Name: Tang Mengting
Degree: Master of Science
Dept: Department of Computer Science
Thesis Title: Geon-Based Classification of 3D Objects in 2D Real Images

Abstract

Pose-invariant recognition of 3D objects from single 2D images is a long-standing difficult problem in computer vision. To recognize an object in an image, one must know how the object appears in the image. To solve such a problem, model-based techniques are usually used. In this thesis we adapt the component-model approach. This approach represents objects in terms of a small number of standard components. Due to occlusion, only partial information is available for inferring the shapes of the objects in an image. To overcome this difficulty, our method performs simultaneous matching at two levels. At the component level, partial information about object parts is matched with the components in known models. This matching process is usually ambiguous. At the object level, structural information about the model components is used to constrain the matching process. Once a good match is obtained, the shapes of the object parts can then be inferred from the shapes of the model components. The matching algorithm can be used for object classification and image retrieval.

Keywords: Geon
Component modeling
Color segmentation
3D model matching
Object classification
Image retrieval

GEON-BASED CLASSIFICATION OF 3D OBJECTS IN 2D REAL IMAGES

TANG MENGTING

NATIONAL UNIVERSITY OF SINGAPORE

2001

GEON-BASED CLASSIFICATION OF 3D OBJECTS IN 2D REAL IMAGES

TANG MENGTING

(B.Sc. (E.E.)(Hons, ZSU))

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2001

Acknowledgments

I wish to express my gratitude to Dr. Leow Wee Kheng for his help and guidance in the research that leads to this thesis. Dr. Leow has been very patient and helpful during our discussions from which much insight is gained. I also wish to thank a group of people who gave me precious opinions and helps for my research, they are: Fan Lixin, Terence Tay, Annie Hui, Zhang Yong, Yi Lan, Wang Jun, Zhou Luping, Li Rui and all the others whom I have forgotten. Last but not least, I wish to thank my family members whose care and support are invaluable during the course of my research.

Table of Contents

List of Figures

List of Tables

Summary

Pose-invariant recognition of 3D objects from single 2D images is a long-standing difficult problem in computer vision. To recognize an object in an image, one must know how the object appears in the image. To solve such a problem, model-based techniques are usually used. In this thesis we adapt the component-model approach. This approach represents objects in terms of a small number of standard components. Due to occlusion, only partial information is available for inferring the shapes of the objects in an image. To overcome this difficulty, our method performs simultaneous matching at two levels. At the component level, partial information about object parts is matched with the components in known models. This matching process is usually ambiguous. At the object level, structural information about the model components is used to constrain the matching process. Once a good match is obtained, the shapes of the object parts can then be inferred from the shapes of the model components. The matching algorithm can be used for object classification and image retrieval.

Chapter 1

Introduction

1.1 Motivation

Pose-invariant recognition of 3D objects from single 2D images is a long-standing difficult problem in computer vision. With a single 2D image, there is no depth information for deducing the 3D structure of the object in the image. Moreover, due to occlusion, some parts of the object are not visible. The faces of the parts that connect to each others are not visible as well. Therefore, only partial information is available for inferring the shapes of the object and its parts. Although texture and shading variations can be used to deduce surface orientations, this method requires accurate recovery of texture or shading information. This requirement is difficult to meet for general images of objects.

To overcome these problems, object recognition from single 2D images usually adapts a model-based approach. Models of target objects of interest are constructed. Given the image of an object, it is matched with the known object models, and the best matching model is identified.

There are two approaches to object recognition: qualitative recognition and quantitative recognition. Qualitative object recognition is a coarse-scaled object recognition task that solves the object categorization problem, e.g., distinguishing a helicopter from an airplane. Quantitative object recognition is a detailed object recognition task which takes detailed distinctions into account, e.g., distinguishing a Boeing 747 from a Boeing 767. In many practical applications, qualitative recognition already satisfies the requirement of the applications. This thesis focuses on the problem of qualitative recognition of objects in real images.

1.2 Objectives

The objectives of this research are:

1. Represent 3D object models by model components.
2. Given a 3D object model and a 2D object in an image, compute the similarity between them.
3. Apply the method to geon-based object classification.

This project concentrates on recognizing *significant objects* in the images. A significant object is an object in the image that complies with the following three criteria:

1. It has a large enough size and a distinctive shape.

(a)

(b)

Figure 1.1: Significant object in image

2. Most parts of the object are visible in the image.

3. It is not enclosed in a larger significant object in the image.

To make it more clear, let's look at the images in Figure 1.1. Figure 1.1(a) contains a house with a garden in front of it. The green grass in the garden cannot be regarded as a significant object since it does not have a distinctive shape. The pathway leading to the house is not a significant object since it stretches out of the image, which cannot provide distinctive shape information. The house is a significant object because most parts of it are visible in the image. The windows are not considered as significant objects since they are enclosed in the house. For the same reasons, in Fig. 1.1(b) only the tower is regarded as a significant object.

Some parts of the definition of significant object are ambiguous because they relate to fuzzy concepts such as "large enough", "sufficient" and "most parts". The

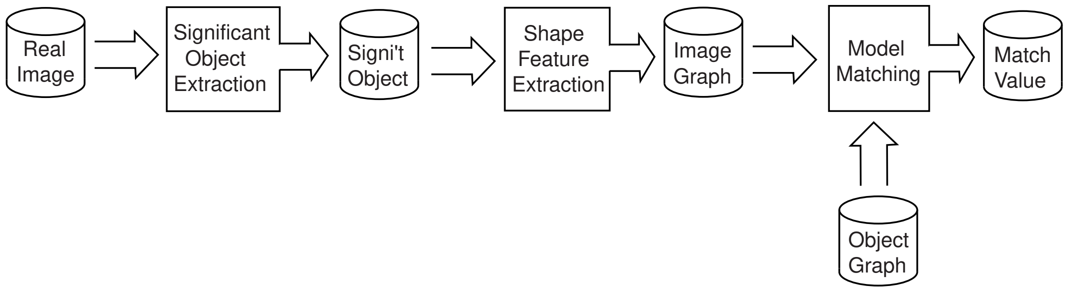


Figure 1.2: Main flowchart of the system

use of these fuzzy terminologies is caused by the inherent subjective nature of the problem. Different observers will have different judgements of what are significant objects. Nevertheless, they provide useful guidelines for making the solution to the problem possible.

1.3 Overview of the System

The pose-invariant recognition system is divided into three main modules, namely, significant object extraction, shape feature extraction and model matching. Figure 1.2 shows the flowchart of the system.

The Significant Object Extraction Module (Chapter 4) extracts the significant objects in an input image. We first segment the input image into regions based on color homogeneity. Following a background removal process, possible regions containing significant objects are highlighted and extracted. Currently only the outer boundary of the largest significant object is extracted from an image.

The Shape Feature Extraction Module (Chapter 5) extracts qualitative shape features from the significant object. Qualitative shape features are viewpoint-invariant shape features such as parallelism, symmetry and continuity. This is a model-based shape analysis, so that the shape features we extract from the significant object correlates to the shape characteristics of the models we chosen. The task of this module is to detect these features from the object boundary contour and extract the respective qualitative 2D object parts. These 2D object parts form a graph, whose nodes are the object parts and whose edges describe the 2D spatial relationships among different object parts. This graph is called the *image graph* and serves as the output of the Shape Feature Extraction Module.

The Model Matching Module (Chapter 6) computes the similarity between the 2D object and the 3D object model. The 3D object model consists of model components that form a model graph, whose nodes are the model components and whose edges describe the 3D spatial relationships among different model components (Chapter 3). Thus, the matching process can be accomplished by a graph matching algorithm.

In the case of object classification, object models can be constructed interactively from predefined primitive 3D shapes. For images in databases, the first and second modules are performed only once to produce preprocessed image records. Given a preprocessed significant object, the third module matches the 2D significant objects with the pre-constructed 3D object models. The matching results can then be used to rank the significant objects in terms of their similarities to the object models.

Chapter 2

Related Work

2.1 Model-Based Object Recognition

Existing model-based object recognition approaches can be categorized into four categories: rigid, deformable, statistical and component model approaches.

Rigid model approach [17, 20, 22, 25, 26, 44, 46] uses 3D models that capture the exact geometry of the objects. Given a 2D image, a 3D model is scaled, rotated and projected to a 2D plane to align and match with relevant shape features in the 2D image. The object is recognized if the match is sufficiently good. This approach is effective when the exact object geometry is known and the object is rigid. When the geometrical variation of the object is large, many rigid models are needed to represent the different variations. If the geometrical variation of the object is too large, the number of rigid models involved will make the searching and matching process intractable.

The deformable model approach [21, 24, 27, 37, 42, 43] allows a model to undergo local deformation while matching a 2D image. Given a 2D image, points in the model are attracted by the corresponding data points in the image by forces derived from (1) image properties such as gradient and color, (2) model properties such as mass, stiffness and damping, and (3) additional deformation tendency in the image provided by the users. The amount of deformation is modeled as energy and the matching process looks for a minimum energy deformation. When the exact geometry of the object cannot be determined, deformable model becomes a good choice. This approach is typically applied to matching 2D models instead of 3D models although it is possible to construct 3D deformable models [42]. A variant of the deformable model represents 2D shapes of objects as algebraic curves and applies a fitting process to recognize the objects [41].

The statistical model approach is different from the other types of model-based approaches in the construction of the models. Statistical models are constructed from training examples. Properties such as the eigenvectors of the training set are extracted [7]. The training set is then approximated and represented using techniques such as polynomial approximations [39] and polar co-ordinates [18]. This modeling approach is effective for classifying specialized objects such as human hands and faces [7, 8, 15, 40]. Its performance is very dependent on the training samples. To perform pose-invariant recognition, training samples in various poses must be included, which will make the training process complicated.

The component model approach represents an object by its components. Different kinds of objects are represented by combining a small number of components.

Depending on the method used, the components may be represented as generalized cylinders [4, 45, 48], geons or geon-like volumetric primitives [1, 2, 10, 11], or superquadrics [33, 42]. Due to the difficulty arising from occlusion discussed earlier, this approach has been applied to recover shapes mainly from line drawings, range data, and 2D images of simple geometric objects. Our method belongs to the component model approach. The following section provides a more detailed discussion of related techniques in the component model approach.

2.2 Component-Based Object Recognition

There are three common types of components, namely, generalized cylinders, deformable superquadrics, and geons. An object part can be represented by one of these models. Deformable superquadrics and generalized cylinders are quantitative components that provide detailed description of the objects. Geon (geometrical ions), on the other hand, being qualitative components, mainly provide coarse representations of 3D objects.

Generalized cylinders were first proposed by Binford in 1971 [4]. A generalized cylinder is a solid with a planar area sweeping along a 3D space curve. Simple examples of objects that are easily modeled by generalized cylinders include cylinders, cones, and prisms. Take cone for an example. It is formed by sweeping a circular disc along a straight axis. Generalized cylinders allow easy representation and calculation of a number of mathematical properties of objects such as volume and surface normals. Many researchers have utilized generalized cylinders successfully in their work [6, 32, 45, 48].

Deformable superquadrics were proposed by Pentland in 1986 [33]. Superquadrics are a 3D extension of basic quadric surfaces formulated by a set of equations. Continuous material properties such as mass, stiffness and damping are integrated. A useful property of superquadrics is that they can represent 3D shapes in different resolution by modifying relevant parameters. By using superquadrics, 3D components can be represented mathematically and can further be deformed and glued to model articulated objects [16, 23, 29, 38, 42].

The geon model was first proposed in 1985 by Biederman [2]. There is a total of 36 geons which are qualitative 3D shapes containing viewpoint invariant features such as parallelism, symmetry, and continuity. We choose geons as our basic components, the detailed description of geons and their applications will be given separately in Section 2.3.

2.3 Geon Theory

Biederman summarized five viewpoint invariant features as "nonaccidental features": curvature, collinearity, symmetry, parallelism and cotermination. They are rarely produced by accidental alignments of viewpoint and object features, and consequently are generally unaffected by slight variations in viewpoint (Witkin & Tenenbaum, 1983). For instance, a straight line (collinearity) in a 2D image is

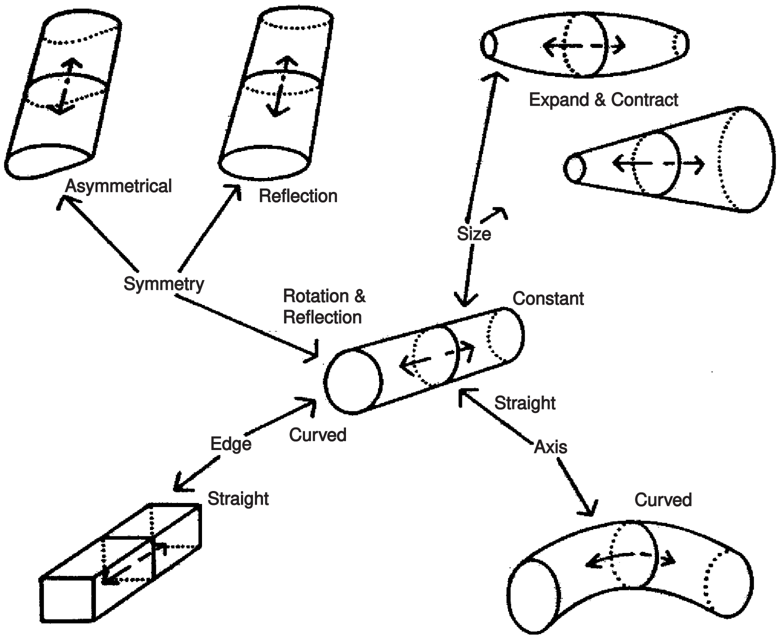


Figure 2.1: Nonaccidental feature variation. An illustration of how variations in (1) cross-section edge curvature, (2) cross-section symmetry, (3) cross-section size, and (4) axis curvature generate a set of geons differing in nonaccidental features.

more likely to be produced by a straight line in 3D space than an accidental alignment between the camera and a curved edge. Similarly, curved lines in 2D images are more likely to arise from curved lines in 3D space. These nonaccidental features give rise to four 2D attributes: cross-section edge curvature, cross-section symmetry, cross-section size, and axis curvature. Figure 2.1 shows the process of creating different geons through variation in the nonaccidental relations in cross-section and axis. We can see that constant-sized cross-sections have parallel sides whereas expanded or expanded and contracted cross-sections have non-parallel sides. A complete set of 36 geons are generated from the Cartesian product of the dichotomous and trichotomous variation of these four attributes.

Biederman claimed that the geon primitives can represent almost all the objects in the world through their combinations [3]. He estimated that the potential number of common basic-level object categories (common natural and man-made objects) is approximately 3,000, whereas the estimated number of readily discriminable objects (objects distinct from each other) is 30,000. The representational capacity of 2 geons out of the total of 36 will cover 74,649 different objects. Table 2.1 shows an example of how to estimate the representational capacity of two geons. The random combination of two geons gives rise to 36×36 possible choices. In the same way, relative size (bigger than or equal), relative location (top or bottom or side), nature of joint (end-to-end or end-to-side), joined surface (long or short) give rise to a total of $36 \times 36 \times 3 \times 2.4 \times 2 \times 2 \times 2 = 74,649$ different combinations. In the same way, 3 geons can represent $74,649 \times 36 \times 57.6 = 154$ million possible objects.

Nevertheless, one major criticism of geon theory is that geons cannot represent all objects. Many natural objects such as animal bodies cannot easily be described by an assembly of a few volumetric parts. And some objects or regions of an object such as bushes and flowers are complex or highly irregular and cannot be described by geons. When details must be taken into account for object recognition, other models have to be involved. In our application only qualitative recognition is required, hence geons' representation capability is sufficient to achieve this goal.

Table 2.1: Representation power of geons. The following analysis shows the representative power (74,649 different objects) of 2 geons.

Value	Component
36	First component (G_1)
$\times$	
36	Second component (G_2)
$\times$	
3	Size ($G_1 \gg G_2, G_1 \ll G_2, G_1 = G_2$)
$\times$	
2.4	G_1 top or bottom or side (represented for 80% of the objects)
$\times$	
2	Nature of joint (end-to-end (off center) or end-to-side (centered))
$\times$	
2	Join at long or short surface of G_1
$\times$	
2	Join at long or short surface of G_2
Total: 74,649 possible two-geon objects	

2.4 Geon-Based Object Recognition

Most of the works based on geon theory concern (1) how to represent geons and object and (2) how to extract geon features from an image for object recognition. There are two main trends in object representation methods: qualitative representation and quantitative representation. Qualitative representation concentrates mainly on non-accidental geon features such as parallelism, symmetry, cotermination, etc. while quantitative representation concentrates mainly on specific shape features such as angle, size ratio, joint type, etc. These two methods are sometimes combined to enhance the geon-based system's performance. First we introduce works involving qualitative representation.

Bergevin and Levine [1] developed the PARVO system in 1988. Their goal was to test the validity of geon-based applications. This system performed a fast and generic recognition of unexpected 3D objects (represented by geons) from single 2D line drawings. For object representation, the system first generated a coarse part-based volumetric description of the object. Each part was then labeled as one member of the geon set using a hypothesis and test procedure. The object models to be recognized were specified in advance via a generic graph representation: the geon type was represented as a graph node, and relations between the geons were represented as graph arcs. The recognition problem then became a sub-graph isomorphism problem. Although PARVO system did not solve the low level feature extraction problem because line drawings were manually segmented, it was very successful for showing the strength of the geon-based approach.

Dickinson et al. [13, 14] developed the OPTICA system to recover object from 2D real image. Geons were represented in a hierarchical structure with different levels, i.e., geons, aspects, faces, boundary groups and edges. In each level, graph representation were used for the mapping between levels. In the recognition process, edges were grouped into boundary groups. A set of boundary groups were grouped and classified as one of the faces, a set of faces were grouped and classified as one of the aspects, and so on. Between the mapping of each level in the hierarchy conditional probabilities were used to capture the ambiguity of mappings. The system performed some simple low-level processing to extract shape features of simple input image. The works of Dickinson et al. will be described in details in Section 2.5 because our system adapts some of their hierarchical inference concepts.

Besides the graph representation method, Fairwood [30, 31] developed his primitive recognition system based on logic description. For representing objects, he wrote declarative semantics for (1) primitive models, (2) 3D/2D projection inference network, and (3) 2D input primitives of an image. The category of input image object and predefined models then could be related through the inference network. In his later works the inference network was integrated with causal probability to assign confidence measures to nonaccidental properties. For its semantic nature, the system did not concern with low-level processing and did not consider inputs with complicated structure, which would be required for a full complex-object recognition system.

Hummel and Biederman's neural network system JIM [19] was a seven-layer connectionist network that took as input a representation of a line drawing of an object and, as output, activated a unit representing the identity of the object. Geon identification was achieved in layer three with cells representing different geon attributes. Activated by the input, cells in layer three would fire in synchrony, representing the corresponding geon. Low-level image (line drawing) processing was achieved automatically in the first layer, which was a mosaic of orientation-tuned cells with overlapping receptive field, extracting useful line features. JIM performed the complete object recognition task from low-level image process to high-level object recognition. But due to the nature of its network structure, it only worked well for predefined objects. Adding or reducing object cells in the output layer would lead to major changes to the structure of the network.

Unlike the above methods, most methods involving quantitative representation adapted the technique of deformable model. Narayan and Jain [36] recovered geons with straight axis from range data. For representing objects, they used superquadrics to quantitatively model geons. Through the fitting process similar to Pentland's work [33, 34] on superquadrics, real geon-based objects in range image could be recovered. Their work explored the possibility of geon-based application in range data by including Pentland's superquadrics. Their work proved that superquadrics are excellent models for range data applications.

Dickinson et al. [10] further developed the OPTICA system for recognizing objects in range data. There were two recognition phases in the system: qualitative recognition and quantitative recognition. In the qualitative recognition phase the identity of geons were recognized and represented using a hierarchical structure. In the quantitative recognition phase, detailed information about the geons were extracted and represented by superquadrics similar to those of Narayan and Jain. We can see that qualitative and quantitative representations can be used together in a system to achieve better results.

Pilu et al. [35] developed an approach to recognize geons using 2D deformable

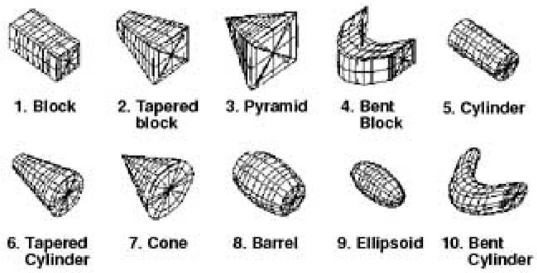


Figure 2.2: Geons used in Dickinson's works.

models in 2D real images. They modified 3D superquadric models into 2D parametric deformable contour models, which were the visible contours of superquadrics' 2D projections. Edge detection were used to extract low-level object features from the image. Represented by 2D deformable contours, initial geons were manually placed on the object parts of the image. The contours were continuously deformed until the fitting between the contours and the object features in the image was optimized. Thus, quantitative recognition of geons was achieved.

2.5 Works of Dickinson et al.

The works of Dickinson and his colleagues provide solid foundation for our system. In this section we describe the details of their works and compare the differences between our work and theirs.

For the sake of simplicity, most of the geon-based methods use a subset of the 36 geons. For example, Dickinson et al. selected 10 geons for their systems (Fig. 2.2).

Most works of Dickinson et al. [9, 10, 11, 12, 13, 14] on geon recognition used an aspect hierarchy (Fig. 2.3) to accomplish the high level inference of geons after recovering boundary groups from input images. This aspect hierarchy consists of four levels: (1) geon, (2) aspect, (3) face, and (4) boundary group. An aspect is a topologically distinct 2D view of a geon in terms of its visible surface. A face is a surface of a geon. A boundary group is the boundaries of a face. The relationship between a higher level component, e.g., an aspect, and its corresponding lower-level parts, e.g., faces, is represented by a graph.

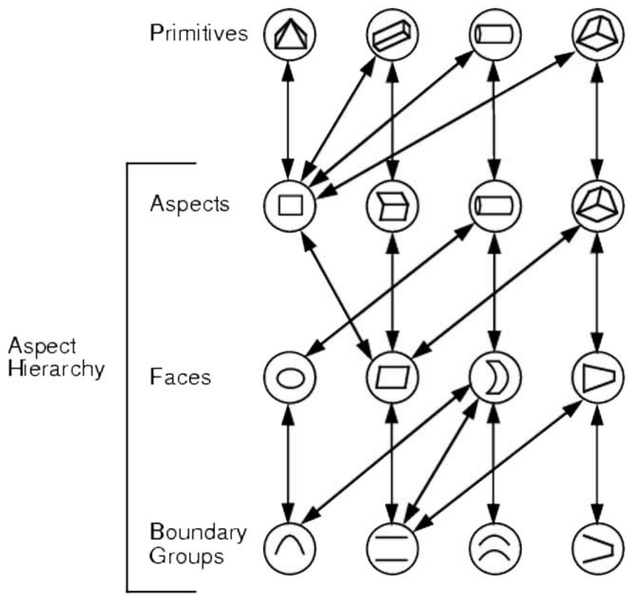


Figure 2.3: Part of Dickinson's inference hierarchy. Starting from boundary groups, low-level features are used to infer high-level features until geon primitives are found.

For geon recognition, an input image was first segmented into regions. Contours of the resulting regions were partitioned at significant curvature discontinuities. The segmented contours were labeled and represented as contour graphs, e.g., graphs of boundary groups. Faces were inferred from these contour graphs and represented as face graphs. In the same way aspects were inferred from face graphs and primitives were inferred from aspect graphs. Relative locations between primitives were inferred from the position of the joint between two surfaces and the angles at which they joined.

We adapt their hierarchical method and develop a primitive inference hierarchy similar to the aspect hierarchy in OPTICA, but with a different concept: primitives are inferred from partial shape features and their spatial relationships. The method is called Multi-Level Object Matching (MOM). It performs simultaneous matching at two levels. At the component level, partial information about object parts is matched with the components in known models. This matching process is usually ambiguous. At the object level, structural information about the model components are used to constrain the matching process. Once a good match is obtained, the shapes of the object parts can then be inferred from the shapes of the model components.

2.6 Summary

Most of the existing works tried to recover the complete geons before simple object recognition can be achieved. This lies in the fact that their model/object matching schemes require complete geon recovery. However, due to those long-standing difficulties with low-level image processing, recovery of complete geons from real image containing complex objects is often impossible. Moreover, the geon faces that touch each others cannot be detected at all, which makes geon detection ambiguous. This is why most of the existing works were tested with simplified inputs such as images with manual segmentation, artificial input data or unrealistically simple objects.

To solve these problems, we look at the issue from another point of view. As a matter of fact, partial shape information from real image is what we can obtain through current low-level image processing techniques. We start from an object model consisting of geons. Through a graph matching process, image object is directly compared with the object model instead of inferring geon identity separately. In this way, even when no geon can be exactly identified, we can still tell the degree of match between the image object and the model.

Chapter 3

Model Representation

MOM models objects in terms of their components. In this chapter, we describe the representation of the components in terms of shape attributes.

3.1 Simplified Geon Set

For simplicity, we picked 10 out of total 36 geons in a similar way as Dickinson et al. [12] (Fig. 3.1). Note that this subset is almost the same as that of Dickinson's except that the "barrel" geon (Fig. 2.2) is replaced by our "truncated ellipsoid" (last geon in Fig. 3.1). The change is made under the consideration that "barrel"

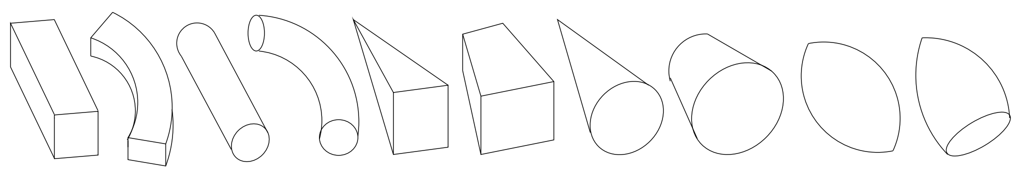


Figure 3.1: The set of geons used in MOM.

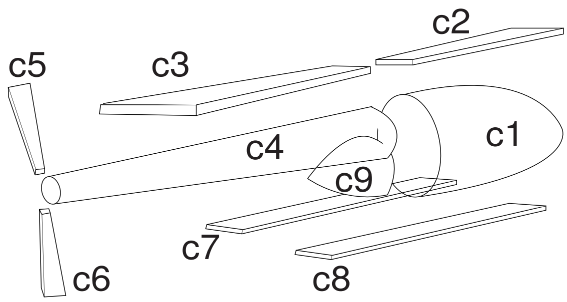


Figure 3.2: An object composed of geons.

is not distinct from "cylinder" while "truncated ellipsoid" cannot find a correspondence in Dickinson's geon set. These ten geons can already represent many objects [3]. Figure 3.1 shows a helicopter model composed of geons. It has a relatively complicated structure and yet can be modeled using only four different types of geons.

To achieve qualitative representation, we accept shapes with a certain degree of deviation from the 10 standard geons in MOM. Figure 3.3 shows an example of the variants allowed. As long as the main features (the two parallel curves of the bent cylinder or the two longest edges of pyramid in Fig. 3.3) of the variant shapes match those of the standard geons, they can be represented by the closest geon.

3.2 Aspects

In MOM, 3D geons are represented by 2D geon aspect contours. We adapt the aspect concept (Sec. 2.5) used in the works of Dickinson et al. [9, 10, 11]. Aspect

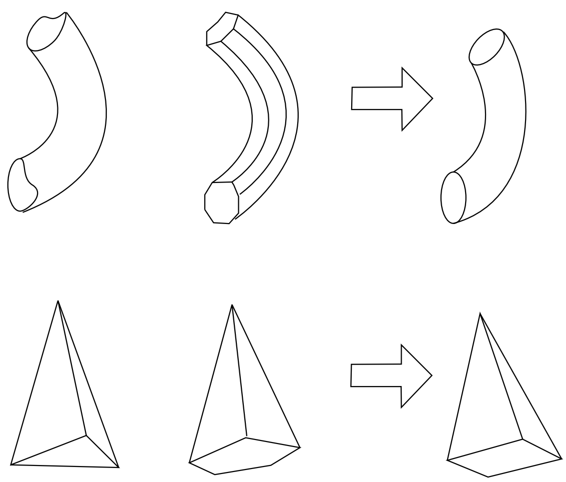


Figure 3.3: Variants of standard geons (left side of arrow) are represented by standard geons (right side of row).

is defined by (1) the number of its faces and (2) the shape of its faces. Actual size and orientation of faces are ignored, so that every geon has only a small number of aspects. The complete set of aspects can be found in Appendix B.1.

Figure 3.4 shows a geon (bent block) with 9 aspects. There is a total of 47 aspects derived from 10 geons. An aspect may belong to more than one geon, so that specific geon cannot be inferred from a single aspect. This is an ambiguous relationship in such a representation structure.

3.3 Aspect Contours

Aspects are represented in terms of aspect contours. Aspect contour is the outer boundary of an aspect, which is defined by (1) the number of lines, (2) the type of lines (straight or curved) and (3) the relationship (adjacency, parallelism) among

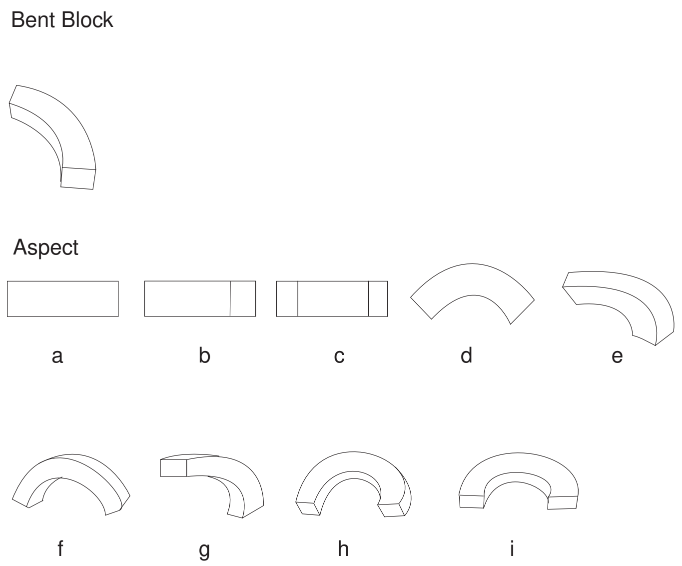


Figure 3.4: The bent block has 9 aspects.

those lines. Again, size, angle, and pose are ignored. Figure 3.5 shows the complete aspect contours of the bent block. Although the bent block has a large number of aspect contours, five of them can be simplified into Figure 3.5(c) by discarding information about the parallelism between some of the straight lines. This is because in practice no aspect contour of other geons has such a combination of lines. A total of 19 distinct aspect contours is derived from the 47 geon aspects defined in the system, resulting from the fact that some aspects have the same aspect contours. For example, aspect a, b, and c of the bent block (Fig. 3.4) share the same aspect contour a (Fig. 3.5).

From the definition of geons (Sec. 2.3), we can see the relationships between the lines of the aspect contours should include adjacency, parallelism and symmetry.

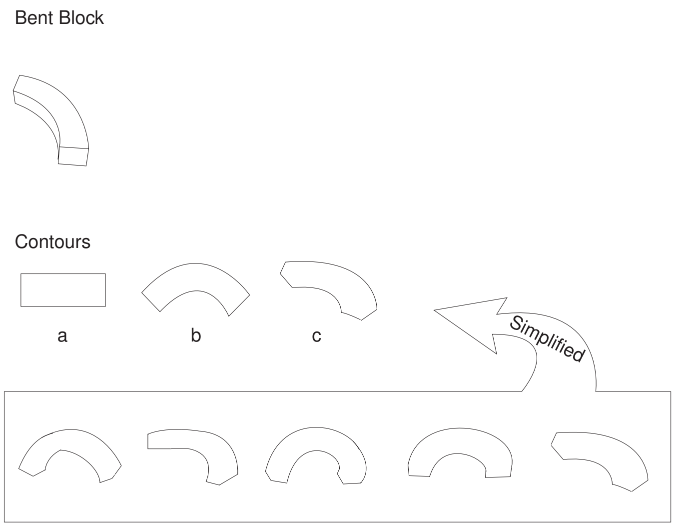
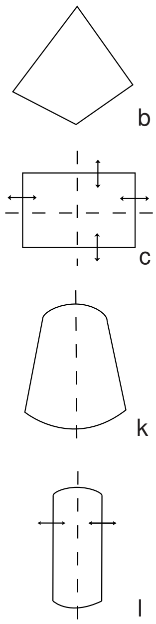


Figure 3.5: The bent block has 7 aspect contours. Five of them can be simplified into contour c.

However, MOM does not use symmetry since the 10 geons used in MOM can be uniquely identified without the information of symmetry. Figure 3.6 shows an example of why symmetry information can be ignored. From the figure we can see that, without symmetry, all the aspect contours can still be distinct from each other because they have different combination of number of lines, straight lines, curved lines, and parallel pairs. Without parallelism, k and l will be confused. Similarly, without line type (straight line or curve), none of the aspect contours can be uniquely defined. Interested readers can refer to Table B.1 in the Appendix, which gives the complete feature comparison among these contours. The final simplified aspect contours are shown in Figure 3.7.

In practice, the line types (curve and straight line) in aspect contours are



	Number of Lines	Straight Lines	Curved Lines	Parallel Pairs	Symmetric Axes
b	4	4	0	0	0
c	4	4	0	2	2
k	4	2	2	0	1
l	4	2	2	1	1

Figure 3.6: Aspect contour distinction. Dash line indicates axis of symmetry. Parallel arrow pair indicates the lines under them are parallel to each other. Without information of symmetry, these aspect contours can still be distinct from each other. Without information of parallelism, k and l will be confused. Without information of straight line or curve, none of the aspect contours can be uniquely defined.

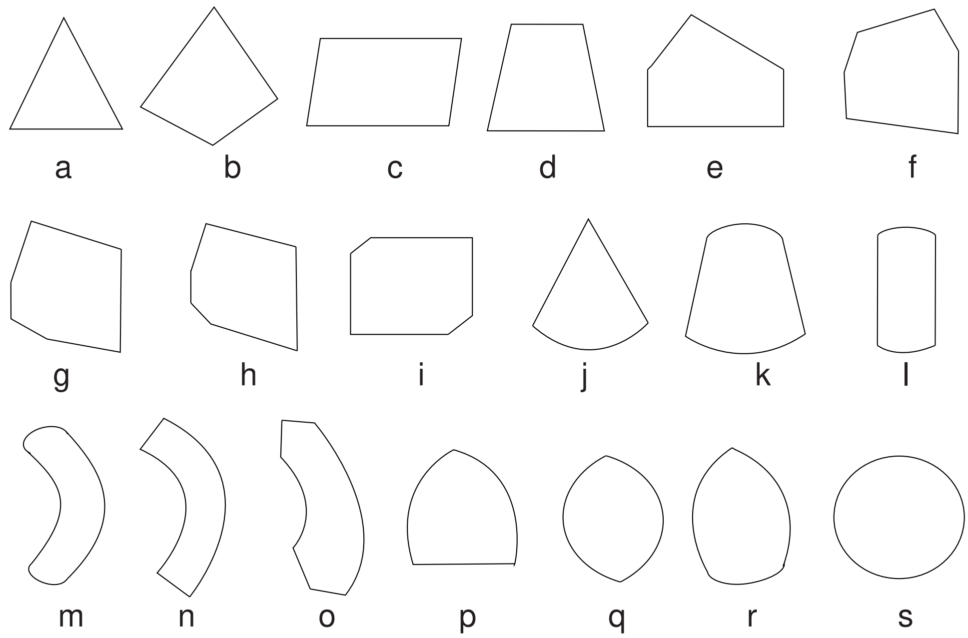


Figure 3.7: Complete set of aspect contours.

represented by its curvature: 0 for straight edge, 0.5 for moderately curved edge, and 1 for highly curved edge. In Figure 3.7 all curves have curvature 0.5 except contour s, which consists of one curve with curvature 1.

3.4 Partial Aspect Contours

In MOM, aspect contours are represented in terms of partial aspect contours. Partial aspect contours are parts of aspect contours. Figure 3.8 shows two aspect contours and their corresponding partial aspect contours. The simplest partial aspect contour is a straight line or a curve. From the figure we see that different aspect contours may share the same partial aspect contour. Therefore, a partial

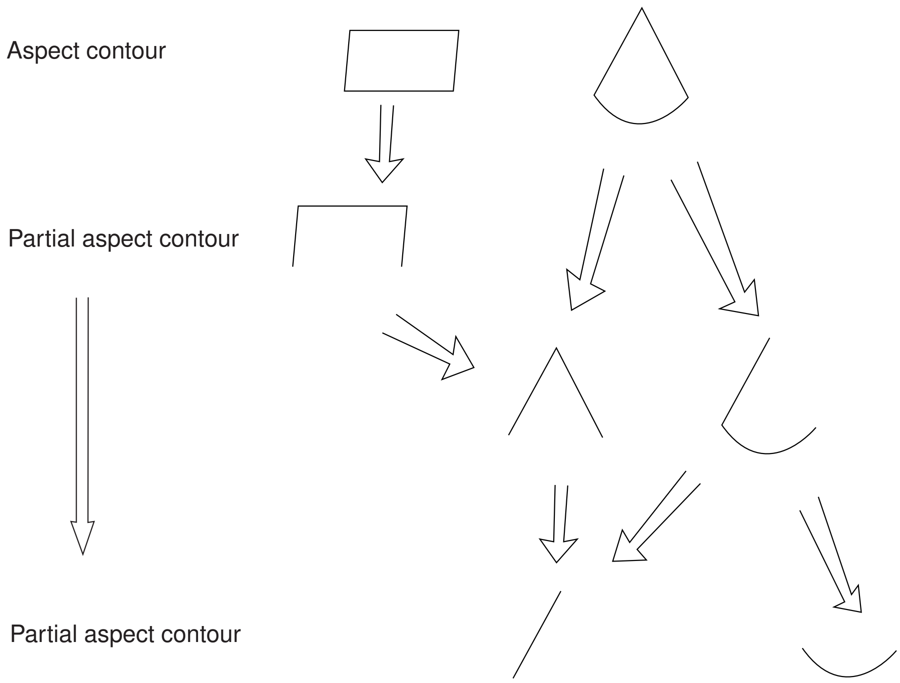


Figure 3.8: Partial aspect contours. An aspect contour is composed of partial aspect contours. The simplest partial aspect contour is a straight line or a curve.

aspect contour cannot be associated with a unique aspect contour. This is another ambiguity factor that exists in this representation method. There is a total of 28 different partial aspect contours defined in MOM (Fig. 3.9).

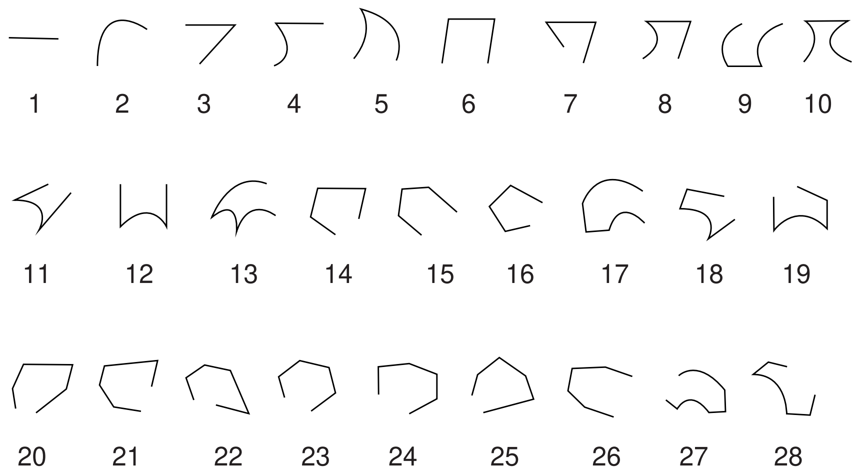


Figure 3.9: Complete set of partial aspect contours.

Chapter 4

Significant Object Extraction

To extract the significant object from an image is to (1) identify significant object regions so that (2) the boundary contour of the significant object can be extracted. Figure 4.1 shows the structure of the significant object extraction procedure. The input image is first segmented into regions based on color. Then the background is separated from the image. Finally, regions corresponding to significant objects are extracted based on the location, size and background information.

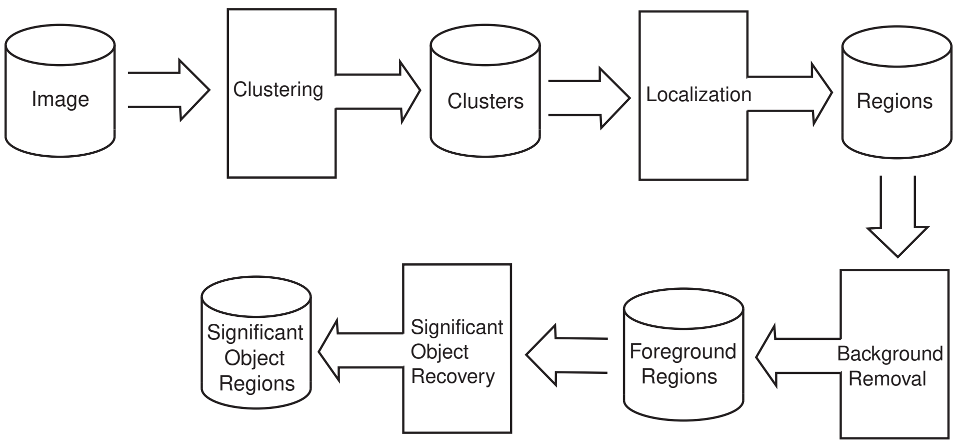


Figure 4.1: Flowchart of significant object extraction.

4.1 Color Clustering

The colors of the image pixels are converted from RGB to CIELUV before clustering (Appendix A) because the color difference computed as the Euclidean distance in CIELUV space matches human perception closely. The color clustering algorithm is governed by two parameters: the maximum radius r of a cluster and the minimum separation s between clusters. The algorithm is as follows:

1. Create first cluster with the first pixel.

2. Repeat until all pixels are clustered.

   2.1. Compute Euclidean distance $d(i, j)$ between each cluster i and each un-grouped pixel j.

   2.2. For each $d(i, j)$ in increasing order, while pixel j is un-grouped,

      2.2.1. If $d(i, j) \leqslant r_i$, group pixel j into cluster i. Update cluster's characteristics.

      2.2.2. If $d(i, j) \geqslant s_i$, create a new cluster k, group pixel j into cluster k, update cluster characteristics.

   2.3. Relax parameters r_i and s_i of each cluster i. Update cluster status.

This algorithm ensures that each cluster is as compact as possible and as far away from other clusters as possible. The details of the algorithm are as follows.

(1) Creating first cluster with first pixel $\mathbf{P_1}$

Characteristics of cluster 1 is initialized as follows:

- Cluster size: $N_1(0) = 1$
- Cluster center: $\vec{c}_1(0) = \vec{p}_1$
- Cluster "spread" (standard deviation): $\sigma_1(0) = 0, \quad \delta_1(0) = ||\vec{p}_1||^2.$
- Cluster maximum radius: $r_1(0) = \sigma/m$
- Cluster minimum separation: $s_1(0) = 4\sigma/m$

Here $\vec{p}_1$ is the color vector of pixel P_1, m is the expected number of clusters after the clustering process and σ is the standard deviation of the colors in the input image. δ is an auxiliary variable for calculating σ incrementally. Equations for initializing r_1 and s_1 are developed under the consideration that after relaxing r_1 and s_1 during the clustering process, they will finally approach each other and stay in between the value $3\sigma/m$ to $3.5\sigma/m$. This criterion is derived from the consideration that 99% of the colors in the cluster will fall within a range of $\pm 3\sigma$ from the mean color $\vec{c}_i$ of the cluster assuming normally distributed colors.

The first cluster starts the clustering cycle. To make the algorithm efficient, the first cluster should not end up as a dead cluster, which is a cluster with fewer than N_{min} pixels. Since the first pixel of most input images belongs to the background, which form large clusters, the first pixel is suitable for forming the first cluster center. The above equations are also applied to initialize the other new clusters during clustering.

(2) Updating cluster characteristics

When a new pixel P_j (with $\vec{p_j}$ as its color vector) is grouped into cluster C_i, the characteristics of C_i is updated:

$$\begin{aligned} N_i(t) &= N_i(t-1)+1 && (4.1.1) \\ \vec{c_i}(t) &= \frac{N_i(t-1)\vec{c_i}(t-1)+\vec{p_j}}{N_i(t)} && (4.1.2) \\ \delta_i(t) &= \frac{N_i(t-1)\delta_i(t-1)+\|\vec{p_j}\|^2}{N_i(t)} \\ \sigma_i(t) &= \sqrt{\delta_i(t)-\|\vec{c_i}(t)\|^2}\,. && (4.1.3) \end{aligned}$$

Here $t-1$ is the cycle before adding P_j, t is the cycle after adding P_j, δ_i is an auxiliary variable for calculating σ_i incrementally.

(3) Updating cluster status

Here r_i and s_i are updated differently for different types of clusters. There are altogether three types of clusters: (a) normal, (b) saturated, and (c) dead clusters. After every clustering cycle, the clusters' status are updated. For the t-th cycle, the status of cluster C_i is updated as follows:

When $r_i(t) < s_i(t)$

1. If $N_i(t) \leqslant N_{sat}$ or $\sigma_i(t) \leqslant \sigma_{sat}$, status = normal.

2. If $N_i(t) > N_{sat}$ and $\sigma_i(t) > \sigma_{sat}$, status = saturated.

3. If $N_i(t) < N_{min}$ and $t > 1$, status = dead.

Here N_{sat}, N_{min}, and σ_{sat} are status parameters determined by the size and color distribution of the input image. Saturated clusters are those clusters which have a large number of pixels ($N(t) > N_{sat}$) and a large color distribution ($\sigma(t) > \sigma_{sat}$). Dead clusters are those clusters which have too few pixels ($N(t) < N_{min}$) after the first clustering cycle ($t > 1$). Normal clusters are those clusters which are neither saturated nor dead.

The follows is the procedure for updating r_i and s_i:

- If status = normal:

$$\begin{aligned} r_i(t) &= r_i(t-1) + \eta(s_i(t-1) - r_i(t-1)) \\ s_i(t) &= s_i(t-1) - \eta'(s_i(t-1) - r_i(t-1)) \end{aligned} \tag{4.1.4}$$

- If status = saturated:

$$\begin{aligned} r_i(t) &= r_i(t-1) \\ s_i(t) &= r_i(t) \end{aligned} \tag{4.1.5}$$

- If status = dead, then delete cluster C_i and release all its pixels.

For the normal clusters, their r_i and s_i are updated according to Equation 4.1.4. Here $r_i(t)$ and $s_i(t)$ approaches each other until $r_i(\infty) = s_i(\infty)$. The approaching speed is adjusted by the parameters η and η'. They are set so that $\eta < \eta' < 1$, allowing $s_i(t)$ to change faster than does $r_i(t)$. This is necessary since $|s_i(0) - \sigma_f| >$

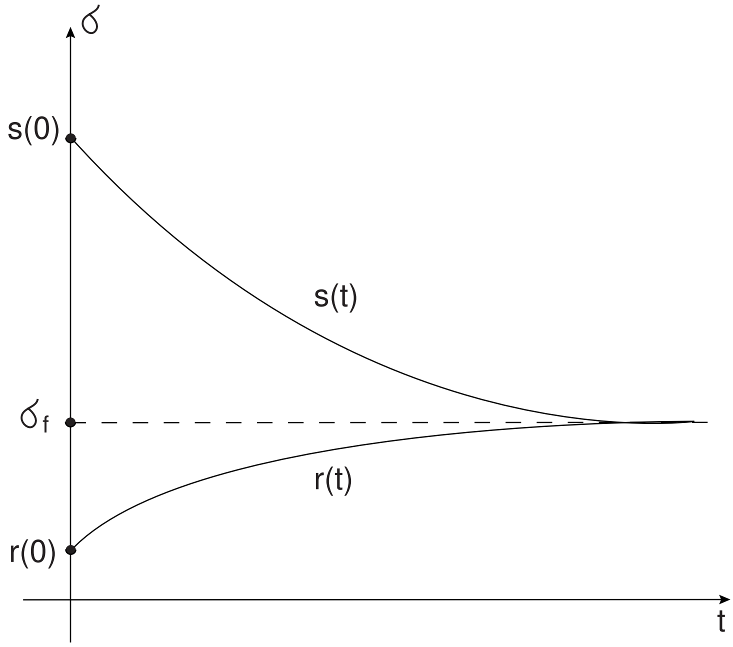


Figure 4.2: Update of r_i and s_i. Here σ_f is the expected final value of $r_i(t)$ and $s_i(t)$.

$|r_i(0) - \sigma_f|$, where σ_f is the expected final value of $r_i(t)$ and $s_i(t)$ (Fig. 4.2).

Saturated clusters spread out widely in the color space ($\sigma_i(t) > \sigma_{sat}$), their $r(t)$ will remain the same as their previous values so as to keep them more compact. Their $s(t)$ will be reduced to the value of $r(t)$, thus making room for the creation of new clusters around them.

Dead clusters, with too few pixels ($N_i(t) < N_{min}$), will be deleted to make room for creating new clusters with new cluster centers.

(4) Alternative Scanning Direction

During the clustering process, the pixels are read according to a certain scanning sequence, e.g., top to bottom and left to right, or bottom to top and right to left, etc. A certain fixed scanning sequence will cause the problem that some erroneous cluster separations will be produced along the scanning direction. This problem can be resolved by applying alternative scanning directions between different clustering cycles. For instance, at the first cycle the scanning sequence is top to bottom and left to right. At the second cycle, it changes to bottom to top and right to left.

4.2 Significant Object Recovery

The output of the clustering procedure are just color clusters. To detect regions which are connected pixels belonging to the same color clusters, we apply a simple 4-neighbor detection algorithm with the following two constraints:

1. Pixels in the same region should be connected and belong to the same cluster.

2. Pixels belonging to adjacent regions do not fall into the same color cluster.

After detecting regions, a layering process is performed to group all regions with the same color cluster into layers. We sort the layers in decreasing order based on the layers' largest region size.

Normally, when there exists a distinct object in an image, the background of the image is distinct too. Background should satisfy two criteria: (1) it touches the image border and (2) background pixels have significantly different colors than object pixels.

Some of the images have border frames. If a border frame exists, it should be taken as one of the background regions. Another similar situation is that due to segmentation inaccuracy, there will be some small narrow regions at the edges of the image, which should also be taken as background regions. We first detect and remove these regions before detecting the background according to the two criteria mentioned in the previous paragraph.

If the background can be detected successfully, the recovery of significant object is straightforward: locate the biggest region which is not a background region, grow it by adding adjacent non-background regions until no such regions can be found. Note that this algorithm detects only the biggest significant object in the image.

4.3 Testing Results

In this section we show the results of the algorithms described in this chapter step by step. Figure 4.3 shows the first three cycles of the clustering procedure. The standard deviation σ of the colors in the input image is 13.929. The dark areas in Figure 4.3(b) and Figure 4.3(c) are the pixels which are not yet clustered. As the clustering cycle continues, the number of un-clustered pixels decreases. The related data for Figure 4.3(b), Figure 4.3(c), and Figure 4.3(d) are described in Table 4.1. Note that s and r approach each other gradually during clustering.

Figure 4.4 shows the clustering and localization results of Figure 4.3(a) under different maximum number of cycles M. The η and η' in Equation 4.1.4 are

(a) (b) (c) (d) (e)

Figure 4.3: Clustering stages. (a) The original image. (b–d) Clustering results after cycle 1, 2, and 3. Pixels belonging to different clusters are marked with different colors. Pixels belonging to ambiguous regions are marked with black color. (e) Cluster localization result after merging small regions. Pixels belonging to different regions are marked with different colors.

Table 4.1: Parameters of a normal cluster at various clustering stages. Clustering rate is the percentage of clustered pixels.

Fig.4.3	b	c	d
Cycle	1	2	3
r	2.978	2.978	2.978
s	36.533	19.756	2.978
Cluster Number	4	10	77
Clustering Rate	76.77%	89.26%	100.0%

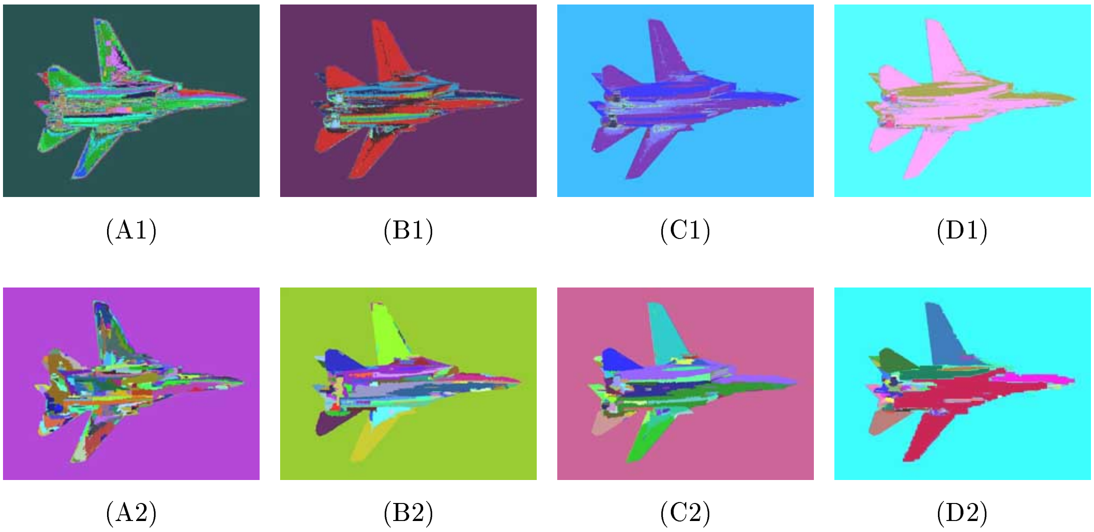


Figure 4.4: Results under different maximum number M of cycles. (1) The first row shows the clustering results and the second row shows the localization results. The M values are: (A) 2, (B) 3, (C) 4, and (D) 5.

adjusted with M so that when the cycle number reaches M, $r_i(t)$ and $s_i(t)$ will converge, causing the clustering to terminate. From the results we can see that too few or too many clustering cycles will not produce good results. The related data are described in Table 4.2. From these data we can see that performance with 2 cycles is not sensible because too many segments are produced. Performance with 3 cycles improves dramatically. There is not much difference between the performance with 3 and 4 cycles (102 vs. 93 regions). Result with 5 cycles are somewhat at the edge of losing detailed information while costing more execution time. Most of the images being tested show the same property. Considering the accuracy and efficiency factors, we set 3 as the default M value.

Table 4.2: Clustering results under different maximum number of cycles. Relevant data of images in Figure 4.4.

Fig. 4.4	Cycle	Cluster Number	Region Number
A	2	185	258
B	3	77	102
C	4	40	93
D	5	26	60

Figure 4.5 shows the results of the layering procedure. We can see that some of the layers already carry meaningful information. Figure 4.5(1) is the original image. Figure 4.5(2)–(6) are the resulting layers. They are put in decreasing order based on the size of the largest region of the particular layer. Figure 4.5(A3) is the significant object layer, and Figure 4.5(B2) is the background layer. For most test images, the background and significant object regions fall into different layers.

Figure 4.6 shows several significant object detection results. If the background is too complicated (e.g., the color distribution is too large) or too small, the system will fail to find the right background. In this case the significant object detection process will probably fail. The system fails to detect the significant object in Figure 4.6(B) due to the complexity of the object and the background. Currently we use only the images whose significant objects can be extracted. The system can successfully extract significant objects from 50% of the test images.

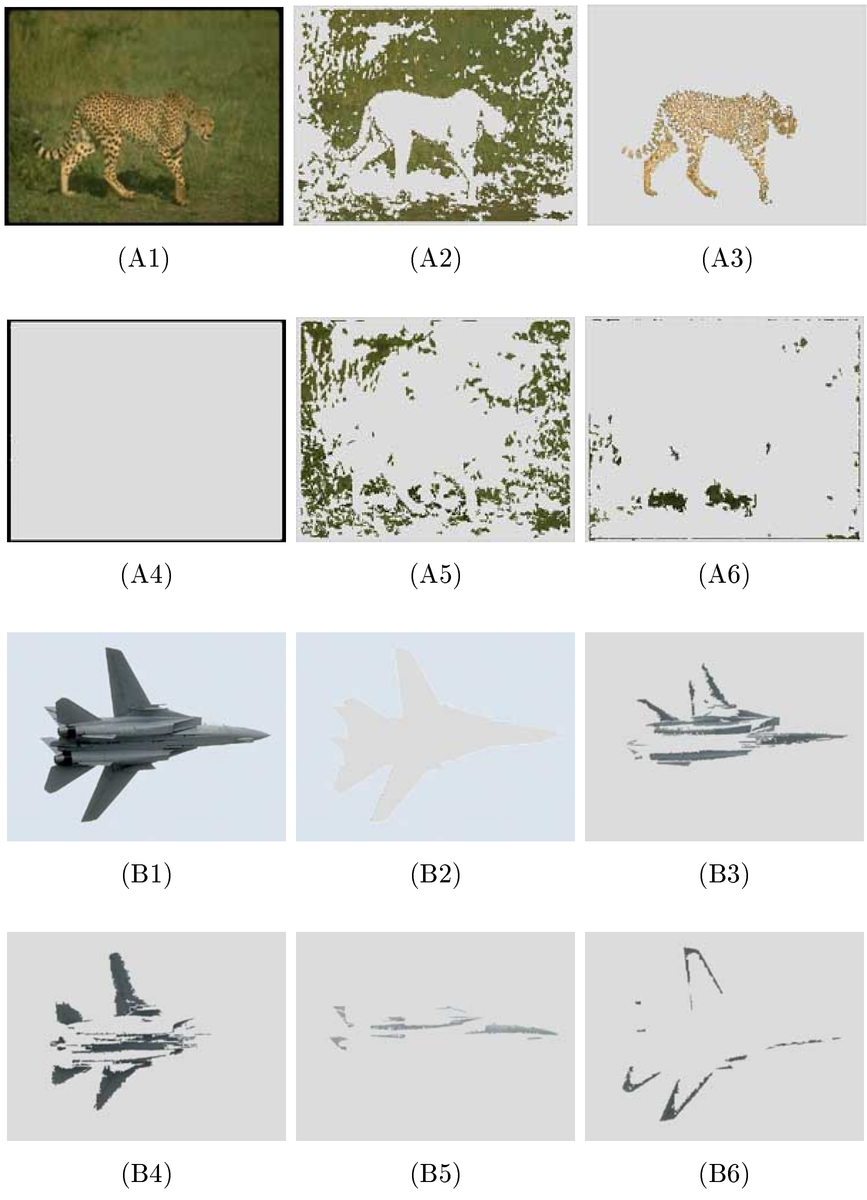


Figure 4.5: Layering results. (1) is the original image. (2)–(6) are the resulting layers. They are arranged in decreasing order of the size of the largest region in the particular layer. (A3) is the significant object layer and (B2) is the background layer. For most test images, the background and significant object regions fall into different layers.

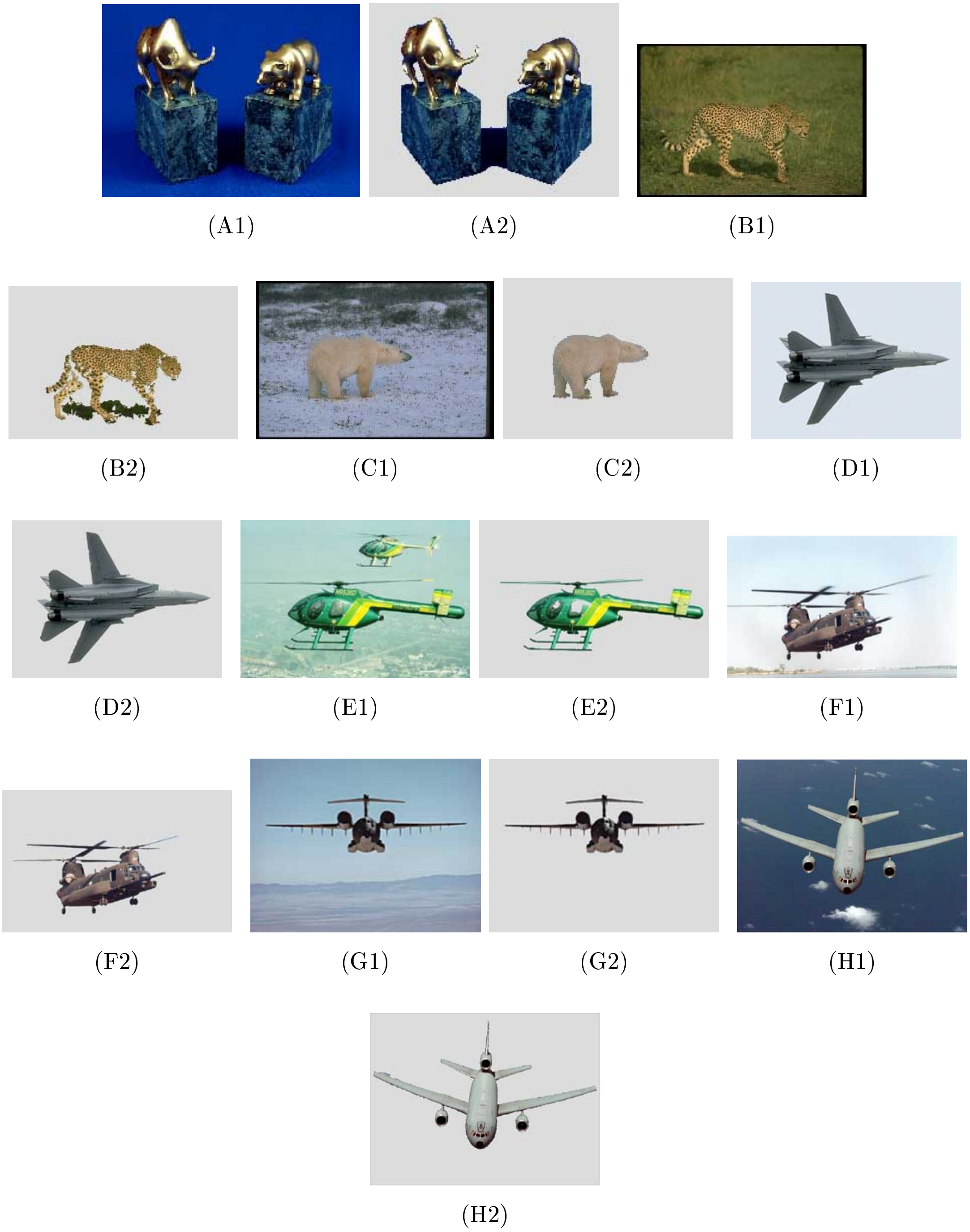


Figure 4.6: Significant object detection results. (1) are original images and (2) are the significant objects detected.

4.4 Summary

Our significant object detection module is based mainly on color segmentation. A clustering process is used to group pixels into clusters with homogeneous colors. Clusters are localized into regions, regions are separated into foreground and background, and finally significant object regions are extracted from the foreground.

Chapter 5

Contour Segmentation

After extracting the significant object from the input image, the next stage is to extract edges from the object's contour. These edges will be used in subsequent stages for shape analysis and object classification. There are three main steps in this module: contour extraction, contour segmentation and edge grouping (Fig. 5.1). The boundary of significant object regions is extracted and the bounding contour is segmented into straight and curved edges. The edges are grouped into edge groups based on relationships such as adjacency and parallelism among them. The edge groups correspond to parts of the aspect contour of model components.

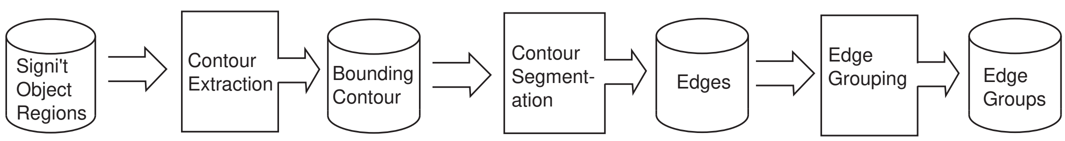


Figure 5.1: Flowchart of contour segmentation.

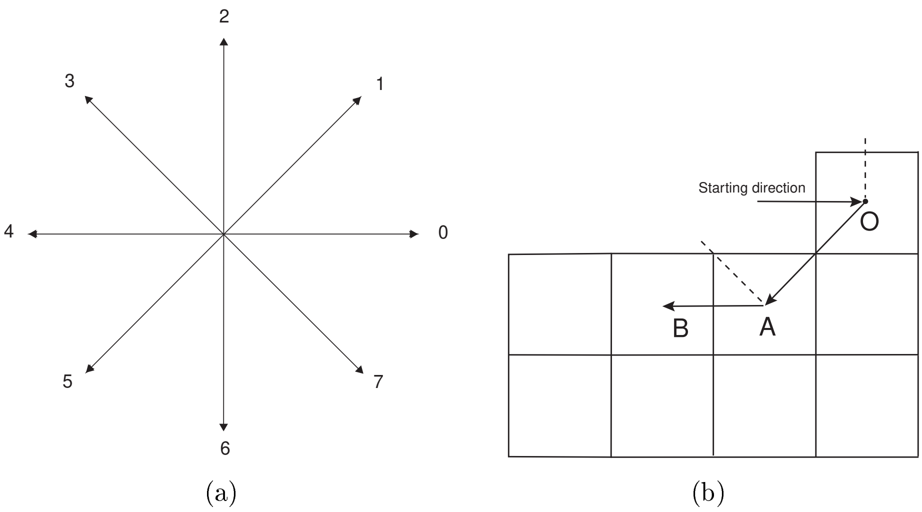


Figure 5.2: Boundary tracing. (a) Direction notation. (b) An example of boundary tracing. Solid arrows stand for the heading directions of detected boundary points and the dashed lines show the searching directions for subsequent boundary points.

5.1 Contour Extraction

The contour of a significant object's boundary is extracted using the boundary tracing algorithm developed by Zhang Yong [47]. It first finds a starting point on the object's boundary, then searches the next contour point in a 3×3 neighborhood. The search continues until the contour is closed at the starting point. The boundary tracing algorithm consists of three steps:

1. The first contour point is found by searching the image from top left to bottom right until a significant-region pixel is found. The direction from the previous pixel to the first boundary point is recorded as the heading direction (Fig. 5.2(b)).

2. Starting with the pixel perpendicular to the current heading direction and pointing outward from the significant region, the eight neighbors around the current boundary point is checked one by one along the counterclockwise direction until a significant-region is found. Mathematically, the first neighbor checked is at the direction (heading direction + 2) mod 8. The first significant-region pixel found is marked as a new boundary point. The heading direction is updated as the direction from the previous boundary point to the current boundary point.

3. If the last detected boundary point is the same as the first boundary point, then stop; else go to step 2 and continue.

An example of boundary tracing is shown in Figure 5.2. In this case, suppose pixel O is the first boundary point met and the heading direction is 0 (solid line). The next search begins at a pixel in the direction (dashed line) perpendicular to the heading direction. This corresponds to a neighbor at direction 2. Neighboring pixels are searched along the counterclockwise direction, i.e., along direction 3, 4, 5 ... until a significant-region pixel is met. Here pixel A is found at direction 5. In the same way, starting from pixel A at direction 3 (dashed line), pixel B is found to be a boundary pixel. This procedure continues until pixel O is reached again.

5.2 Contour Segmentation

After extracting the boundary of the regions, we will next perform contour segmentation to parse the boundary into edge groups. In MOM, we do not require accurate contour segmentation. A simple segmentation algorithm with noise insensitivity is developed. It is described as follows:

Given a closed boundary **C**,

1. Compute the normals and gradients of normals at the boundary points.
2. Segment boundary into edge segments.
3. Merge adjacent edge segments with similar characteristics to form edges.

The following sections explain the details of the contour segmenting algorithm.

5.2.1 Normal Computation

Unit normal vector $\vec{u}_i$ at each pixel p_i along the contour is computed by applying Principle Components Analysis (PCA) on the pixel positions in the neighborhood of pixel p_i. The unit normal vector is given by the second eigenvector of PCA and is oriented outward from the interior of the object region.

After computing the unit normal vectors, a symmetric nearest-neighbor non-linear filter is applied to smoothen the normal vectors along the object contour to reduce noise. For a pixel p_i, its normal vector $\vec{u}_i$ is smoothened as follows:

$$\begin{aligned}
\vec{u}_i &= \frac{\sum\limits_{j\in\Lambda} w_j \vec{u}_j}{||\sum\limits_{j\in\Lambda} w_j \vec{u}_j||} \\
w_j &= \begin{cases} 1, & \text{if } |\frac{\vec{u}_j - \vec{u}_i}{||\vec{u}_j - \vec{u}_i||}| < |\frac{\vec{u}'_j - \vec{u}_i}{||\vec{u}'_j - \vec{u}_i||}| \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{5.2.1}$$

Here Λ is the set of neighboring points of p_i, $\vec{u}_j$ and $\vec{u}'_j$ are the normal vectors of the opposite neighbors of p_i. Equation 5.2.1 performs the following: (1) select the neighbors of p_i whose normals $\vec{u}_j$ are closer to $\vec{u}_i$ than their symmetrical counterparts $\vec{u}'_j$ are, and (2) assign to p_i the averaged normal of the selected neighbors. Figure 5.3 shows an example of smoothing the normal value of point A with a sampling window of size 3. The normal of point $L3$ is closer than that of point $R3$ is to that of point A. So, the normal of point $L3$ is selected for the computation of *A's* mean normal. In the same way point $R1$ and $R2$ are selected. Consequently, the normal of A is the averaged normal of points $L3$, $R1$ and $R2$. In MOM the sampling window size is set as 5 neighbors on each side of pixel p_i.

An ordinary linear filtering which averages all the neighbors' normals will blur sharp corners. On the contrary, the symmetric nearest-neighbor nonlinear filtering will preserve the corners and compute the normals more accurately.

After smoothing the normals, the gradients g_i of the normal at each pixel p_i is

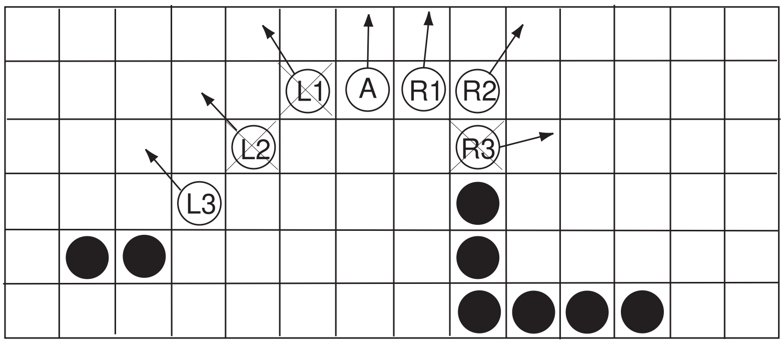


Figure 5.3: Symmetric nearest-neighbor nonlinear filtering. The normal value of point A is smoothed with a sampling window of size 3. Solid arrows are normal directions. Point $L3$, $R1$ and $R2$ are selected for calculating the normal of point A because their normals are closer than those of their symmetric counterparts (point $R3$, $L1$ and $L2$) are to that of point A.

computed as follows:

$$g_i = \frac{1}{|\mathcal{L}_i|} \sum_{j \in \mathcal{L}_i} w_{ij} \vec{u}_j - \frac{1}{|\mathcal{R}_i|} \sum_{j \in \mathcal{R}_i} w_{ij} \vec{u}_j \tag{5.2.2}$$

where $\mathcal{L}_i$ and $\mathcal{R}_i$ are the left and right neighborhoods of pixel p_i, and the weight w_{ij} is inversely proportional to the distance between pixels p_i and p_j and $|.|$ is the number of pixels in the neighborhood. Note that $\Lambda = \mathcal{L} \cup \mathcal{R}$.

5.2.2 Line Segmentation

In this step, the boundary is segmented into edge segments using the following algorithm.

1. Find the set $\mathbf{S}$ of pixel p_i with locally maximum normal gradient g_i.

   To accurately determine the local maxima of g_i from the whole boundary pixels, we use non-maxima suppression with a window size of 5 pixels.

2. Select pixels from $\mathbf{S}$ with $g_i > \Gamma$. Segment boundary into edge segments at these points.

   The boundary is segmented at locations with significant change of normal $\vec{u}$, i.e., large gradient g_i. Pixels along straight lines and smooth curves have small g_i. Locations such as corners have large g_i. The value of Γ is set so that significant corners are detected. Hence the boundary is over-segmented into edge segments.

3. Calculate the characteristics of edge segments.

   The characteristics of edge segments include: (1) length L, (2) average normal $\vec{n}$, and (3) curvature ω. Length $L(e_i)$ of edge segment e_i is approximated by the number of pixels of the segment. Average normal $\vec{n}_i$ of e_i is the average of the normals of the pixels along the segment. PCA is performed over the locations of the pixels along e_i to estimate the curvature ω_i of e_i as $\omega_i = \lambda_2/\lambda_1$, where λ_1 and λ_2 are the first two eigenvalues of PCA. The value of ω_i ranges between 0 and 1.

4. Merge edge segments with similar characteristics into edges.

   Adjacent edge segments e_i and e_j are merged if (1) they have similar curvature ($\omega_i \approx \omega_j$) and (2) they have similar average normal direction ($\vec{n}_i \approx \vec{n}_j$). The characteristics of newly merged edge segments are updated.

5.3 Edge Grouping

Edge grouping consists of three steps. The first step groups edges into edge groups. The second step detects characteristics such as parallelism and cotermination among the edges in an edge group. The third step identifies the type of edge group.

5.3.1 Initial Edge Grouping

We group edges into edge groups at convex points since the edges of the geons' aspect contours are always connected at convex points. Consequently, each edge group contains a connected set of edges which form a convex opened boundary that begins and ends at concave corners. In the case that an object contour contains at most one concave corner, the entire closed contour is grouped into a single edge group. Insignificant edge groups are discarded. An edge group is considered insignificant if the total length of the edges that it contains is small compared to the total length of the object contour.

Given two adjacent edges e_A and e_B, we determine whether their joint is concave by their normal vectors $\vec{n}_A$, $\vec{n}_B$ and their relative locations (Fig. 5.4). Let $\vec{AB}$ denote the unit vector pointing from point A to point B.

If $\vec{n}_A \cdot \vec{AB} > 0$ and $\vec{n}_B \cdot \vec{AB} < 0$, a concave corner is found.

If $\vec{n}_A \cdot \vec{AB} < 0$ and $\vec{n}_B \cdot \vec{AB} > 0$, a convex corner is found.

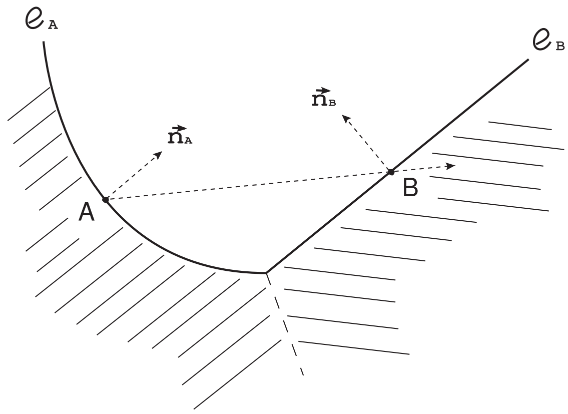


Figure 5.4: Concave corner detection. $\vec{n}_A$, $\vec{n}_B$ are the normal vectors of edges e_A and e_B. $\vec{AB}$ is a unit vector pointing from point A to point B. When $\vec{n}_A$ and $\vec{n}_B$ point toward each other, a concave corner is found.

5.3.2 Edge Group Characteristics Detection

After grouping edges into edge groups, the characteristics of edge groups are then extracted in order to identify them. As mentioned before, geon contours differentiate from each others by (1) the number of edges, (2) the type of edges (straight or curved), and (3) the relationship among those edges (adjacency and parallelism). Among these characteristics, the number of edges and parallelism between them are determined immediately after edge grouping. On the other hand, the type of edges and the adjacency relationship are determined at the matching process in next stage (Sec. 5.3.3).

After edge grouping, parallelism between the edges in each group is determined so as to identify parallel edges that fall on opposite sides of an object part. Two edges e_i and e_j with mean normal vector $\vec{n}_i$ and $\vec{n}_j$ are parallel and opposite if their normal vectors are parallel and opposite: $\vec{n}_i \approx -\vec{n}_j$, and their curvatures are roughly the same: $\omega_i \approx \omega_j$.

5.3.3 Edge Group Identification

Edge groups correspond to partial aspect contours of a significant object in an image. There is a total of 28 standard partial aspect contours derived from our geon set (Fig. 3.9). Edge group identification is to match the retrieved edge group with the closest partial aspect contour.

Edges in edge group are connected to each other, hence the possible number of matching alternatives between an edge group and a partial aspect contour is limited. The edges are matched with the lines of a partial aspect contour by comparing their curvatures and adjacency relationships. Details are given in the following paragraph.

Let h denote a mapping from a partial aspect contour q to an extracted edge group g such that $h(e_i) = e'_j$ for some lines e_i in q and edges e'_j in g, and that $h(e_i)$ is connected to $h(e_j)$ if e_i is connected to e_j. Let q' denote a part of q which is matched with g. The match value between g and q is defined as follows:

$$M(g,q) = \max_{q' \subseteq q} M(g,q') \tag{5.3.1}$$

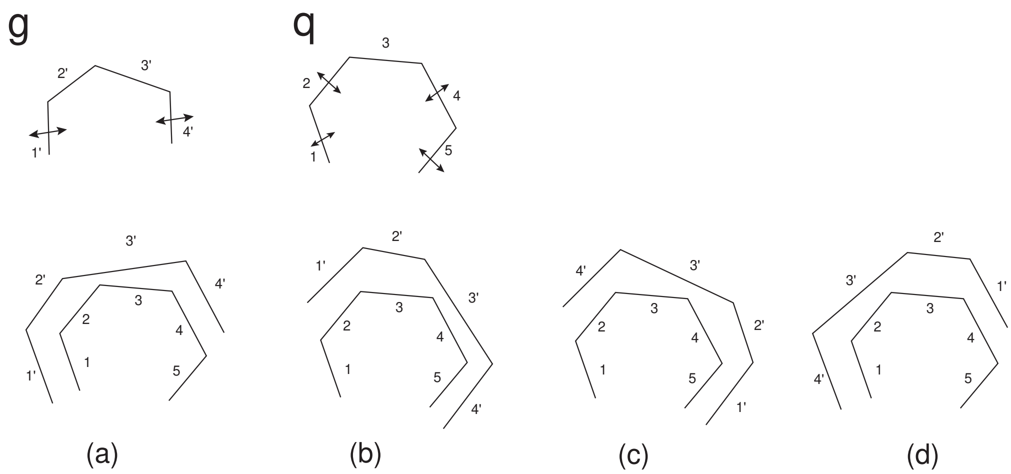


Figure 5.5: Matching process in edge group identification. Parallel arrow pair indicates the lines under them are parallel to each other. There is a total of four possible matches between g and q (a–d).

where

$$M(g,q') = \frac{1}{\sqrt{|g||q|}} \sum_{e_i \in q'} [\, 1 - |\omega(e_i) - \omega(h(e_i))| \,] \tag{5.3.2}$$

In Equation 5.3.2, $|g|$ and $|q|$ are the number of edges in g and q, $\omega(.)$ is the curvature of an edge. The value of $M(g,q')$ ranges from 0 to 1, reaching 1 when g and q are exactly matched. The term $\sum_{e_i \in q'} [\, 1 - |\omega(e_i) - \omega(h(e_i))| \,]$ measures the degree of match: the more the matched edges and the closer is the curvature between e_i and $h(e_i)$, the bigger is the match value.

All possible matches between g and q, i.e., all possible q', are considered and the maximum match value is chosen to be the match value between g and q (Eq. 5.3.1). Figure 5.5 shows an example. An important criterion for determining q' is whether

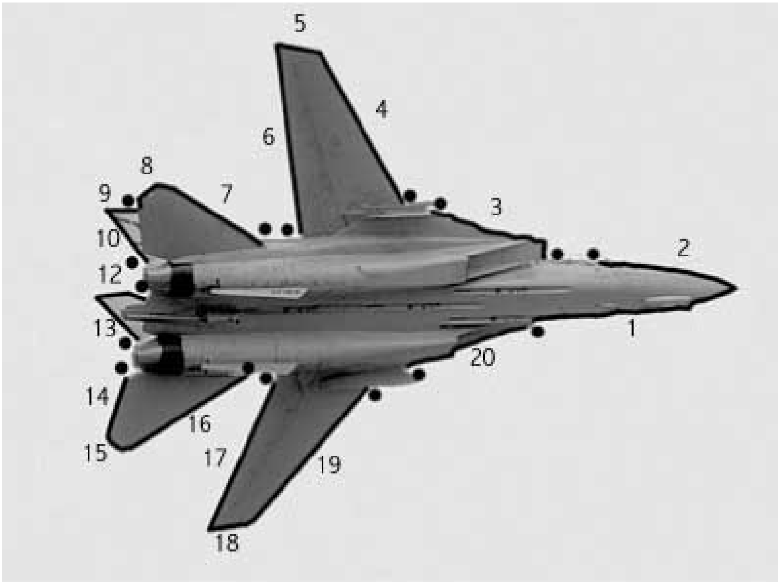


Figure 5.6: Edge segmentation result. Dots indicate the detected concave corners which begin or end an edge group. Numbers indicate the detected significant edges.

the mapped counterparts e_i and $h(e_i)$ have the same parallelism relationship among the other edges. For example, in Figure 5.5(a), Edge $1'$ is parallel to edge $4'$. The matching line 1 in q is also parallel to line 4. Therefore, the connected edges $1', 2', 3', 4'$ of g match the connected lines 1, 2, 3, 4 of q. Similarly, g and q can be matched in three other ways (Fig. 5.5(b), (c), (d)). Note that edge type, length, and angle are not used in this matching stage. They are used in subsequent stages to assist in component-level matching (Chapter 6).

5.4 Testing Results

Figure 5.6 shows an example of the contour segmentation result. There is a total of

Table 5.1: Edge groups for the edges in Figure 5.6. The edge group ID corresponds to the partial aspect contour ID in Figure 3.9.

Edge Group	g1		g2	g3			g4		g5	
Edge Group ID	5		2	7			3		3	
Edge Member	1	2	3	4	5	6	7	8	9	10

Edge Group	g6			g7			g8			g9
Edge Group ID	7			7			7			1
Edge Member	11	12	13	14	15	16	17	18	19	20

9 edge groups detected, after discarding insignificant edge groups. The edge group ID and their edge members are shown in Table 5.1. The edge group ID number corresponds to the partial aspect contour ID in Figure 3.9.

Chapter 6

Pose-Invariant 3D Model Matching

After extracting the edge groups from an input image, we obtain partial shape information of the object parts. With the object modeled by geons, MOM compares the shape of the significant object in the image with those in the 3D model. MOM will produce a match value telling how close is the shape of the significant object to the shape of the model.

6.1 Similarity Measurement

To perform model matching, we have to first define the similarity between a model and an object. Recall that a 3D geon model can have several 2D aspects and thus several aspect contours. An edge group forming a closed contour would match a single aspect contour completely. On the other hand, an edge group that forms an opened contour could match parts of several different aspect contours. Suppose that an aspect contour c_i has n_i edges and an edge group g_j matches m_i edges of c_i, then the degree of match $M(c_i, g_j)$ between c_i and g_j is defined as m_i/n_i. Since the identity of edge groups are already determined during edge grouping, the above value is easy to obtain. We don't directly combine the match procedure in edge grouping with the matching procedure here; otherwise the computation complexity will increase exponentially. The goodness of match $M(v, g_j)$ between primitive 3D shape v and edge group g_j is given by the largest goodness of match with the aspect contours of v:

$$M(v, g_j) = \max_{c_i \in v} M(c_i, g_j) \ . \tag{6.1.1}$$

Adjacency relationships between connected components are represented in a graph. For a model, the model graph contains vertices each representing a model component. Each vertex is, thus, associated with a primitive 3D shape which has several aspect contours. Two vertices are connected by a graph edge if the corresponding model components are connected to each other (e.g., Fig. 6.1c). For an image object, the object graph contains vertices each representing a 2D edge group. Two vertices are connected by a graph edge if the corresponding edge groups are connected to each other. Since the edge groups are connected only to adjacent edge groups along the object contour, the object graph is always a circular graph (e.g., Fig. 6.1d). Having represented a model and the object as graphs, the model matching problem can now be cast as a graph matching problem.

Given a model graph $G(V, E)$ with vertex set V and edge set E, and an object

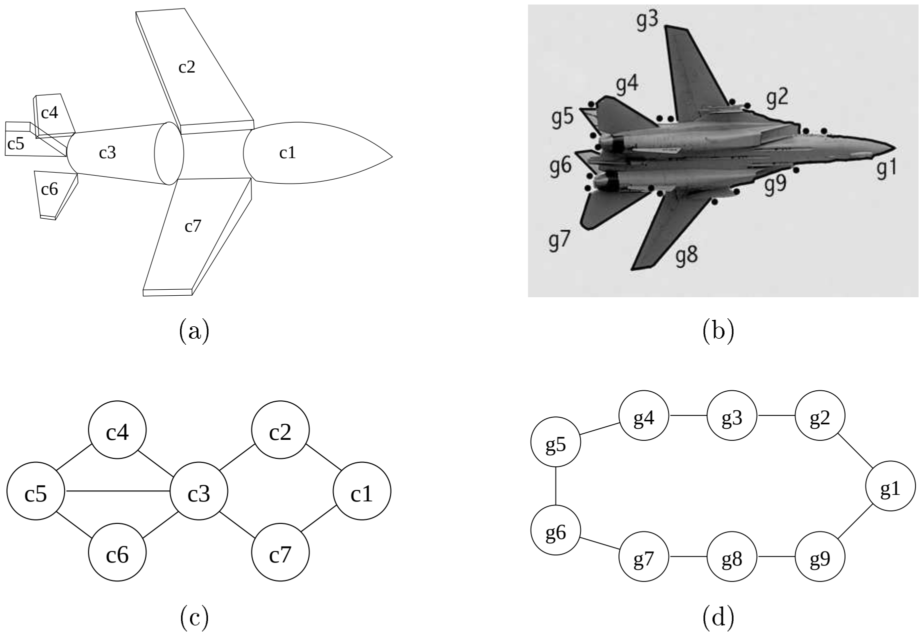


Figure 6.1: Airplane geon model. (a) An airplane model with 7 components. (b) An image of an aircraft. Each significant edge group is denoted by a black line. The black dots denote the ends of the edge groups. (c) The model graph of (a). (d) The object graph of (b).

graph $G'(V', E')$ with vertex set V' and edge set E', an *admissible mapping* from G to G' is a mapping f that satisfies the following properties:

1. For each vertex v in V, $f(v)$ is a vertex in $V' \cup \{\phi\}$. If $f(v) = \phi$, the null vertex, then there is no available match for v. Conversely, for each vertex v' in V', $f^{-1}(v') \in V \cup \{\phi\}$.

2. For any two vertices u and v in V connected by an edge e in E, if $f(u) \neq \phi$ and $f(v) \neq \phi$, then $f(u)$ and $f(v)$ are connected by an edge $f(e)$ in E'.

3. For each sequence of vertices $v_1, \ldots, v_n$ in V connected by edges $e_1, \ldots, e_{n-1}$ in E such that $f(v_1) \neq \phi$, $f(v_n) \neq \phi$, and $f(v_2) = \cdots = f(v_{n-1}) = \phi$, there exists a sequence of vertices $v'_1, \ldots, v'_{n'}$ in V' connected by edges $e'_1, \ldots, e'_{n'-1}$ in E' such that $f(v_1) = v'_1$, $f(v_n) = v'_{n'}$, $f^{-1}(v'_2) = \cdots = f^{-1}(v'_{n'-1}) = \phi$, and $f(\sum_i e_i) = \sum_i e'_i$, where $e_i + e_j$ denote concatenation of edges e_i and e_j.

Property 3 states that some connected sequences of vertices in the model and object graphs may not have matching counterparts. Property 2 is a special case of property 3 with $n = n' = 2$. These properties are required because, in practice, some objects may have additional parts not presented in a model while other objects have missing parts due to occlusion.

The overall goodness of match $M(G, G')$ between model graph G and object graph G' is defined with respect to an admissible mapping f from G to G':

$$M(G, G') = \left[\sum_{v_i \in V} W(v_i)\, M(v_i, f(v_i))\right] \left[\sum_{v_i \in V} L(f(v_i))\right] . \tag{6.1.2}$$

The term $M(v_i, f(v_i))$ is the degree of match between model component v_i and its best matching object part $f(v_i)$ (Eq. 6.1.1). If $f(v_i) = \phi$, then $M(v_i, \phi) = 0$. The term $W(v_i)$ is a weighting factor encoding the significance $S(v_i)$ and relative size $L(v_i)$ of model component v_i:

$$W(v_i) = S(v_i)[\,1 - |L(v_i) - L(f(v_i))|\,] . \tag{6.1.3}$$

The significance $S(v_i)$ encodes how significant a component is in contributing to a good match. For instance, the head and wings are the most distinctive parts of an airplane, and they can clearly distinguish an airplane from other objects. Therefore, the significance of these parts are larger than those of other parts. The significance values of the components are normalized over the entire model: $\sum_{v_i \in V} S(v_i) = 1$. The relative size $L(v_i)$, $0 \leq L(v_i) \leq 1$, of model component v_i is measured relative to the whole model. Similarly, the relative size $L(f(v_i))$ of the matching object part $f(v_i)$ is measured relative to the whole object. A good match requires that the relative sizes of the model component and the matching object part be roughly equal. The summation of $L(f(v_i))$ in Eq. 6.1.2 measures the amount of matched object parts relative to the whole object. The more the matched parts and the larger the parts, the higher is the resulting degree of match between the model and the object. Note that $0 \leqslant W(v_i) \leqslant 1$, $0 \leqslant \sum_{v_i \in V} L(f(v_i)) \leqslant 1$, hence $0 \leqslant M(G, G') \leqslant 1$.

6.2 Matching Algorithm

The model matching problem can now be stated as the following optimization problem:

> Given a model graph $G(V, E)$ and an object graph $G'(V', E')$, find the admissible mapping $f : G \to G'$ that maximizes the overall goodness of match $M(G, G')$.

This problem is solved using a variant of Messmer and Bunke's error-tolerant subgraph isomorphism algorithm [28]:

1. Create null graphs g and g', and insert the pair (g, g') into the backtrack queue Q_b.

2. Repeat n times:

   2.1. Remove first pair (g, g') from Q_b.

   2.2. Empty normal search queue Q_n, insert (g, g') into Q_n.

   2.3. Repeat:

   (a) Remove first pair (g, g') from Q_n.

(b) If all vertices in V have been considered for connecting to g, then break out of inner repeat loop, i.e., go back to Step 2.1.

(c) For each pair $(v, v') \in V \times V'$ such that v can connect to g and v' can connect to g' directly and indirectly (i.e., with and without vertex deletion):

i. Connect v to g, v' to g', and compute $M(g, g')$.

ii. Insert (g, g') into Q_n in decreasing order of $M(g, g')$.

iii. If a component in V has at least one neighbor which has the same shape as v and $|g| > 1$, insert (g, g') into Q_b in increasing order of $|g|$ and decreasing order of $M(g, g')$. ($|g|$ is the number of vertices in g.)

2.4. Save (g, g') and $M(g, g')$ as a candidate solution.

2.5. If Q_b is empty, break out the outer repeat loop, i.e., go to Step 3.

3. Return the candidate solution with the largest $M(g, g')$.

The idea is to connect vertices to matching model and object subgraphs until all the vertices in the model graph have been considered for connection. Then, the matching model and image subgraphs are saved as a candidate solution.

When the object model contains components with the same type and same size, if they are adjacent to each other, their match goodness to the image edge groups will be the same, which arises match ambiguity. Due to this possible ambiguity of the model components, initial erroneous matches that happen to have large match values may lead the search algorithm down the wrong search paths. So, the backtrack queue Q_b is used to keep other possible matches when an ambiguous component is encountered. Once a candidate solution is found and saved, the normal search queue Q_n is emptied and a new search from another subgraph is initiated. The backtrack is always performed from the initial matching stage which is suspected to produce misleading match value. This method allows the algorithm to escape from local maxima of the solution space. This procedure is repeated n times, and the best solution among them is returned as the final matching result. Here the maximum possible value of n is the number of subgraph pairs in Q_b. In practice the subgraph bearing a local maximal match value will not deviate too far away from the subgraph of the global maximal value if we trace back to the initial erroneous match point. That's why by default we set n as the number of the components in object model.

6.3 Test Results And Discussion

In this section we will show that the match value produced by MOM can be used for object classification and can also become the backbone for a image retrieval system.

6.3.1 Details of Model Matching

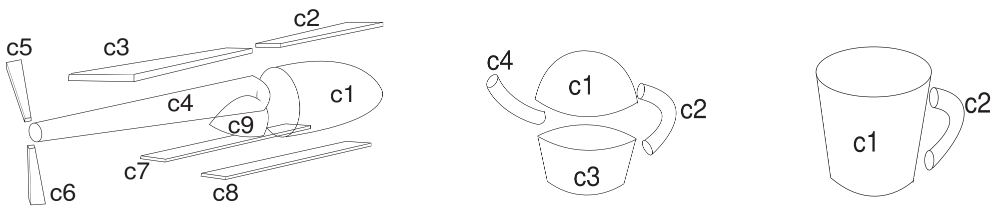
Figure 6.1(a) and (c) illustrate a model of an airplane composed of 7 components and the corresponding model graph. Figure 6.1(b) shows the edge groups extracted from a real image of an aircraft. Each edge group is denoted by a black line. The corresponding object graph is shown in Fig. 6.1(d). The detailed results of matching the model and the object in Fig. 6.1 are shown in Table 6.1. Take object part g1 for example. It has 2 edges and it matches a 3-edge aspect of model component c1 best. Therefore, its match value is 2/3. The weights $W(v_i)$ indicate the relative importance $S(v_i)$ of the model components and how similar the matched pairs are in terms of relative size $|L(v_i)-L(f(v_i))|$ (Eq. 6.1.3). For this example, the nose (c1) and the wings (c2, c7) have higher weights. The overall match value is obtained with Equation 6.1.2. Here, $\sum_v L(f(v_i)) = 0.901$, thus $M(G, G') = 0.571$.

6.3.2 Model Matching Results

A large number of images were collected for various types of objects ranging from simple objects such as mug and teapot to complex objects such as airplane and helicopter (Fig. 6.1(a) and Fig. 6.2). Each object model was regarded as a query object and was matched against all the images. Figure 6.3 and 6.4 illustrates 40 sample images for each of four object types: airplane, helicopter, teapot, and mug. Each set of 10 images include objects at various poses, objects that are very similar

Table 6.1: Results of matching the input object in Fig. 6.1(b) with the model in Fig. 6.1(a).

v_i	$f(v_i)$	$M(v_i, f(v_i))$	$S(v_i)$	$W(v_i)$
c1	g1	2/3	0.2	0.198
c2	g3	3/4	0.2	0.196
c3	ϕ	0	0.1	0.000
c4	g4	3/4	0.1	0.090
c5	g6	3/4	0.1	0.087
c6	g7	3/4	0.1	0.099
c7	g8	3/4	0.2	0.197



(a) (b) (c)

Figure 6.2: Sample object model. (a) helicopter, (b) teapot, (c) mug.

to the models, objects with missing parts due to occlusion, and ambiguous objects (e.g., object A10 is a mixture of airplane and helicopter). Table 6.2 tabulates the matching results.

As shown in Table 6.2, the airplane model matches images A1 to A8 well because all these images contain enough object parts that result in good matches with the airplane model. A9 and A10 have lower match values because A9 has only one wing and one tail plane, while A10 is a mixture between an airplane and a helicopter. The helicopter model matches images H1 to H9 well. It matches H10 less

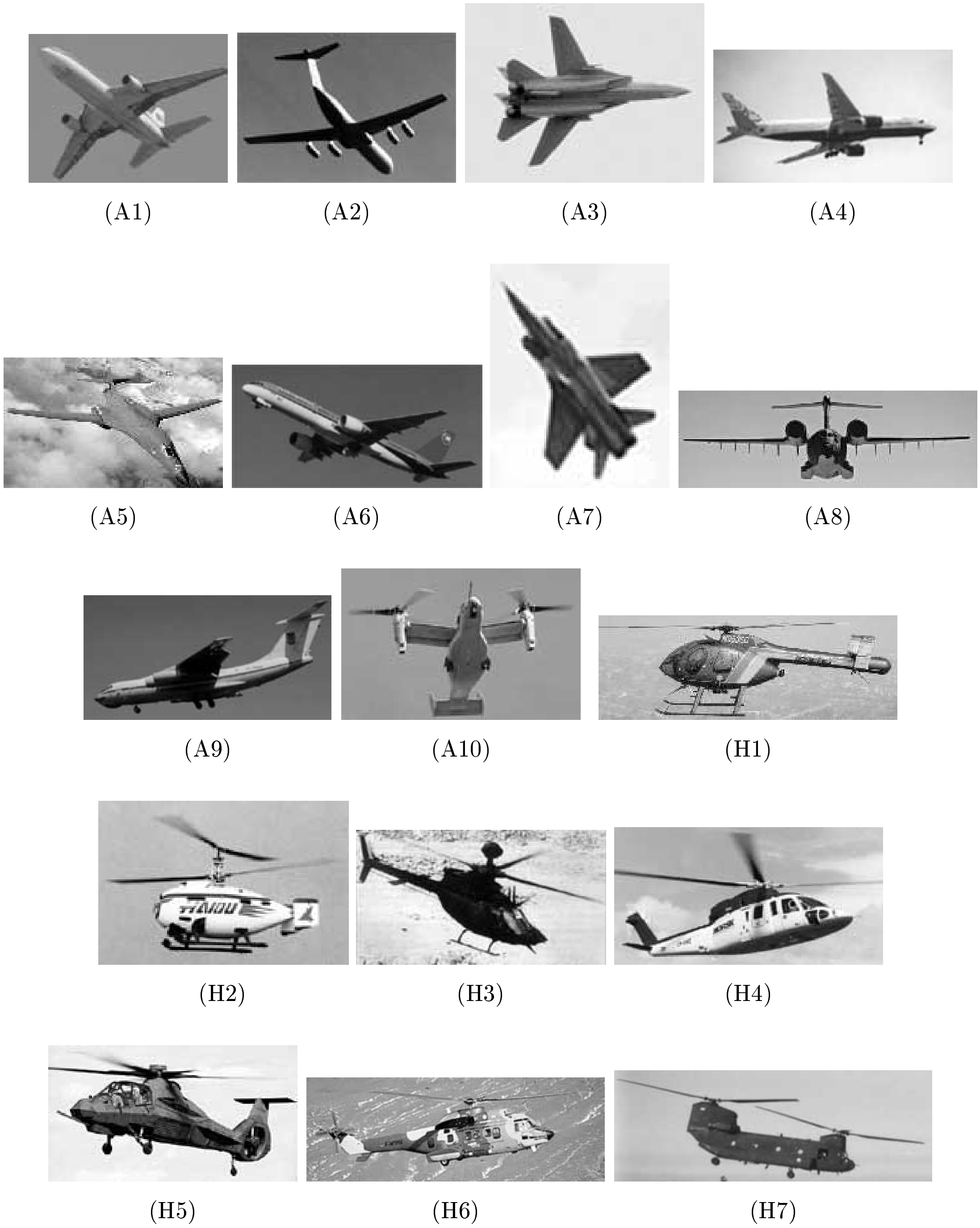


Figure 6.3: Samples of testing images.

Figure 6.4: Samples of testing images.

well since H10 is also a mixture between an airplane and a helicopter. It confuses A8 as a helicopter and produces a high match value because the high-mounted tail planes of the airplane resemble the rotor blades of a helicopter. Matching the airplane and helicopter models with images of teapot and mug yields very low match values (last two rows of Table 6.2).

The teapot model matches images P1 to P7 well. It matches P8 to P10 poorly because P8 has only 2 of the 3 model components; P9's handle is mounted on top, enclosing the cap of the teapot, instead of on the side; and P10 is oriented such that the contours of both its handle and spout cannot be detected. The mug model matches M1 to M10 well even though M9 and M10 have missing handles. It confuses P8 as a mug precisely because the teapot does not have a spout and thus resembles a mug. Matching the mug and the teapot model with images of airplane and helicopter yields extremely low values (last two rows of Table 6.2).

Figure 6.5 shows the relationship between match value and the percentage of matched components relative to the total number of components of a model and an image object. The match value is high when both model and object have high percentage of matched components. Otherwise, the match value is low. This is consistent with the principle that an object is similar to a model only if both have similar overall structure. If a model matches only a small part of an object or an object matches only a small part of model, then the overall structure of the object is different from that of the model. In this case, our method gives a correct low

Table 6.2: Results of matching 3D models to various images. The last two rows show that largest match values among the images of the specific types. (A_i: images A1–A10, H_i: images H1–H10, P_i: images P1–P10, M_i: images M1–M10.)

Image	Model		Image	Model	
	Airplane	Helicopter		Teapot	Mug
A1	0.590	0.289	P1	0.535	0.242
A2	0.586	0.200	P2	0.521	0.245
A3	0.571	0.212	P3	0.520	0.261
A4	0.542	0.270	P4	0.508	0.271
A5	0.541	0.274	P5	0.498	0.251
A6	0.538	0.239	P6	0.481	0.262
A7	0.461	0.214	P7	0.457	0.217
A8	0.405	0.470	P8	0.239	0.438
A9	0.387	0.274	P9	0.232	0.204
A10	0.266	0.244	P10	0.010	0.363
H10	0.276	0.373	M10	0.060	0.400
H9	0.223	0.428	M9	0.263	0.410
H8	0.218	0.475	M8	0.286	0.527
H7	0.201	0.481	M7	0.286	0.546
H6	0.254	0.509	M6	0.246	0.552
H5	0.240	0.518	M5	0.282	0.552
H4	0.225	0.594	M4	0.269	0.559
H3	0.185	0.604	M3	0.282	0.569
H2	0.267	0.623	M2	0.264	0.575
H1	0.227	0.626	M1	0.025	0.625
T_i max.	0.042 P6	0.021 P6	A_i max.	0.061 A7	0.079 A8
M_i max.	0.032 M6	0.022 M3	H_i max.	0.078 H2	0.058 H2

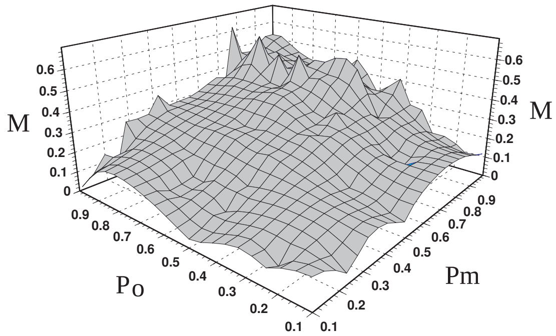


Figure 6.5: A plot of mean matched value M vs. percentages of matched components of the model P_m and the object P_o.

match value (see Fig. 6.6(a) and (b)). The plot shows that the match value are higher for images that are more similar to the models. Therefore, the degree of match can be used to rank the images for image retrieval.

Figure 6.6(c) and (d) illustrate cross-sections of the 3D plot in Fig. 6.5 at $P_o = 1$ and $P_m = 1$. In Fig. 6.6(c), a match value M above point B always corresponds to a percentage of matched component P_m of 0.74 or more, which is a satisfactory percentage for indicating a good match between object and model. Similarly, a match value below point A always corresponds to a low matched percentage which indicates a poor match. On the other hand, a match value between points A and B is ambiguous because it corresponds to several matched percentages that differ drastically. For example, a matched value of 0.35 corresponds to matched percentages of 0.42 and 0.69. This large difference in matched percentage corresponds to large difference in the quality of match between an object and a model. The same phenomenon exists in Fig. 6.6(d). According to these observations, two

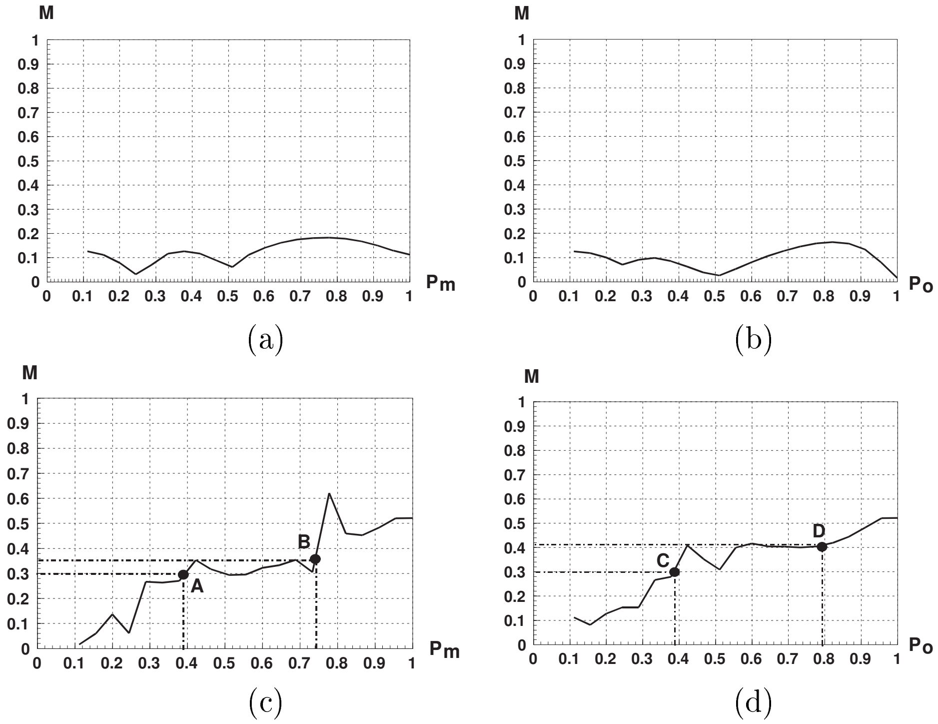


Figure 6.6: Cross-sections of the 3D plot in Fig. 6.5 at (a) $P_o = 0.1$, (b) $P_m = 0.1$, (c) $P_o = 1$, and (d) $P_m = 1$.

thresholds can be identified: match values above threshold $\Gamma_g = 0.42$ correspond to good matches, and match values below threshold $\Gamma_p = 0.30$ correspond to poor matches. Match values between these two thresholds will correspond to ambiguous matches. The thresholds correlate closely to human's perception of the similarity between the images in Fig. 6.3 and 6.4 and the models. Take the airplane model for example. Images A1 to A7 are perceptually similar to the airplane model, A8 and A9 are ambiguous, and A10 and the rest of the images are not similar.

Chapter 7

Discussion and Conclusion

7.1 Discussion

The MOM method described in this thesis is most similar to the PARVO system of Bergevin and Levine [1] and the geon-based method of Dickinson et al. [10, 11]. Nevertheless, there are major differences between these methods.

For low-level feature extraction, PARVO extracts many features such as T-junctions, parallelism, symmetry, etc. These features are very useful for identifying the underlying 3D primitive shape. However, it is almost impossible to extract them reliably from natural images. Therefore, the original PARVO in [1] requires manual input of line segments. In contrast, MOM acknowledges the difficulty of extracting such complex features from natural images. Instead, it extracts only the edges along the objects' boundary contours. Although these edges constitute only partial information about the object parts, MOM can still match them with the object model successfully.

The approach of Dickinson et al. defines a strict hierarchy of features from object, through 3D primitives, 2D aspects, 2D faces, boundary groups, to edges. In the recognition process, edges are grouped into boundary groups. A set of boundary groups are grouped and classified as one of the faces; a set of faces are grouped and classified as one of the aspects, and so on. That is, correct object classification relies on the successful recovery of each face, aspect, and 3D primitive of the object. On the other hand, MOM does not attempt to recover faces, aspects, and 3D primitives. Instead, it directly matches the edge groups extracted from an object's boundary contour to the possible aspect contours of the model components. Since the edge groups contain only partial information about the object parts, they can usually match several aspect contours. Therefore, this matching at the component level is ambiguous. The ambiguity is resolved by searching for the best match between each component of a model with the corresponding edge group of an object, taking into account the spatial relationships between the model components.

Dickinson's approach represents each level of the model hierarchy by a set of graphs. PARVO represents both a model and an input object as graphs. In the matching process, an initial filtering is first applied to identify the models that contain all the 3D primitives extracted from the input image. Then, an exhaustive method is used to find the model that best matches the input object. MOM's graph representation is more similar to that of PARVO than that of Dickinson. MOM uses an error-tolerant subgraph isomorphism to match the model and the object.

To speed up the matching process, the subgraph isomorphism is implemented as best-first search with backtracking. This algorithm can be extended using the method illustrated in [28] to simultaneously match multiple model graphs with an input object graph. This extension will greatly improve the efficiency of the algorithm for matching many images in a large database.

Due to the need to extract complicated features and to fully recover individual 3D primitive, PARVO and Dickinson's method have been applied only to images of simple geometric shapes, line drawings, and range data. In contrast, MOM can be applied to natural images containing complex objects such as airplanes and helicopters.

7.2 Future Work

The error tolerant subgraph isomorphism algorithm presented in Chapter 6 is designed for matching a model and an object graph. The edge groups extracted from an image are matched with the corresponding partial aspect contour (Chapter 5), which produces a match value between the edge group and the partial aspect contour. Currently this match value is only used to determine the identity of the edge group. Errors in this process can affect subsequent subgraph matching procedure. To solve this problem, the match values in edge grouping can be used directly by the subgraph matching procedure.

The current implementation of MOM extracts shape features only from the bounding contour of an object. When an object is oriented such that its major parts do not appear along the bounding contours (e.g. Fig. 6.4, P10), the parts cannot be extracted and matched with the model. One way to overcome this problem is to extract edges within the bounding contour and to include these edges in the edge grouping process. This method will allow MOM to extract additional features for making more accurate matching.

The MOM method is currently applicable to images whose background can be easily removed. This is achieved with a simple color segmentation algorithm when the background is relatively homogeneous. Otherwise, a more sophisticated segmentation algorithm has to be applied. Bradshaw's regional image retrieval [5] scheme provides a possible solution to this problem. In his work regions in an image can be classified semantically as man-made or natural. With proper modification of his algorithm, regions belonging to complicated background which cannot be separated by MOM currently can then be extracted. For example, complicated background always contains natural structures in the form of different textures, such as trees. Bradshaw's method can detect such natural textures pretty accurately. Combining his method with MOM can then extract significant object from images with more complicated natural background.

Another possible modification of MOM is to feedback object classification system to the low-level image segmentation process, using shape and other a priori knowledge to enhance the performance of low-level modules.

7.3 Conclusion

This thesis presented a method for matching 3D models with complex objects in 2D images. It overcomes the partial information problem by performing multi-level matching. At the component level, partial information about object parts is extracted from the image and is matched with possible model components. This matching is usually ambiguous. At the object level, structural information such as relative size and position of object parts are used to constrain the matching process.

The model matching problem is cast as a graph matching problem and is solved using error-tolerant subgraph isomorphism algorithm. Test results show that the method can match 3D models with 2D objects at various poses. Moreover, the match values correlate well with the similarity between the objects and the models. For object-based image retrieval, the match values can be used to rank object images in decreasing order of similarity with the query model.

Compared to existing methods that perform detailed model fitting, our method is much more flexible because it does not require detailed model information. Instead, it performs qualitative matching of model components with object parts and can tolerate addition, deletion, and variation of object parts.

Our method also introduces a new way for a user to construct a query for image retrieval. Given a set of 10 3D primitive shapes, the user can put the primitive shapes together like LEGO bricks to construct a required model. This method is very intuitive and easy for a novice user to construct a query for retrieving images of specific objects from the database.

Experimental results show that MOM is a successful system for component-model-based object classification and that multi-level object matching is an effective method for real image object classification. With the possible enhancements, MOM will be useful for other applications such as object-based image retrieval.

Appendix A

Color Space Conversion

Several mathematical color spaces such as RGB, CIELUV, CIE(XYZ), YIQ, HSI, etc. exist. We use the CIELUV color space in our project. An image's colors are converted from the RGB space to CIELUV. We briefly describe these two color spaces and the conversion formula.

RGB color space is the most popular color space for digital pictures. In this model colors are represented by three components: R, G and B. To reduce the effect of varying light intensity, a *normalized RGB space* is developed as:

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}. \tag{A.0.1}$$

Note that $r + g + b = 1$.

The CIELUV space is developed by the CIE committee (Commission International de l'Eclairage—the International Commission on Illumination). It was verified that this color space is roughly perceptually uniform and the measurement of color distance by *Euclidean distance* is consistent with human's perception.

Among all the different color spaces only CIELUV, CIELAB, and Munsell HSV have this property. To convert from RGB space to CIELUV space, we first convert RGB space into CIE(XYZ) space.

$$\begin{aligned} X &= 0.607r + 0.174g + 0.200b \\ Y &= 0.299r + 0.587g + 0.114b \\ Z &= 0.000r + 0.066g + 1.117b \end{aligned} \tag{A.0.2}$$

The formulas for transforming CIELUV space from CIE(XYZ) space are:

$$\begin{aligned} &L = 116\sqrt[3]{100Y/Y_0} - 16, && \textit{if } Y/Y_0 > 0.008856 \\ &L = 903.3(Y/Y_0), && \textit{if } Y/Y_0 \leqslant 0.008856 \\ &u = 13L\,[\,4X/(X + 15Y + 3Z) - 0.20092\,], \\ &v = 13L\,[\,9Y/(X + 15Y + 3Z) - 0.46083\,] \end{aligned} \tag{A.0.3}$$

where $Y_0 = 1.0$ is the XYZ value for the reference white.

Appendix B

Geon Model Analysis

B.1 Detailed Contour Analysis

Every geon has a limited number of aspects. Each aspect has a boundary contour which is defined by (1) the number of lines, (2) the type of lines (straight or curved) and (3) the relationship among these lines (parallelism and adjacency). Although there are around 50 aspects, there are fewer distinct aspect contours (Fig. 3.7) because many aspects have the same aspect contours. Figure B.1 to Figure B.10 show the complete list of aspects and contours of each geon used in MOM.

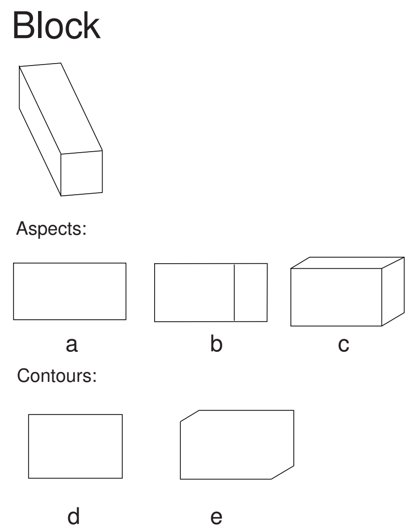


Figure B.1: Block

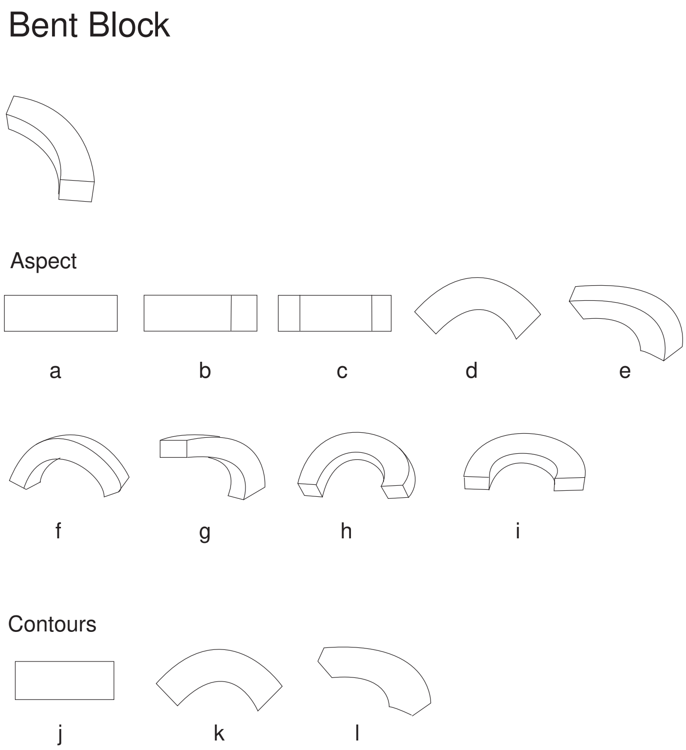


Figure B.2: Bent Block.

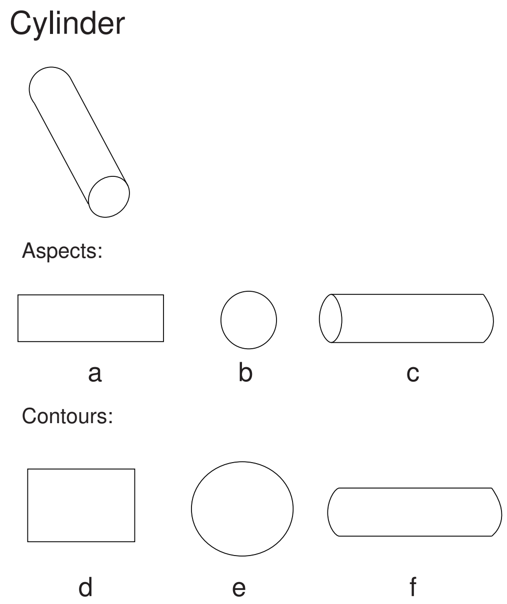


Figure B.3: Cylinder

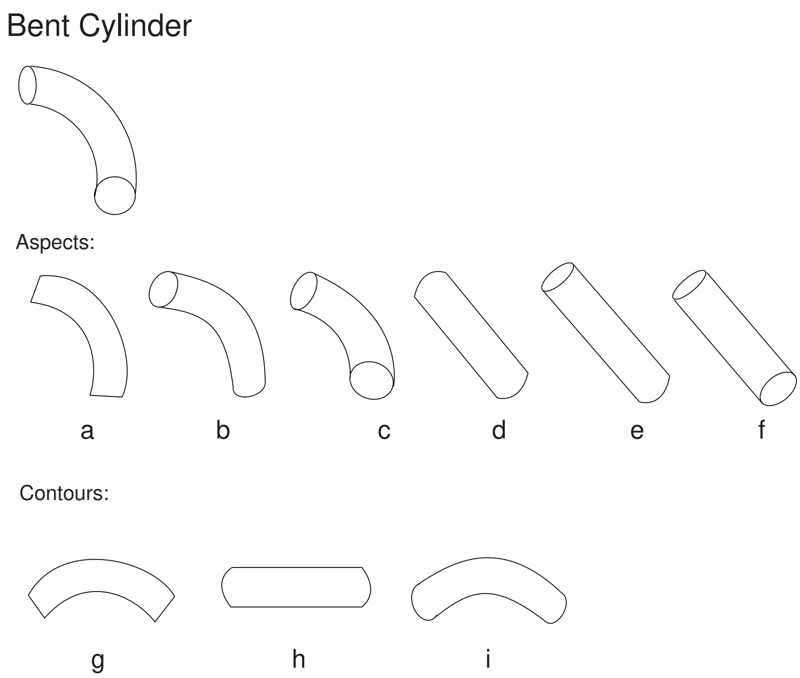


Figure B.4: Bent Cylinder

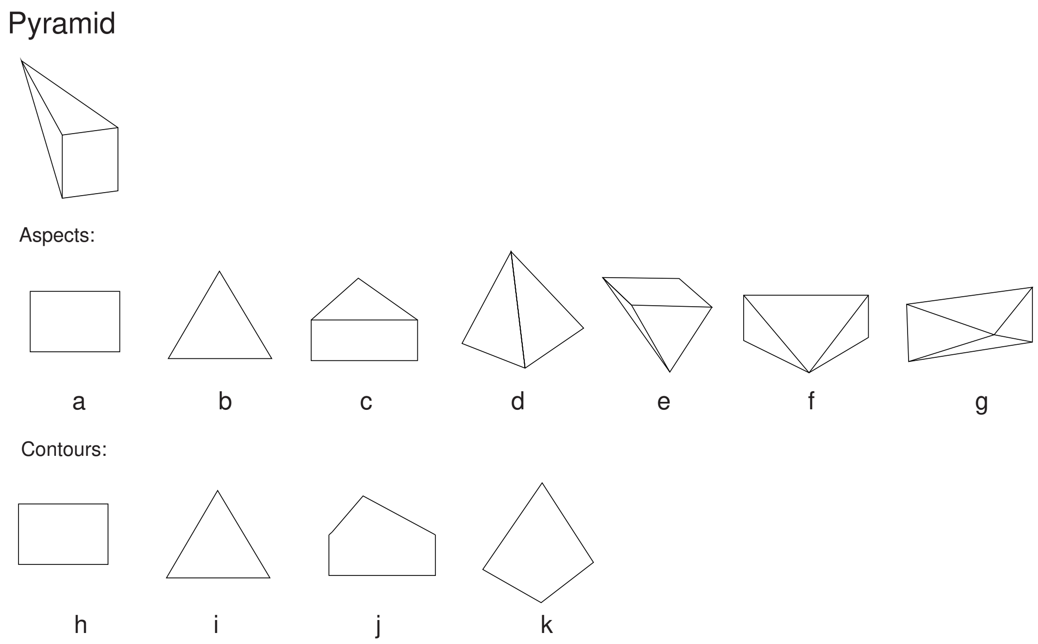


Figure B.5: Pyramid

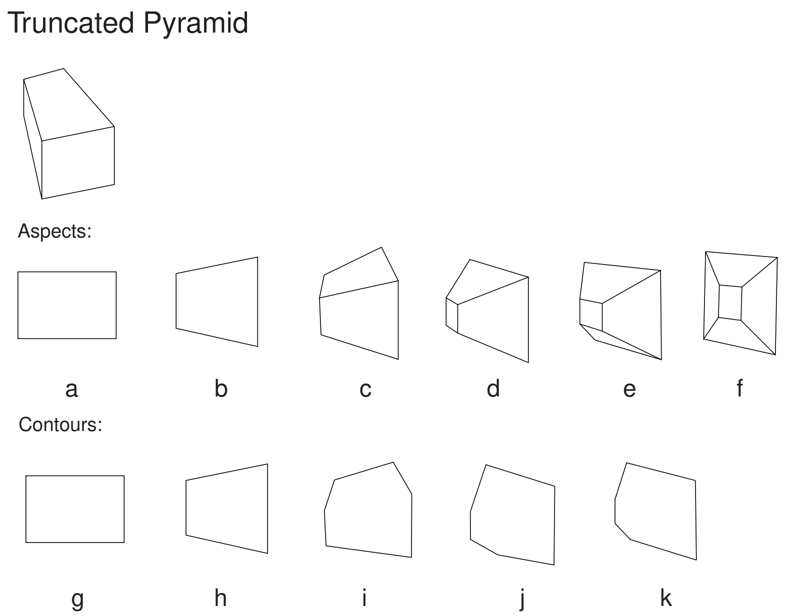


Figure B.6: Truncated Pyramid

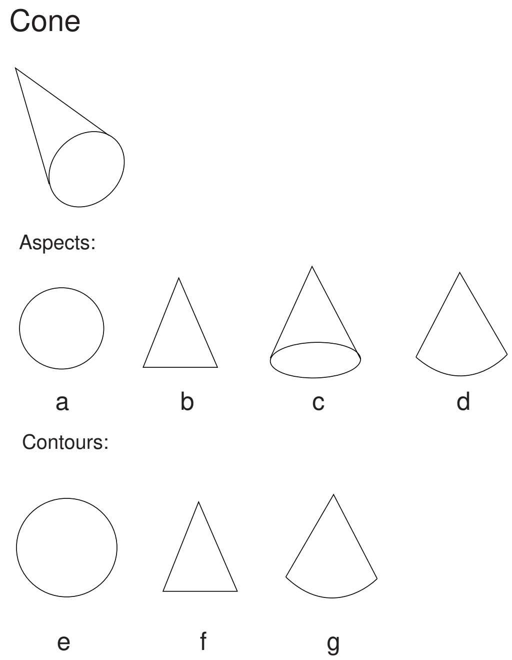


Figure B.7: Cone

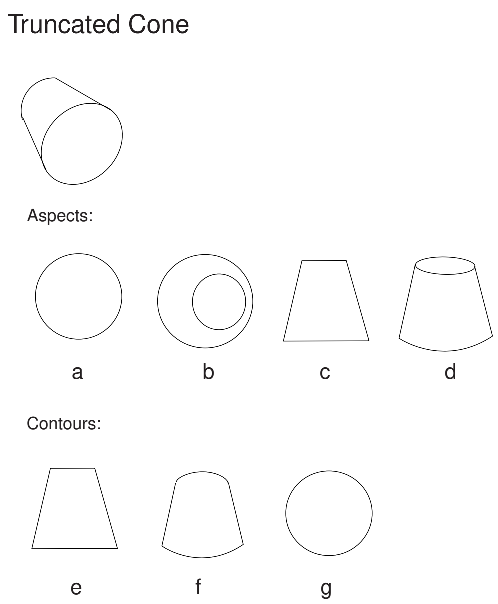


Figure B.8: Truncated Cone

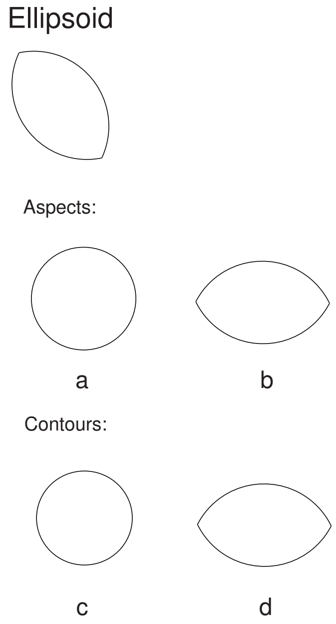


Figure B.9: Ellipsoid

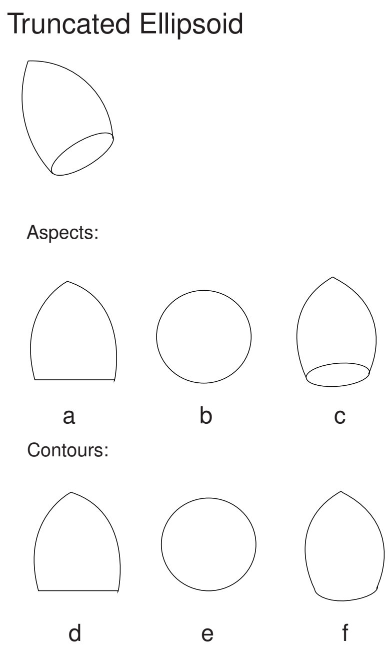


Figure B.10: Truncated Ellipsoid

B.2 Aspect Contour Features

Table B.1 lists the complete feature combination of each aspect contour. From this table we can see that the axis of symmetry is redundant because the other features are enough to differentiate the contours. Note that contours l and n have different features in that l's parallel pair are straight lines while n's are curves. Contours g and h are distinguished by their different parallel pairs.

Table B.1: Features of aspect contours in Figure 3.7.

Contour ID	Total Lines	Straight Lines	Curve Lines	Parallel Pairs	Symmetric Axis
a	3	3	0	0	1
b	4	4	0	0	0
c	4	4	0	2	2
d	4	4	0	1	1
e	5	5	0	1	0
f	6	6	0	1	0
g	6	6	0	2	0
h	6	6	0	2	0
i	6	6	0	3	0
j	3	2	1	0	1
k	4	2	2	0	1
l	4	2	2	1	1
m	4	0	4	1	1
n	4	2	2	1	1
o	6	4	2	1	0
p	3	1	2	0	1
q	2	0	2	0	1
r	3	0	3	0	1
s	1	0	1	0	∞

Bibliography

[1] R. Bergevin and M.D. Levine. Generic object recognition: Building coarse 3D descriptions from line drawings. *IEEE Trans. on PAMI*, 15(1):19–36, 1993.

[2] I. Biederman. Human image understanding: Recent experiments and a theory. *Computer Vision, Graphics, and Image Processing*, 32:29–73, 1985.

[3] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[4] T.O. Binford. Visual perception by computer. In *IEEE System Science and Cybernetics Conference*, Miami, Florida, December 1971.

[5] B. Bradshaw. Semantic based image retrieval: A probabilistic approach. In *ACM Multimedia*, pages 167–176, 2000.

[6] R.A. Brooks. Model-based three dimensional interpretations of two dimensional images. *IEEE Trans. on PAMI*, pages 140–150, March 1983.

[7] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In *Proc. on ECCV*, volume 2, pages 484–498, 1998.

[8] T.F. Cootes, G.V. Wheeler, K.N. Walker, and C.J. Taylor. Coupled-view active appearance models. In *Proc. on British Machine Vision Conference*, volume 1, pages 52–61, 2000.

[9] S. Dickinson, I. Biederman, A. Pentland, J.O. Eklundh, R. Bergevin, and R. Munck-Fairwood. The use of geons for generic 3D object recognition. In *IJCAI*, pages 1693–1699, 1993.

[10] S. Dickinson and D. Metaxas. Integrating qualitative and quantitative shape. *IJCV*, 13(3):1–20, 1994.

[11] S. Dickinson, D. Metaxas, and A. Pentland. The role of model-based segmentation in the recovery of volumetric parts from range data. *IEEE Trans. on PAMI*, 19(3):259–267, 1997.

[12] S. Dickinson, A. Pentland, and A. Rosenfeld. A representation for qualitative 3D object recognition integrating object-centered and viewer-centered models. In K. Leibovic, editor, *Vision: A Convergence of Disciplines*. Springer Verlag, New York, 1990.

[13] S. Dickinson, A. Pentland, and A. Rosenfeld. 3D shape recovery using distributed aspect matching. *IEEE Trans. on PAMI, Special Issue on Interpretation of 3D Scenes*, 14(2):147, 1992.

[14] S. Dickinson, A. Pentland, and A. Rosenfeld. From volumes to views: An approach to 3D object recognition. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(2):130–154, 1992.

[15] L.X. Fan and K.K. Sung. Model-based varying pose face detection and facial feature registration in video images. In *ACM Multimedia*, pages 295–302, 2000.

[16] F.P. Ferrie, J. Lagarde, and P. Whaite. Darboux frames, snakes and superquadrics: Geometry from the bottom up. *IEEE Trans. on PAMI*, 15(8):711–784, 1993.

[17] D. Gennery. Visual tracking of known three-dimensional objects. *IJCV*, 7(3), 1995.

[18] T. Heap and D. Hogg. Extending the point distribution model using polar coordinates. In *Computer Analysis of Images and Patterns*, pages 130–137, 1995.

[19] J.E. Hummel and I. Biederman. Dynamic binding in a neural net model for shape recognition. *Psychological Review*, 99:480–517, 1992.

[20] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *IJCV*, 5(2):195–212, 1990.

[21] M. Kass, A. Witkin, and D. Terzopolous. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988.

[22] Y. Lamdan, J. Schwartz, and H. Wolfson. On recognition of 3D objects from 2D images. In *Proc. on IEEE International Conference on Robotics and Automation*, pages 1407–1413, 1988.

[23] A. Leonardis, A. Jaklic, and F. Solina. Superquadrics for segmenting and modeling range data. *IEEE Trans. on PAMI*, 19(11):1289–1295, 1997.

[24] L. Liu and S. Sclaroff. Deformable shape detection and description via model-based region grouping. In *Proc. on CVPR*, volume 2, pages 21–27, 1999.

[25] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, 1985.

[26] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on PAMI*, 13(5):441–450, 1991.

[27] K.V. Mardia, W. Qian, D. Shah, and K.M. A. Desouza. Deformable template recognition of multiple occluded objects. *IEEE Trans. on PAMI*, 19(9):1035–1042, 1997.

[28] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. on PAMI*, 20(5):493–504, 1998.

[29] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Trans. on PAMI*, 15(6):580–591, 1993.

[30] R. Munck-Fairwood. Recognition of generic components using logic-program relations of image contours. *Image and Vision Computing*, 9(2):113–122, 1991.

[31] R. Munck-Fairwood and G. Barreau. A belief network for the recognition of 3D geometric primitives. In *6th International Conference on Image Analysis and Processing*, Como, Italy, 1991.

[32] T. O'Donnell, T. Boult, X. Fang, and A. Gupta. The extruded generalized cylinder: A deformable model for object recovery. In *Proc. on CVPR*, pages 174–181, 1994.

[33] A. Pentland. Perceptual organization and the representation of natural form. *AI*, 28:293–331, 1986.

[34] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Trans. on PAMI*, 13(7):715–729, 1991.

[35] M. Pilu and R.B. Fisher. Recognition of geons by parametric deformable contour models. In *Proc. on 4th ECCV, Volume I of LNCS*, pages 71–82, Berlin, 1996.

[36] N. Raja and A. Jain. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, 10(3):179–190, 1992.

[37] R. Ronfard. Region-based strategies for active contour models. *IJCV*, 13(2):229–251, 1994.

[38] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Trans. on PAMI*, 12(2):131–147, 1990.

[39] P. Sozou, T. Cootes, C. Taylor, and E. Di-Mauro. A non-linear generalisation of PDMs using polynomial regression. In *Proc. on British Machine Vision Conference*, volume 2, pages 397–406, York, UK, 1994.

[40] K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. on PAMI*, 20(1):39–51, 1998.

[41] J.P. Tarel and D.B. Cooper. The complex representation of algebraic curves and its simple exploitation for pose estimation and invariant recognition. *IEEE Trans. on PAMI*, 22(7):663–674, 2000.

[42] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Trans. on PAMI*, 13(7):703–714, 1991.

[43] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *AI*, 36:91–123, 1988.

[44] D. Thompson and J. Mundy. Model-directed object recognition on the connection machine. In *Proc. on DARPA Image Understanding Workshop*, pages 93–106, Los Angeles, CA, 1987.

[45] A.J. Vayda and A.C. Kak. A robot vision system for recognition of generic shaped objects. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 54(1):1–46, 1991.

[46] J. Wu, R. Rink, T. Caelli, and V. Gourishankar. Recovery of the 3D location and motion of a rigid object through camera image: An extended Kalman filter approach. *IJCV*, 3:373–394, 1989.

[47] Z. Yong. Preprocessing and feature tracking for reconstruction of 3D models. Master's thesis, National University of Singapore, 2000.

[48] M. Zerroug and R. Nevatia. Segmentation and recovery of shapes from a real intensity image. In *Proc. on 3rd ECCV (LNCS 800)*, pages 319–330, 1994.