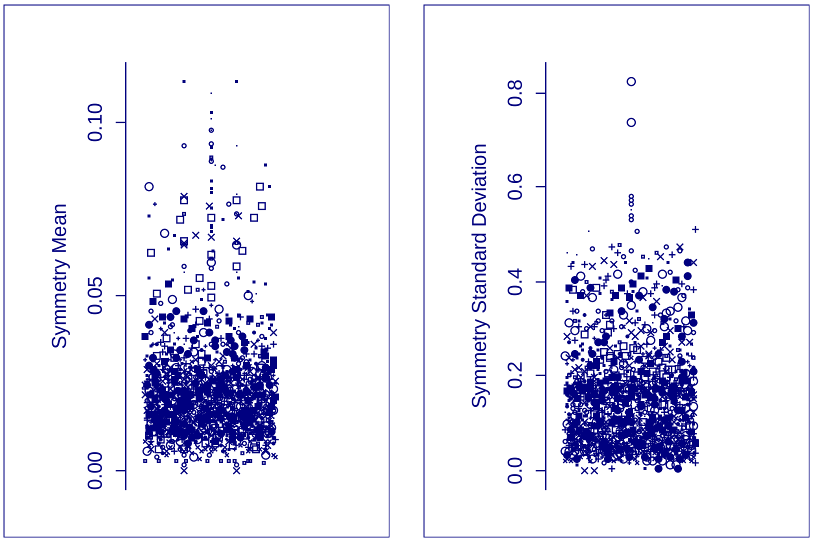



Convex Deficiency




Compactness

Figure 6.6: Scatterplot of aspect ratio, convex deficiency and compactness.




Symmetry Mean Symmetry Variance

Figure 6.7: Scatterplot of symmetry mean and standard deviation.

Another indicator for the usefulness of the attributes described in this section is that only two are correlated to any degree. This is the aspect ratio and the compactness measures. Figure 6.8 displays a 2D plot of these two attributes. The correlation evident between the two measures is expected since the aspect ratio and compactness are closely related for convex regions - as the aspect ratio increases, the compactness decreases. However, the relationship is not true for non-convex regions, and although the convex deficiency attribute helps to discriminate between non-convex regions, the aspect ratio and compactness measures can also contribute.

### 6.1.3 Arboricity Measures

There are many variations that can be made to the general shape attributes approach in the hope of improving the recognition performance of the system. One such extension is the use of arboricity measures. Arboricity measures attempt to capture more complex symmetrical features which may occur in some shapes. They rely heavily on the notion of symmetrical axes evident within a shape and in this case are calculated from the principal axis of a region which is extracted as described

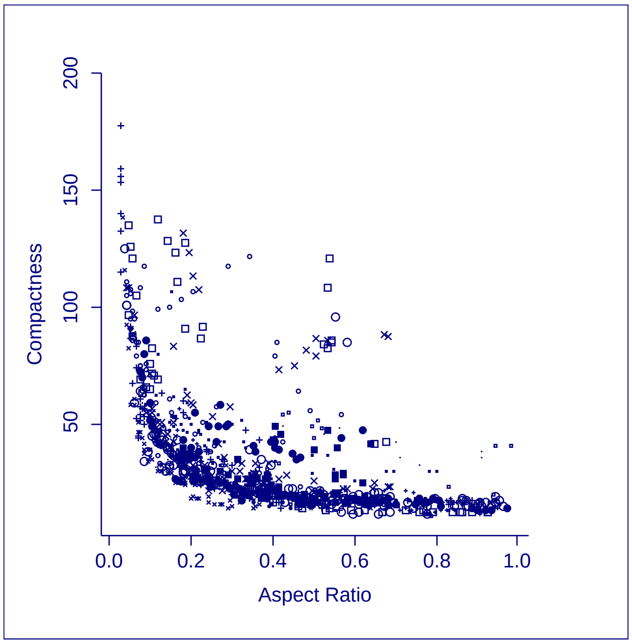



Figure 6.8: x-y plot of aspect ratio versus compactness.

above. After attributes have been extracted from the whole region, it is split into two regions (perpendicular to the principal axis) and further measurements are made on the regions thus created. Figure 6.9 depicts the essence of the idea.

The attributes used for each of the stages are the same as those described above in Section 6.1.2, thus hopefully providing a more powerful representation of each part. The distributions of each attribute are the same as those reported in Section 6.1.2, so these plots will not be repeated here. The question still remains about how many levels to split the parts into (all parts must be split the same amount to give comparable representations across different parts). Care must be taken here as using too many splits quickly introduces the "curse of dimensionality". With 5 base attributes, even 1 splitting level results in 15 attributes - 5 for the top level and 5 for each of the two sub-levels. With two levels the number of attributes is 35, so parts should definitely not be split more than 2 levels. Intuitively, this would seem to be enough to represent most shapes (since the shapes typically are not too complex).

### 6.1.4 Reconstructionist Approaches

An alternative to the descriptive methods described above are reconstructive measures. There are a number of ways in which a shape can be explicitly represented -

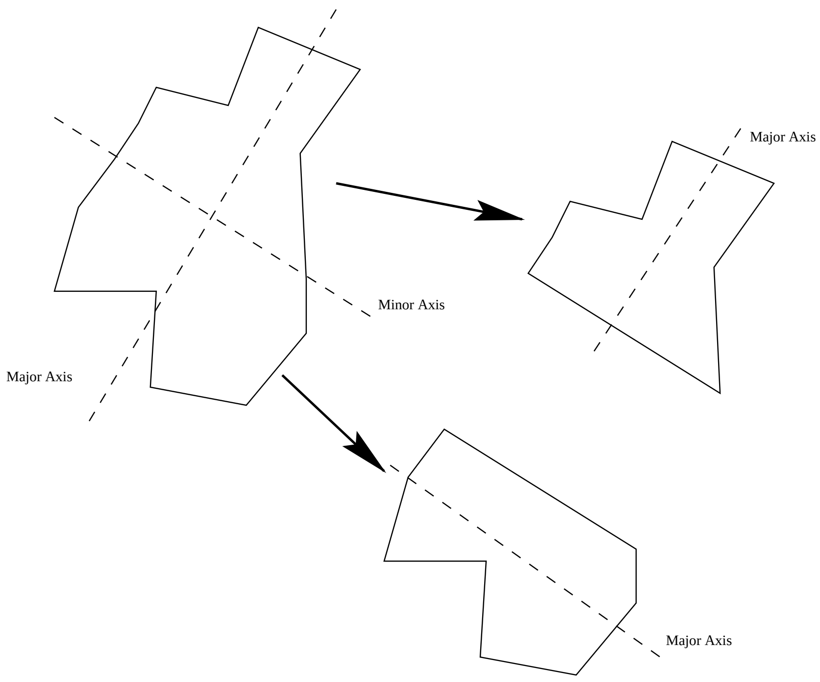



Figure 6.9: Arboricity measurements: measurements are taken initially about the major axis, the region is then split into two along the minor axis. For the two subregions, the major axis is computed and more measurements are taken. This procedure can continue for as long as required in a recursive manner.

most notably via region or boundary properties. 2D shapes (which is what we are dealing with here) are more compactly represented using boundary descriptors as boundaries typically contain less redundant information than whole areas. Boundaries can be represented via a curvature function such as the cumulative curvature function. Here, points along the boundary of a curve are represented by the difference between the tangent at the current point and the tangent at the starting point on the curve (wherever that is defined). Figure 6.10 displays the definition of cumulative curvature. Tangents can be calculated from discrete data by measuring the angle made by the current point and a point on the curve $k$ points in front or behind the current point.

Figure 6.11 displays an example region and Figure 6.12 displays its corresponding cumulative curvature function. In this example, the curve is closed (which will be true for all curves studied here since they are the boundaries of regions), starting at the minimum y-value and continuing in a clockwise direction around the boundary, resulting in a net cumulative angle of $-2\pi$. The x-axis has also been normalised to

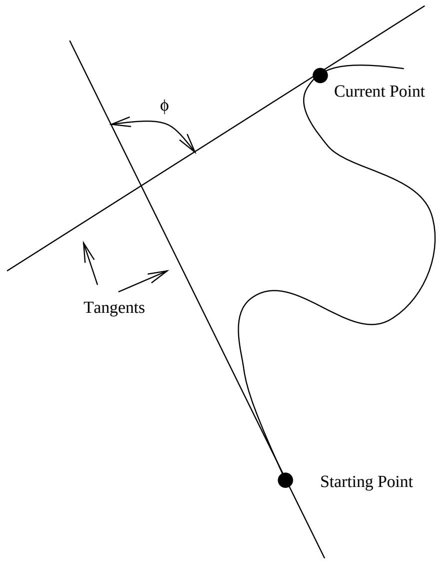



Figure 6.10: Definition of Cumulative Curvature.

$2\pi$ to aid in the visualisation of the curve, and x-axis normalisation also makes the representation scale independent. As can be seen from Figure 6.12, the curvature function is very block-like. That is, it is mostly constant along areas of the curve which are mostly straight lines, with large jumps at points of high curvature (corner points). The standard method for approximating curves of this nature is to use Fourier descriptors and this has been described by Zahn and Roskies ([125]), and Bennett and Macdonald ([8]) among others. Nevertheless, other methods of approximation also exist, including Hadamard descriptors, and sinusoidal descriptors, which will also be discussed.

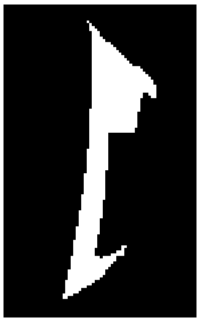


Figure 6.11: An example region.

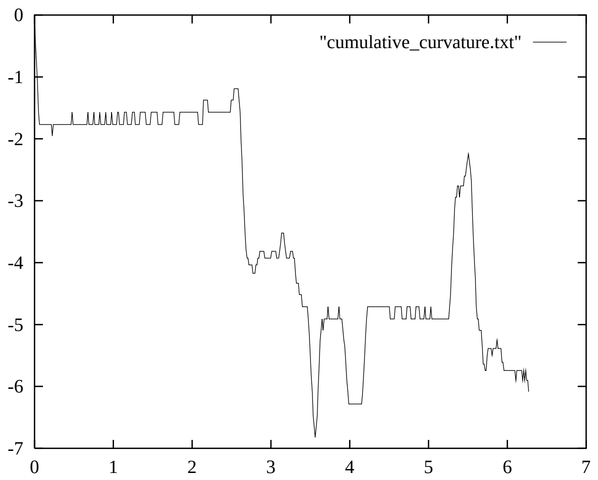



Figure 6.12: An example of the cumulative curvature function.

### 6.1.5 Fourier Descriptors

The idea of using Fourier descriptors for representing the curve stems from the Fourier transform. In the discrete domain the Fourier transform can be written as:

$$F_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{2\pi ikn/N} \tag{6.14}$$

where $i = \sqrt{-1}$, and $n$ is defined over the same range as $k$. The idea of using Fourier descriptors is to truncate the Fourier series down to the first few coefficients, and use these to represent the curve. The curve can be reconstructed from the $L$ lower order coefficients using the following formula:

$$f_k = \sum_{n=0}^{L-1} F_n e^{-2\pi ikn/N} \tag{6.15}$$

It should be noted that all the values of $f_k$ and $F_n$ are complex numbers, so if $L$ coefficients are used to represent a curve, then $2L$ attributes will be needed (the real and imaginary parts of the $F_n$s). To obtain the initial sequence $f_k$, the cumulative curvature function is sampled at 256 data points along the curve. If there are less than 256 points in the curve, then linear interpolation is used to fill in the gaps.

Figure 6.13 display the reconstruction of the cumulative curvature function using 4, 8, 16 and 64 coefficients. The reconstruction of the curve gives us some idea of how good the curve can be represented using Fourier descriptors. As can be seen from the figures, using as few as 4 coefficients could be useful for recognition, but little seems to be gained by using greater than 16. In fact 8 coefficients (16 attributes) is probably sufficient as with more coefficients the problem of the "curse of dimensionality" becomes more pronounced. It is also evident from the plots that some amplitude information is lost with the lower order frequency components - this should not be a major problem as most of the gross shape of the function is preserved.

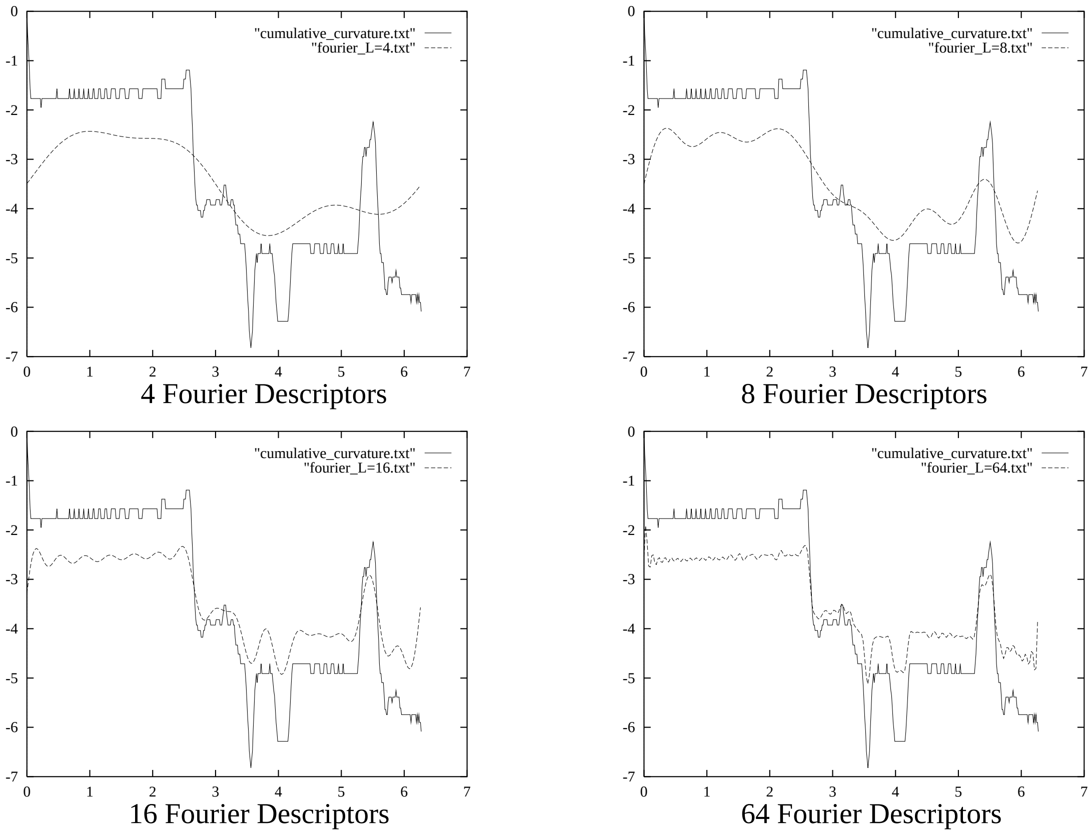



Figure 6.13: Fourier descriptors: The original and reconstructed cumulative curvature function using 4, 8, 16, and 64 Fourier descriptors

The Fourier descriptors extracted from the curves are fairly well spread across the allowable range and in general are not highly correlated with each other, with one exception. The imaginary part of $F_0$ of a real function is, by definition, always 0. Since the cumulative curvature function is a real function, this descriptor is omitted from the attributes used in the experiments.

### 6.1.6 Sinusoidal Descriptors

There are a number of problems with using the lower order Fourier descriptors for representing the cumulative curvature function. Firstly, as can be seen from Figure 6.13, considerable amplitude information is lost. Secondly, there is some redundancy in using Fourier descriptors to represent real functions - the function is transformed from a real function containing $N$ information points to a complex function containing $2N$ information points. These problems can be overcome by utilising the sine transform which was introduced by Jain [65]. The sine transform (forward and inverse) is defined by:

$$S_n = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} \sin\left[\frac{(n+1)(k+1)\pi}{N+1}\right] \tag{6.16}$$

The original curve can be reconstructed from the $L$ lower order sine descriptors in a similar way to Equation 6.15. Figure 6.14 displays the reconstruction of the curve using the lower order sine descriptors. It should be noted here that 8 sine descriptors involves using the same number of attributes as 4 Fourier descriptors, since the Fourier descriptors are complex numbers (thus requiring two real numbers to represent them) whereas the sine descriptors are real numbers. Taking these facts into account, it is obvious from the figures that the sine descriptors approximate the cumulative curvature function much better than using the corresponding number of Fourier descriptors.

It is worth remarking that the sine transform was arbitrarily chosen in preference to the related cosine transform ([3]). However, a few points about the relative advantages of each transform are pertinent. Firstly, it has been shown by Yip and Rao ([122]) that the discrete sine transform can be computed more efficiently than the discrete cosine transform, which would be of importance if building a real time system. However, in general, the discrete cosine transform performs better than the discrete sine transform in terms of decorrelating a signal based on a first order Markov random process ([123]), and as such has become a common method for image and signal compression ([51], pg. 144). In light of these facts, it would be interesting to evaluate the effectiveness of the discrete cosine transform in terms of recognition performance, and compare it to the other attribute sets discussed here.

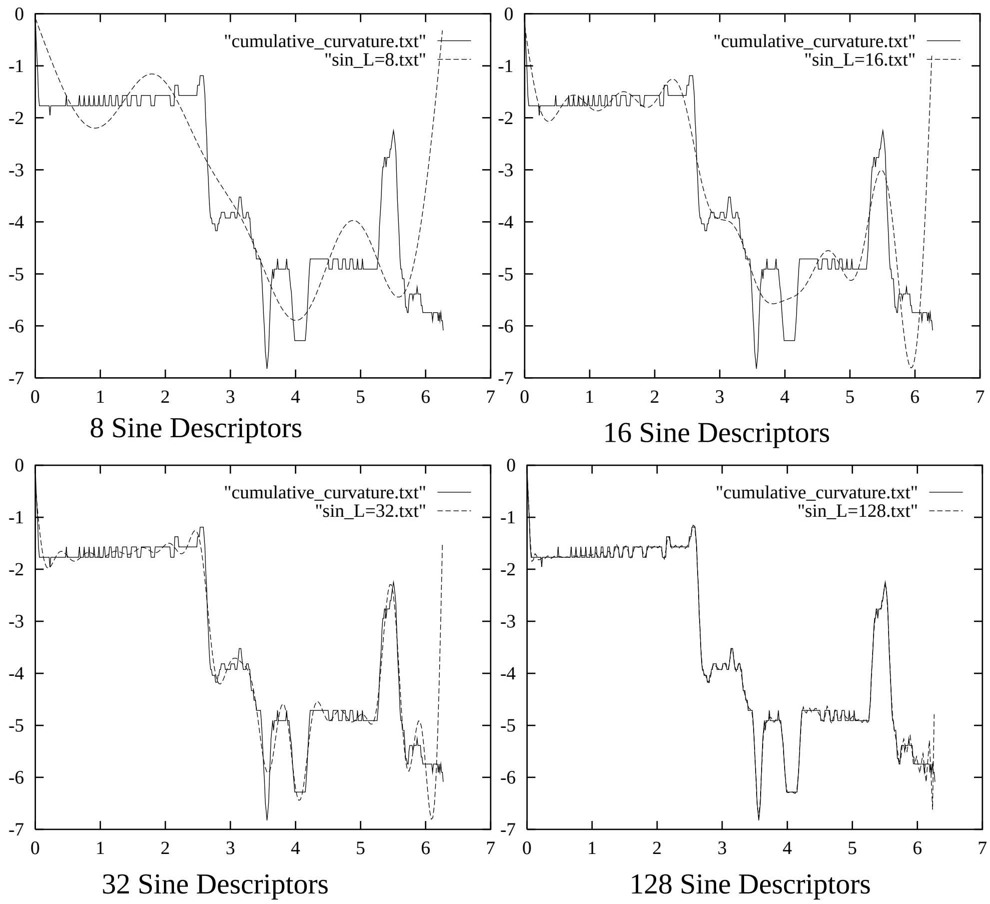



Figure 6.14: Sine descriptors: The original and reconstructed cumulative curvature function using 8, 16, 32, and 128 sine descriptors

This has not been done, however, and remains a possible area of future work.

### 6.1.7 Hadamard Descriptors

As can be seen from the plots displayed above, the cumulative curvature function for shapes with mostly straight edges (which are overwhelmingly evident in the current data set), are predominantly step-like in nature. That is, they are characterised by curves which are constant in all areas except near corner points, where sharp discontinuities occur. Keeping this fact in mind, it would be more appropriate to approximate such a function using square waves rather than sines and cosines as in the Fourier transform. One such transform is called the Hadamard Transform. A good description of the Hadamard Transform is given in Pratt ([95], pg 209-212). The Hadamard transform can be used to perform the decomposition of a function into a set of rectangular waveforms and is defined by the following equation:

$$H(u) = \frac{1}{N} \sum_{k=0}^{N-1} h(k)(-1)^{q(k,u)} \tag{6.17}$$

where

$$q(k, u) = \sum_{i=0}^{n} g_i(u) j_i \tag{6.18}$$

and

$$g_0(u) = u_{n-1} \tag{6.19}$$

and

$$g_l(u) = u_{n-l} + u_{n-l-1}; \qquad l > 0 \tag{6.20}$$

and $x_i$ refers to the $i^{th}$ bit of the binary representation of $x$. The inverse Hadamard transform is also computed using equation 6.17.

Figure 6.15 displays the reconstruction of the cumulative curvature function using Hadamard descriptors with a number of different values of $L$. As can be seen from the plots displayed in these figures, the Hadamard descriptors are performing at least as well as the sine descriptors and probably marginally better, in terms of reconstructing the curvature function. It remains to be seen which of attributes perform best in terms of recognition performance.

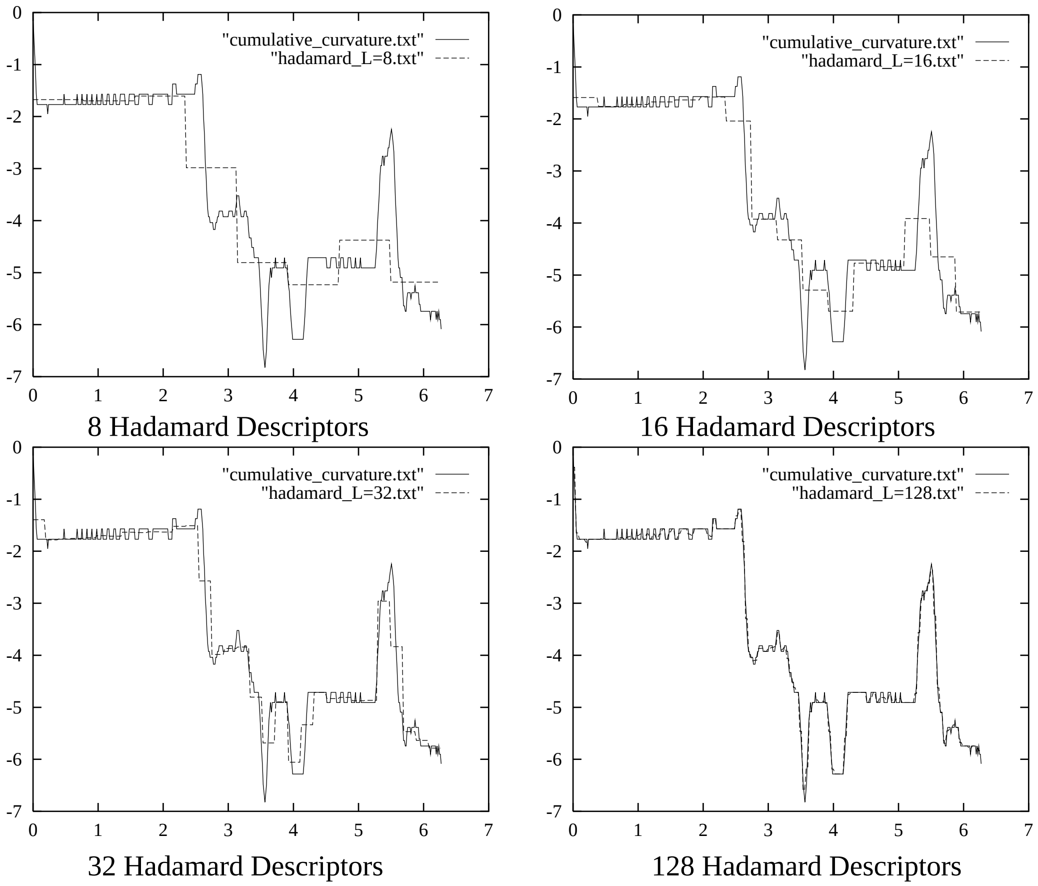



Figure 6.15: Hadamard descriptors: The original and reconstructed cumulative curvature function using 8, 16, 32 and 128 Hadamard descriptors

## 6.2 Binary Attributes

Although the unary attributes are used to discriminate between different types of regions, quite often it is the relationships between the regions which identifies a particular class. This is especially true when considering man-made objects. Most parts of such objects can be described from a limited library of shapes (circle, quadrilateral, triangle, ellipse etc) and will tend to produce similar unary attributes. Hence, extracting binary relationships between object parts is important for performing successful object recognition. It would be useful to use binary attributes which uniquely characterise this relationship as closely as possible whilst maintaining invariant properties. In this implementation we have used:

- Relative position of one region with respect to the other.
- Relative orientation of the major axes.
- Relative size of the two regions.
- Percentage overlap of the boundaries of the two regions.
- Maximum 2D distance between the two regions.
- Normalised maximum 3D distance between the two regions.
- Normalised minimum 3D distance between the two regions.
- 3D angle difference.

The relative position of one region with respect to another gives an indication of the distance between the two regions relative to the size of the regions, but is more descriptive as it gives values in two directions. This attribute is calculated using the centroid of the first region as the origin, the major axis as the reference axis, and the relative position, $(u, v)$, of the centroid of the second region is calculated in the following manner:

$$u = \frac{(\vec{p}_3 - \vec{p}_1) \cdot (\vec{p}_2 - \vec{p}_1)}{|\vec{p}_2 - \vec{p}_1|^2}, \tag{6.21}$$

and

$$v = \frac{(\vec{p}_3 - \vec{p}_1) \cdot (\vec{p}_2 - \vec{p}_1)^{\perp}}{|\vec{p}_2 - \vec{p}_1|^2}, \tag{6.22}$$

where $\vec{p}_1, \vec{p}_2, \vec{p}_3, \vec{p}_4$, are the centroid and boundary points along the major axis of the two regions and are displayed in Figure 6.16, and $(\vec{p}_2 - \vec{p}_1)^\perp$ is $(\vec{p}_2 - \vec{p}_1)$ rotated by 90° in the image plane. The relative orientation of the two regions is calculated using the relative orientations of the major axes in the following manner:

$$\theta = \cos^{-1}\left(\frac{(\vec{p}_2 - \vec{p}_1)\cdot(\vec{p}_4 - \vec{p}_3)}{|\vec{p}_2 - \vec{p}_1||\vec{p}_4 - \vec{p}_3|}\right). \tag{6.23}$$

These three attributes were used because they completely characterise the major axes of the two regions (except for relative size), and were shown by Burns et al [93] ([29]) to be relatively invariant over a large range of 3D affine transformations. The relative size of the regions is also affine invariant and is calculated simply as:

$$r = \frac{\mathcal{A}(R_1)}{\mathcal{A}(R_2)}, \tag{6.24}$$

where the function $\mathcal{A}$ indicates the area of the region and is represented in this implementation as the number of pixels belonging to a region. Finally, the percentage overlap of the two boundaries is calculated using the following scheme:

$$o = \frac{\mathcal{L}(\mathcal{P}(R_1) \cap \mathcal{P}(R_2))}{\mathcal{L}(\mathcal{P}(R_1))}, \tag{6.25}$$

where $\mathcal{P}(R)$ is the perimeter of a region, and $\mathcal{L}(P)$ is the length of a curve and is approximated here by the number of pixels in the curve. This attribute gives a measure of the connectedness of two regions and is potentially very important in distinguishing between differing objects.

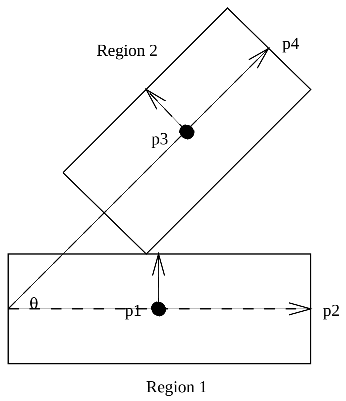



Figure 6.16: Binary attributes: Extracting binary attributes from two object parts

The maximum 2D distance between two regions is the only attribute which is not largely invariant across different images. In the current application, this is unimportant since different images are formed under the same conditions (eg same distance from the camera). However, for a generally applicable system, this attribute would need to be normalised, possible by the size of the largest region involved in the binary relationship. Alternatively, it would be possible to use the stereo data to calculate the true value of this attribute.

The final three binary attributes which have been used in this study rely on information extracted using the stereo algorithm described in Chapter 5. For each part/region in an image, the stereo data relevant to that region is mapped onto the region, including the data on the region's boundary. Theoretically, data on the region boundary may not be valid due to edge ownership problems caused by occluding surfaces ([40]), but for complex scenes it is extremely difficult to accurately determine edge ownership due to errors in the edge formation process etc. Hence, this problem has been ignored for the current work which may degrade performance somewhat. This procedure produces very sparse depth data for each part, which is somewhat difficult to use in its current form. Noting that most of the objects used in the current work consist of plane surfaces (indeed more complex surfaces can be successfully approximated with plane surfaces anyway), it was decided to fit a plane to the depth data using a least squares fit, for greater ease of use. The regions in question were then projected onto the fitted plane (see Figure 6.17), and the minimum and maximum depth distances were calculated based on this projection. These distances were then normalised by the length of the major axis of the bigger region, thus providing measures which are invariant to scenes taken at different depths with the same camera setup. The 3D angle difference was simply calculated as the angle between the normals of the two planes.

The final issue which must be addressed with regard to attribute extraction is that of the range over which the binary relationships are extracted and used for learning and recognition. This issue is not a simple one and there are many possible approaches. Firstly, binary relationships between all parts in an image could be extracted, but this would cause too many relationships to be formed in both the learning and recognition phases, making it very difficult to extract meaningful information from a scene. Alternatively, only those parts which actually touch in the

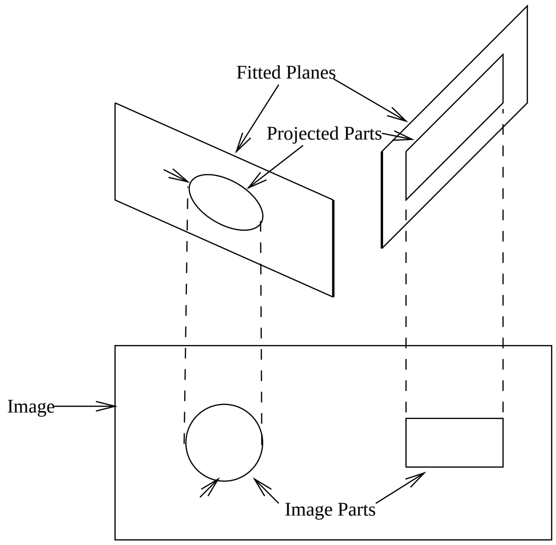



Figure 6.17: Plane projection: An example of two image parts being projected onto planes fitted from the depth data within those regions. The 3D attributes are then extracted from these projected parts.

scene could be used for extracting binary attributes. Unfortunately, with the problems inherent in the segmentation / part extraction phase described above, many parts which should be touching do not - after part extraction. Also, important relationships between non-touching, but discriminating, parts would not be extracted. To overcome these problems, a simple scheme has been employed which extracts binary relationships if the minimum distance between any two parts is less than a threshold (currently 40 pixels with respect to a 256x256 image format).

# Chapter 7

# Learning and Classification

At this stage in processing, we have unary and binary attributes related to all parts for each instance of each object we wish to learn. One method of learning the objects is to build a representation of the unary and binary attributes from which recognition can be performed. The CRG and FCRG classifiers are tree based representations designed specifically for this problem. Following is a description of the original CRG classifier, along with comments about its strengths and weaknesses, followed by a description of an extension to the CRG classifier - the FCRG classifier.

## 7.1 The Conditional Rule Generation (CRG) Classifier

### 7.1.1 The Learning Phase

In the learning phase it is assumed that each object is presented separately as a number of training views. All training views are presented serially, but not necessarily in any order. The training views are in the form of intensity images from which regions are extracted. The regions are used as the object parts from which unary and binary attributes are extracted. The process of region extraction and the attributes used are discussed more fully in Chapters 4 and 6.

The CRG technique works by grouping unary attributes from all parts, of all views of all objects into a unary feature space $U$. Each point in the unary feature space is representative of a single part. The feature space $U$, is then partitioned into a number of clusters $U_i$. Some clusters may be unique with respect to class membership (object type - eg stapler, iron, etc), and these clusters uniquely define which class the corresponding part belongs to. Other clusters will contain multiple classes and these need to be examined further. For the non-unique clusters, the binary features between a part and other parts in the view, are calculated to form a binary feature space, $UB_i$. The binary feature space, $UB_i$, is clustered into a number of clusters $UB_{ij}$. Again, some clusters will be unique with respect to class membership while others will not. The unary features of the second part, $p_s$, are used to form another unary feature space $UBU_{ij}$, which is again clustered to form clusters $UBU_{ijk}$. This process can continue until all clusters are unique, resulting in clusters at the $UBUB, UBUBU, UBUBUB, ...$ levels, but, in some cases, clusters may be unresolvable at any level. In this case, the clusters need to be refined at some level, either by reclustering a particular feature space into more clusters, or by splitting clusters. This will be discussed in the next section. An example of the clustering process is shown in Figure 7.1 ([17]).

The clustering method which is used at each stage is not critical to the success of the CRG technique, although some clustering methods may work better than others. Bischof and Caelli ([17]) have employed a simple splitting based clustering method which produces disjoint axis parallel hyper-rectangular clusters. This procedure works by expanding the cluster tree to the maximum level (an input parameter, typically equal to 5 or 7), and then splitting clusters as necessary. Initially, this means that all points are clustered into the same unary cluster $U$, which is expanded to $UB$, $UBU$, etc. At each level, all points belong to the same cluster. When the maximum height of the tree is reached, the splitting procedure is performed to hopefully form better clusters. This expanding and splitting procedure is continued until all clusters are unique (ie. contains only parts of one class). Other techniques are also possible. For example, a clustering algorithm (such as Leader Cluster) could be used to form initial clusters, which could then be expanded or split as required.

A point, $p_n$, in a cluster, $UBUB_{...ijkl...}$, represents not only a part of a certain view of an object, but also a part path, $\{p_i, p_j, ..., p_n\}$, within that view. It is im-

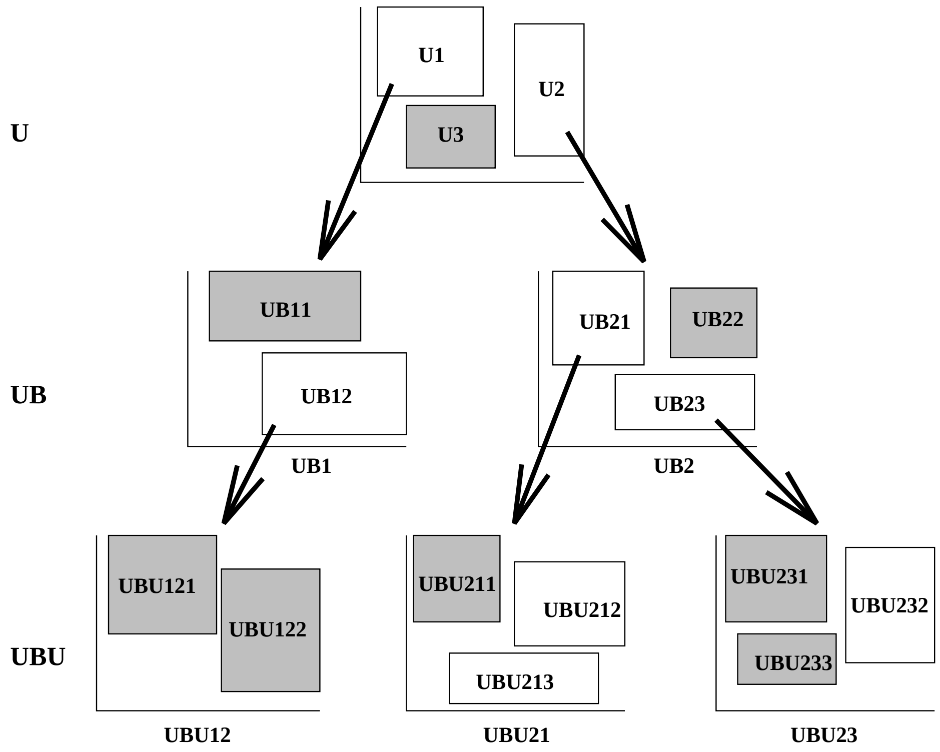



Figure 7.1: An example CRG tree: Displays the tree-like structure of the CRG classifier (taken from [17]). In this figure, grey clusters have zero entropy and are therefore not expanded. White clusters are expanded to the next level.

portant when choosing extra parts for further binary feature spaces not to choose parts already involved in the path. This would form a cyclic path which could be expanded ad infinitum. Also, if all part paths are included in the tree, then significant redundancy will result. For example, both paths $\{p_1, p_2, p_3\}$, and $\{p_1, p_3, p_2\}$ will be included in the cluster tree. Bischof and Caelli [94] proposed a scheme for ordering paths containing the same parts and analysed only the path with the highest order. This removed the redundancy from the cluster tree and made it more compact. In 3-D object recognition from intensity images, such a procedure is not appropriate. Often, parts are incorrectly extracted from an image, or not extracted at all. If this part is one of the earlier parts in the highest order path, then the object will most likely not be classified. Alternatively, if all paths were included, then classification becomes far more likely. For this reason, all permutations of all paths are included for learning of the cluster tree.

An important issue in building the cluster tree is deciding when tree expansion should cease and cluster refinement be employed. Cluster refinement must be considered because tree expansion may never lead to unique clusters, and also reduces the probability of clustering partial data, which is an extremely important consideration with respect to 3-D object recognition. However, cluster refinement leads to a more complex feature space partitioning and a reduction in generalisation ability of the cluster tree. Clearly, a balance between the two techniques must be struck. Bischof and Caelli ([17]) perform tree expansion up to a maximum level and non-unique clusters at this level are refined using the method discussed in the following section.

#### 7.1.1.1 Cluster Refinement

Cluster refinement can be performed in a number of ways. One method is to group the feature space into a larger number of clusters. However, sibling clusters of the unresolved cluster are lost, and must be recomputed, which is inefficient. Another method involves cluster splitting. Here, clusters are split along some optimal boundary hopefully improving the recognition ability of the tree, while not affecting sibling clusters. Also, given that clusters are axis parallel hyper-rectangles in feature space, it is reasonable to restrict split boundaries to also be axis parallel. This greatly simplifies finding the optimal split boundary.

Consider splitting an unresolved cluster, $C$, along feature dimension, $F$. Possible split points are those midway between successive feature values of points in $C$. Choosing an optimal boundary involves choosing the best split point along one of the feature dimensions. To evaluate which split point is optimal, [17] utilise the concept of cluster entropy, $H$:

$$H_i = -\sum_j q_{ij} \ln q_{ij} \tag{7.1}$$

where $q_{ij}$ defines the probability of elements of cluster $i$ belonging to class $j$, and is typically represented by the relative frequency of parts from class $j$ appearing in cluster $i$. Using the concept of cluster entropy, partition entropy, $H_P(T)$, is defined as:

$$H_P(T) = n_1 H(P_1) + n_2 H(P_2) \tag{7.2}$$

To find the optimal split boundary of a cluster, we need to evaluate $H_P(T)$ at each possible split point along each of the feature dimensions, and choose the point with the lowest partition entropy. Splitting can occur at any level in the cluster tree. To find the best cluster to split, each ancestor cluster of the unresolved leaf cluster is examined. The optimal split entropy for each of these clusters is calculated and the cluster with the lowest split entropy is chosen for splitting. If the cluster to split is not a leaf cluster, then it may be necessary to re-expand the cluster tree from the split cluster. All previous descendants of the split cluster are discarded.

If the partition entropy described in equation 7.2 is examined, then it becomes clear that clusters with fewer points will, on average, produce a smaller partition entropy due to the multiplication factors $n_1$ and $n_2$. Such a bias has no effect if only different splits of the same cluster are being compared, but if partition entropies of separate clusters are being compared, then this bias will produce skewed results. In fact, it was found that using equation 7.2 caused the splitting of clusters to occur at the initial unary cluster level (U) almost always, which makes sense, since there are typically fewer unary points than binary points. To overcome this bias, equation

7.2 can be modified in the following way:

$$H_P(T) = (n_1 H(P_1) + n_2 H(P_2))/(n_1 + n_2) \tag{7.3}$$

In summary, CRG generates part-indexed decision trees where each node is defined by specific parts or relations with associated attribute bounds. Typically for vision applications the binary attributes are calculated between adjacent parts or between parts within a restricted neighbourhood.

### 7.1.2 The Recognition Phase

Recognition of objects in a scene typically involves locating regions in a scene, which possibly contains multiple objects, and recognising which class the region belongs to. Hence, there are two problems which need to be solved - scene partitioning and object recognition. The CRG classifier can be used for performing both tasks using the following steps ([17]):

1. Unary features are extracted for all scene parts, and binary features are extracted for adjacent parts.
2. An adjacency graph is extracted from the scene and all non-cyclic paths up to a maximum length, *maxlength*, are extracted. The value of *maxlength* is equal to the maximum depth of the tree which was generated during the training stage. These paths are termed evidence chains or chains.
3. Each chain, $S = < p_i, p_j, ..., p_n >$, is classified using the classification tree generated during the learning phase.
4. The evidence vectors of all chains starting at $p_n$ determine the classification of part $p_n$.

Figure 7.2 displays the relationship between the chains extracted from the scene, and the cluster paths through the classification tree. On the left hand side of the figure is a line drawing of a stapler which is to be recognised using the classification tree on the right. Let us consider chain $< p_1,\ p_2,\ p_3 >$. The unary attributes for part $p_1$ are calculated and it is found that $p_1$ falls in cluster $U_1$. Since cluster $U_1$ is non-unique, the binary attributes for the relationship, $p_1 - p_2$, are calculated which falls in cluster $UB_{12}$. Again, this cluster is non-unique, so the unary attributes for part $p_2$ are used and fall into cluster $UBU_{121}$. This continues with the binary attributes from $p_2 - p_3$ and finally with the unary attributes from part $p_3$. The final cluster, $UBUBU_{12121}$ is also non-unique, because, in this case, the tree has not been further expanded, nor have the clusters been refined. If we assume we have a two class problem, the stapler and the mouse, for example, and the relative frequencies for the training views in cluster $UBUBU_{12121}$ being 0.6 and 0.4 respectively. Then the evidence vector for chain $< p_1,\ p_2,\ p_3 >$ would be $(0.6,\ \ 0.4)^T$. Another example is shown in Figure 7.2 for chain $< p_7,\ p_8,\ p_{10} >$.

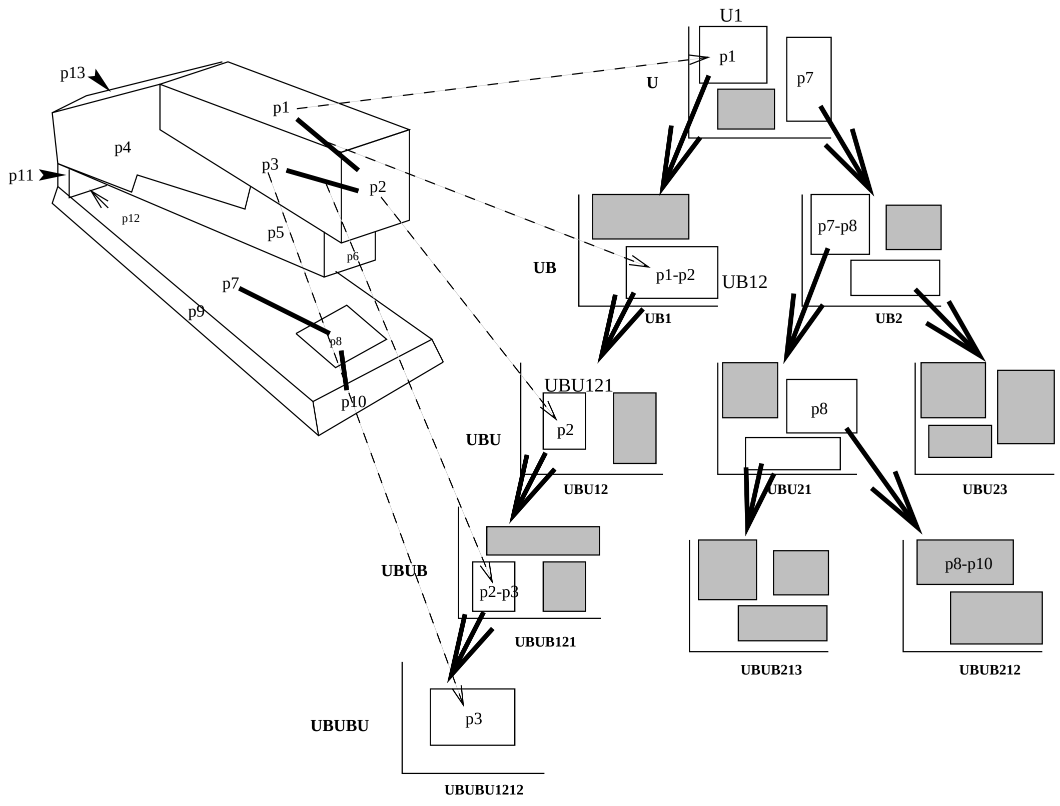



Figure 7.2: The relationship between scene chains and cluster tree paths: Parts are labelled from $p_1$ to $p_3$. Example chains on the object are indicated by thick lines joining two part labels (eg $p_1 - p_2 - p_3$), and the mapping between the scene parts and their relationships and the cluster tree is indicated by dashed lines.

To determine the final classification for part $p_1$, for example, all chains beginning at $p_1$ (eg $< p_1,\ p_2,\ p_3 >$, $< p_1,\ p_3,\ p_5 >$, $< p_1,\ p_3,\ p_4 >$, etc) are classified using the classification tree, and the evidence vectors from these chains need to be combined in some manner. The problem is somewhat complicated since evidence vectors for a given part may be incompatible, or non-unique. Incompatibility occurs when two evidence vectors indicate two separate class labels for the given part, while non-uniqueness indicates that a rule has been partially instantiated (eg path is only UB, while rules are UBU), or a cluster is non-unique. Further, the problem of how best to utilise such evidence vectors of the chains to provide an optimal and consistent labelling of scene parts is non-trivial. In the case where only one object is presented at a time for recognition, the problem is greatly simplified since, aside from erroneous part extraction, all chains extracted from the scene are formed from only one object. Hence, most chains should be classified correctly, and a simple averaging of evidence vectors should prove adequate for correctly recognising the object. If multiple objects are present in a scene, however, then an added level of complexity is introduced and this has not been adequately addressed in the literature. Approaches to this problem, in the literature, fall into two categories. In the first approach, candidate "objects" are selected using perceptual grouping principles and the like, and this was a motivation for implementing the procedures described in Chapter 3. However, the results of those procedures were not adequate for use here. The second approach is the one adopted here as an extension of recent work by Bischof and Caelli ([19]) and partially used by Fan et al ([45]). In this approach evidence is propagated from parts and relations via compatibility measures, as described below.

### 7.1.3 Compatibility Analysis

In their original solution, Bischof and Caelli ([19]) utilised a relaxation labelling technique for updating evidence vectors by weighting the evidence vector of a chain, $S = < p_i, p_j, ..., p_k >$, by the current evidence vectors of the parts, $p_i, p_j, ..., p_k$ using a compatibility function $w_{intra}$. The initial evidence vectors, $\vec{E}(p_i)$, for each part, $p_i$, are found by averaging the evidence vectors of each chain starting at part $p_i$. This method was based on the idea that for chains which do not cross object boundaries, the evidence vectors of the parts belonging to the chain should be similar, and dissimilar for chains which cross object boundaries. More formally, given a chain $S_i = < p_{i1}, p_{i2}, ..., p_{in} >$, the similarity of all intra-chain evidence vectors can be given by:

$$w_{intra}(S_i) = \frac{1}{n(n-1)} \sum_{k=1}^{n} \sum_{l=1 l \neq k}^{n} \vec{E}(p_{ik}) \cdot \vec{E}(p_{il}). \tag{7.4}$$

The relaxation labelling method can be stated formally as:

$$\vec{E}^{(t+1)}(p) = \Phi \left[ \frac{1}{Z} \sum_{S \in S_p} w_{intra}^{(t)}(S) \vec{E}(S) \right], \tag{7.5}$$

where $Z$ is a normalising factor:

$$Z = \sum_{S \in S_p} w_{intra}^{(t)}(S), \tag{7.6}$$

and $\Phi$ is a non-linear transducer function. Iterative computation of Equation 7.5 is necessary because the recomputation of $\vec{E}(p)$ affects the value of the intra-chain compatibility $w_{intra}(S)$.

Compatibility of evidence vectors is a weak constraint for updating part evidence vectors. Bischof and Caelli ([18]) have recently described stronger constraints for updating part evidence vectors. Four rules for detecting and eliminating "crossing" chains have been described by Bischof and Caelli ([18]). These are the chain permutation constraint, the single classification constraint, inter-chain compatibility analysis, and intra-chain compatibility analysis. The first two rules are deterministic rules, and the last two are probabilistic.

### 7.1.4 Chain Permutation Constraint

Each CRG rule has a set of model or learning parts which are associated with it. The chain permutation constraint is based on the fact that permutations of the same scene chain should instantiate the same model parts. For example, if we consider two chains $S_1 =< p_1 p_2 >$ and $S_2 =< p_2 p_1 >$, then in chain $S_1$, part $p_1$ will instantiate model parts at the $U$ level ($M(U|p_1)$), and part $p_2$ will instantiate parts at the $UBU$ level ($M(UBU|p_2)$). Similarly, in chain $S_2$, part $p_2$ will instantiate model parts at the $U$ level, ($M(U|p_2)$), and part $p_1$ will instantiate model parts at the $UBU$ level, ($M(UBU|p_1)$). In the chain permutation constraint, the chains $S_1$ and $S_2$ are considered valid iff $M(U|p_1) \cap M(UBU|p_1) \neq \phi$ and $M(U|p_2) \cap M(UBU|p_2) \neq \phi$, where $\phi$ is the empty set. If either of the intersections are empty, then both of the chains are invalid.

Although theoretically sound, the practical applications of such a constraint are limited especially in the domain of 3D object recognition from 2D intensity views. The major reason for this is that it is difficult to tell if the same model parts have been instantiated unless an explicit correspondence is given between model parts in different views. In practice, it will only be possible to determine if the same model part from the same training view has been instantiated. Hence, if two chain permutations instantiate the same model parts, but from different training views (which is quite likely in the current domain), then these chains will be wrongly rejected, and this will adversely affect the recognition performance of the system.

### 7.1.5 Single Classification Constraint

The single classification constraint is based on the assumption that at least one chain starting at a scene part does not cross an object boundary and that at least one instantiated rule indexes the correct model parts. This constraint is somewhat difficult to apply given the possibly large number of chains starting at each part. However, if there exists a part that initiates only a single chain, $S_i$, and this chain instantiates a single rule, then the constraint can be applied to all chains which touch $S_i$. The constraint is applied in a similar manner to the chain permutation constraint. That is, all chains touching $S_i$ must instantiate the same set of model parts as $S_i$, or it is considered to be inconsistent with $S_i$ and is removed.

Since this constraint is applied in the same manner as the chain permutation constraint, it suffers from the same problem as described in Section 7.1.4. However, it also suffers from a further limitation. If the assumption upon which the constraint is based is not true, then the constraint cannot be applied. In other words, if the single classification chain is incorrectly classified, then the recognition performance will be adversely affected. In fairly complicated scenes, however, it is unlikely that a part will initiate a single chain only, and hence this constraint seems to have little effect on the overall recognition performance.

### 7.1.6 Inter-chain Compatibility Analysis

The inter-chain compatibility analysis is based on the idea that the less compatible the evidence vector of a chain, $S_i$, is with the evidence vectors of all touching chains, the more likely it is that $S_i$ crosses an object boundary. A measure can again be formulated based on the model parts which are instantiated by the chains. Let $S_i =< p_{i1}, p_{i2}, ..., p_{in_i} >$ and $S_j =< p_{j1}, p_{j2}, ..., p_{jn_j} >$ be touching chains (ie $p_{ik} \equiv p_{jl}$, for some $k$, $l$), and let $T_{ij}$ be the set of scene parts common to $S_i$ and $S_j$, ie $T_{ij} = \{p | \exists k, p \equiv p_{ik} \text{ and } \exists l, p \equiv p_{jl}\}$. The compatibility of $S_i$ and $S_j$ is defined as:

$$C(S_i, S_j) = \frac{1}{\#T_{ij}} \sum_{p \in T_{ij}} \frac{\#(\mathcal{M}(p|S_i) \cap \mathcal{M}(p|S_j))}{\#(\mathcal{M}(p|S_i) \cup \mathcal{M}(p|S_j))} \tag{7.7}$$

Where the function # defines the cardinality of a set and $\#(T_{ij}) > 0$. The overall inter-chain compatibility of chain $S_i$ is then defined with respect to the set of all chains touching $S_i$, $S_T = \{S_j | \#(T_{ij}) > 0\}$, as follows:

$$w_{inter}(S_i) = \frac{1}{\#(S_T)} \sum_{S \in S_T} C(S_i, S). \tag{7.8}$$

Equation 7.8 can then be used to form the initial average evidence vectors of parts in the following way:

$$\vec{E}(p) = \frac{\sum_{s \in S_p} w_{inter}(S) \vec{E}(S)}{\sum_{s \in S_p} w_{inter}(S)}, \tag{7.9}$$

where $S_p$ is defined as the set of all chains starting at part $p$.

Again, this measure is theoretically sound, but since it, too, uses the model parts which are instantiated, it is also limited by the same problems as described above. Namely, it is difficult to tell if two given model parts are the same part unless they are in fact the same part from the same training view. Also, since it is likely that a significant amount of chains will cross object boundaries in reasonably complex and cluttered scenes, then it is also likely that the value calculated from Equation 7.8 will be low regardless of whether the current chain crosses an object boundary or not.

### 7.1.7 Intra-chain Compatibility Analysis

The final rule for detecting chain crossings is similar to the original method described above in Section 7.1.3, and is based on the idea that for chains which do not cross object boundaries, the evidence vectors of the parts belonging to the chain should be similar, and dissimilar for chains which cross object boundaries. More formally, given a chain $S_i =< p_{i1}, p_{i2}, ..., p_{in} >$, the similarity of all intra-chain evidence vectors can be given by:

$$w_{intra}(S) = \frac{1}{n(n-1)} \sum_{k=1}^{n} \sum_{l=1, l \neq k}^{n} \vec{E}(p_{ik}) \cdot \vec{E}(p_{il}). \tag{7.10}$$

Since this measure does not make use of compatibilities between instantiated model parts, it does not suffer from the same problems as the other three compatibility rules. However, as was mentioned above, this rule is rather weak for finding a consistent set of labelled parts.

### 7.1.8 The Resultant Compatibility Scheme

In their scheme, Bischof and Caelli ([18]), apply each of the four rules in series to produce a hopefully consistent set of labelled parts. The chain permutation constraint and the single classification constraint are first applied to remove inconsistent chains from the set of all chains. The initial average evidence vectors are then calculated using the inter-chain compatibility from Equation 7.9. Finally, the intra-chain compatibility measure is incorporated into a relaxation labelling scheme by iteratively computing evidence vectors for each part in the following manner:

$$\vec{E}^{(t+1)}(p) = \Phi \left[ \frac{1}{Z} \sum_{S \in S_p} w_{inter}(S) w_{intra}^{(t)}(S) \vec{E}(S) \right], \tag{7.11}$$

where $Z$ is a normalising factor:

$$Z = \sum_{S \in S_p} w_{inter}(S) w_{intra}^{(t)}(S), \tag{7.12}$$

and $\Phi$ is a non-linear transducer function. Iterative computation of Equation 7.11 is necessary because the recomputation of $\vec{E}(p)$ affects the value of the intra-chain compatibility $w_{intra}(S)$.

### 7.1.9 General Comments and a New Compatibility Measure

As has been mentioned, the first three constraints (Sections 7.1.4, 7.1.5, 7.1.6) described above have theoretical merit, but are limited in their practical application due to the problem of finding correspondence between the same model parts in different training views. Hence, model part correspondence is limited to the trivial correspondence between a model part in a particular training view, and itself. The rules may still be useful if part extraction is particularly stable, but this is typically not the case in most real world environments. In attempting to apply these rules to non-synthetic data, in the guise of 3D object recognition from 2D intensity images, it was found that very few chains remained after all inconsistent chains were deleted. In fact, many parts could not be recognised at all because of this factor. Hence, these rules were not used in the current object recognition system.

The fourth constraint (7.1.7) for chain compatibility analysis does not suffer from the same problem as it doesn't rely on correspondence between instantiated model parts, and hence is more applicable to the problem domain. However, the usefulness of this compatibility measure is somewhat limited. This can be attributed to the fact that each chain evidence vector is weighted by a single multiplicative factor which depends only on the compatibility of the average evidence vectors of each part in the chain. It does not depend on the compatibility of the chain's evidence vector with the average evidence vectors of the parts, resulting in the weighting of chain evidence vectors independently of how well the chain represents the overall evidence for those parts. This is best described by an example. Consider the chain $S =< A, B, C >$ in a two class problem, with average evidence vectors of each of the parts:

$$\vec{E}_{av}(A) = (0.5,\ 0.5)^T, \tag{7.13}$$

$$\vec{E}_{av}(B) = (0.8,\ 0.2)^T, \tag{7.14}$$

$$\vec{E}_{av}(C) = (0.9,\ 0.1)^T, \tag{7.15}$$

and the evidence vector for $S$:

$$\vec{E}(S) = (0.2,\ 0.8)^T. \tag{7.16}$$

The intra-chain compatibility measure for $S$, according to Equation 7.10 is:

$$w_{intra}(S) = \frac{1}{3(3-1)} \left[ \begin{array}{ll} \vec{E}_{av}(A) \cdot \vec{E}_{av}(B) & + \\ \vec{E}_{av}(A) \cdot \vec{E}_{av}(C) & + \\ \vec{E}_{av}(B) \cdot \vec{E}_{av}(C) & \end{array} \right] = 0.29. \tag{7.17}$$

This compatibility measure is quite high, and it should be obvious that the average evidence vectors for each of the parts are compatible. However, the compatibility is used for weighting the evidence vector of $S$ in Equation 7.11, which is clearly incompatible with the evidence vectors of the parts $B$ and $C$. The result of this is that the updated evidence vector for part $A$ will probably tend to favour class 2, whereas it is more likely to belong to class 1.

So the current intra-chain compatibility measure does not seem to capture enough information about the compatibility between the chain's evidence vector and the evidence vector's of its constituent parts. A more salient description of the compatibility between parts of a chain would involve a measure of the compatibility of each class of the evidence vector's of the constituent parts with the evidence vector of the chain. In other words, the measure would be a vector quantity and not simply a scalar quantity. More formally, this measure can be characterised by the following equation, for a chain $S_i =< p_{i1}, p_{i2}, ..., p_{in} >$:

$$\vec{w}_{intra}(S_i) = \frac{1}{n}\sum_{k=1}^{n} \vec{E}(p_{ik}) \tag{7.18}$$

This compatibility measure can be used directly with the relaxation labelling scheme given above in Equation 7.11. The modified relaxation labelling scheme becomes:

$$\vec{E}^{(t+1)}(p) = \Phi\left[\frac{1}{Z}\sum_{S\in S_p}\left(\vec{w}_{intra}^{(t)}(S)\otimes\vec{E}(S)\right)\right], \tag{7.19}$$

where all quantities are defined in the same way as above and the binary operator $\otimes$ is defined as a component-wise vector multiplication in the following way:

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ bd \end{pmatrix}. \tag{7.20}$$

Redoing the above example gives the following compatibility measure:

$$\vec{w}_{intra} = \frac{1}{3}[\vec{E}_{av}(A) + \vec{E}_{av}(B) + \vec{E}_{av}(C)] = (.73,\ .26)^T. \tag{7.21}$$

As can be seen from the resultant compatibility measure, chain $S$ is more likely to belong to class 1 than class 2, even though the initial evidence vector suggests otherwise. So, when the component-wise multiplication is performed in Equation

7.19, the evidence used due to chain $S$ is minimised because of the apparent incompatibilities and becomes less significant as more iterations are completed.

This measure for calculating compatibilities has proved very successful for finding a more consistent set of part labels for each of the scene parts and this fact will be shown in Chapter 10. It should be noted that the measure can be used in conjunction with the other three compatibility rules if desired. However, due to the unsuccessful application of these rules, it was found to work very well with the relaxation labelling scheme alone.

## 7.2 The Fuzzy CRG Algorithm

From the above discussion, it can be seen that the problem of recognising 3D objects in complex scenes from 2D intensity images using an empirical learning approach is very difficult. At the learning stage, each object is presented to the classifier from a variety of viewpoints. Images formed from different viewpoints of the same object may have parts in common, or they could be completely different to other views. Even when views of the same objects have common parts, these parts are likely to project differently due to the distortions of perspective projection. For a recognition system to perform well in such a domain, it must be able to gracefully handle such distortions.

Although CRG handles the label compatibility problem well, and is an efficient means of recognition, it does have some limitations in handling distortions in the input data, whether at the learning stage or the recognition stage. That is, the generalisation capabilities of CRG are somewhat limited. This deficiency in CRG can be attributed to the fact that at each level, CRG makes use of hard or crisp clustering algorithms to cluster the data. If an object is presented to the system at recognition time from a novel view, then it is possible that the feature values of the extracted parts will fall outside the formed cluster, thus reducing the recognition performance of the system. A more natural approach would be assign each part, a degree of membership in each of the clusters and choose the clusters with the largest membership values as the most likely clusters. Such a scheme is in the realm of fuzzy clustering which was introduced by Ruspini ([102]).

### 7.2.1 The Learning Stage

The fuzzy CRG algorithm is very similar to the original CRG tree building algorithm. The major difference is in how the clusters are formed and the way in which a data point is considered to be a member of the clusters. The basic algorithm remains the same and is given in Algorithm 7.2.1. Similarly, a block diagram of the overall learning system is given in Figure 7.3.

In the original implementation of the CRG classifier, the cluster tree was formed using an expand and split procedure. That is, the cluster tree was expanded to the maximum level of expansion, and then if non-unique clusters exist, a cluster is split, and the tree is re-expanded. This process continues until all clusters are unique. Any clustering method can be used to form the initial clusters, but Bischof and Caelli ([17]) chose to use a single cluster approach. That is, all points are initially grouped into one cluster and the splitting procedure is relied upon to generate discriminating clusters (see Section 7.1.1.1). Although this method has been used with some success, it does have a number of limitations. Firstly, it can be an expensive way to build the cluster tree. One solution to this problem is described in Section 7.2.1.1. Secondly, the splitting procedure can cause imbalanced clusters and is not directly applicable to a fuzzy clustering algorithm. This problem is more fully described in Section 7.2.1.2 where a new splitting strategy is introduced.

#### 7.2.1.1 The Fuzzy Clustering Strategy

With the original CRG tree building strategy, the cluster tree is first expanded and then clusters are split if necessary. When a decision is made to split a cluster (and this can occur at any level in the tree), then all children of that cluster are discarded and expansion must be performed on the two clusters formed from the split. Since the number of initial clusters is 1, and the number of clusters which provides a good tree is possibly many, there is potentially a great deal of tree building which is discarded, resulting in an inefficient learning process. A better way of building the cluster tree involves performing a more discriminatory clustering whilst initially

**Algorithm 7.2.1** *Fuzzy CRG*

```
function fuzzy_crg(parts, entropy_threshold)
inputs:
        parts   /* all parts in the database */
        entropy_threshold
endinputs
variables:
           paths   /* list of paths to be expanded */
           clist   /* a list of clusters */
           Q   /* a queue of clusters to be expanded */
           tree   /* the cluster tree */
endvariables
begin
  paths ← all unary parts
  tree → clusters ← fuzzy_cluster(paths)
  Q ← Q ∪ tree → clusters
  while ¬(empty(Q)) do
         cluster ← dequeue(Q)
         if (cluster → entropy > entropy_threshold) then
           if (cluster → level > maxlevel)∨
                (can't expand paths) then
              clist ← fuzzy_split(cluster)
              Q ← Q ∪ clist
           else
                /* go down to next level ie UB... */
                new_paths ← expand_paths(cluster)
                cluster → clusters ← fuzzy_cluster(new_paths)
                Q ← Q ∪ cluster → clusters
           endif
         endif
  endwhile
  return(tree)
end
```

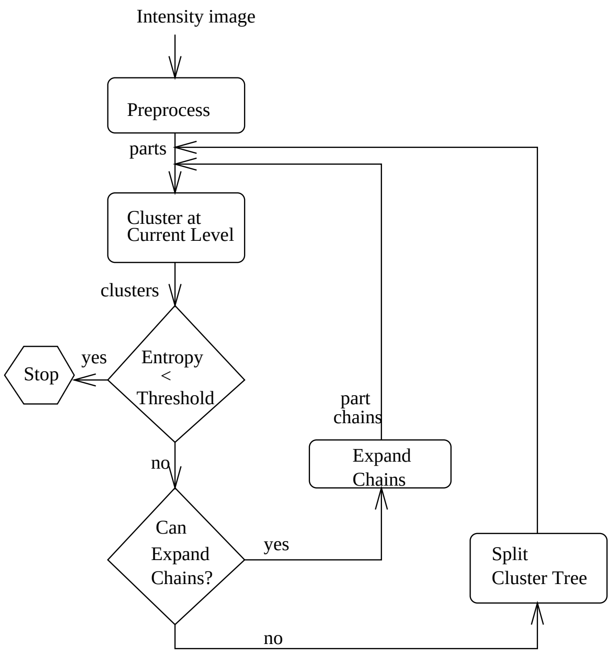



Figure 7.3: The overall learning system.

building the tree instead of relying on the splitting procedure. This technique should significantly reduce the amount of tree building which is discarded due to the splitting procedure. However, it is a non-trivial problem to estimate the number of clusters which are most representative of a given feature space.

The basic clustering algorithm used in this implementation is the fuzzy K-means algorithm ([14]) which was chosen because of its simplicity. The fuzzy K-means algorithm minimises the following objective function using an iterative optimisation scheme:

$$J_q(U,V) = \sum_{j=1}^{N}\sum_{i=1}^{K}(u_{ij})^q d^2(X_j, V_i); \quad K <= N \tag{7.22}$$

where $q$ is any real number greater than 1 and controls the fuzziness of the clusters, $X_j$ is the $j$th $m$-dimensional feature vector, $V_i$ is the centroid of the $i$th cluster, $u_{ij}$ is the degree of membership of $X_j$ in the $i$th cluster, $d^2(X_j, V_i)$ is any inner product measure (distance between $X_j$ and $V_i$), $N$ is the number of data points and $K$ is the number of clusters. The fuzzy clustering algorithm is given in Algorithm 7.2.1.1.

A number of comments about the fuzzy K-means algorithm are pertinent here. Firstly, the general shape of each of the clusters is determined by which distance measure, $d(X_1, X_2)$, is used. In the current implementation, a simple Euclidean

**Algorithm 7.2.2** *Fuzzy K-means*

```
function fuzzy_kmeans(X, V)
inputs:
        X /* array of input attribute vectors */
        K /* number of clusters to form */
endinputs
outputs:
        V /* array of K cluster centres */
endoutputs
begin
  V_i ← K random points
  while (error between subsequent u_ij's or V_i's < threshold) do
        /* compute the degree of membership of all feature vectors
        in all clusters: */
```

$$u_{ij} = \frac{\left[\frac{1}{d^2(X_j,V_i)}\right]^{1/(q-1)}}{\sum_{k=1}^{K}\left[\frac{1}{d^2(X_j,V_k)}\right]^{1/(q-1)}}$$

```
        /* compute new centroids: */
```

$$V_i = \frac{\sum_{j=1}^{N}(u_{ij})^q X_j}{\sum_{j=1}^{N}(u_{ij})^q}$$

```
  endwhile
  return(V)
end
```

distance measure is employed resulting in spherically shaped clusters. This method has been found to work well, but other methods do exist. Most notably, Gustafson and Kessel ([56]) have incorporated a covariance matrix into the distance measure, enabling the formation of elliptical clusters. Also, in the fuzzy K-means algorithm, the sum of the membership values of a feature vector in all clusters, must sum to unity. More formally, this means $\sum_{j=1}^{N} u_{ij} = 1$. Such a constraint is not strictly required of fuzzy memberships. It is used here, however, as a convenience. This constraint is not enforced at the recognition stage as a different measure of fuzzy membership is used. This will be described more fully in Section 7.2.1.1.

Most clustering algorithms require the need for input parameters which explicitly or implicitly determine the number of clusters which are to be formed. For example, the K-means clustering algorithm requires the number of clusters to be explicitly provided as an input parameter. Alternatively, the Leader clustering method requires a distance threshold for implicitly determining the number of clusters. In general, it is up to the problem-solver to determine what these values should be. In a simple clustering environment, this may be the simplest and best method for determining the number of clusters, but in a more complex setting, such as the CRG classifier, such an approach is untenable as the number of clusters required may be different at each node in the cluster tree. A better solution is to automatically determine the number of clusters which are most representative of the data. One way of doing this is to cluster the data using a number of different input parameters (in the case of K-means, different numbers of clusters), and then determine the best clustering from this set similar to what has been developed by Jain and Dubes ([64]). Such an approach has been employed here, and introduces the cluster validity problem.

The issue of cluster validity is one that has received a considerable amount of attention in the fuzzy clustering literature ([12], [13], [116], [117], [71], [118], [55]). However, many of these cluster validity criterion do not take into account the geometrical properties of the resultant clustering. An approach which does account for geometric properties has been introduced by Xie and Beni ([121]). The measure introduced involves a measure of the compactness and separation of the fuzzy clusters. The reasoning being that the more compact and separate the clusters, the more likely it is that the partitioning is valid. The compactness and separation validity function was defined by Xie and Beni ([121]) as follows:

$$S = \frac{\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{q}||V_i - X_j||^2}{n\left(\min_{i,j}||V_i - V_j||^2\right)} \tag{7.23}$$

where $c$ is the number of clusters, $n$ is the number of data points, $\mu_{ij}$ is the fuzzy membership of point $j$ in cluster $i$, $V_i$ is cluster centre $i$, and $X_j$ is data point $j$. The measure $\frac{\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{q}||V_i-X_j||^2}{n}$ refers to the compactness of the fuzzy partitioning, while the value of $\min_{i,j}||V_i - V_j||^2$ is a measure of the separation of the clusters. The measure for the separation of clusters is not necessarily indicative of the overall separation of the clusters, but refers to the minimum separation and hence does not give a true indication of the cluster separation. For this reason the compactness and separation validity function was modified to include an overall measure of cluster separation. This overall cluster separation measure can be described as the average minimum separation between all the clusters. The new validity function becomes:

$$S = \frac{\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{q}||V_i - X_j||^2}{n\sum_{i=1}^{c}\min_{j}||V_i - V_j||^2/c}. \tag{7.24}$$

In applying the measure shown in Equation 7.24 to the problem of finding the optimal number of clusters in a data set, it was found that $S$ decreased monotonically with $c$. This results in choosing the number of clusters as the greatest number tried. For this reason a penalty function was used which varied in inverse proportion to the number of points in each cluster, so larger clusters are favoured over smaller. The penalty function was defined as follows:

$$P = \frac{1}{c}\sum_{i=1}^{c}\frac{1.0}{1.0 + \#V_i} \tag{7.25}$$

where $\#V_i$ indicates the number of points in cluster $V_i$ and is calculated using a hard (non-fuzzy) K-means algorithm. The final validity function then becomes:

$$S' = \frac{\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{q}||V_i - X_j||^2}{n\sum_{i=1}^{c}\min_{j}||V_i - V_j||^2/c} + \frac{1}{c}\sum_{i=1}^{c}\frac{1.0}{1.0 + \#V_i} \tag{7.26}$$

To apply this function to estimating the optimal clustering of the data it is necessary to perform the K-means fuzzy clustering algorithm using different numbers

**Algorithm 7.2.3** *Fuzzy Clustering*

```
function fuzzy_clustering(X)
inputs:
        X /* array of attribute vectors */
endinputs
outputs:
        V_i /* array of cluster centres */
endoutputs
begin
  if (#X < 4) then return(X_av)
  for (i = 2 to 20) do
      V_i = fuzzy_kmeans(X, i)
  endfor
  return(V_i | S'_i < S'_j; i ≠ j)
    /* S' comes from equation 7.26 */
end
```

of clusters at each run. The method used here is a simple linear search starting at 1 cluster and terminating at either 20 clusters or $n/4 + 1$, whichever is less. This ensures that the optimal clustering does not consist of the trivial case (one cluster per data point) and that the execution time of the algorithm is not prohibitive. The final algorithm for clustering at each stage in the cluster tree is described in Algorithm 305. Although such an algorithm is quite expensive, it is a closed loop solution. That is, the researcher is not required to discover the best number of clusters, which may be different at different levels in the tree anyway. So, although the procedure is expensive, nothing is lost in terms of processing time, since no parameters have to be estimated by the researcher as often happens with other techniques.

#### 7.2.1.2 The Fuzzy Splitting Strategy

Although the fuzzy clustering algorithm extends the original CRG expand and split procedure, there is still need for a splitting or reclustering strategy to improve the performance of the cluster tree. However, the splitting strategy used in the original CRG algorithm is applicable only to "hard" clustering environments, and cannot be applied to the fuzzy clustering environment used here. Also, the original splitting method used is based on a minimum entropy procedure, which has a number of disadvantages. In the original method, a cluster is split into two distinct clusters if the optimal split for that cluster has the minimum split entropy (Equations 7.2 and 7.3) for all clusters in the leaf to root cluster chain. The major problem with this method is that the split point for a particular cluster is determined entirely by the point which produces the minimum split entropy. No consideration is given to the distribution of points within a cluster. For example, consider the cluster given in Figure 7.4, it has two well separated "natural" clusters. Unfortunately, the right cluster also consists of two linearly separable clusters. The minimum split entropy results in a split at partition A, forming two clusters that are unequally distributed in their respective cluster bounds, with relatively large areas of empty space. A more natural split for this cluster would be at partition B, resulting in two well formed clusters. If necessary, the right cluster could be further divided at a later stage in processing. Although this is a contrived example, practical experience has shown that similar situations often occur.

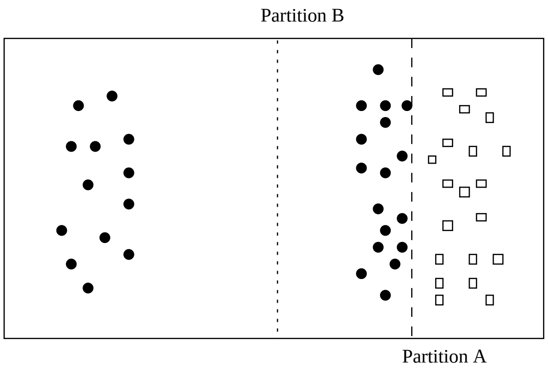



Figure 7.4: The problem with partition entropy: The small filled in circles and small unfilled boxes represent different instances from two different classes. The two "natural" clusters within the original cluster are not found due to the partition entropy criterion (Partition A) leaving a cluster which has large amounts of free space in it. A better splitting rule would choose Partition B.

Taking these problems into account, it was decided not to employ the original splitting strategy in the fuzzy CRG algorithm, though the idea of splitting clusters remains. The splitting strategy employed here is quite simple. If a leaf cluster is found to be non-unique, then all clusters in the leaf to root cluster chain are candidates for splitting (just as in the original algorithm). Each of these clusters are reclustered into two distinct clusters using the fuzzy K-means algorithm, and the split entropy of the cluster is calculated according to Equation 7.3. The cluster with the lowest split entropy is chosen as the one to split, and two new clusters are formed. All ancestors of the split cluster must be discarded and the cluster tree must be re-expanded. Although this procedure is very similar to the original splitting procedure it does not suffer from the problem described above (Figure 7.4) because the split point in each cluster is not chosen based on cluster entropy alone, but rather, is based upon the geometric structure of the data. This should remove the problems mentioned above and provide a more natural clustering of the data.

A problem which has been overlooked until now, is the method of determining cluster entropies. How can entropy be calculated from a fuzzy cluster? In the current implementation, for comparison purposes, cluster entropies are calculated by forming a hard partition around each of the cluster centroids using the crisp K-means algorithm. In other words, each data point is assigned to belong to the nearest cluster centroid for entropy calculation. A more natural extension to fuzzy cluster entropies would be to include the fuzzy membership of each data point in the calculation of the entropy, but this becomes computationally expensive as the number of points increases, and was not used in the current implementation.

### 7.2.2 The Recognition Stage

Again, the basic algorithm used in the recognition stage is the same as that used in the original CRG algorithm. In this case, however, due to the fuzzy nature of the cluster tree, we can descend more than one branch of the tree at a time, by choosing the best $N$ clusters to descend, rather than just one cluster as in the crisp case. The basic algorithm is given in Algorithm 7.2.2, and a block diagram of the recognition system is given in Figure 7.5.

In the current implementation, the value of N given above is 2. This means that at each level of the cluster tree, two clusters are chosen from all the clusters and the path is expanded along both those clusters. The best clusters at a particular

**Algorithm 7.2.4** *Scene Classification*

```
inputs:
        cluster_tree /* tree generated by Fuzzy CRG */
        scene_parts /* all parts in the scene */
        N /* number of branches to traverse at each node */
endinputs
begin
  paths ← all paths in the scene
  foreach (path ∈ paths) do
    clist, plist ← best_clusters(path, cluster_tree, N)
    cluster_queue ← cluster_queue ∪ clist
    path_queue ← path_queue ∪ plist
  endforeach
  while ¬(empty(cluster_queue)) do
        cluster ← dequeue(cluster_queue)
        path ← dequeue(path_queue)
        path.classification ← fuzzy_membership(cluster, path)
        clist, plist ← best_clusters(path, cluster, N)
        cluster_queue ← cluster_queue ∪ clist
        path_queue ← path_queue ∪ plist
  endwhile
end
```

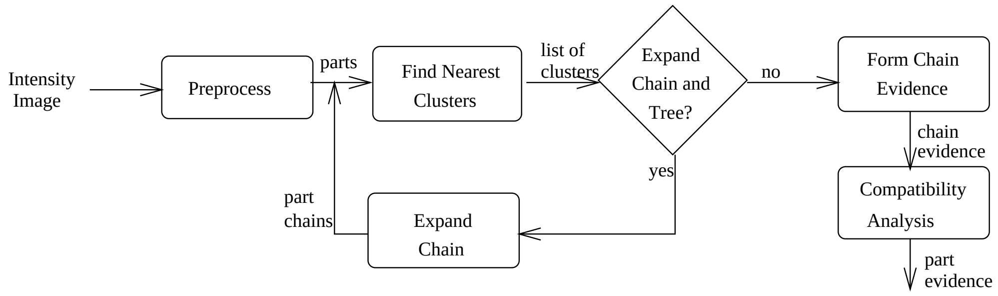



Figure 7.5: The overall recognition system.

level are chosen by evaluating a fuzzy membership function. The most obvious fuzzy membership function to use is the one given above in Figure 7.2.1.1. However, as was briefly mentioned above, such a measure has one major disadvantage: the sum of the fuzzy memberships in each cluster, for a particular data point, is restricted to sum to 1. The problem here is best demonstrated by an example. Consider Figure 7.6, where there are two clusters represented by their respective cluster centroids, and two data points. Both data points are equidistant from both cluster centroids, but data point A is much closer to the centroids than data point B. If the probabilistic membership function is used for evaluating fuzzy memberships, then both of the data points will receive a fuzzy membership value of 0.5 in both clusters. Such a membership function is adequate if we only wish to rank the membership of a data point in each cluster, but if we wish to compare memberships of different data points, then such a measure becomes unacceptable.

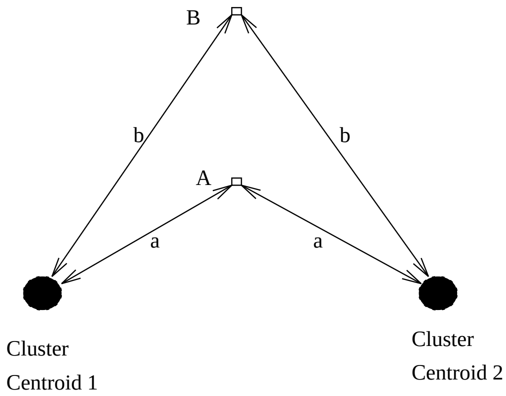



Figure 7.6: Equal fuzzy memberships: Using a probability model, points A and B are both given fuzzy memberships of 0.5 in both clusters. A more natural model would give higher fuzzy membership to point A (eg possibilistic model)

To overcome this problem, one can employ a fuzzy membership function based on the distance a point is from the cluster centroid. Such a possibilistic membership function was introduced by Krishnapuram and Keller ([70]) and is defined as:

$$u_{ij} = \frac{1}{1 + \left(\frac{d_{ij}^2}{\eta_i}\right)^{1/(q-1)}}, \tag{7.27}$$

where $u_{ij}$, $d_{ij}$, and $q$ are defined as before, and $\eta_i$ determines the distance at which the membership value for a point in a cluster becomes 0.5. In the current implementation, $\eta$ has been arbitrarily set to equal 1, and this appears to work quite well. A more sophisticated method could be to set $\eta$ to be proportional to the overall size of each cluster, but this has not been used in the current implementation for simplicity.

#### 7.2.2.1 Scene Understanding

The description given above provides a good way for evaluating the evidence vectors of each chain extracted from the test scene. These evidence vectors can then be used for computing the final labelling of scene parts using any of the techniques described in Section 7.1.3 in exactly the same way as the crisp CRG algorithm. However, with the fuzzy CRG method, we have more information than just the relative frequencies of classes in a given cluster. At each level in the classification tree, each part has an associated fuzzy membership with each cluster at that level in the tree. This membership value can be used to weight the initial evidence vectors of the chains, to give a more descriptive representation of the probable class a chain belongs to, and to provide a better evaluation for comparing the evidence vectors of different chains.

According to Chang and Pavlidis ([35]) the decision value of a particular node in a fuzzy decision tree, is the product of the decision values of the branches composing the path from the root to the node. Consider a chain $S =< p_1, p_2, ..., p_n >$, being classified in the decision tree along the cluster path $P =< U_i, UB_{ij}, ..., (UBU)^n_{ij...} >$, the fuzzy evidence vector for $S$ is given by:

$$\vec{E}_f(S) = \vec{E}(S) \bigwedge_{k = U_i}^{(UBU)^n_{ij...}} u_{kl}, \tag{7.28}$$

where $\vec{E}$ is the original evidence vector of chain $S$ characterised by the relative frequencies of each model part in the cluster $(UBU)^n_{ij...}$, and $u_{kl}$ is the fuzzy membership function of part $p_l$ in cluster $k$ given by Equation 7.27. There are two different interpretations of the product in Equation 7.28 - a real number product in the probability model, or as a minimum function in the max-min model ([35]). Either definition can be used but there is a distinct disadvantage in using the probability model in this case. The problem arises when comparing membership values at different levels in the cluster tree. Due to the multiplication, the decision value at nodes lower in the tree is likely to be smaller than nodes higher in the tree. Hence, it could be argued that comparing decision values at different levels in the tree is not reasonable. For example, if we have a tree with 7 levels and a chain that is recognised at each part in the tree with a fuzzy membership of 0.9, then the final membership value for that chain would be 0.48 ($0.9^7$). Contrast this to a chain with only one part which is recognised with a membership value of 0.5 at the initial level of the tree, but still is considered more likely than the longer chain with membership values of 0.9 at each level. For this reason, it was decided to use the max-min model of fuzzy memberships, which allows memberships to be reasonably compared regardless of which level in the tree they occur at.

A side effect of the above scheme is that a single chain produces many classifications due to the number of clusters which are expanded at each level. In fact, the number of classifications is exponential in the length of the chain, (in this case, exponential to base 2). So, for each part, which may initiate a large number of chains, there is potentially a large number of possible classifications. In combining these classifications, the same mechanism as described in Section 7.1.3 could be used. However, because of the potentially large number of chains, it is possible that the classifications from the most likely chains are overwhelmed by the many classifications of the less likely chains. This is possible because the initial evidence vector of a part is calculated as the average evidence vector of all chain evidence vectors. To remove the possibility of this occurring, the classifications of chains whose total fuzzy evidence (defined by the sum of all fuzzy evidences in the evidence vector) is less than 10% of the maximum chain evidence for that part are deleted. In this way, only classifications of chains which significantly contribute to the overall classification of the part are considered.

# Chapter 8

# Hypothesis Verification

The output of the FCRG classifier involves evidence vectors of each of the parts in the scene - representing what FCRG considers the most likely object class. However, the evidence of each of the parts may not be consistent (even with the compatibility analysis), and in the case of some parts, the correct class label may be ranked second or third according to FCRG, indicating that FCRG would be most useful as the hypothesis stage of an hypothesis-and-test ORS. A simple hypothesis verification scheme has been used here to demonstrate the usefulness of the FCRG algorithm for this task. It should be noted that the hypothesis verification scheme used here is not intended to be a practical system, it is intended only for demonstrating the applicability of the FCRG algorithm.

The hypothesis verification system is a correlation based template matching approach. For each of the parts in the scene, the most likely classes are found using the FCRG algorithm. In this case the number of classes used was set to two. That is, two classes are chosen as the most likely for each scene part, according to the two best scores in the part's evidence vector. These classes are then matched against the parts using a correlation based scheme, and the class which returns the best match for that part is selected. It was found that the classical measure of correlation in the computer vision literature - the correlation coefficient ([51], pg 584) - was particularly sensitive to noise in the images. For this reason a more robust measure of correlation, which exhibits a lower sensitivity to noise, based on the $L_1$ norm was used. This measure due to Brunelli and Messelodi ([28]) is defined as:

$$g(x,y) = 1 \; - \; \frac{\sum_i |x_i - y_i|}{\sum_i |x_i| + |y_i|} \tag{8.1}$$

where $x$ and $y$ can be interpreted as the intensity values for the model and scene image respectively, when the regions in question are placed on top of one another.

The question remains about how to determine the most likely pose of the scene object to maximize the correlation between model and scene, and how to perform this actual matching. Theoretically, the results from the FCRG algorithm give pose information about the scene objects. Each cluster in the FCRG tree potentially contains information about the training parts that belong to the cluster. From this data, it is possible to determine the most likely object and pose of a particular scene part. Unfortunately, each scene part can be involved in a number of different clusters at many different levels of the tree - due to the likelihood that scene parts are involved in more than one chain and at different parts of each chain. This problem is further compounded by the fuzziness of the FCRG tree which allows a single chain to belong to numerous clusters - possibly two child clusters for each parent cluster in the current implementation, allowing for up to $2^7 = 128$ clusters in a tree of height 7. Because of the large number of clusters each scene part can be involved in, significant confusion can arise over which model pose is the most likely, making it difficult to accurately perform correlation matching based on this information. So, since the number of model poses is small in the current application (typically 6 poses with corresponding left and right views), and utilising the fact that left and right views of a given pose are very similar, the correlation matching for a scene was performed using all left model views of a class and choosing as the most likely the one which gave the highest correlation. This was performed for the two most likely classes and the best class and pose was chosen from this set. It is important to note that the resulting hypothesis verification system is not scale and rotation invariant - nor was it intended to be, since it was only a demonstration system.

A similar argument for estimating pose can also be made for estimating which model part a particular scene part is. Again, the same problems arise due to the large numbers of clusters involved. To overcome this problem, each model part in a given model pose is placed completely over the scene part to perform the correlation and again the highest correlation is used to determine the most likely model part from that model pose. The correlation is calculated over the entire model view, affecting the recognition of all scene parts which touch the shifted model view, possibly causing parts to be classified as a class which wasn't one of the best two classes hypothesised by the FCRG algorithm. This effect is desirable as it results in a more globally consistent interpretation of the scene. The algorithm used for performing hypothesis verification is given in Algorithm 8.

Figure 8.1 displays the workings of the hypothesis verification algorithm for a simple object and model. In this case, the FCRG algorithm would have found a likely match between the scene part, $S_1$, and the model object (ie the model object is one of the two most likely matches according to FCRG). The hypothesis verification algorithm works by correlating each of the parts of the model object with the scene part, $S_1$, and choosing the best match, in this case model part $M_1$. This process is repeated for all left training views of the model object, across all parts in the scene. The whole process is repeated for the second most likely object (according to FCRG), and the most likely match for each scene part is chosen based upon these correlations.

Typically, there is in the order of 10 model parts per model view, and we have tested 6 model views on each scene part (the left model views for a given class). Resulting in performing 60 correlations for each scene part. This number can be reduced by only performing the correlations when the model and scene regions are close to each other in size (one region can be no more than twice the size of the other). Nevertheless the number of correlations being performed is still quite high, and a more efficient method would be required for practical use of the system. In fact, since we are performing a lot of learning off-line anyway, it may be possible to develop techniques which would optimally discriminate between the two possible classes hypothesised by the FCRG method, by pre-compiling discriminating methods between all possible pairs of classes. Of course, such a system would be impractical for very large databases ($> 1000$), but may be useful for databases with less than 100 objects. In any case, other methods such as graph matching may be more

**Algorithm 8.0.5** *Hypothesis Verification*

```
function hypothesis_verification(S, E)
inputs:
        𝒮/* list of scene parts */
        E/* Evidence for each part from FCRG */
endinputs
variables:
            S/* scene part */
            C/* list of object classes */
            V/* model view */
            P/* model part */
            CM/* correlation matrix : numparts X numclasses */
endvariables
begin
  foreach S ∈ 𝒮 do
    for the two most likely classes,C do
        foreach V ∈ C do /* each model view belonging to class C */
          foreach P ∈ V do /* each model part belonging to view V */
            corr ← correlation_match(P, C)/* use equation 8.1 */
            Keep note of which other scene parts are affected by the
            correlation match (ie scene parts touching the model view).
            if (corr > M) then
              M ← corr
              CC ← C
            fi
          endforeach
        endforeach
        correlation_matrix[S][CC]+ = M
    endforeach
  endfor
  foreach S ∈ 𝒮 do
    S.class ← maxj_j correlation_matrix[S][j]
  endforeach
end
return(S)
```

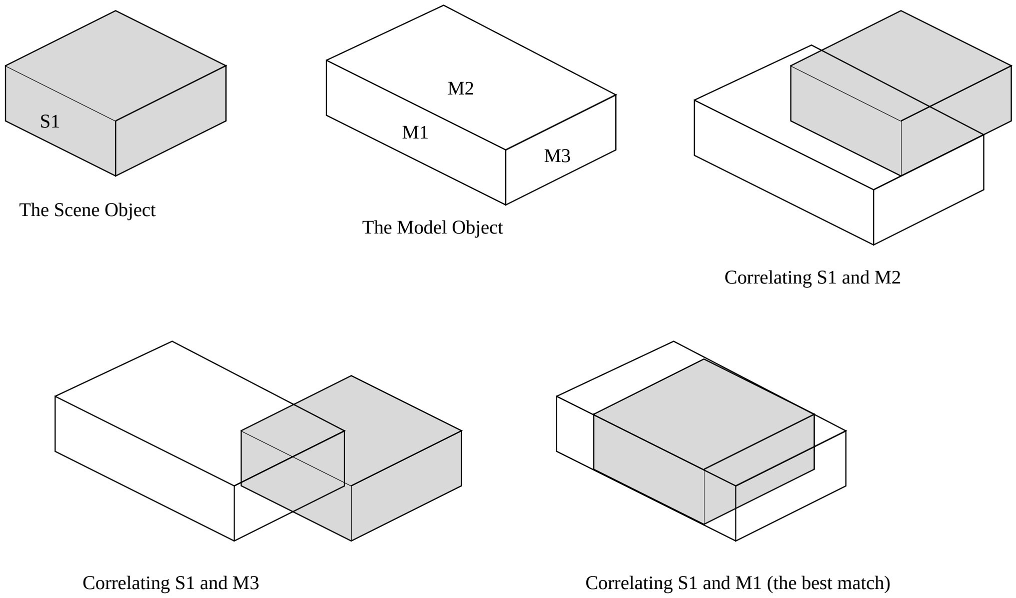



Figure 8.1: An example hypothesis verification.

appropriate.

# Chapter 9

# Experimental Design

This chapter describes the equipment, data sets and the experimental method used to obtain the results reported in Chapter 10.

## 9.1 Equipment

All images for all of the experiments described here were taken using a Pulnix TM-6CN high resolution CCD camera, attached to a frame-grabber on an IBM compatible personal computer. Only one camera was available, so this was mounted on a tripod with a sliding arm which can be configured to slide in a horizontal direction (see Figure 9.1). In this way, stereo images of each of the test scenes were taken. The distance from the camera to the objects is in the range 60-90cm, and the separation of the camera position for taking the two images is approximately 10cm. Obviously, such a setup is subject to many errors, including differing camera separations between different stereo pairs, vertical registration problems between a stereo pair, different object to camera distance for differing views of the same object etc. So, the environment, although controlled, is not too controlled.

## 9.2 Data Sets

For the experiments conducted here, two data-sets were used based upon:

1. Hand edited image segmentation (data set 1).

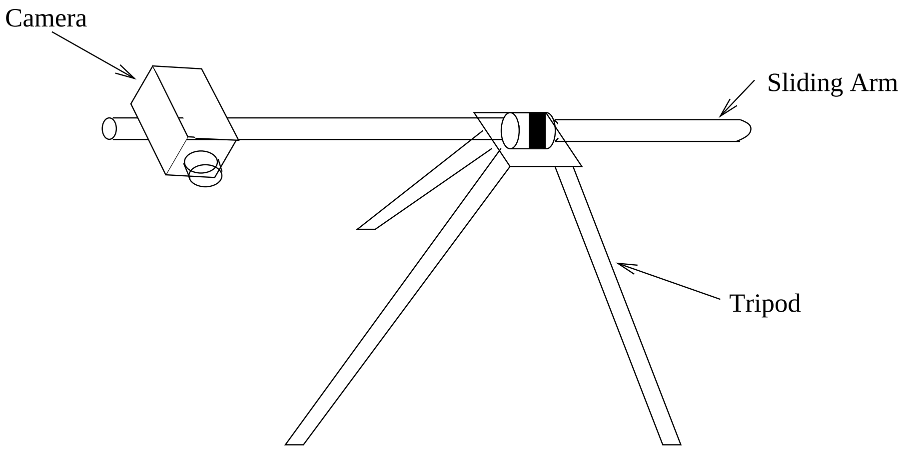



Figure 9.1: The camera setup for the experiments.

2. Automatic image segmentation (data set 2).

Data set 1 was used for some initial evaluations of attributes and learning techniques mostly due to the lack of a reasonable segmentation algorithm. This was rectified by the development of a new segmentation algorithm which was described in Chapter 4.

Both data sets consist of intensity images of a number of different objects. Each object was presented with between 5 and 8 different poses over an approximate angular variation of 90°, and a stereo pair of images was taken for each pose. This resulted in 10 to 16 different images of the same object. Figure 9.2 displays a top view of the different poses and camera positions used for the experiment (not to scale). Note that no attempt was made to span the entire view sphere, rather experiments were limited to poses within this range. This does not reduce the significance of the results however, as within this pose range there exists at least two topologically different poses (the two extreme poses). Of course, if the entire viewing sphere were covered, then more topologically different poses would be evident, but the current experiments give a good indication of these more general results. Also note that the experiments are not precise, and nor were they designed to be. In many possible applications, it is not reasonable to insist that objects must be presented at a fixed set of locations with fixed poses. A general system needs to be able to function in a less precise environment.

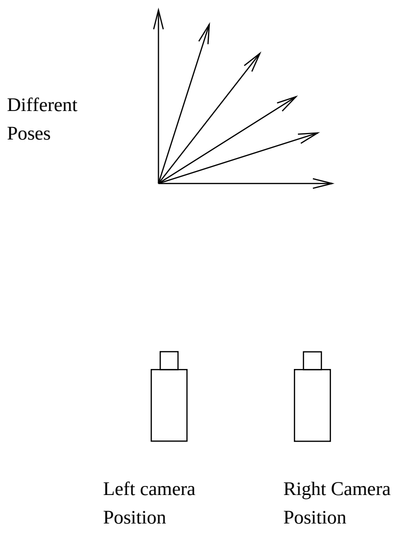



Figure 9.2: Example poses: A top view displaying the different poses and camera positions used in the experiments (not to scale).

Data set 1 consisted of a set of 14 objects resulting in a total of 180 training samples. The training data was created by the following method:

1. The edge detector developed by Shen and Castan ([105, 106]) and distributed with Khoros ([88], routine $vgef$) was applied to the intensity images to form edge images.

2. Closed regions were then formed using an edge closing routine. ([88], routine $vclose$).

3. The edge images were then hand edited to form near "perfect" edge images.

Figure 9.3 displays the effects of the manual editing of the intensity images. As can be seen from the image the edges are improved significantly in some places (the teapot for example), but have little effect in most areas. Figures 9.4 and 9.5 display all the objects and their corresponding regions respectively, from one of the poses. Although the problem has been simplified by the hand editing scheme, it is still not an easy problem as is displayed in the figures. Figure 9.6 displays all the poses used for the stapler object - other objects were presented using similar poses.

Data set 2 is a more comprehensive data set containing 18 objects (including all the objects from data set 1) resulting in a total of 220 training samples, and was

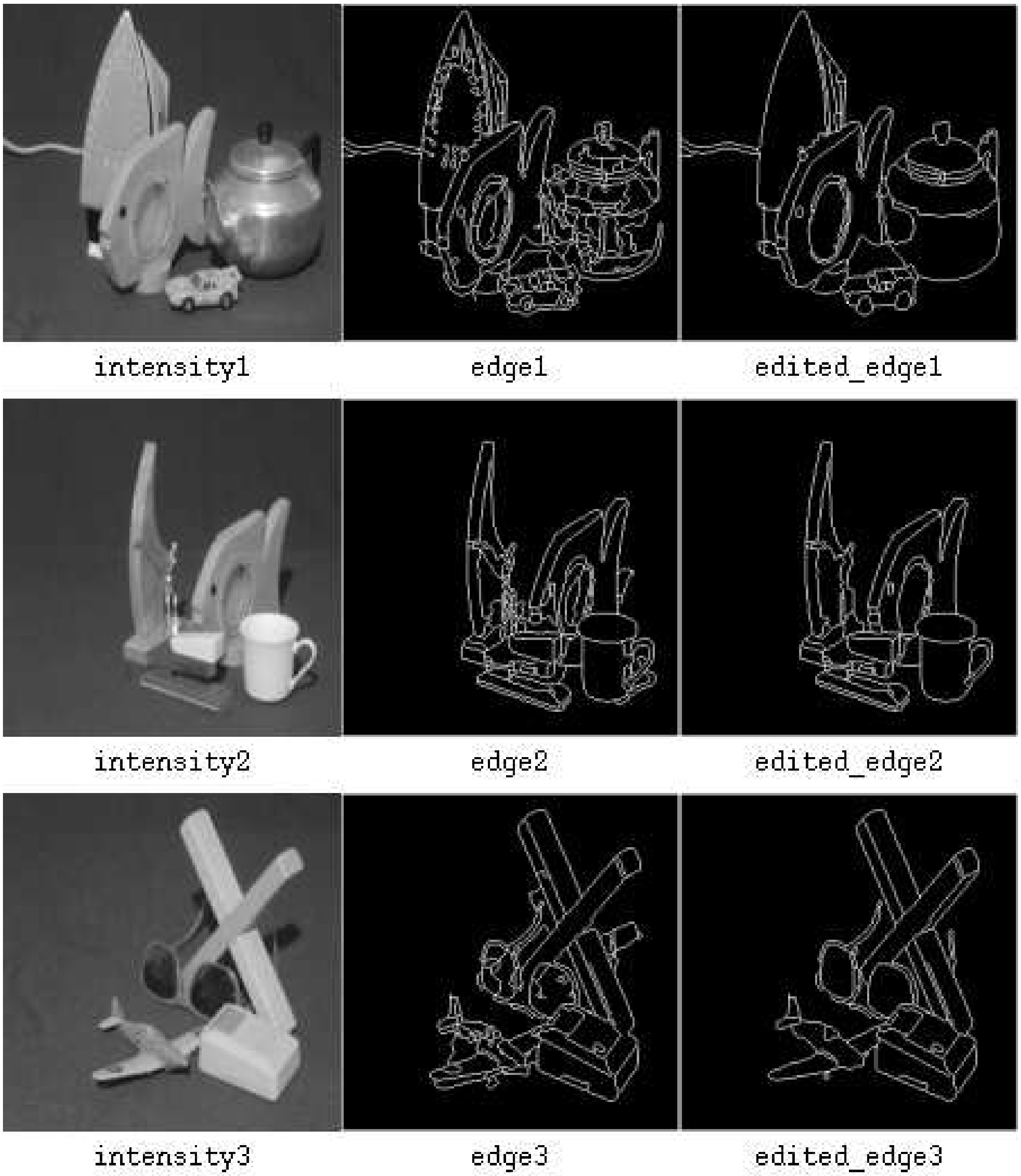



Figure 9.3: Examples of the extracted edge images after manual editing: As can be seen from the images, editing has little effect for many objects, but has obvious effect for objects which are difficult to segment successfully due to surface properties etc (eg the teapot).

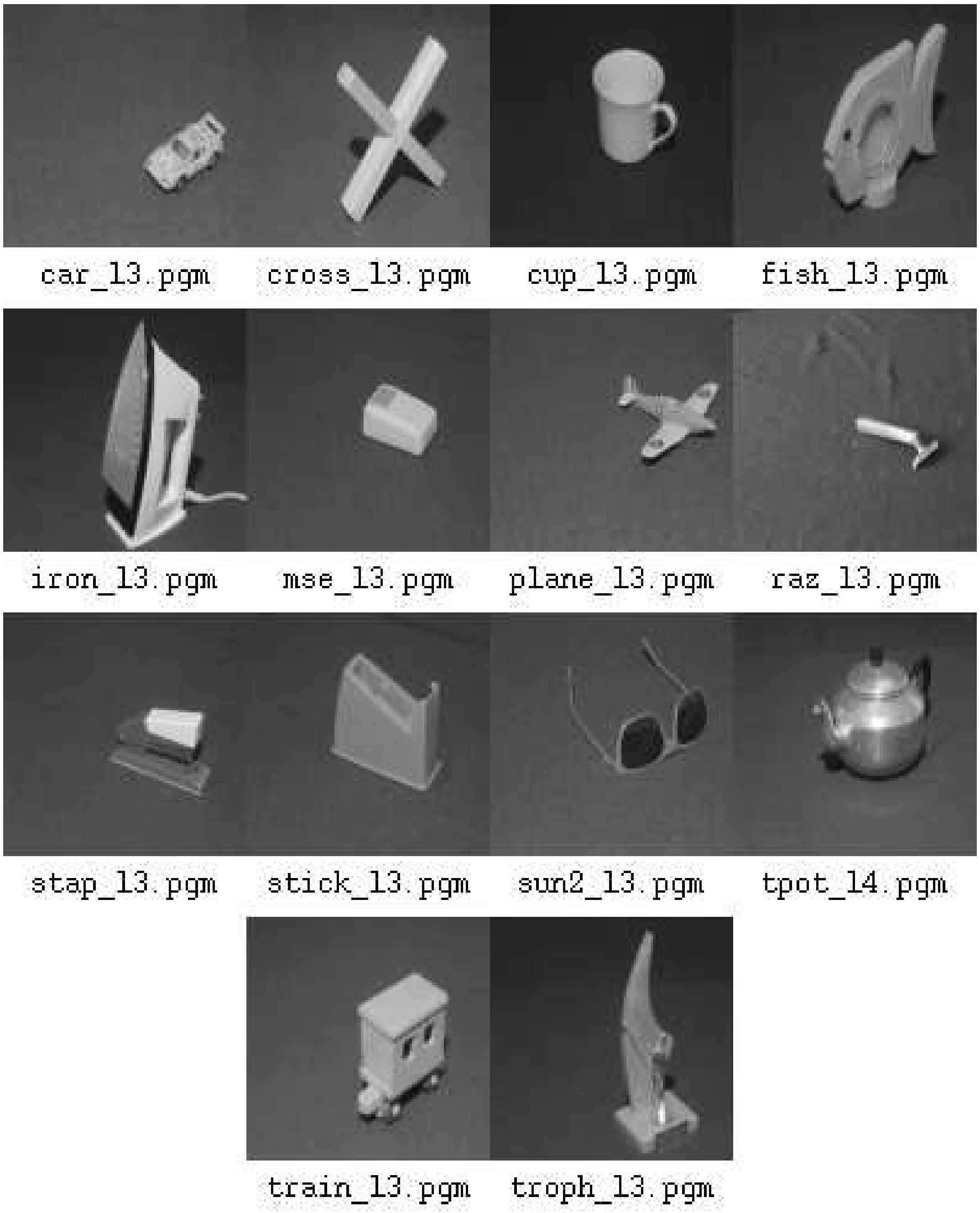



Figure 9.4: Data set 1 - intensity images

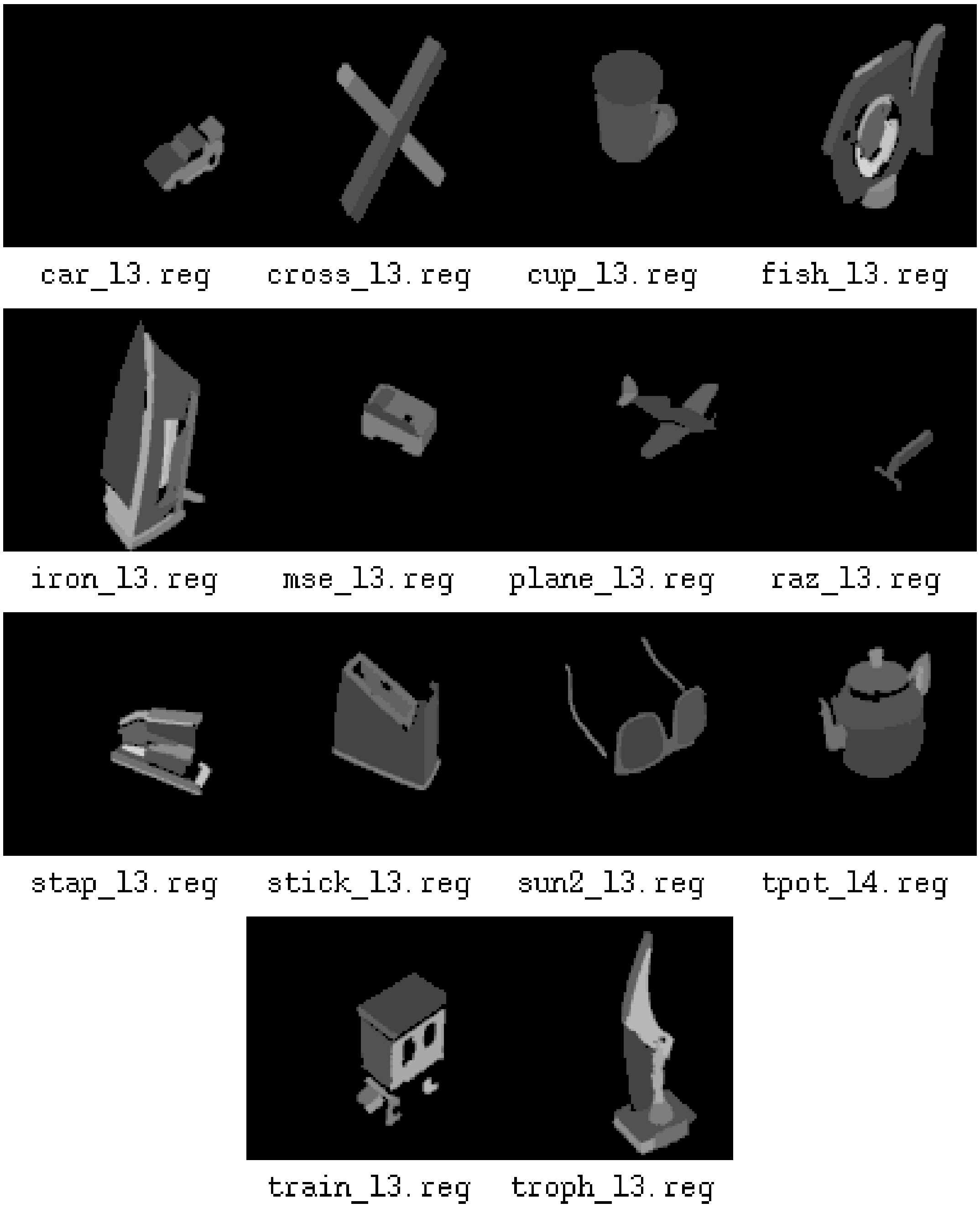



Figure 9.5: Data set 1- extracted regions

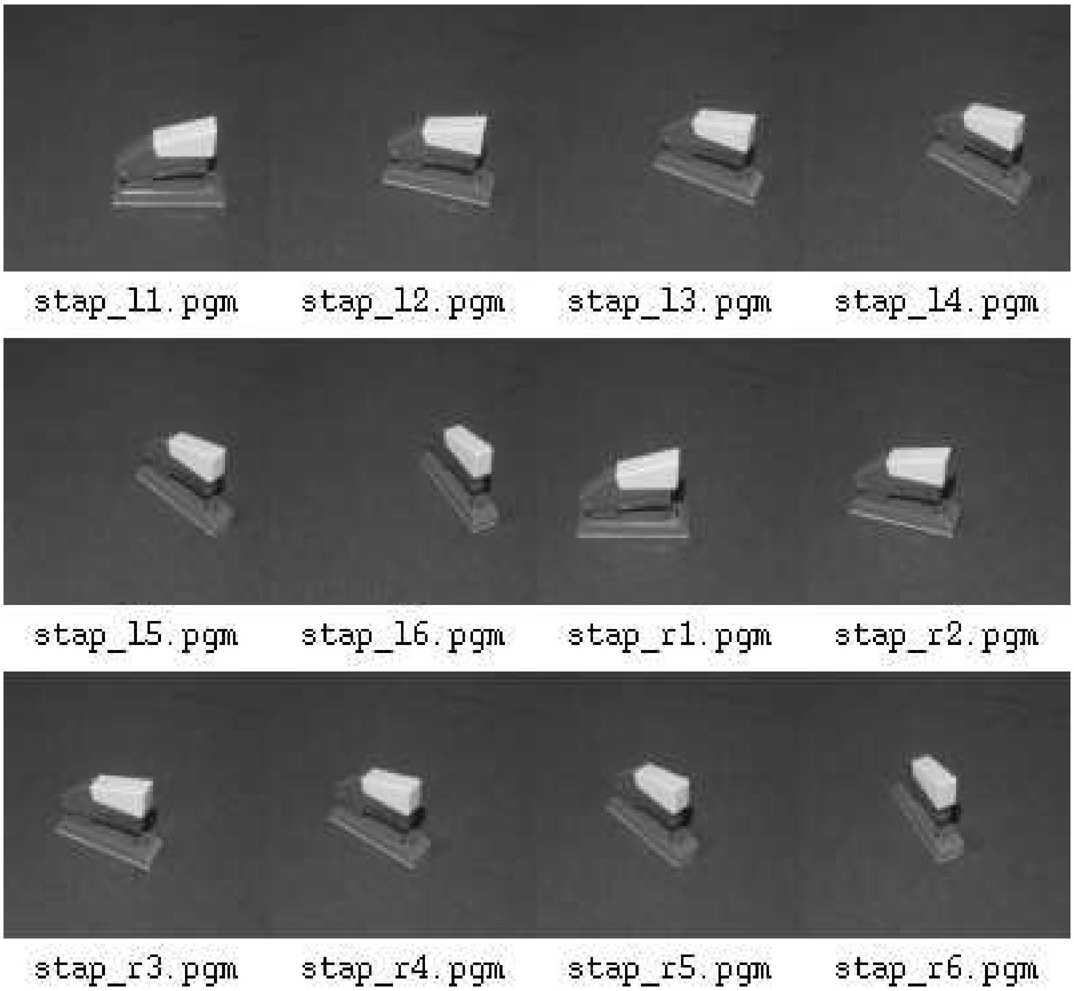



Figure 9.6: All views of the stapler.

created using the segmentation algorithm described in Chapter 4. Such a data set is more realistic than the hand-edited data set as the segmentation is performed in an automatic fashion. Of course, this has its disadvantages as the corresponding data set is somewhat more complex than the hand-edited data. Figures 9.7 and 9.8 display all the objects and their corresponding extracted regions used in data set 2.

## 9.3 Experiments

Two types of experiments were performed. These are the single object scene experiments and the multi-object scene experiments. The first set of experiments performed involved single object classification. These involved training the FCRG classifier on all objects at all poses except one. The object pose which was not used for training was then used to test the accuracy of the tree for that object. This procedure is repeated across all poses of all objects to give an overall measure of the accuracy of the classifier (a type of cross-validation procedure). The procedure is summarised in Algorithm 9.3. In this procedure, both left and right views for each pose are removed from the training set and used for testing at the same time. There is a practical reason for removing both the left and right views of an object's position from the training data. In a realistic situation, when recognising objects, if stereo pairs of views are used, then it is likely that neither of these views would have been seen before.

The cross-validation procedure has been used to evaluate which attributes and learning techniques work best on the data sets used here. It does not necessarily follow that the same techniques will work just as well on other data sets, but since the current data sets are reasonably diverse with respect to the objects contained in the sets, it is reasonable to expect that the results found here will generalise to other situations.

The other set of experiments which were performed are the multiple object experiments. In these experiments, an image containing a number of objects (between 2 and 5) was presented to the system for recognition. A database of over 100 multiple

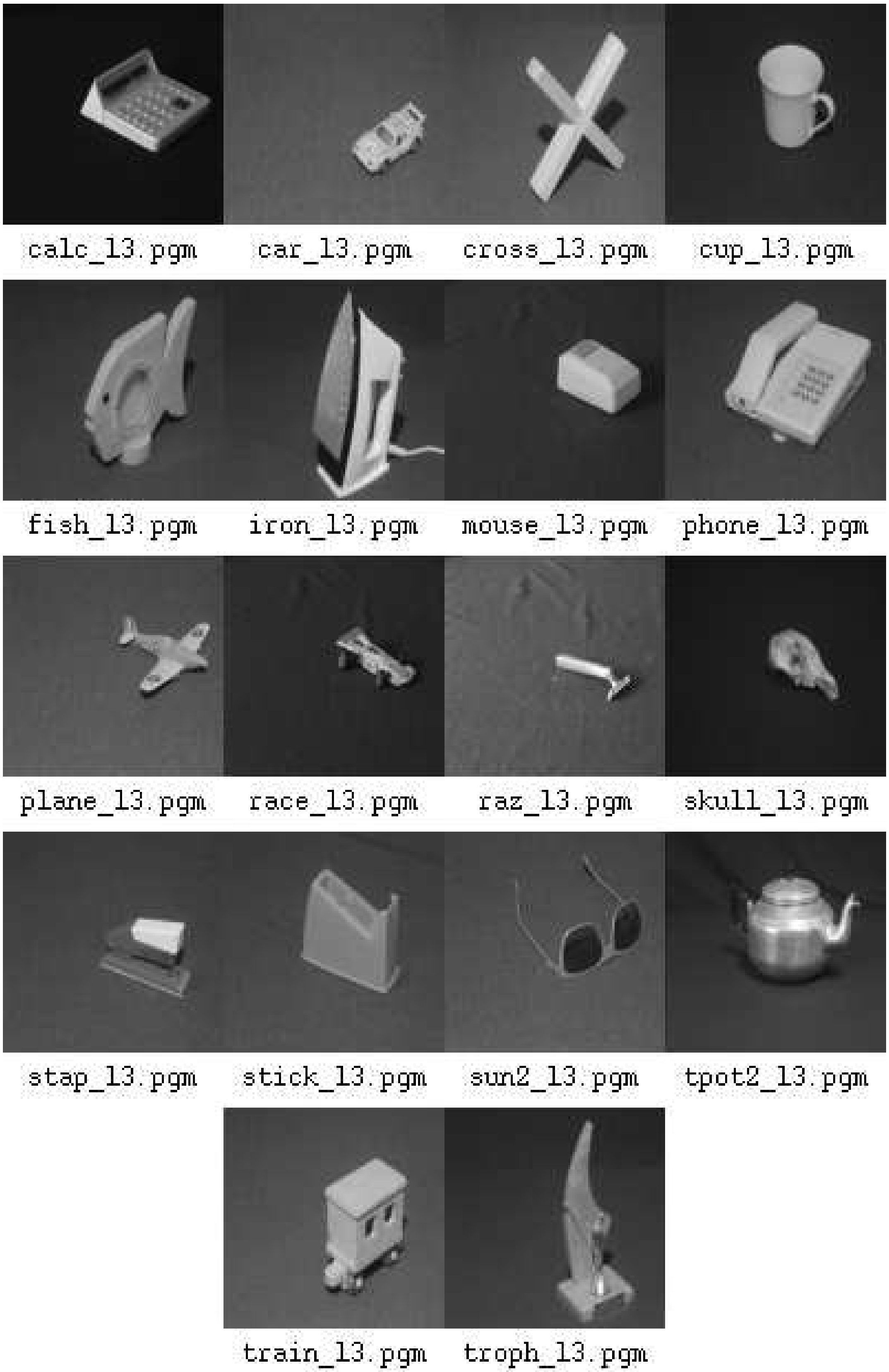



Figure 9.7: Data set 2 - intensity images

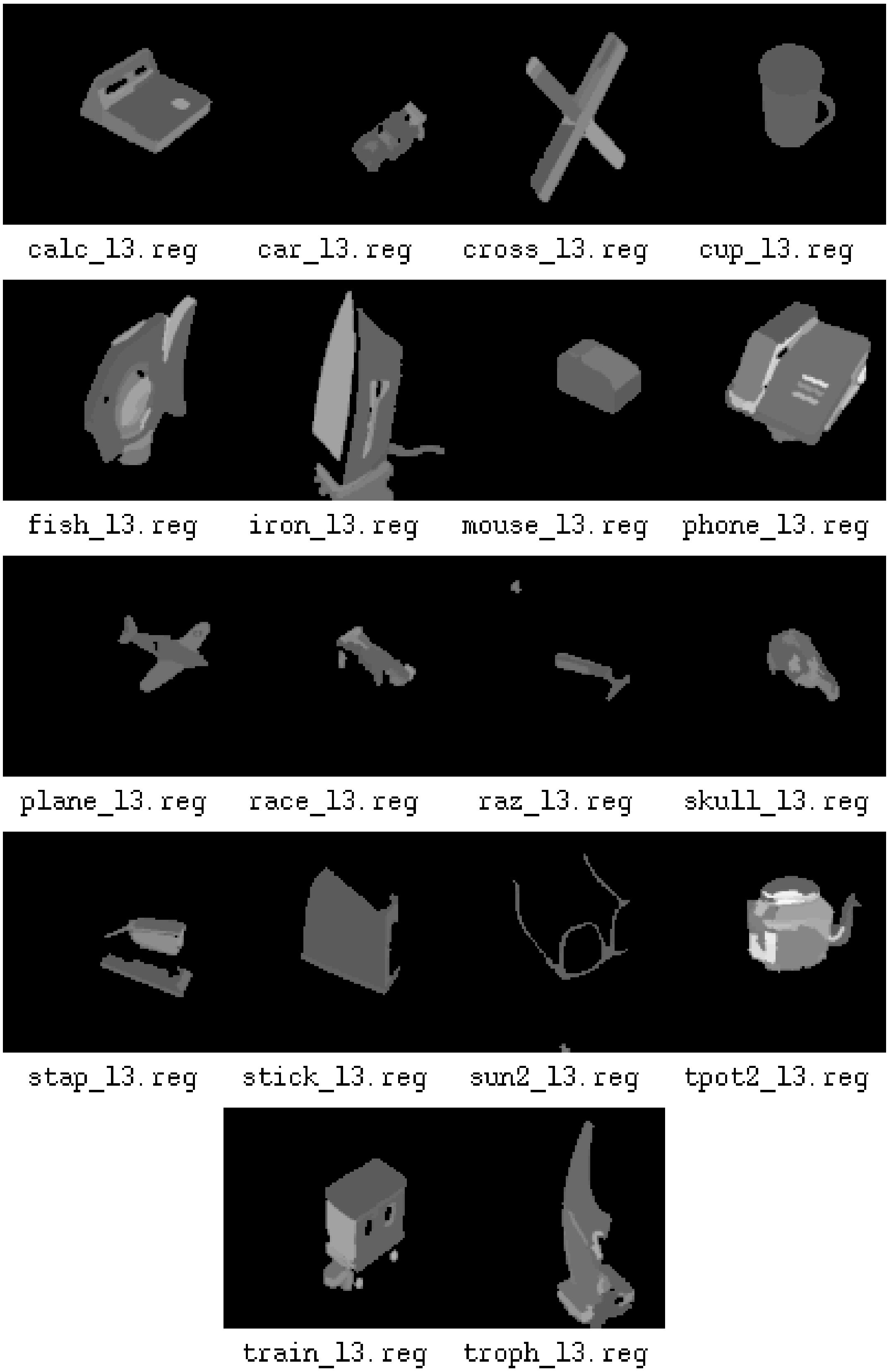



Figure 9.8: Data set 2 - extracted regions

**Algorithm 9.3.1** *Algorithm for Cross Validation*

**function** *cross_validation*$(\mathcal{V}_l, \mathcal{V}_r)$
**inputs:**
  $\mathcal{V}_l$ /* set of all left views of all objects */
  $\mathcal{V}_r$ /* set of all right views of all objects */
**endinputs**
**begin**
  **for** all poses, $i$ **do**
    $v_l \leftarrow \mathcal{V}_l[i]$
    $v_r \leftarrow \mathcal{V}_r[i]$
    $\textit{training\_data} \leftarrow \{\mathcal{V}_l + \mathcal{V}_r - v_l - v_r\}$
    $\textit{FCRG\_tree} \leftarrow \textit{FCRG\_learn}(\textit{training\_data})$
    $\textit{classify\_scene}(v_l, \textit{FCRG\_tree})$
    $\textit{classify\_scene}(v_r, \textit{FCRG\_tree})$
  **endfor**
**end**

object images was used at this stage to test the usefulness of the system.

For all the experiments, the parameters used for the FCRG classifier are as follows:

- minimum entropy equal to 0.1. That is, stop growing the tree at a cluster whose entropy is less than 0.1.
- fuzziness of clusters equal to 1.5, where a value of 1.0 is not fuzzy at all, and a value of $\infty$ is maximally fuzzy. The value of 1.5 is a common value to choose.
- a binary distance of 10 pixels with respect to a $256 \times 256$ pixels image. That is, binary attributes are extracted from parts within 10 pixels of each other.
- the maximum depth of the classification tree was set to 7, allowing for chains up to 4 parts long.
- all input data were z-score transformed (ie transformed such that each attribute had a mean of zero and a variance of 1).

These values were found to produce good results for the current experiments but may not be applicable to all situations. In particular, the binary distance parameter is highly dependent on the application area, while the other parameters should be mostly application independent.

# Chapter 10

# Results

This Chapter describes the results obtained from the experiments performed here. This includes a comparison between the performance of CRG and FCRG, evaluations of the goodness of the unary and binary attributes and the final results obtained from the best set of attributes. Also reported are the results obtained from the multiple object scenes.

## 10.1 Comparison of CRG and FCRG

It is certainly useful to see if FCRG does in fact perform better than the CRG classifier. To test the relative performance of each, we use Data Set 1 as the input data, the general shape descriptors and mean and variance of intensity as the unary attributes, and the binary attributes without depth were used. No hypothesis verification is performed with either classifier, as we just want to test how each classifier performs relative to the other. The CRG classifier achieved a recognition performance of 56%, compared to 72% for the FCRG classifier. The performance of the CRG classifier used here was the best performance found whilst optimising over the input parameters. The parameters used were to set the minimum entropy per cluster to 0.1 (ie don't split a cluster if the entropy is less than 0.1), and the minimum size of a cluster to 0.3 along all attribute axes (ie don't split a cluster along an attribute axis, if the size of the cluster along that axis is less than 0.3). The parameters for the FCRG classifier were those reported in the previous chapter. For each test scene the proportion of correct scene regions was calculated (the number of correctly recognised regions divided by the total number of regions), and this was averaged across all test scenes to produce the overall performance. The cross-validation procedure described in Algorithm 9.3 was used to generate these results.

So, the FCRG classifier has performed significantly better in this case, but it is useful to test the significance of the improved performance using a statistical significance test. In this case, we have two random variables, $X_{CRG}$ and $X_{FCRG}$, which correspond to the proportion of correctly classified scene regions in each test scene for each of the two classifiers. We don't know if $X_{CRG}$ and $X_{FCRG}$ are randomly distributed, but since we have 180 samples of each variable (one for each test scene), we can assume the sample means are normally distributed by the Central Limit Theorem ([49], pg 423), and the variances of each variable can be reliably estimated.

We want to determine, the likelihood that $\bar{X}_{FCRG}$ is greater than $\bar{X}_{CRG}$ given the above experiment. The null hypothesis, then, is:

$$H_0 : \mu_{CRG} = \mu_{FCRG} \tag{10.1}$$

with the alternative hypothesis:

$$H_1 : \mu_{FCRG} - \mu_{CRG} > 0. \tag{10.2}$$

Under the null hypothesis, the statistic:

$$Z_{\bar{X}_{FCRG}-\bar{X}_{CRG}} = \frac{\bar{X}_{FCRG} - \bar{X}_{CRG}}{\sqrt{\frac{\hat{\sigma}^2_{FCRG}}{N_{FCRG}} + \frac{\hat{\sigma}^2_{CRG}}{N_{CRG}}}} \tag{10.3}$$

is normally distributed with a mean of 0, and a variance of 1. Substituting the known values in the above equation gives us the following:

$$\begin{aligned} Z_{\bar{X}_{FCRG}-\bar{X}_{CRG}} &= \frac{0.72 - 0.55}{\sqrt{\frac{0.15}{180} + \frac{0.12}{180}}} \\ &= 4.13. \end{aligned}$$

Looking up a cumulative normal probabilities distribution using 4.13, we find that the probability of the mean of the FCRG classifier being greater than the mean of the CRG classifier is:

$$P(\mu_{FCRG} > \mu_{CRG}) > 0.99. \tag{10.4}$$

Alternatively, we can reject the null hypothesis (or accept the alternative hypothesis) with a significance of $< 0.01$, indicating that it is highly likely that the FCRG classifier has performed better than the CRG classifier in this case.

Other than recognition performance, it is also useful to discuss other relative advantages and disadvantages between the CRG and FCRG classifier. In general, the CRG classifier is faster than the FCRG classifier in learning and recognition times, although this is dependent on the parameters used for both classifiers. The CRG classifier learns quickly using default parameters, but often does not perform well (only 39% recognition with the above experiment) with these parameters, depending on the input data. The FCRG classifier, on the other hand, tends to perform fairly well using default parameters across a greater range of problems, since it estimates many of the inherent parameters of the problem by itself (eg the number of clusters per level, size of clusters etc). So, although CRG is quicker in terms of running time, the experimenter typically has to adjust the CRG parameters and repeat the experiment a number of times, to find a good set of parameters. So, in effect, the CRG classifier has no real advantage over the FCRG classifier in terms of execution time.

## 10.2 Evaluating the Unary Attributes

The unary attributes described in Chapter 6 have been evaluated by performing cross-validation tests (Algorithm 9.3) on Data Set 1. The binary attributes which were used are irrelevant except to say that the same binary attributes were used for all tests involving the unary attributes. For the results quoted here, the binary relationships were calculated between parts which were within 10 pixels (on a 256 by 256 image) of each other, and no hypothesis verification was performed. It is known that better performance for this data set can be achieved using a larger binary range, but since we are evaluating the unary attributes, it is the relative scores between the different unary attribute sets that are important.

Figure 10.1 displays a graph comparing the performance of each of the attribute sets. For the relevant attribute sets, Set 1 in the graph uses the first 16 descriptors, Set 2 the first 8 and Set 3 the first 4, except for the Fourier descriptors which do not make use of the imaginary part of the first descriptor (since its always 0). In the case of the Fourier descriptors, Set 1 makes use of the 15 non-zero attributes and likewise for Set's 2 and 3.

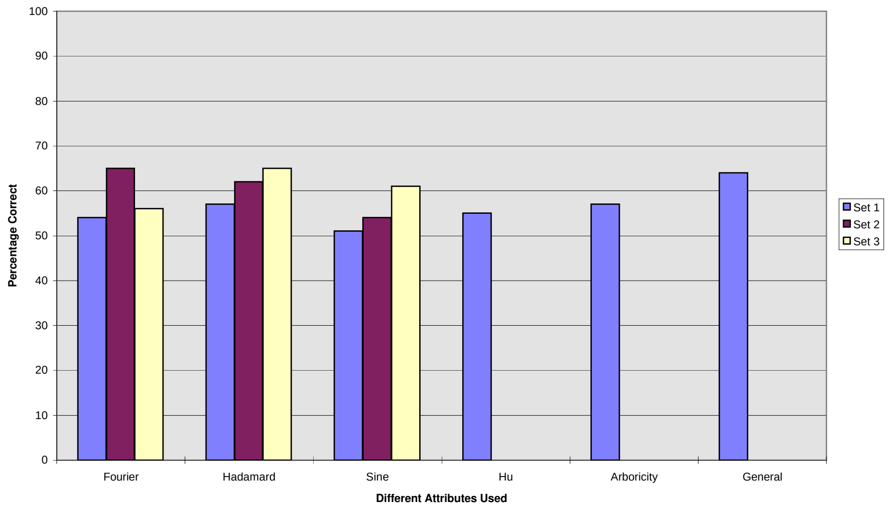



Figure 10.1: A comparison of the performance of a number of attribute sets: For the first 3 distinct attribute sets, Set 1 involves 16 descriptors, Set 2 involves 8, and Set 3 involves 4 descriptors.

As can be seen from the figure, the Fourier, Hadamard and Sine descriptors all perform approximately the same for the optimal number of descriptors, with the Hadamard descriptors performing the best overall (not surprising considering the data). Also, the general shape descriptors perform just as well as the best of the reconstructionist approaches. So, it is unclear from this data which would be the best set of attributes to use. The best set of Fourier descriptors makes use of 7 descriptors which begins to make the dimensionality of the unary attribute space too high. The Hadamard and Sine descriptors, on the other hand, perform best with only 4 descriptors, but the Sine descriptors do not work quite as well as the others. The Hadamard descriptors have somewhat limited applicability, since they are inefficient at representing shapes with smooth curves, and although most of the objects used here are well represented by Hadamard descriptors, it was thought best to use the most general attributes. The Hu invariants and the arboricity measures are not as effective as the others. Because of these reasons, and since it appears not to be too critical a decision anyway, it was decided to make use of the general shape descriptors for the remainder of the experiments.

## 10.3 Evaluating the Binary Attributes

The binary attributes are evaluated using a cross-validation procedure (Algorithm 9.3) and using Data Set 2. Hypothesis verification was used here and the general shape descriptors were used for the unary attributes. The mean and variance of the intensity values inside a region were also used to augment the unary attributes. This amounts to the best results that have been obtained thus far using the complete ORS with Data Set 2. The results for the cross validation tests are shown in Table 10.1. In the table, the percentage of correctly classified regions are displayed for each of the objects, for each of the different tests which were carried out. The tests involved using different combinations of the binary attributes related to depth values (obtained from the depth-from-stereo procedure described in Chapter 5) to test which were the most appropriate for the recognition task. The attribute sets which were used (and are listed in the table) include:

1. All binary attributes ($A$).
2. All attributes except the depth attributes (maximum depth difference, minimum depth difference, plane angle, $ND$).
3. All attributes except maximum depth difference and plane angle ($NXA$).
4. All attributes except minimum depth difference and plane angle ($NMA$).
5. All attributes except plane angle ($NA$).
6. All attributes except 2d maximum distance ($N2D$).

The average correct classification of object regions for the single object scenes ranges from 74.1% to 78.2% depending on which feature set was used. It is interesting to note that these results include attempting to recognise the views on the extremes of the poses (ie the first and last poses - $0°$ and $90°$) which is particularly

| Object | Recognition results for different attributes | | | | | |
|---|---|---|---|---|---|---|
| | $A$ | $ND$ | $NXA$ | $NMA$ | $NA$ | $N2D$ |
| Calculator | 92.5 | 92.5 | 94.3 | <u>**99.4**</u> | 97.1 | 94.3 |
| Model Car | 73.1 | 66.7 | 66.7 | <u>**83.9**</u> | 82.8 | 52.7 |
| Wooden Cross | 84.8 | 79.3 | 80.4 | <u>**90.2**</u> | 85.9 | 88.0 |
| Cup | 80.0 | 66.7 | 57.8 | 60.0 | <u>**93.3**</u> | 53.3 |
| Wooden Fish | 48.7 | 46.2 | 48.7 | 49.6 | <u>**58.8**</u> | 46.2 |
| Iron | 78.0 | 81.9 | 75.6 | 74.8 | 75.6 | <u>**83.5**</u> |
| Mouse | 88.1 | 89.5 | 90.8 | <u>**90.8**</u> | 89.4 | 76.3 |
| Phone | 95.9 | 90.7 | <u>**97.4**</u> | 95.9 | 96.9 | **97.4** |
| Model Plane | 62.1 | 47.9 | 54.6 | <u>**70.6**</u> | 49.6 | 58.8 |
| Model Car 2 | 62.2 | 47.9 | 54.6 | <u>**70.6**</u> | 50.0 | 58.8 |
| Razor | 40.0 | 38.9 | 40.0 | 43.3 | 42.2 | <u>**44.4**</u> |
| Wallaby Skull | <u>**85.1**</u> | 80.0 | 83.9 | 82.6 | 80.6 | 83.9 |
| Stapler | 81.3 | 75.0 | 79.2 | 81.3 | <u>**83.3**</u> | 80.2 |
| Tape Holder | <u>**66.2**</u> | 64.7 | <u>**66.2**</u> | 64.7 | <u>**66.2**</u> | 64.7 |
| Sun Glasses | 9.3 | 9.3 | 9.3 | 9.3 | 9.3 | 7.4 |
| Tea Pot | 85.2 | <u>**86.0**</u> | 85.6 | 85.6 | 84.8 | 85.6 |
| Model Train | 85.7 | 82.9 | 86.7 | <u>**87.6**</u> | 86.7 | 81.0 |
| Trophy | 68.1 | 86.2 | 79.0 | 79.7 | <u>**92.7**</u> | 79.7 |
| Total | 75.4 | 74.1 | 74.7 | 77.3 | **78.2** | 74.2 |

Table 10.1: Results for the single object tests

difficult since these views are outside of the pose ranges of all the views being used for training. If the recognition results of these views are not included in the analysis, then the best recognition achieved is 86.7%. This is not an unexpected result and possibly a clearer indication of the performance of the technique, since it unreasonable to expect a system to classify an object in a pose which is outside the pose range of the training data.

It is clear from Table 10.1 that certain binary attributes are more useful than others. Specifically, looking at the depth attributes, without any depth attributes, the recognition results are 74.1% recognition rate. Using all attributes results in a recognition rate of 75.4% - a marginal improvement. However, if the plane angle is removed from the attribute space, then the recognition performance improves to a 78.2% recognition result - a significant improvement. Clearly, the plane angle binary attribute reduces the performance of the classifier, whereas the other binary depth attributes (maximum and minimum depth distances) improve the performance.

The current results indicate that sparse depth data obtained from stereo can be useful for object recognition, but these results are not conclusive. Certainly, it is unclear why the plane angle does not aid in recognition performance. However, there are a number of possibilities:

1. The plane fitting routine did not produce an accurate fit due to the sparseness of the data.

2. The angle between parts (ie planes) is not a useful attribute for recognition (given the current data set).

3. The depth data produced by the current stereo system is too sparse to extract truly useful information for recognition.

Any of the above reasons seem likely, with points 1 and 3 the most likely of the three. Although the other depth attributes did improve recognition performance using the same plane data, this may be an artifact of the test data. Certainly, the use of sparse depth data needs to studied in more detail before any conclusive statement can be made about the viability of such data for recognition purposes.

Most of the objects were recognised extremely well. Notably, the most complex objects were often recognised the best, including the phone, the teapot and the wallaby skull. The teapot and the wallaby skull especially highlight the advantages of using the approach taken here over more traditional approaches (eg CAD modelling). Firstly, it would be difficult to accurately model the skull and since it consists of complex, often hollow, components, recognition would be particularly difficult. Secondly, the teapot could be easily modelled, but because of the highly specular nature of the teapot's surface, it is very difficult to correctly segment an image of the teapot into regions which represent whole surfaces. The alternative approach, which was used in this study, was to model an object based on its appearance, hence, it is not important to segment an object into regions representing any sort of underlying object structure, the segmentation merely has to be reasonably stable across a number of views and be able to produce a discriminating model.

The objects which were recognised the least were the sunglasses and the razor. The sunglasses were very difficult to recognise for two reasons: the difficulty in extracting the lenses as regions in the images (since they are transparent), and a deficiency with the hypothesis verification routine. The hypothesis verification routine does not perform exhaustive template matching, rather the centroids of regions are aligned, and the correlation is calculated from this match. If the regions are skinny and irregular (as for the sunglasses), then it is likely that aligning the centroids of regions will cause the actual regions to be misaligned. For this reason, the hypothesis verification routine failed to recognise the sunglasses. In fact, FCRG performed better on the sunglasses without hypothesis verification, achieving a recognition rate of 43% for the tests using the NA attribute set. If a better hypothesis verification method were used, one would expect to perform significantly better than this. The razor was not recognised well for a number of reasons. Firstly, some of the important features of the razor (eg the blade) were too small to be correctly extracted using the above region extraction techniques, and, in addition, the attribute extraction techniques lose their discriminating power with small regions (the regions are not big enough to produce reliable discriminating attributes due to the quantisation effect of the pixels). Secondly, the (rectangular) regions which were extracted occurred in many of the objects and hence there were not enough binary relationships to resolve ambiguities.

## 10.4 Multiple Object Scenes

In this experiment, a single scene was presented for classification involving a number of objects (typically 2-4). The image was preprocessed in the same way as described above. The objects were presented in similar, but not necessarily identical, pose to the training data. This problem is considerably more difficult than the single object scenes for a number of reasons:

1. The segmentation of multiple object scenes can lead to single regions which cross object boundaries making it impossible to correctly recognise the whole region.

2. Chains are often formed which cross object boundaries and these cause confusing evidence.

3. There are generally more chains in a multiple object scene which makes recognition more difficult.

4. Two similar objects in a scene can create extra confusion especially if chains cross over these objects boundaries.

5. Specifically related to the current experiments - often objects in a multiple object scene are of different size than the training views, which confused the hypothesis verification stage of the system (which is based on correlation).

Despite these difficulties, the classifier does perform well in some cases, although there are many cases for which it does not perform well.

Figures 10.2 to 10.9 display the recognition results for a number of different scenes containing various objects. In the images the flecked pattern indicates a correctly recognised region, while the lattice pattern indicates an incorrectly recognised region. The images display scenes of varying complexity with the simpler images generally being recognised better than the more complex ones as would be expected. In some of the images (10.3, 10.8), the objects were correctly recognised except for regions which overlapped other objects as in the top of the stapler in both of the mentioned images. This is due to the correlation effect of the hypothesis verification procedure. Figure 10.5 was recognised fairly well except for the model car and the middle band of the teapot. The middle band of the teapot was incorrectly recognised due to incorrect segmentation of the intensity image - as can be seen from the segmented image of the scene, the middle band of the teapot and the face of the iron were considered to be the same region due to the absence of an obvious edge between the two regions probably due to the reflective properties of the teapot's surface. The model car was incorrectly recognised in this image due to the small size of the car in the image and the subsequent lack of detail of the car and its segmentation. Figure 10.6 is the most complex of the images used for testing (containing 5 objects) and was somewhat poorly recognised. Again, the main body of the plane was incorrectly recognised as it was considered to be part of the main region of the sticky tape holder. The sunglasses were incorrectly recognised for reasons discussed in Section 10.3, and the model car was largely misclassified due to crossing chains with the sticky tape holder. In Figure 10.9 the calculator is classified as the phone object. This is not an unexpected result as the calculator and the phone are quite similar objects, and the presence of crossing chains between the two objects would cause further confusion.

The overall performance of the recognition algorithm for multi-object scenes was fairly poor. For 33 multi-object scenes, of a similar order of complexity as the scenes shown in figures 10.2 to 10.8, segmented regions (ie parts) were recognised at a rate of 39%. Clearly, more work is required in the area of recognising objects in complex scenes.

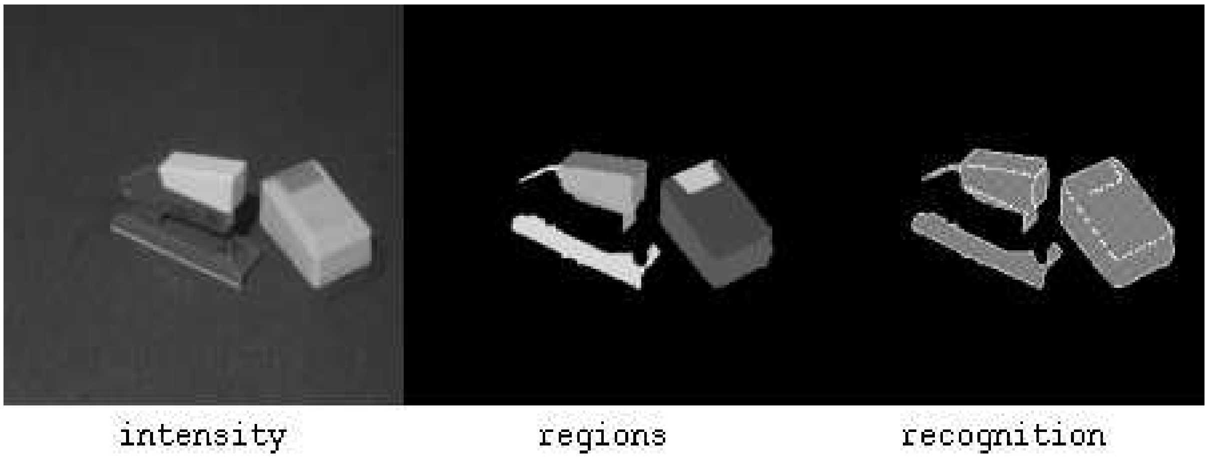



Figure 10.2: The mouse and the stapler.

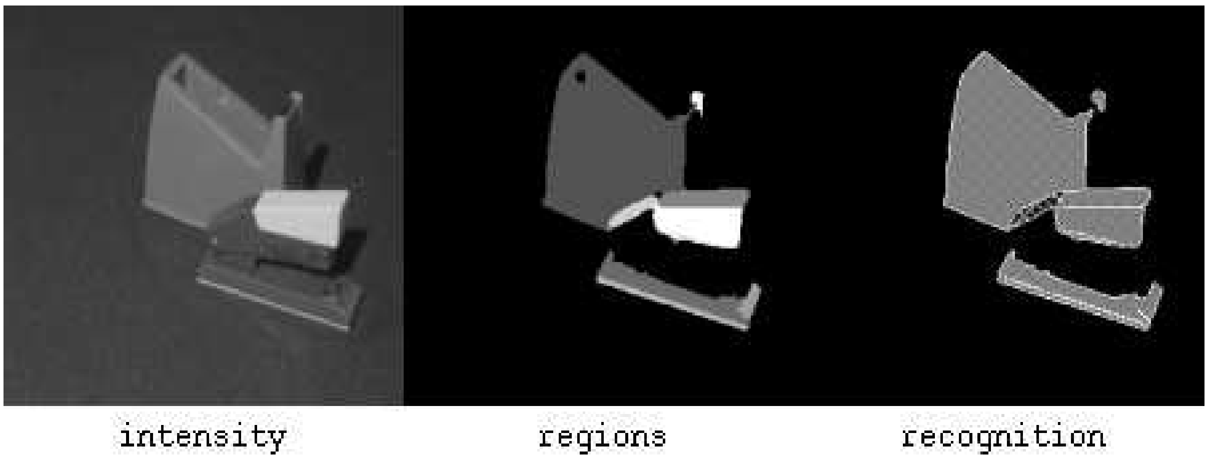



Figure 10.3: The stapler and the sticky tape holder: The back end of the stapler was recognised as part of the sticky tape holder due to the hypothesis verification algorithm

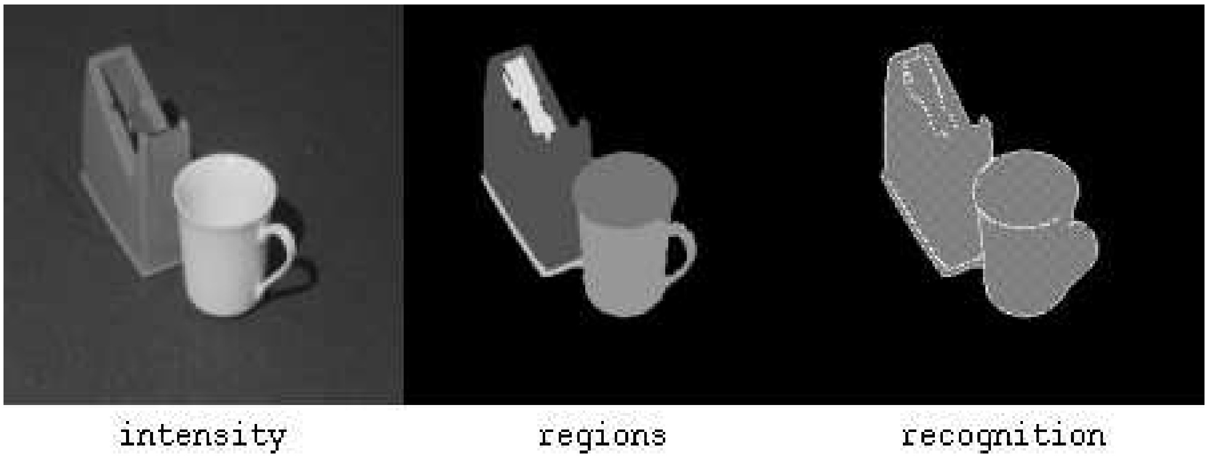



Figure 10.4: The cup and the sticky tape holder.

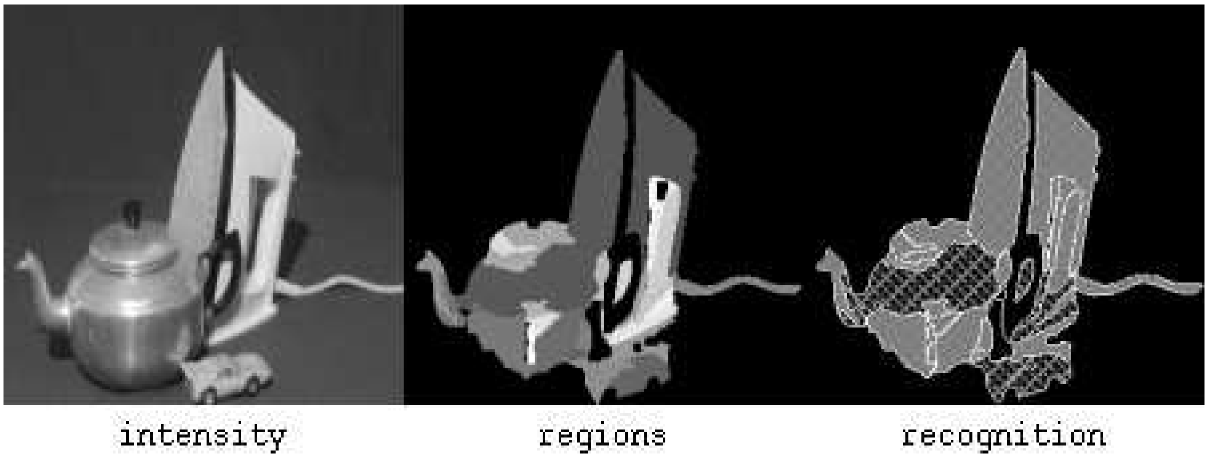



Figure 10.5: The teapot, iron and the model car: The middle section of the teapot was misclassified because of incorrect segmentation and the car was misclassified due to lack of detail.

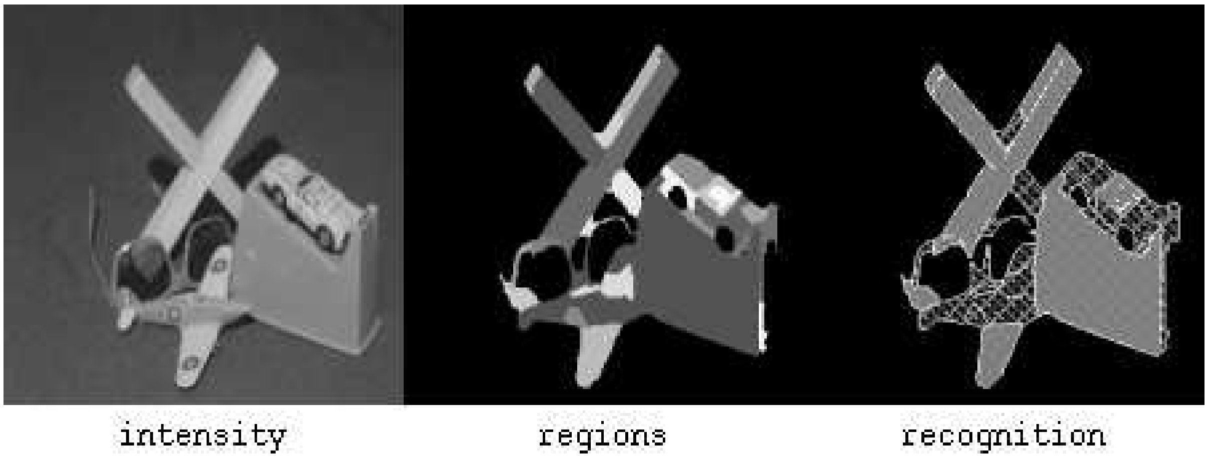



Figure 10.6: The cross, sunglasses, plane, car and the sticky tape holder.

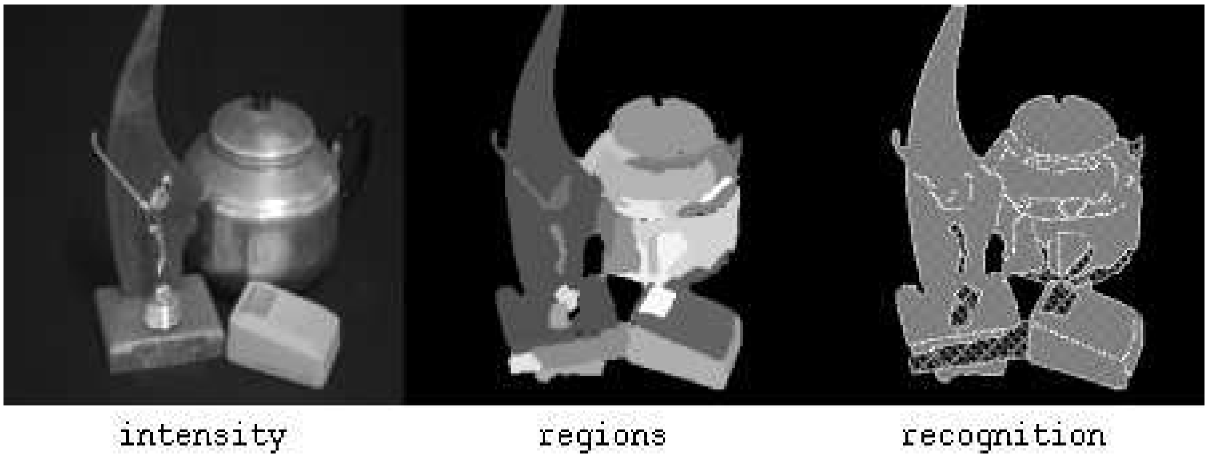



Figure 10.7: The mouse, trophy and teapot.

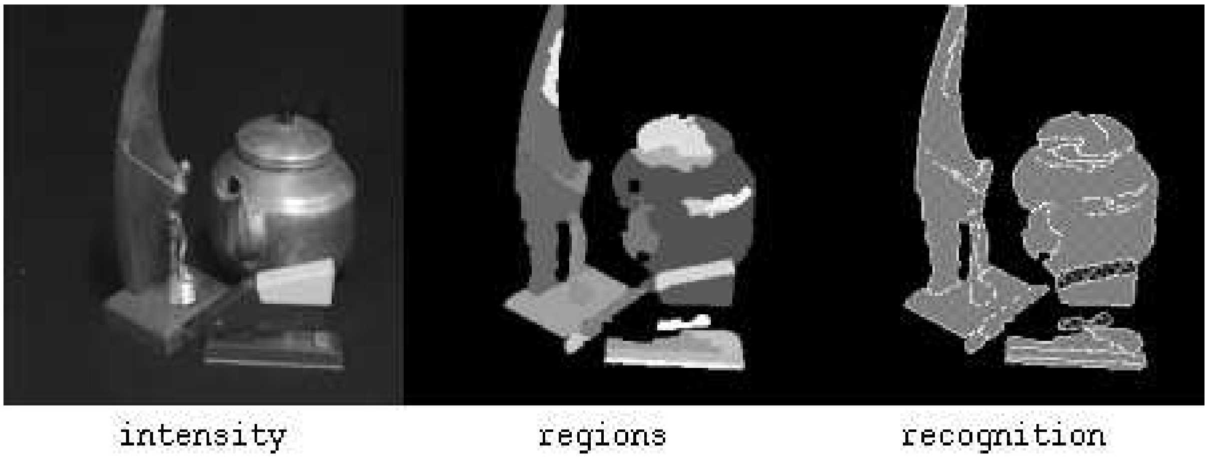



Figure 10.8: The stapler, trophy and the teapot.

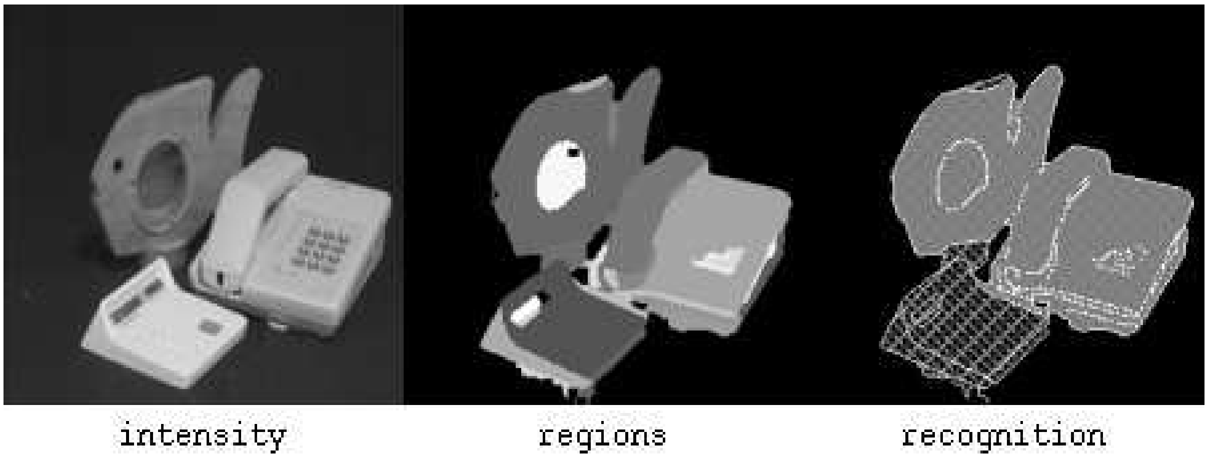



Figure 10.9: The calculator, fish and phone: The calculator was misclassified as the phone - not surprising since the two objects are quite similar.

# Chapter 11

# Conclusions

To date, current 3D object recognition systems which attempt to solve the problem of recognising 3D objects from 2D intensity images typically do so within a very limited domain. Some systems solve the problem with only one or two objects in the object database (eg ACRONYM [25, 27], SCERPO [74]), others solve a simplified problem (the wireframe objects of Poggio and Edelman [92]), whilst others assume parts of the problem are already solved (Bergevin and Levine [9] in their PARVO system). Also, most systems that have a non-trivial database of objects assume that the objects are already partitioned in the image (ie the system needs only to recognise one object at a time), assuming that this problem will be solved in the future. Although valid for developing the recognitive aspects of such systems, and for use in specialised domains involving only one object per scene, for general vision, this is a naive approach. It seems to be an impossible problem to expect that objects in a complex intensity image can be separated from each other without first having knowledge about the objects. In other words, objects can't be separated until they are recognised, and for these systems, objects can't be recognised until they are separated. No systems thus far have attempted to address all of the problems inherent in such a domain, from a machine learning perspective.

In this dissertation, a 3D object recognition system has been described which can learn to recognise 3D objects from 2D intensity images. The system is a significant contribution to computer vision research since it begins to address the problem of 3D object recognition from an holistic point of view rather than dealing with only a sub-problem as has been the predominant approach in the computer vision literature. The system as presented: makes use of a novel segmentation algorithm to find the object parts in a 2D intensity image; incorporates sparse depth data obtained from a novel depth-from-stereo algorithm to aid in learning and recognition; and the CRG classifier was extended to a fuzzy machine learning classifier, which is more appropriate than the original crisp algorithm, for learning 3D objects from 2D intensity images. The primary contributions of the dissertation are the development of the novel segmentation and depth-from-stereo algorithms, the extension of the CRG classifier to a fuzzy machine learning classifier, and the integration of these routines into a complete object recognition system - something which has not been previously reported in the context of learning 3D objects from 2D intensity images.

The system as presented is quite successful with a recognition rate of 78% from a database of 18 non-trivial 3D objects, for single object scenes. Although the system is not as successful for multiple object scenes, it does demonstrate that such a system is possible, and that it is an area of useful research.

## 11.1 Future Work

There are many areas where the system can be improved and extended. Most notably, a more sophisticated hypothesis verification system could be implemented which would hopefully improve the performance of the system especially on the multiple object scenes. Currently, the hypothesis verification routine is a simple template matching correlation based approach, and as such is not size or rotation invariant. This is a particular problem with the multiple object scenes as the objects often appear at different sizes than in the learning scenes, thus reducing the performance of the system with the multiple object scenes. Since graph-matching techniques are typically size and rotation invariant, an obvious improvement could be obtained by implementing one of the many approaches reported in the literature ([6], [110], [68], [45], [119]).

The FCRG classifier is another area which requires significant research effort to fully explore its capabilities and shortcomings. Specifically, it could be extended by including the capacity for non-spherical (eg ellipsoidal) fuzzy clusters in the fuzzy clustering algorithm in a manner similar to that reported by Gustafson and Kessel ([56]). Such an approach would improve the generality of the cluster formation process, allowing for long thin clusters where appropriate.

Also, the tree building algorithm could possibly be improved by examining alternative stopping criterion like the minimum message length approach developed by Wallace [114, 115]. The minimum message length idea is to allow expansions only if the total message length of the data structure decreases. In the FCRG case, this would include the storage requirements for the tree itself and the requirements for storing the particular data points in the tree. This criterion could be applied in a number of ways. Firstly, the number of clusters at each level of the tree could be determined by choosing the optimal number of clusters using the minimum message length criterion. Secondly, the decision to expand a particular cluster could be determined depending on whether the message length of the tree decreases with cluster expansion. Such a procedure would produce more compact tree representations, thus improving generalisation abilities of the tree and hopefully improving recognition performance.

## 11.2 Some Final Comments

In performing the research which was reported in this dissertation, it is clear to me that computer vision research has to move towards integrated systems that address the entire problem rather than treating each part independently. Different sub-systems can then be evaluated in terms of their required application rather than some arbitrary measure of goodness. There are two primary types of systems which can be constructed. An approach similar to the one described here can be taken, with essentially a linear control mechanism where the output of one module is used as the input to another with complete confidence. Alternatively, a more complex control structure could be used where modules can make use of different forms of input depending on what's available, and where confidence levels are assigned to each of the results produced by the sub-systems, which other sub-systems can make use of. Blackboard systems are examples of frameworks from which such systems can be built. It is believed that such systems will be necessary if the ultimate goals of computer vision (vision systems that approach human performance) are to be obtained.

# Bibliography

[1] Yaser S. Abu-Mostafa and Demetri Psaltis. Recognitive aspects of moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):698–706, November 1984.

[2] Yaser S. Abu-Mostafa and Demetri Psaltis. Image normalisation by complex moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):46–55, January 1985.

[3] H. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computing*, C-23:90–93, 1974.

[4] N. Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *International Journal of Computer Vision*, pages 107–131, 1987.

[5] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, New Jersey, Zip: 07632, 1982.

[6] Bassam Bamieh and Rui J.P. De Figueiredo. A general moment-invariants/attributed-graph method for three-dimensional object recognition from a single image. *IEEE Journal of Robotics and Automation*, RA-2(1):31–41, March, 1986.

[7] S.O. Belkasim, M. Shridhar, and M. Ahmadi. Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recognition*, 24(12):1117–1138, 1991.

[8] John R. Bennett and John S. Macdonald. On the measurement of curvature in a quantised environment. *IEEE Transactions on Computers*, C-24(8):803–820, August 1975.

[9] Robert Bergevin and Martin D. Levine. Generic object recognition: building and matching coarse descriptions from line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):19–36, January, 1993.

[10] Robert Bergevin and Martin D. Levine. Extraction of line drawing features for object recognition. *Pattern Recognition*, 25(3):319–334, March, 1992.

[11] Paul J. Besl and Ramesh C. Jain. Invariant surface characteristics for 3D object recognition in range images. *Computer Vision, Graphics and Image Processing*, 33:33–80, 1986.

[12] J.C. Bezdek. Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology*, 1:57–71, 1974.

[13] J.C. Bezdek. Mathematical models for systematics and taxonomy. In G Estabrook, editor, *Proceedings 8th International Conference on Taxonomy*, pages 143–166, San Francisco, CA, 1975. Freeman.

[14] J.C. Bezdek. *Pattern recognition with fuzzy objective function algorithms.* Plenum, New York, 1981.

[15] Irving Biederman. Human image understanding: recent research and a theory. *Computer Vision, Graphics and Image Processing*, 32:29–73, 1985.

[16] T.O. Binford. Visual perception by computer. In *IEEE Systems Science and Cybernetics Conference*, Miami, December 1971. IEEE. Invited Paper.

[17] Walter Bischof and Terry Caelli. Learning structural descriptions: a new technique for conditional clustering and rule generation. *Pattern Recognition*, pages 689–697, May 1994.

[18] Walter F. Bischof and Terry Caelli. Scene understanding by rule evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Under revision.

[19] Walter F. Bischof and Terry Caelli. Visual learning of patterns and objects. *IEEE Transactions System, Man and Cybernetics*. In Press.

[20] H. Blum. Biological shape and visual science. *Journal of Theoretical Biology*, 38:205–287, 1973.

[21] C. Bouman and B. Liu. Multiple resolution segmentation of textured images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):99–113, 1991.

[22] Kevin Bowyer and David Eggert. Computing the generalised aspect graph for objects with moving parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):605–610, June 1993.

[23] M Brady. Criteria for representation of shapes. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 39–84. New York, Academic, 1983.

[24] M. Brady and H. Asada. Smoothed local symmetries and their implementation. *International Journal of Robotics Research*, 3:36–61, Fall 1984.

[25] Rodney A. Brooks. Symbolic reasoning among 3D models and 2D images. *Artificial Intelligence*, 17:285–348, 1981.

[26] Rodney A. Brooks. A layered robust control system for a mobile robot. *Journal of Robotics and Automation*, 2(1):14–25, March 1986.

[27] Rodney A. Brooks. Model-based 3D interpretations of 2D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, March, 1983.

[28] Roberto Brunelli and Stefano Messelodi. Robust estimation of correlation with applications to computer vision. *Pattern Recognition*, 28(6):833–841, 1995.

[29] J. Brian Burns, Richard S. Weiss, and Edward M. Riseman. View variation of point-set and line-segment features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):51–68, January, 1993.

[30] P. Burt and B. Julesz. A disparity gradient limit for binocular fusion. *Science*, 208:615–617, 1980.

[31] Terry Caelli and Ashley Dreier. Variations on the evidence-based object recognition theme. *Pattern Recognition*, 27(2):185–204, 1994.

[32] John F. Canny. Finding edges and lines in images. Technical Report AI-TR-720, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA, 1983.

[33] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November, 1986.

[34] I-Cheng Chang and Chung-Lin Huang. Aspect graph generation for non-convex polyhedra from perspective projection view. *Pattern Recognition*, 25(10):1075–1096, 1992.

[35] Robin L. P. Chang and Theodosios Pavlidis. Fuzzy decision tree algorithms. *IEEE Transactions Systems, Man and Cybernetics*, SMC-7(1):28–35, January 1977.

[36] Chen-Chau Chu and J.K. Aggarwal. The integration of image segmentation maps using region and edge information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1241–1252, December 1993.

[37] Umesh R. Dhond and J.K. Aggarwal. Structure from stereo - a review. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1489–1510, Nov/Dec 1989.

[38] Sven J. Dickinson, Alex P. Pentland, and Azriel Rosenfeld. 3D shape recovery using distributed aspect matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):174–198, February, 1992.

[39] Sven J. Dickinson, Alex P. Pentland, and Azriel Rosenfeld. From volumes to views: an approach to 3D object recognition. *Computer Vision, Graphics and Image Processing: Image Understanding*, 55(2):130–154, Mar, 1992.

[40] Craig Dillon and Terry Caelli. Generating complete depth maps in passive vision systems. In *11th IAPR International Conference on Pattern Recognition*, pages 562–566. International Association for Pattern Recognition, IEEE Computer Society Press, 1992.

[41] Craig Dillon and Terry Caelli. Shape from virtual aperture focus. Technical report, Department of Computer Science, University of Melbourne, October 1992.

[42] Bruce A. Draper. *Learning object recognition strategies.* PhD thesis, University of Masachusetts, May 1993.

[43] S Edelman and D Weinshall. A self-organising multiple-view representation of 3D objects. *Biological Cybernetics*, 64:209–219, 1991.

[44] David Eggert and Kevin Bowyer. Computing the perspective projection aspect graph of solids of revolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):109–127, February 1993.

[45] Ting-Jun Fan, Gerard Medioni, and Ramakant Nevatia. Recognising 3D objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, November, 1989.

[46] T.J. Fan. *Describing and recognising 3D objects using surface properties.* PhD thesis, University of Southern California, Aug 1988.

[47] Patrick J. Flynn and Anil K. Jain. BONSAI: 3D object recognition using constrained search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1066–1075, 1991.

[48] Patrick J. Flynn and Anil K. Jain. 3D object recognition using invariant feature indexing of interpretation tables. *Computer Vision, Graphics and Image Processing: Image Understanding*, 55(2):119–129, March 1992.

[49] Harry Frank and Steven C. Althoen. *Statistics: Concepts and Applications.* Cambridge University Press, 1994.

[50] Pascal Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. *Vision*, pages 1292–1298, 1991.

[51] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley, 1992.

[52] Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[53] W. Eric L. Grimson. A computer implementation of a theory of human stereo vision. *Philosophical Transactions Royal Society London*, B 292:217–253, 1981.

[54] W. Eric L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, January 1985.

[55] R. Gunderson. Applications of fuzzy isodata algorithms to start tracker pointing systems. In *Proceedings 7th Triennial World IFAC Congress*, pages 1319–1323, Helsinki, Finland, 1978.

[56] Donald E. Gustafson and William C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In James C. Bezdek and Sankar K. Pal, editors, *Fuzzy Models for Pattern Recognition*, pages 117–122. IEEE Press, 1992.

[57] John F. Haddon and James F. Boyce. Image segmentation by unifying region and boundary information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):929–948, October 1990.

[58] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29:100–132, 1985.

[59] S. Haring, M.A. Viergever, and J.N. Kok. Kohonen networks for multi-scale image segmentation. *Image and Vision Computing*, 12(6):339–344, 1994.

[60] J.J. Hopfield and D.W. Tank. Neural networks and physical systems with emergent collective computational abilities. *Proceedings National Academy of Science*, 79:2554–2558, April 1982.

[61] B.K.P. Horn and M.J. Brooks. The variational approach to shape from shading. *Computer Vision, Graphics and Image Processing*, 33(2):174–208, 1986.

[62] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions Information Theory*, IT-8:179–187, February 1962.

[63] R.A. Hummel and S.W. Zucker. On the foundations of relaxation labeling process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(3):267–287, May 1983.

[64] A Jain and R.C. Dubes. *Algorithms for clustering data.* Englewood Cliffs: Prentice Hall, 1988.

[65] Anil K. Jain. A fast Karhunen-Loeve transform for finite discrete images. In *Proceedings of the National Electronics Conference*, pages 323–328, Chicago, IL, October 1974.

[66] Anil K. Jain and Richard Hoffman. Evidence-based recognition of 3D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):783–802, November, 1988.

[67] Nak H. Kim and Alan C. Bovik. A contour-based stereo matching algorithm using disparity continuity. *Pattern Recognition*, 21(5):505–514, 1988.

[68] Whi-Yul Kim and Avinash C. Kak. 3D object recognition using bipartite matching embedded in discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):224–251, March, 1991.

[69] G. Koepfler, C. Lopez, and J.M. Morel. A multi-scale algorithm for image segmentation by variational method. *SIAM Journal of Numerical Analysis*, 31(1):282–299, Feb 1994.

[70] Raghu Krishnapuram and James M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, May 1993.

[71] G Libert and M. Roubens. New experimental results in cluster validity of fuzzy clustering algorithms. In Janssen J., J.F. Macrotorchino, and J.M. Proth, editors, *New trends in data analysis and applications*, pages 205–218, Amsterdam, The Netherlands, 1983. North Holland.

[72] Wei-Chung Lin, Fong-Yuan Liao, Chen-Kuo Tsao, and Theresa Lingutla. A hierarchical multiple-view approach to three-dimensional object recognition. *IEEE Transactions on Neural Networks*, 2(1):84–92, January 1991.

[73] J.Q. Liu and Y.H. Yang. Multiresolution color image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):689–700, 1994.

[74] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[75] Song De Ma, Shao Hua Si, and Zhi Yi Chen. Quadric curve based stereo. In *International Conference on Pattern Recognition*, volume A-1, pages 1–4, 1992.

[76] Sidhartha Maitra. Moment invariants. *Proceedings of the IEEE*, 67(4):697–699, April 1979.

[77] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings Royal Society of London*, B 207:187–217, 1980.

[78] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings Royal Society London*, B 204:301–328, 1979.

[79] J.E.W. Mayhew and J.P. Frisby. Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, 17:349–385, 1981.

[80] Brendan McCane, Terry Caelli, and Olivier de Vel. Learning and recognising 3D objects from 2D views. Technical Report 95/12, James Cook University of North Queensland, 1995.

[81] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics and Image Processing*, 31:2–18, 1985.

[82] Rakesh Mohan and Ramakant Nevatia. Perceptual organisation for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):616–635, June, 1992.

[83] B.S. Morse, S.M. Pizer, and A. Liu. Multi-scale medial analysis of medical images. *Image and Vision Computing*, 12(6):327–338, 1994.

[84] Joe L. Mundy, R. Curwen, J. Liu, C. Rothwell, Andrew Zisserman, and D. Forsyth. MORSE: An architecture for 3D object recognition based on invariants. In *Second Asian Conference on Computer Vision*, volume II, pages 16–21. School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, December 1995.

[85] Joe L. Mundy, C. Huang, J Liu, W. Hoffman, D.A. Forsyth, C.A. Rothwell, Andrew Zisserman, S. Utcke, and 0. Bournes. MORSE: A 3D object recognition system based on geometric invariants. In *23rd Image Understanding Workshop*, pages 1393–1402, 1994.

[86] Nasser M. Nasrabadi. A stereo vision technique using curve segments and relaxation matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(5):566–572, May, 1992.

[87] Ramakant Nevatia. Personal Communication, February 1996.

[88] University of New Mexico. Khoros. Software System, 1992. Copyright.

[89] Theo Pavlidis and Yuh-Tay Liow. Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, March 1990.

[90] Alex Paul Pentland. A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:523–531, July 1987.

[91] Sylvain Petitjean, Jean Ponce, and David J. Kriegman. Computing exact aspect graphs of curved objects: algebraic surfaces. *International Journal of Computer Vision*, 9(3):231–255, 1992.

[92] T Poggio and S Edelman. A network that learns to recognise 3D objects. *Nature*, 343:263–266, 18th January 1990.

[93] S.B. Pollard, J.E.W. Mayhew, and J.P. Frisby. PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1981.

[94] J. Porrill, S.B. Pollard, T.P. Pridmore, J.B. Bowen, J.E.W. Mayhew, and J.P. Frisby. TINA: a 3D vision system for pick and place. *Image and Vision Computing*, 6(2):91–99, May 1988.

[95] William K. Pratt. *Digital Image Processing*. John Wiley and Sons, 2nd edition, 1991.

[96] K. Prazdny. Detection of binocular disparities. *Biological Cybernetics*, 52:93–99, 1985.

[97] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and monographs in computer science. Springer-Verlag, 1985.

[98] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C - the art of scientific computing*. Cambridge University Press, 2nd edition, 1992.

[99] Luc Robert and Olivier D. Faugeras. Curve based stereo: figural continuity and curvature. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition '91*, pages 57–62, June 3-6, 1991.

[100] Paul Rosin. Representing curves at their natural scales. *Pattern Recognition*, 25(11):1315–1325, 1992.

[101] Paul L. Rosin and Geoff A.W. West. Segmenting curves into elliptic arcs and straight lines. *International Conference on Computer Vision*, 3:75–78, 1990.

[102] Enrique H. Ruspini. A new approach to clustering. *Information Control*, 15(1):22–32, July 1969.

[103] Michael Seibert and Allen Waxman. Adaptive 3D object recognition from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):107–123, Feb, 1992.

[104] Shujun Shang, Geoff D. Sullivan, and Keith D. Baker. The automatic construction of a view-independent relational model for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):531–543, June 1993.

[105] J. Shen and S. Castan. An optimal linear operator for edge detection. *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition '86*, 1986.

[106] J. Shen and S. Castan. Edge detection based on multi-edge models. *Proceedings of The International Society for Optical Engineering (SPIE) '87*, 1987.

[107] Doron Sherman and Shmuel Peleg. Stereo by incremental matching of contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1102–1106, November 1990.

[108] Yoshiaki Shirai. *Three-Dimensional Computer Vision*. Symbolic Computation. Springer Verlag, 1st edition, 1987.

[109] M . Spann and A.E. Grace. Adaptive segmentation of noisy and textured images. *Pattern Recognition*, 27(12):1717–1733, 1994.

[110] Fridtjof Stein and Gerard Medioni. TOSS - a system for efficient three dimensional object recognition. *Image Understanding Workshop*, pages 537–543, 1990.

[111] Fridtjof Stein and Gerard Medioni. Structural indexing: efficient 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–145, Feb, 1992.

[112] Michael Reed Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, August 1980.

[113] C.S. Wallace. MML inference of predictive trees, graphs and nets. Australian Winter School in Computer Science, Magnetic Island, 2-4 July 1995. James Cook University of North Queensland.

[114] C.S. Wallace and P.R Freeman. Estimation and inference by compact coding. *The Journal of the Royal Statistical Society, Series B. Methodology*, 49(3):223–265, 1987.

[115] C.S. Wallace and P.R. Freeman. Single factor analysis by MML estimation. *The Journal of the Royal Statistical Society, Series B. Methodology*, 54(1):195–209, 1992.

[116] M.P. Windham. Cluster validity for fuzzy clustering algorithms. *Fuzzy Sets and Systems*, pages 177–185, 1981.

[117] M.P. Windham. Cluster validity for fuzzy c-means algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4), July 1982.

[118] M.P. Windham, H. Bock, and H.F. Walker. Clustering information from convergence rate. In *Proceedings 2nd Conference International Federation Classification Society*, page 143, Washington, DC, 1989.

[119] Andrew K.C. Wong, Si W. Lu, and Marc Rioux. Recognition and shape synthesis of 3D objects based on attributed hypergraphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):279–289, March, 1989.

[120] Daniel M. Wuescher and Kim L. Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):41–51, Jan 1991.

[121] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(8):841–847, August 1991.

[122] P. Yip and K.R. Rao. A fast computational algorithm for the discrete sine transform. *IEEE Transactions on Communications*, COM-28(2):304–307, 1980.

[123] P. Yip and K.R. Rao. On the computation and the effectiveness of discrete sine transform. *Computer and Electrical Engineering*, 7:45–55, 1980.

[124] H. Zabrodsky, S. Peleg, and D. Avnir. A measure of symmetry based on shape similarity. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition '92*, pages 703–706, 1992.

[125] Charles T. Zahn and Z. Roskies, Ralph. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, C-21(3):269–281, March 1972.