Technical Report 963

Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-Parts

Gil J. Ettinger

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-parts

by

Gil J. Ettinger

Abstract

This thesis describes the development of a model-based vision system that exploits hierarchies of both object structure and object scale. The focus of the research is to use these hierarchies to achieve robust recognition based on effective organization and indexing schemes for model libraries. The goal of the system is to recognize parameterized instances of non-rigid model objects contained in a large knowledge base despite the presence of noise and occlusion. Robustness is achieved by developing a system that can recognize viewed objects that are scaled or mirror-image instances of the known models or that contain component sub-parts with different relative scaling, rotation, or translation than in the models. The approach taken in this thesis is to develop an object shape representation that incorporates a component sub-part hierarchy—to allow for efficient and correct indexing into an automatically generated model library as well as for relative parameterization among sub-parts, and a scale hierarchy—to allow for a general to specific recognition procedure. After analysis of the issues and inherent tradeoffs in the recognition process, a system is implemented using a representation based on significant contour curvature changes and a recognition engine based on geometric constraints of feature properties. Examples of the system's performance are given, followed by an analysis of the results. In conclusion, the system's benefits and limitations are presented.

Revised version of a thesis submitted to the Department of Electrical Engineering and Computer Science of the Massachusetts Institute of Technology on May 15, 1987 in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering and Computer Science.

This empty page was substituted for a blank page in the original document.

Acknowledgments

I would like to first and foremost express my gratitude to my adviser, Eric Grimson. His dedication, guidance, and many helpful comments are much appreciated. Thanks also to Mike Brady who, with his enthusiasm and ingenuity, sparked my interest in the field. These people and many others helped make the MIT AI Lab an enjoyable place to work and learn.

I thank my family for their love, support, and inspiration, and my friends for making my MIT experience a memorable one. In particular, I would like to thank one special friend, Linna, whose help, understanding, and devotion are dearly appreciated.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the System Development Foundation and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

This empty page was substituted for a blank page in the original document.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A main goal of robotics research is to impart in machines an ability to intelligently interact with their environments. This ability requires sensing the surroundings, accurately interpreting the signals from these sensors, and acting on these interpretations appropriately. Machine vision is concerned with the interpretation link of this process. While the name of the field implies that only optical sensors are used, it also applies to other forms of sensing modalities such as range or tactile data. Optical sensors, such as TV cameras, are often used since they are readily available and provide a rich and dense source of information about viewed scenes. The main aim of machine vision is to develop a symbolic description of a scene that may be effectively used in accomplishing a particular task such as inspection, path planning, target classification, or part identification. Many of these tasks require recognition of objects in the scene. This process is normally comprised of identifying objects in the environment and localizing them by computing their transformation from model coordinates to scene coordinates.

The recognition process can only be accomplished if the system has some prior knowledge of the type of objects it is viewing. The characterization of the knowledge depends on many factors such as the type of sensors used to extract information from the environment, the method used to process the available data, and the task to be performed. As a result, the knowledge base can consist of many properties of objects, a few of which might be shape, functionality, texture, reflectance, stiffness, or heat conductivity. A general recognition system would combine all of these properties with information from many input sources, such as light intensity, color, stereo, and motion, in order to reason about a viewed scene. Unfortunately, such a general system is not possible yet due to the existence of many unresolved questions encountered in solving

much smaller problems. One such problem is the shape recognition problem. This problem involves recognizing a 2D object based on the arrangement of its outline segments or a 3D object based on the arrangement of its visible surfaces or their projection onto a 2D image. Since shape is an intrinsic property of objects and one normally associates the definition of an object with its shape, it forms an appealing basis for an object recognition system. The shape recognition problem is motivated by the well-developed human ability to recognize objects when presented solely with their bounding contours. In addition, most recognition tasks are intended to identify aerial scenes or man-made objects, such as tools, machined parts, and communication symbols, where object shape is the primary descriptive medium as opposed to color, texture, or surface reflectance. As a result of these factors, shape recognition has been the subject of much recent research activity and some advancements in this field, especially in low level processing such as edge detection, have already been made.

Prior to any recognition attempts, one must instill the shape knowledge of the known objects into the system. The model objects must be internally represented in such a way that these representations can be matched to representations produced from a viewed scene. This knowledge acquisition process may be accomplished by manual training, by a separate learning module, or by the same module that is used for the actual recognition task. Manual training involves the off-line generation of the representations to be used by the system. These representations may originate from a CAD/CAM database or may be processed by hand to precisely extract the relevant features that the system can employ. Manual training leads to the most accurate models, but is a lengthy, time consuming process. A more effective knowledge acquisition procedure would consist of the system constructing the representations automatically and possibly interacting with users to correct mistaken labelings and to query questionable features. Such a system would be far more independent and easier to use and modify. Furthermore, knowledge acquisition by using the same recognition module would be advantageous to using a separate learning module since the same object would be guaranteed to have the same representation whether viewed as a model or as a component in an unknown scene. As well, any biases in the system would show up equally well in the models and in the scene representations. In addition, new models could be learned or old ones could be modified at the same time that the system is performing its recognitions tasks.

Based on the knowledge in its model library and the information given by its input sensors, a recognition system attempts to interpret a scene by searching through the

space of possible matches between model features and scene features. The interpretation process attempts to identify the objects in the scene that gave rise to the sensor signals. This problem is ill posed since not all the information about the viewed objects is preserved by the input sensors. The sensors provide intensity or depth values derived from the scene without knowledge of the objects' identity or configuration, or the scene factors giving rise to these values. These scene factors include presence of occlusion, attitude of light sources, attitude of viewer, reflectance of surfaces, and sensor noise and distortion. In order to limit the large number of possible interpretations of a scene and to provide accurate results, natural as well as assumed constraints are applied by the system. These constraints consist of such real world knowledge as continuity of surfaces, and such assumed knowledge as background color. The application of these constraints aids in the identification of the parameters that cannot be recovered from the input sensor data. The selection of appropriate constraints constitutes a large research effort in the field.

1.1 An approach to Model-Based Recognition

In designing a model-based recognition system, one must first decide on the desired domain and type of performance. Some of the issues of interest include correctness, efficiency, number and class of objects recognized, complexity of scenes, stability of interpretations, independence from outside intervention, and extensibility into larger and more complex domains. While it is desirable to optimize each of these goals, it will become apparent that they tend to conflict with each other and as a result the tradeoffs must be analyzed. While, in general, a large domain of applicability is desirable, it is often restricted by efficiency and stability requirements. These factors are further counteracted by the ability to recognize objects under diverse conditions without being overly noise sensitive. These issues are studied in detail in Chapter 3.

In general, the task of a recognition system consists of identifying and localizing all instances of all objects in a scene. This problem is studied in this thesis in the domain of two-dimensional objects where shape information is the primary source of data. For example, we would like to identify the object in the image in Figure 1.1[1] as an instance of the model in Figure 1.2 even though the two objects have different style arrows and

[1]This image as well as all other grey level images shown in this thesis have been halftoned for reproducibility.

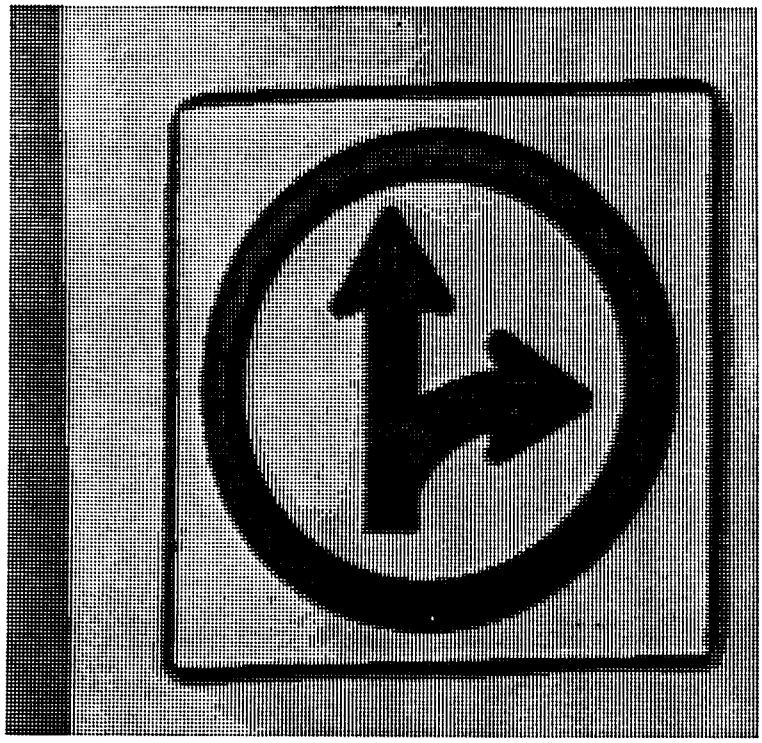


Figure 1.1: **Sample grey level scene presented to a vision system that is attempting to recognize traffic signs.**

different relationships between their components. This task may be performed by the system by breaking the model into its component sub-parts, as shown in Figure 1.3, and allowing some displacement between the sub-parts in the scene. Figure 1.4 shows the configuration of the sub-parts found in the scene and Figure 1.5 overlays the original model on the scene as specified by the identified sub-parts. The superimposition is derived by transforming the model object as specified by the average transformation of all the sub-parts and does not overlay exactly since the two objects are different.

The approach of this research is to design a robust recognition system with the following characteristics:

- Develop an object representation that is both hierarchical in scale (gross to fine features) and structure (whole object to component sub-parts). Features are defined as any identification primitives that capture and abstract some shape information of the object. Features are generally the lowest level primitives that compose the representation. Sub-parts consist of subsets of these features that partition the object into components. These hierarchies allow the system to stress the important features of the object and to reduce the large problem of recognizing the whole

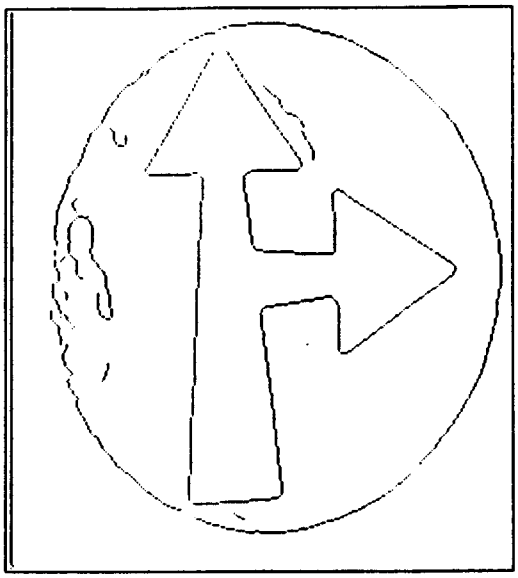


Figure 1.2: **One of the model objects known to the system, indicating no left turn. Bounding contours of the object are shown.**

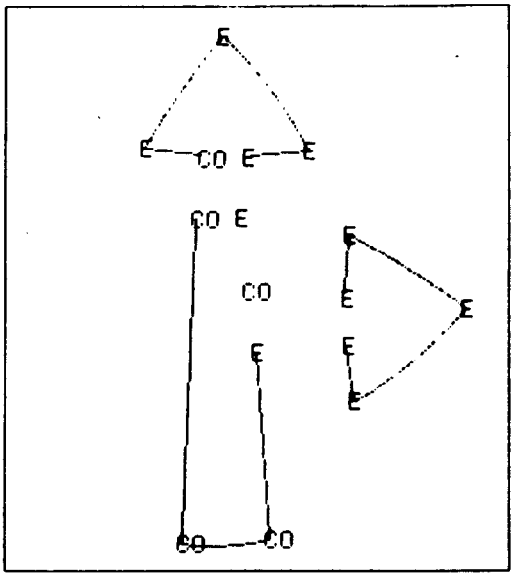


Figure 1.3: **Sub-parts of the *no left turn* sign. Letters indicate the coarse features of the sub-parts: E = End, CO = Corner.**

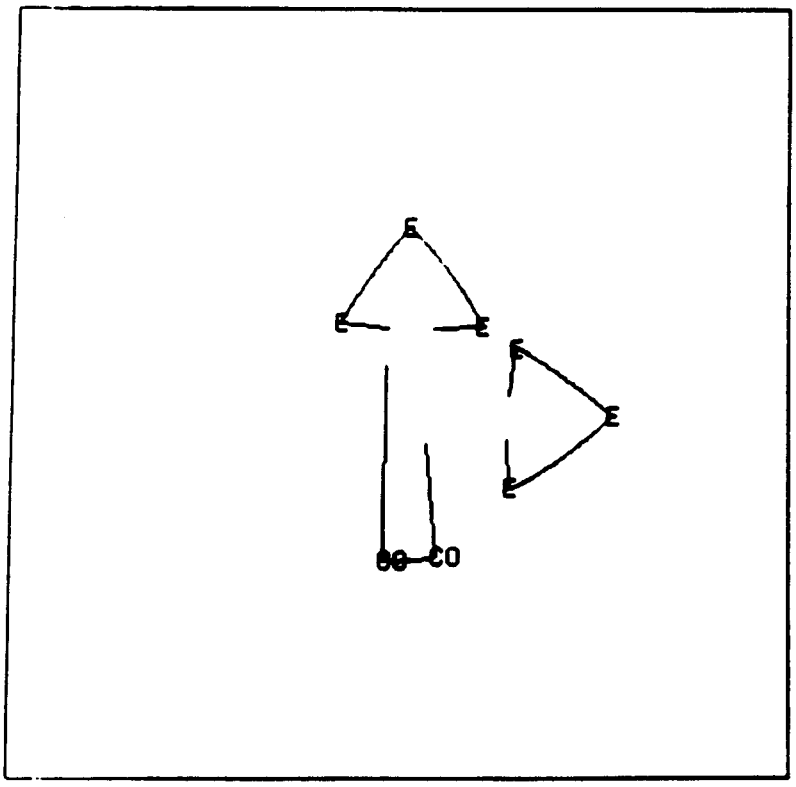


Figure 1.4: **Configuration of model sub-parts found in the scene.**

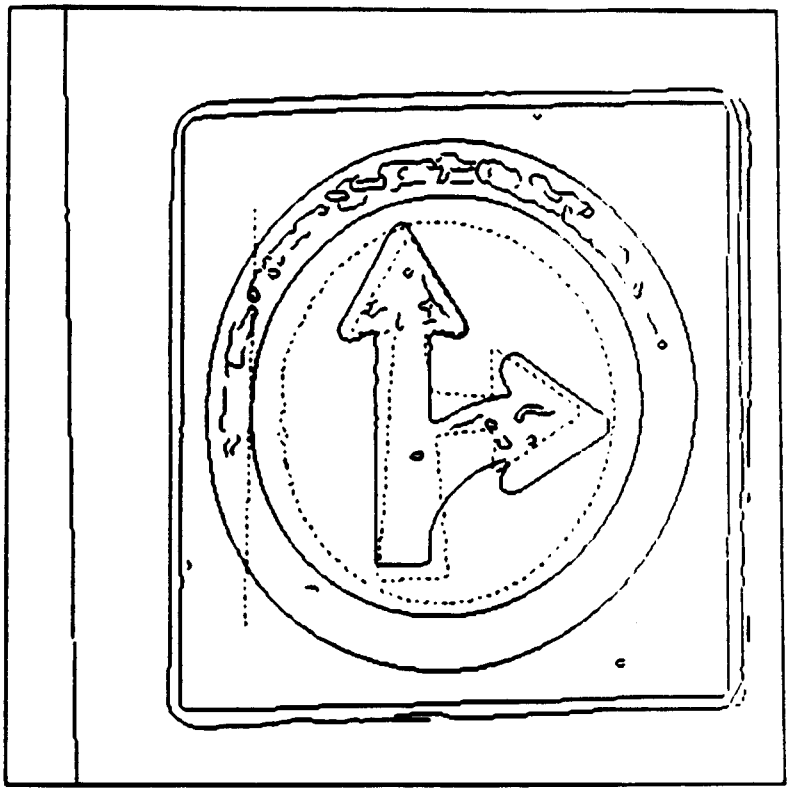


Figure 1.5: **Superimposition of the model on the scene.**

object into several much smaller problems of recognizing the object's sub-parts. The system thus demonstrates a recognition behavior of focusing on the correct interpretation while reducing the combinatorics of the process.

- Create model libraries automatically based on the hierarchical nature of the model representations. A recognition system should be able to identify any of a large set of objects that are stored in its library. The library should drive the recognition process in order to yield efficient retrieval of the correct model objects.

- Recognize all instances of all known model objects. Attempt to find the closest match for unknown objects by using the abstract features of the representation and by varying the sub-parts parameters. Close matches are defined by relatively high percentages of matched abstract features. This behavior allows the system to achieve an important recognition task of non-exact matching. For example, the system should be able to recognize hammers with handles longer than the one(s) in the database.

The novel contributions of such a system are the incorporation of *both* scale and structure hierarchies resulting in recognition of a wide class of objects under complex viewing configurations, and the development of an automatically created, sub-part indexed model library system to drive the recognition process.

In order to demonstrate these contributions, the following goals are set forth. These goals are accompanied by their measures of success that will be used to measure the degrees to which the goals were met.

1. **Efficient derivation of the scene interpretations.**
   The system should attempt to construct as small a search space as possible and then explore as little of the search space as possible without pruning any correct matches. It should narrow in on the final interpretations quickly and not spend much time considering wrong interpretations.

   In order to measure the efficiency of the system, we can confirm attempts to reduce the search space size and measure the ratio of the number of interpretations explored to the total number of interpretations possible.

2. **Efficient and accurate indexing of candidate models from a library of object models.**

The system should be able to automatically construct a model library that provides for fast and accurate indexing of candidate models based on feature cues provided by the image. The library should provide a means of focusing the search for interpretations of the scene and for drastically reducing recognition search times when compared with a non-sub-part, linear library search scheme.

Sample libraries should be constructed by the system and its hierarchical recognition behavior should be studied. The accuracy of the system with increased library size should not be affected and recognition time should increase slowly (in less than linear time) as the size of library grows. The behavior of the sub-part driven system should also be contrasted with the effectiveness of the system using only whole object descriptions.

3. **Correct identification and localization.**
   The system should identify all instances of all objects contained in its database and not form any extraneous interpretations. Close inexact interpretations should be given if they form the best matches. The orientation, translation, and scaling factor of the identified objects should be determined as accurately as possible.

   The accuracy of the system may be verified by checking that all the given interpretations of the scene are correct and all possible interpretations of the scene are given for various degrees of scene complexity. Localization should be checked by overlaying the interpretation on top of the scene.

4. **Tolerance for occlusion and noise.**
   An object should be recognized even when some of its components are not visible due to occlusion caused by itself or other objects, as long as enough features are still visible to accurately recognize it. The additional features introduced by overlapping objects should not interfere with the computation of the correct configuration. The system's performance should degrade gracefully with added noise introduced by sensor imprecision and distortions or by poor lighting conditions.

   Noise sensitivity may be measured by testing the system with various degrees of naturally and artificially perturbed scenes in order to introduce overlapping objects and shadows, reduce resolution, alter lighting conditions, and modify the sensor viewpoint.

5. **Stability in the presence of variations of global and sub-part parame-**

ters.

The system should be able to identify objects despite variations in scale of the whole object or of its sub-parts, or fluctuations in the spatial layout of its features or of its sub-parts. This capability allows the system to recognize non-rigid objects.

In order to measure the stability of the system, it should be tested with a wide array of complex scenes that incorporate mirror images, globally and locally scaled parts, and displaced and rotated sub-parts.

6. **Large domain of applicability.**

   The system should be able to recognize a wide class of objects with varying characteristics and complexity. Objects can consist of both polygonal and curved segments, can contain many concavities, and consist of various degrees of detailed features. The recognition system should also strive to achieve a high degree of independence from outside intervention by performing its tasks automatically.

   The types of objects and the complexity of scenes that the system can analyze should be determined. In addition, the circumstances under which the system fails should be outlined.

1.2 Upcoming Attractions

In order to analyze the work in this thesis, related work is presented in Chapter 2. Chapter 3 then develops a theoretical basis for model-based recognition systems and outlines many of the factors that affect the behavior of these systems. With these issues in mind, the following chapter describes the recognition system developed using the scale/structure hierarchy approach. Chapter 5 presents several examples of the performance of the system using a library of traffic sign objects. The performance is then analyzed in Chapter 6 in the context of the goals outlined in Section 1.1 and the issues described in Chapter 3.

Chapter 2

Overview of Recognition Systems

Numerous systems have been developed to perform model-based recognition. These systems differ in their domains, sensing modalities, representations, recognition engines, interfaces between the representations and the engines, and constraints that they impose on the scenes to be recognized. Many of the systems have been intended to perform specialized tasks and are not flexible enough to be extended into other domains. Other systems have been designed to be general purpose, but have been limited by such factors as: non-descriptive representations that do not capture enough knowledge about the objects in order to perform robust recognition, abstract representations whose complexity prevents them from being interpreted accurately and are therefore unstable, and inefficient recognition engines that suffer from unmanageable combinatorics. For composite overviews of model-based systems see [Binford 82] and [Besl 85].

While object shape is the standard basis for most recognition systems, the methods used to acquire it vary greatly. Bounding contours of an object can be located by finding image intensity gradients computed by an edge finder. Visible surfaces of an object can be determined from laser range or sonar data as well as by computing disparities from stereo or motion sequence images. In addition, active methods, such as tactile sensing, may be used to identify depth at certain points from which the surface structure can be interpolated.

The recognition procedure consists of two main steps: processing the input image in order to obtain the representational primitives that are then used by the recognition engine to match against similar primitives stored with or derived from the set of known models. This idea of separating the knowledge (the object features and primitives) from the control (the recognition engine) is a common one in the field of artificial

intelligence. The representational primitives should abstract some information about the shape in an organized and structured manner. The representation should convey as much knowledge as possible about the object in order to perform accurate and efficient recognition. Different representations and recognition engines have been developed in order to achieve varying degrees of flexibility, accuracy, and robustness. These systems provide distinct advantages and disadvantages and it remains to be seen if they can be combined in ways that achieve a more effective system than they each offer independently. These representations and recognition engines are briefly described followed by a description of some recognition systems that attempt to incorporate an object library approach.

2.1 Current Types of Representations

A common representation for object shape proposed in the machine vision literature is the *generalized cylinder* (or generalized cone) [Hollerbach 75] [Nevatia 77] [Marr 78] [Brooks 81]. This representation is a volumetric description consisting of an axis of symmetry and a sweeping rule that defines a specific volume when run along the axis. Generalized cylinders can be combined to form a large class of shapes, but are best suited for elongated symmetric shapes where the axis and sweeping rule are easily identifiable. A recognition system based on generalized cylinders attempts to identify the symmetry axis and then define some simple properties of the sweeping rule. While generalized cylinders have proven useful for describing how objects are constructed, the problem of extracting from the image the generalized cylinders that comprise the object remains ambiguous. In addition, the determination of cylinders requires global information that is not available under all viewing conditions. Research is progressing, though, in reducing these drawbacks and exploiting the compact and descriptive qualities of generalized cylinders [Ponce 87(a)] [Ponce 87(b)].

Another frequently-used representation is the description of the surfaces of an object using *polyhedral approximations* [Faugeras 84] [Bhanu 84(a)] [Bhanu 84(b)] [Grimson 84] [Hebert 84] [Ayache 86]. The analogous representation in the 2D domain is the use of polygonal approximations to describe the object contour. Since few parameters are required to define polyhedra, they form a compact representation. If the connectivity of the polyhedral primitives is used along with the polyhedral specification, a good shape description is obtained. This shape description can be matched to model descriptions

that were obtained in the same manner, or to CAD/CAM models that contain surface and edge information. A problem with this representation is that the polyhedral approximation for non-polyhedral surfaces is highly sensitive to noise and object attitude and as a result the representation for curved surfaces is not stable. Partially occluded surfaces also present a problem since small surface patches must be matched against large model surfaces, creating a large search space.

An alternative to using polyhedral approximation is the use of *actual surface patches.* This representation has been applied in the 2D domain with the use of contour segments [Knoll 85] [Turney 85], and in the 3D domain with the use of edges and surfaces [Tomita 85]. The contour segments are described as sequences of points and are matched on a point-by-point basis. As a result of this matching procedure, the representation can lead to inefficient recognition. This representation may be adequate for small tasks, but the primitives do not abstract much shape information and are not well suited for general recognition.

Another approach to representation is the use of *predefined distinctive features* [Bolles 82] [Bolles 86] [Lowe 85]. This representation consists of the spatial relationships of semantically rich features that are distinctive in the context of the object. With the appropriate selection of reliably identifiable features and the incorporation of their relationships, accurate and efficient recognition is feasible. In the Local Feature Focus method, by Bolles and Cain, the recognition process attempts to locate sets of distinguishing features, such as holes and corners, in an ordered manner that is determined from a precompilation of the knowledge of the model. The selection of these distinguishing features, though, can be difficult since it is dependent on the recognition domain.

A descriptive global representation is the *extended Gaussian image* (EGI) [Horn 84] [Brou 84] [Little 85]. This representation consists of vectors on the Gaussian sphere whose orientation corresponds to the orientation of a face of the polyhedron and whose magnitude corresponds to the area of that face. The EGI is a descriptive representation for 3D objects, but is only unique for convex objects. This representation separates the problem of orientation determination from translation determination and is therefore useful for coarse grain matching or pruning of the search space. It is limited, though, in its applicability to recognition since it is removed from much of the shape knowledge of the objects and is limited by occlusion, sensitivity to orientation variations, and domain of application.

A representation that has been used to describe natural form is the *superquadric* [Barr 81] [Barr 84] [Pentland 85] [Bajcsy 87]. This representation consists of parameterized solids that may be deformed in various ways and combined by boolean and fractal operators. The representation has proved to be well suited for building complex scenes, especially of natural objects. Its complexity, though, prevents it from being used in a recognition system in a straightforward manner since the recovery of the large number of parameters used in describing the primitives, and the segmentation of the scene into these primitives are difficult tasks.

2.2 Current Types of Recognition Engines

One of the earliest recognition engines was the *Hough transform* [Ballard 82] [Turney 85]. Various versions of the Hough transform exist, but they all perform some type of histogramming operation to find probable configurations. A common form of its use, termed the extended Hough transform, is to match all possible model and scene features to each other and for each match compute the transformation bringing the model into correspondence with the proposed object in the scene. The transformations are mapped onto a histogram where peaks suggest actual transformations that should be verified. The Hough transform thus provides a coarse filter on the search space and is best used in directing the search or for speeding up a more exact recognition scheme.

A popular form of recognition engine is the *hypothesis-prediction-verification paradigm* [Brooks 81] [Hebert 84] [Lowe 85] [Tomita 85] [Ayache 86]. This control mechanism incorporates bottom-up (scene-driven) and top-down (model-driven) approaches to take advantage of the system's knowledge of the objects. Based on features found in the scene, the system hypothesizes possible configurations in order to avoid searching the whole space of possible feature matches. Based on the features of the model, the system is then able to predict locations of other features in the scene. The system can precompute a search tree from knowledge of the model features in order to drive the hypothesis step in an efficient manner. The system proceeds by verifying these predictions and modifying its hypotheses accordingly. These systems work well, but are susceptible to errors and missed interpretations since they make assumptions during the hypothesis phase that may be incorrect or misleading. Since these methods do not explore the whole search space, good interpretations may not be found if the hypotheses are not based on descriptive features.

A novel recognition engine is the *constrained search scheme* [Grimson 84] [Grimson 85]. This paradigm considers the whole space matching scene features to model features, but prunes many configurations in the search space early in the search process to achieve favorable combinatorics. The pruning is performed by using simple geometric constraints between pairs of features to remove invalid matching configurations from the search space. Many of the matches are pruned after analyzing the match between very few features. Only the well matched configurations are expanded to match all the features. The constraints compare the following properties between pairs of model primitives and their matching pairs of scene primitives: distance, orientation difference, and direction vectors. The scheme has been shown to work well with sparse input data points, assigning points in the scene to lie on model faces in a consistent manner. It depends, though, on the selection of meaningful input data points that survey the complete shape of the object. The contribution of this method has been the demonstration that simple constraints used in concert lead to very effective pruning of the search space, even in the presence of noise [Grimson 86].

Another approach to recognition is the *relaxation scheme* [Bhanu 84(a)] [Bhanu 84(b)]. In this method, an initial assignment of model features to scene features is made and these assignments are modified as the validity of the overall match is propagated along the set of features. The process iterates and the interpretation is refined until an accurate configuration is found. This recognition engine depends on effective propagation along the matching nodes, good initial assignments, and convexity of the configuration space and the domain. The convexity requirements ensure the system does not identify local maxima/minima as global and are usually difficult requirements to satisfy.

Rule-based production systems have also been used as recognition engines [Nagao 79] [McKeown 85]. These systems incorporate extensive context knowledge of certain types of scenes in their rule base instead of using geometric knowledge of the models. The rules are then used to evaluate hypotheses and ensure their consistency. While the incorporation of specific domain knowledge aids in the derivation of viable interpretations, these types of systems are limited by their control structure and efficiency. The control mechanism of production systems is usually very weak and does not lend itself well to more powerful processes that exploit shape information to constrain the interpretation process. The result of this weakness is that these systems are often very inefficient. Production systems are also difficult to develop since the construction of

the knowledge base is a lengthy process. Another drawback is the dependence on an initial segmentation phase that decomposes the scene into regions that are used as the primitives to be input into the system. Segmentation is often difficult to perform at a low level since it is greatly affected by noise and viewing conditions. As a result, errors generated by the segmentation step can be propagated all the way to final interpretation. While production systems may be suited for situations where shape knowledge is limited, their benefits may be best realized by combining them with more powerful systems that benefit from the incorporation of contextual knowledge, such as ACRONYM [Brooks 81].

2.3 Library-Based Recognition Systems

Most of the developed recognition systems are designed to identify a single model object at a time. The issue of effectively recognizing objects from a large library of models has been generally neglected due to its complexity. The problems encountered in this task are how to organize the library and how to index into it in order to avoid a linear search of all the model objects. Despite these open-ended problems, many researchers point out the need for developing well structured object libraries to be used for quick identification of model instances as well as of related objects. Without this capability a general recognition system cannot be achieved. As the ability of recognition systems to identify single models in a scene improves, the problems of library-based recognition must be explored. Some systems that have addressed this issue are described below.

2.3.1 ACRONYM

One of the first systems to explore recognition of multiple objects is the ACRONYM system [Brooks 81] [Brooks 83]. This system models objects hierarchically in object classes by using relaxed parameters at the higher levels and tighter parameters at the lower levels. A constraint manipulation system is then used to constrain the parameters based on the relation of the features found in the scene and the features of the models. As the parameters are refined the identity of the object is established by progressing down the object class tree. The system performs recognition of 3D objects from single images by using different representations for the models and scenes. The models are represented as collections of generalized cylinders while the scene features consist of projections of the cylinders, namely ribbons and ellipses. While this representation

adapts well to the complex recognition of 3D objects from 2D images, it is global in nature (does not extend well to occlusion), and applies to a limited domain of objects (elongated objects with identifiable axes).

While this innovative approach benefits by its generality and flexibility, it does not display an effective library indexing scheme. ACRONYM attempts to match each object in the library in a linear fashion and is therefore not efficient. In addition, the library must be manually constructed—the criteria used to organize the object classes are externally supplied. In a general recognition system these criteria would have to be known internally.

2.3.2 Schwartz & Sharir

Schwartz and Sharir developed a 2D model-based recognition system [Schwartz 85] [Kalvin 86] that is capable of recognizing a model from a large library of objects (up to 100 in one experiment). The objects are represented by their *footprints*, which are derived from the relationships of the lengths and relative orientations of the segments of the polygonal approximation to the bounding contour. These footprints are then hashed into library slots that point to all the objects that consist of the contained footprints. As a result, the system can quickly derive possible candidate models by hashing footprints processed from the scene. These candidates can then be verified by comparing scene edges with the proposed model edges.

While this system displays impressive library indexing performance, its representation does not symbolically capture much of the shape information of the models. The domain of the representation is limited by a scale invariance restriction and the inability to accurately model objects with concavities since concave corners are used to segment the bounding contour of the scene into its component objects.

2.3.3 Turney, Mudge & Volz

Another system that attempts to incorporate libraries into the recognition process is the system developed by Turney, Mudge, and Volz [Turney 85]. This system automatically builds the library by finding salient features that uniquely identify each object relative to all the other models in the library. The identification of these features allows the system to easily recognize objects if these salient features are found in the scene, even if a large portion of it is occluded. The features used by the system are actual bounding

contour segments so the matching process consists of a pixel by pixel Hough transform match of all possible scene contour segments to model segments. Any model segments matches increase the likelihood of an instance of that model existing in the scene. Since the salient features are weighted much more than the others, their identification leads to (and is required for) the identification of the whole object, while features common to many objects do not affect the match likelihood.

Clearly, this representation does not abstract much of the shape information of an object. As a result, it does not extend well to recognition of scaled objects, mirror images, or sub-part variations. Its lack of compactness results in a combinatorial explosion for complex objects and large libraries. In addition, while the salient features are intended to allow recognition despite occlusion, recognition fails if the salient features are occluded and the rest of the object is not. While this system may perform well for such tasks as anomaly detection where many of the scene parameters are known and the use of the actual bounding contour is preferable, it is not well suited for general recognition tasks. The use of a scale hierarchy is suggested to reduce the system's noise sensitivity and running time, and will likely aid its performance.

2.3.4 Knoll & Jain

Knoll and Jain developed the Feature Indexed Hypotheses method to perform model library indexing [Knoll 85]. Instead of using unique features, features common to several objects in the library are used. The system constructs the library automatically by finding all objects that contain each feature. Assuming that the cost of matching a feature is equal to the cost of verifying a resulting hypothesis, Knoll and Jain show that the ideal number of matches per feature in the library is proportional to the square root of the total number of objects. The recognition process attempts to match features ordered by their commonality relative to the library and by their location relative to other features on their contained objects. Model features matched in the scene lead to hypotheses that may then be verified by checking if locations of predicted model edges are supported by data in the scene.

The representation used by this scheme is again actual bounding contour segments so it too does not extend well to the recognition of globally and locally parameterized objects. While the recognition costs made for this system may be appropriate, they depend on this representation and will not hold in general since the cost of generating good hypotheses is usually much greater than verifying them. The hypothesis generation

step is also purely dependent on the objects in the library and not the features in the scene, a process that may result in much needless search for large libraries. Its library organization is appealing, though, since overlapping object classes are allowed and unique features are not required.

Chapter 3

Recognizing the Issues

Since the recognition process consists of searching for the best interpretation of a scene, much of the work in this area centers around optimizing this search. This work often involves extracting and structuring appropriate knowledge about the model objects in order to effectively limit the search. As has been observed in Chapter 2 many different techniques are possible to both reduce the size of the search space and the fraction of the search space that is actually explored. These methods include top-down and bottom-up approaches, preprocessors to determine specific contexts of scenes, statistical analysis, exploitation of high-level and low-level descriptions, and use of local and global representations. The selection and combination of these methods can drastically affect the performance in terms of correctness, accuracy, efficiency, complexity and stability of interpretations, and applicable domain. This chapter describes how the performance is affected by issues that arise in the development of a recognition system using a scale/structure hierarchical approach to construct large object libraries.

3.1 Recognition System Capabilities

The type of behavior desired of a recognition system is dependent on the application. Specific needs often dictate many of the choices that may be made during the system's development. For example, if the system is to be used for inspection, it probably knows the identity of the object it is viewing and needs to perform very exact matching of the detected edges, while for categorization purposes the system may only need to identify rough features. In order to realize the wide range of possible capabilities, we can explore various questions that may be asked of the system:

- Does the system recognize:
  - a (single) particular object in the scene?
  - if a particular object is not in the scene?
  - all instances of all objects in the scene?
  - related objects as members of the same class?
- Is the aim of recognition:
  - to perform localization of objects only? (e.g. obstacle avoidance)
  - to perform identification of objects only? (e.g. if you see a bear, run)
  - to perform both localization and identification?
- If any incorrect interpretations are generated, is their nature:
  - to not find existing model instances in the scene?
  - to find extraneous model instances in the scene?
  - to misidentify model instances?
  - to inaccurately localize correctly identified model instances?

The most beneficial system would be able to identify and localize large classes of objects and be able to realize quickly when an object does not exist in a scene. The introduction of some errors is usually unavoidable since a system based solely on object shape is bound to make mistakes for certain conditions. Even the much richer human visual system can be fooled into making wrong interpretations. Recognition systems, though, can be designed to be conservative, at one extreme, and not introduce extraneous interpretations at the cost of missing certain interpretations or, at the opposite extreme, ensure that all objects in the scene are identified at the cost of deriving some possibly erroneous configurations.

One aspect of the recognition capability is the number of degrees of freedom allowed in the scenes. This dimensionality of the recognition task can be divided into three categories: 2D objects from single images, 3D objects from depth maps, or 3D objects from single images. Many systems are initially implemented to perform recognition of 2D objects and then extended to recognition of 3D objects from depth data. The

extension requires recovery of more parameters since 2D objects usually have three degrees of freedom while 3D objects have up to six degrees of freedom. The depth map, though, provides a rich source of added constraint to aid in the recovery of the additional parameters. Some systems [Brooks 81] [Lowe 85] [Goad 86] [Huttenlocher 87] have been designed to recognize 3D objects from single images and have demonstrated promising results. Even though 2D images do not usually provide enough constraints to recover the many parameters contained in the scenes, these systems have been shown to work well in some domains.

An important capability criterion is the class of objects that may be recognized. Many variables are used in defining the acceptable object class for a recognition system. One issue is flat versus curved regions. Representations may favor straight edges (or planar surfaces) and either ignore curved edges (or curved surfaces) or, worse, exhibit degraded performance with their introduction. Another variable is convex versus concave objects. Some schemes can only represent convex objects and as a result can use concavities to segment the scene into its component objects. A third point is the generality of the object features. Some systems require objects to contain certain predefined features while others impose complexity restrictions and limitations on the number of features objects may have. It is often the case that representations can be optimized if the class of objects is restricted, while general representations may not be as accurate or as efficient. In addition to the type of objects, the number of objects that may be recognized also defines the system's capabilities. While it is often desirable to recognize many objects of complex nature, the task is often simplified by using few complex models or many simple ones.

Another factor that defines the domain of a recognition system is tolerance for variations in the image introduced by the sensing modality. A system should be robust in the presence of noise introduced by the sensing device. Camera resolution and lens distortion reduce the accuracy of image features. Lighting conditions also introduce shadows that may reduce the effectiveness of the system. As a result, images of the same objects may appear differently and reveal different features when viewed at a different orientation or a different scale. Most visual recognition systems do not attempt to filter out the noise since they use higher level information to abstract major features and employ enough flexibility in matching to allow some variability in the characteristics of the image. High frequency noise filtering and various methods of image enhancements may be used to improve the image quality, but any recognition system should be able

to tolerate some inaccuracies in the model object specification.

Variability in the configuration of the objects themselves should also be tolerated. Such variations may arise from interference from other objects or modifications to the actual object of interest. Occlusion, whether caused by other objects or by parts of the same object, introduce new features and remove features for which the system may be searching. As a result, systems that handle occlusion must extract local features that remain unchanged when part of the object is hidden from view, and be able to recognize the object correctly when enough (but not all) features are visible to distinguish it from all other possible configurations of objects. A system should not be led astray by the introduction of other model features or features that are created at the point of overlap of two objects.

A robust system should be able to integrate variability in the parameters of an individual object into the recognition process. Such variability may arise from scaling the whole object or parts of the object, mirror images of 2D objects (corresponding to flipping the object to its back side), or relative parameters between parts of the object such as rotation of the blades of a pair of scissors. Many objects contain some parameterization that allows some degrees of freedom while maintaining some of the same basic shape cues. The type and amount of such parameterization form another restriction on the domain of applicability.

While the work in this thesis is directed only towards recognition of 2D objects, or 3D objects in stable configurations, it does aim to achieve many of the robustness criteria outlined above, as described by the goals detailed in Section 1.1. These criteria include recognition of a large number of objects in the presence of noise and occlusion where the models are non-rigid and the viewed objects may be inexact instances of models. Extensions to 3D recognition are considered in Section 6.6.

3.2 Recognition Efficiency

A common goal for recognition systems is to achieve correct interpretations as quickly as possible. Efficiency, thus, plays an important role in the design of these systems. Since the problem is structured in terms of searching for the best matches of model objects to viewed objects in the scene, the aim is to explore as few configurations as possible without sacrificing the desired performance of finding correct interpretations. This goal may be accomplished by, first, structuring the problem in such a way as to

minimize the size of the search space matching all combinations of scene features to all combinations of model features and, second, minimizing the fraction of the search space to be explored. These issues of matching individual models to scene features are analyzed in this section. The efficiency of library-directed model retrieval is addressed in Section 3.3.

3.2.1 Reducing the Size of the Search Space

Since the recognition time is roughly proportional to the size of the search space (using the same recognition scheme, complex objects are harder to recognize than simpler ones), reducing the size of the search space will improve the efficiency of the system. A simple method to reduce the size of the search space consists of using a representation that enforces a one-to-one matching of model features to scene features. For example, a representation consisting of knot points of the bounding contour of the object will match each model feature to exactly one scene feature and vice versa. This matching behavior allows the recognition system to be guided by either the scene data (iteratively matching model features to each scene feature) or the model data (iteratively matching scene features to each model feature). The system can thus select the method that yields the smaller search space. A many-to-one matching scheme, such as one matching scene edge segments to model edges, may match many scene features to one model feature. As a result, the many-to-one scheme results in larger search spaces that must be guided by the scene data. The scene-guided matching restriction can be costly since recognition tasks usually involve more scene features than model features. As shown below, the ability to guide recognition by using the model features instead of the scene features results in a significantly reduced search space. Given α model features and β scene features, the size of the search space is:

- for one-to-one matching: $\begin{cases} \frac{\beta!}{(\beta-\alpha)!} & \text{using model-guided search, } \alpha < \beta \\ \frac{\alpha!}{(\alpha-\beta)!} & \text{using scene-guided search, } \alpha > \beta \end{cases}$
- for many-to-one matching: α^{β} using scene-guided search.

It is apparent that one-to-one matching provides a significantly smaller search space than the many-to-one matching scheme for $\alpha < \beta$. As an example, given 10 model features and 25 scene features, a many-to-one (scene-guided) matching scheme would consist of 10^{25} configurations while a one-to-one (model-guided) matching system would

only need to consider 10^{13} configurations. Since α is usually less than β, the efficiency analysis below assumes model-guided search.

The search space size can also be reduced by dividing the problem of recognizing a single object into several much smaller problems of recognizing the object's component sub-parts. Since the size of a search space basically grows exponentially with the number of features, reducing the number of features reduces the exponential growth of each sub-part's search space while only linearly increasing the number of those sub-spaces. This scheme can achieve favorable results as long as the recognition of the sub-parts can easily be combined to reach an interpretation for the whole object. This check for consistency among the components can usually be done very quickly by checking the relative scaling, rotation, and translation of the sub-parts. The size of the search space for a one-to-one matching scheme of α model features and β scene features (assuming $\alpha < \beta$) using N object sub-parts (assuming uniform distribution of the model features among the N sub-parts) is:

$$N\frac{\beta!}{(\beta-\frac{\alpha}{N})!}.$$

In order to get an idea of the reduction power of sub-parts, increasing the number of sub-parts from 1 to 2 for the 10-model-feature/25-scene-feature example reduces the size of the search space from 10^{13} to 10^{7} configurations.

The most obvious method for reducing the size of the search space is to use a compact representation for the object models in order to reduce the number of representational primitives needed to describe an object. This goal implies that individual features should be complex and capture a significant amount of the shape information of the object. As a result, the features are difficult to derive accurately and efficiently. On the other hand, if the representation consists of a large number of simple features, each of which does not capture much shape information, many features need to be matched in order to derive scene interpretations. This issue defines a major tradeoff in recognition systems and is further analyzed in Section 3.6.2.

3.2.2 Reducing the Fraction of Search Space Explored

Once the size of the search space has been reduced, a recognition system should attempt to minimize the number of configurations in the search space that it explores. One method for reducing the fraction of the search space to be explored is to use a descriptive representation that only allows primitives of similar semantics to be matched. By using

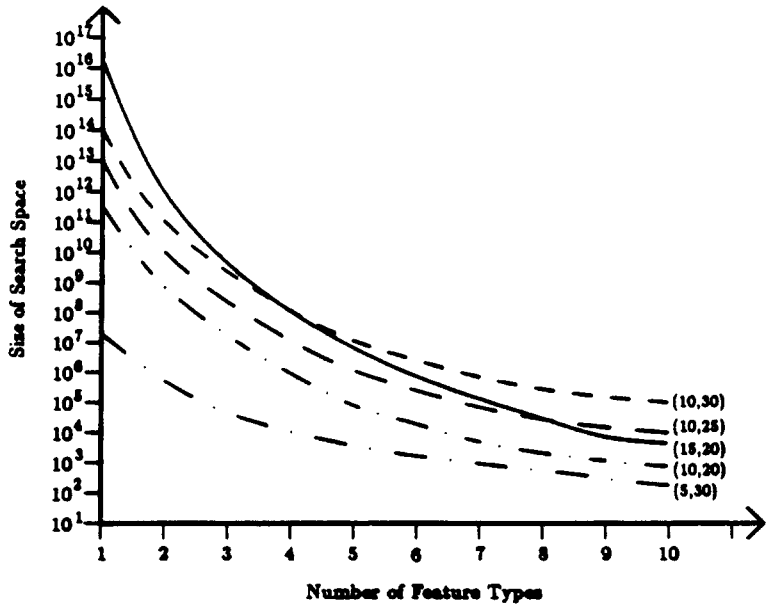


Figure 3.1: **The amount of the search space to be explored as a function of the number of feature types. Several plots are shown for different values of α (number of model features) and β (number of scene features); these values are shown as the pair (α, β).**

different feature types, the system avoids matching all possible combinations of model features to scene features. The benefits of this method can be reaped only if the type of feature can be determined reliably and if ambiguous feature classifications can still be matched. The cost of this scheme is the additional processing that must be performed in order to classify the model and scene features. This processing, though, will generally only grow linearly with the number of features. The combinatoric benefit of feature classification will then manifest itself in the following formula showing the amount of the search space to be explored given α model features, β scene features, and λ feature types (assuming uniform distribution of features among the λ feature types):

$$\left(\frac{(\beta/\lambda)!}{((\beta-\alpha)/\lambda)!}\right)^{\lambda}.$$

An example of this combinatoric advantage is shown in Figure 3.1, which shows the reduction of the number of nodes to be explored with an increasing number of feature types. As an example, only 0.00003% of the search space for a 10-model-feature/25-scene-feature case need be explored when using 5 feature types instead of 1.

Another important way to reduce the number of nodes that are explored is to use a coarse to fine recognition scheme, or a scale hierarchy. The idea is to take advantage of

the coarse shape cues first to derive rough interpretations and then use the finer features to refine them. An object should be described by its coarsest primitives, the ones that abstract the main shape features, at the top level, and by more detailed primitives at lower levels. A recognition scheme can then consider matching only the abstract features of the model to the abstract features of the scene in order to derive likely, but possibly inexact, configurations. The finer features can then come into play in order to differentiate between those configurations and better localize them. The benefit of this scheme is that all the features do not have to be considered at once. Thus, by the time the fine features are being matched vast portions of the search space have already been pruned. This approach works well if the coarse primitives of the representation abstract enough information to remove the burden of initial matching of the detailed features, without abstracting too much information which would lead to spurious and ambiguous configurations.

Another approach aimed at exploring only a small portion of the search space is to hypothesize the branches of the search space that contain the best configurations and then only explore those nodes. This task is performed by running a fast preprocessor or by selecting certain salient features and only exploring configurations where those features are appropriately matched. A powerful and widely used preprocessor is the Hough transform (see Section 2.2) that basically votes for likely paths to take in the search space. Since the preprocessor outputs preconceived notions of the feature assignments, the behavior of the system is to verify those predictions while exploring the search space.

3.3 Effective Libraries

One of the main aims of this research is to study the problem of structuring a large database of model objects. A recognition system should incorporate a vast store of knowledge that it can use to identify and differentiate between a large set of objects. The knowledge base usually consists of the representations of the various objects and the relationships between them. By incorporating different types of objects in the library, various types of recognition performance can be achieved. By entering widely varying model objects in the library, where each model may define a separate class, and by increasing the variation tolerance levels of matched features, the system can achieve an object classification behavior. The recognition system can also display an object

discrimination behavior by differentiating between similar objects entered in the library and by enforcing tight constraints between measured properties of features.

The most difficult problem that arises with library usage is how to retrieve the best matched models without exploring all the possible object matches. This task requires a library design that allows quick indexing of the likely model candidates based on the features observable in the scene. The goal is to achieve correct recognition that grows *sub-linearly* with the size of the library. Two measures are given to assess this goal:

1. The fraction of the composite search space that is explored should decrease as more objects are added to the library. With X objects in the library, the size of the composite search space is approximately $X\Psi$, where Ψ is the average number of explored nodes of the search space of individual objects. The number of nodes that is explored by the system should be smaller than $X\Psi$ since we do not want to attempt to match each object in a linear fashion. Some benefit can be derived by taking advantage of commonality among models to avoid searching similar portions of the search space. But, in addition, we would like the indexing keys of the model database to contain enough shape information to uniquely define subsets of likely model candidates when given observed scene features.

2. The number of configurations explored should grow sub-linearly with the number of objects in the library. This behavior is easily measurable and will serve to further confirm the efficiency of the library system.

Once a preferred library structure is determined, the problem of library creation must be addressed. This issue of learning the model descriptions and relating them to previous knowledge is a complex problem that usually requires some form of tutoring from the user. Some proposals have been made to relate whole model representations to each other in order to build object classes. Connell's system [Connell 85], for example, uses analogies between instances of similar objects to infer the important shape cues common to whole sets of objects and the detailed shape cues that differentiate between related models. A tutor tells the system if objects are related so that the analogy processing can take place. A tutor is needed since he understands the functionality of the objects, which is the usual criterion for object similarity, but functionality is generally difficult to represent directly to the system. After being presented with enough examples from the user, the system can then construct internal canonical representations based on the shape representations of the individual objects and their interrelationships.

Shapiro and Haralick [Shapiro 82] propose clustering objects into classes based on a metric measuring the similarity of the object feature relations and then using the one object closest to the average metric value to represent the whole class. While this approach to canonical representations requires less user interaction, it is unclear as to which types of representations will work well with this type of global metric measure. Their approach does not abstract important shape information and as a result the object classes might not necessarily be meaningful.

The approach taken in this work is to construct the model library automatically by using shape information and as a result avoid the thorny issue of object functionality which is beyond the scope of this thesis. Since the use of global metrics to relate objects is not a robust method, we relate objects by their common sub-parts. Thus object classes are defined by the degree of sharing of their component sub-parts, leading to possibly overlapping object classes. Multiple object classes are useful for converging on the best object match—the iterative identification of shared components among the remaining model candidates continually reduces the candidate list. The use of sub-parts to index into the model library thus leads to an appealing recognition behavior, as outlined in Section 3.4.

The efficiency criteria for libraries outlined above have not been studied in the past in order to show efficient library-driven recognition. In addition, most past library systems, as described in Section 2.3, do not achieve effective recognition of complex objects. They are only shown to work with very simple objects and it is not known if they can be extended to a more complex domain. A goal of this work is therefore to start diminishing these deficiencies and to develop a theory of model library organization.

3.4 Need for Sub-parts

Section 3.2.1 already demonstrated the combinatoric benefits of using sub-parts by showing how their use leads to a reduction in the size of the search space. In addition to this efficiency advantage, sub-parts also lead to two other benefits: indexing keys for the model library, and parameterization of sub-parts. Sub-parts are defined as subsets of object features that partition the object into its components. Many different partitions of the object into sub-parts are possible, but for any one partition the sub-parts are assumed to be non-overlapping. One example of a sub-part partition is given in Chapter 1 where Figure 1.3 shows the sub-parts found for the model in Figure

1.2. Other examples are shown in Section 4.2 and Appendix A. In general only one partition is used to represent the object, but several could be used as well. In addition, each sub-part could be further partitioned into smaller sub-sub-parts. This structure hierarchy can contain an arbitrary number of levels, but is limited by the size of the smallest component. The smallest sub-part can contain just one feature, which cannot be readily broken down into smaller parts since features are generally the lowest level primitives used.

The reason sub-parts are used to index into the model library is that individual features are too small to index a reasonable set of candidate models, while whole objects are too large. Individual features may have many matches with many models and thus do not converge on a set of likely model matches. Using whole objects to index into the library is equivalent to performing a linear search through the library, which is exactly what we are trying to avoid. As a result we would like to use subsets of object features as keys to the objects in the database since they are small enough to match easily, but are large enough to limit the search to a small set of models that contain those sub-parts. Since sub-parts can be shared by several objects, the number of sub-parts generally grows sub-linearly with the number of objects. Thus, common sub-parts further enhance the desired sub-linear recognition behavior relative to the number of objects in the library. If too many objects share the same sub-part, however, the discrimination ability of the sub-part is hindered since the recognition of the sub-part does not significantly limit the number of objects to be potentially matched. To get an idea of the desired degree of sub-part sharing we can extend Knoll and Jain's results for the ideal number of object matches per feature. Under their assumption that the cost of hypothesizing an object match is equal to the cost of verifying that hypothesis, total recognition cost is minimized if the degree of feature sharing is proportional to the square root of the number of objects in the library [Knoll 85]. Although Knoll and Jain do not use a structure hierarchy, and hypothesis generation is generally more expensive than verification, their measure supports our intuitive notion of sharing sub-parts to improve performance.

Even though sub-parts are defined as any subsets of object features, we gain the benefit of local parameterization if the sub-parts are also semantically meaningful. If the sub-parts are separated where variations may occur, the system can explicitly allow for these variations rather than trying to account for them by allowing increased noise fluctuations. These points of variations can be identified by the system by finding such

features as concave corners, as described below. Thus the system can readily identify objects with rotational or translational articulations as well as objects with relative scaling variability among their sub-parts. The recognition of the model in Figure 1.2 in the scene in Figure 1.1 exhibits this type of behavior. While the sub-parts are assumed to be rigid, the rigidity does not apply to the whole object.

The reason that the recognition process itself does not actually place any constraints on the way sub-parts are generated is that only the models need to be decomposed into sub-parts and not the viewed scene. A system does not need to break up both the models and scene into component sub-parts and then compare those sub-parts. Instead the system may attempt to match any model sub-parts anywhere in the scene and then verify that identified instances of the model components are consistent. The scene therefore does not need to be segmented into its component sub-parts prior to the recognition step, a task that is often difficult to accomplish due to occlusion. In addition, an initial segmentation phase is inefficient since we would like to use the knowledge of any identified sub-parts in the scene to help predict the identity of related sub-parts, rather than attempt to independently identify the sub-parts. There are also no constraints on the appearance of sub-parts since they may consist of any configuration of object features. There is no need to define predetermined categories of parts and spatial relations. Instead, the requirement placed on sub-parts is that they capture some shape property of the object that tends to remain consistent—any object variations should occur between sub-parts rather than within them.

The idea of breaking objects into their components for the purpose of object representation has been explored before, but has not been implemented in many recognition systems. Hoffman and Richards [Hoffman 86] argue that the human visual system cuts surfaces into sub-parts in order to interpret scenes. They also propose that sub-parts should be separated at points of concavity since when two arbitrarily shaped surfaces are made to intersect, they usually form a concave corner. An interesting point about their proposal is that objects are decomposed into sub-parts based purely on the bounding contours rather than on properties of the circumscribed region. Their claim can be supported by examining some optical illusions. In Figure 3.2, for example, the dots appear to lie either on the same step or on adjacent steps depending on which side of the staircase corresponds to the foreground of the figure. In one configuration, the concavities separate the step sub-parts such that the two dots are one step. In the other configuration, the two dots are separated by a concavity and thus lie on two dif-

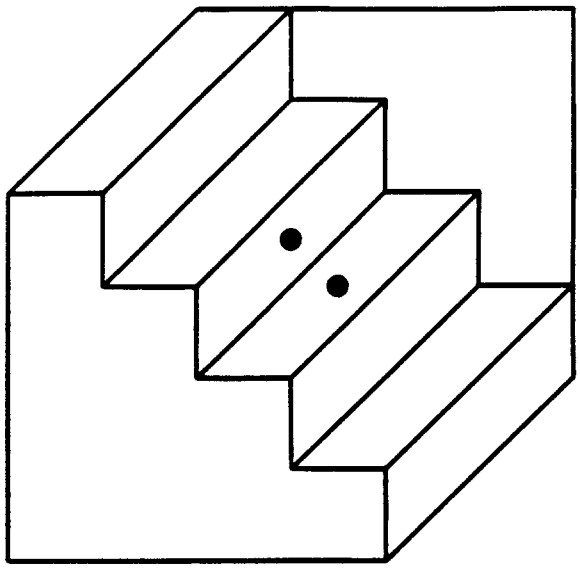


Figure 3.2: The Schroder staircase, published by H. Schroder in 1858, shows that sub-part boundaries change when foreground and background reverse. Rotating the figure helps reverse the orientation. The two dots appear to lie on a single step sub-part in one configuration, and on two adjacent steps in the other.

ferent steps. Figure 3.3 shows a similar behavior for stacked cubes, where the perceived concavities may break down the figure into two different cube configurations.

Shapiro and Haralick [Shapiro 79] also propose to decompose shapes into nearly convex sub-parts in order to extract the simpler parts of the whole object. They use a complex graph based technique to cluster together compact regions. Brady [Brady 84], Heide [Heide 84], and Bagley [Bagley 85] propose a contour and region based sub-part decomposition scheme that analyzes various discontinuities in splines (axes) of objects. These sub-part decomposition techniques point out that sub-parts are usually described as convex components and may be generated automatically by the recognition system.

Once the sub-parts have been generated, the representation must specify the relationship between them. One characteristic of this relationship is the type of connection between the sub-parts. The connection may be the description of the concavity separating the components or the point along each sub-part where the other sub-part intersects it. The geometric relationship between the components should also be specified. This association can be specified symbolically, such as *above*, *to-the-right-of*, or *between*, or, more accurately, by specifying the relative translation, rotation, and scaling factors of the sub-parts. The latter method has the advantages of easy specification and com-

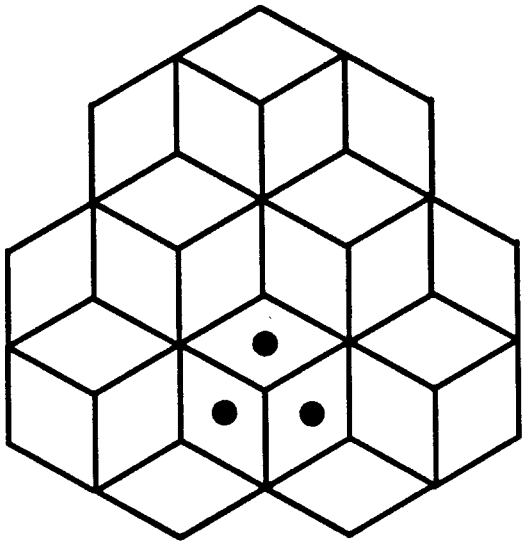


Figure 3.3: **Concavities can appear in two different configurations for this arrangement of stacked cubes. As a result, the three dots may first appear to lie on a single cube, but they may then appear to lie on three different cubes when the figure reverses.**

putation. The transformation between two scene sub-parts can simply be compared against the model transformation to ensure that it falls within a specified threshold. The symbolic sub-part associations, however, may consist of numerous labels in order to be complete and as a result may be difficult to compute and compare for similarity.

3.5 The Hierarchical Nature of Recognition

In order to harness the advantages of the structure and scale hierarchies, a recognition system must satisfy the following criteria:

- automatic generation of the hierarchies,
- incorporation of hierarchies into a single representation paradigm, and
- exploitation of both hierarchies in the recognition engine.

In order to allow the system to function independently it is desirable to generate the structure and scale hierarchies automatically. Section 3.4 supports the feasibility of sub-part separation based on object shape. The methods for generating structure hierarchies can be grouped into two categories: concavity localization and clustering.

Concave points along the contour of a shape can easily be found by measuring the orientation change along the contour. Concavities that are *distinctive* can then be used as sub-part boundaries. These concavities may be identified at the coarse levels of the scale hierarchy in order to extract them reliably. Region based clustering may be accomplished by grouping all features that are *visible* to each other through the interior of the object. This grouping yields compact sub-parts. Contour based clustering may be performed by measuring the spatial proximity of neighboring features and grouping features that are relatively close to one another. This process yields sub-parts that have high feature densities.

Scale hierarchies may be generated by using a scale-space filtering approach [Witkin 86]. This process smoothes a signal with several filters, continually decreasing in size, in order to separate dominant features from less significant ones. In processing bounding contours of shapes, the orientation and curvature of the contour may be convolved with a range of Gaussian filters. These convolutions are of the form:

$$f(s) \otimes \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-s^2/2\sigma^2}\right)$$

where $f(s)$ is the orientation or curvature in terms of arc length along the contour, and σ, the standard deviation of the Gaussian, defines the size of the filter. Larger scale filtering is used for detection of coarse features while the smaller scales detect finer features. Since the features on the various scale levels are associated with each other, they may be linked in order to form a scale-space tree. The location of the coarse features can be traced down the scale-space tree in order to accurately localize them at the fine scales. The coarse primitives cannot be localized accurately at the coarse scales since the large filters smear out the contour so that the primitive locations at those scales might not correspond exactly to their locations on the actual contour. The behavior of the primitives at the various scales can also be used to refine the symbolic labels of those features. For example, a corner at a coarse scale may actually be an end point consisting of two smaller corners at the finer scales and can thus be labelled more accurately as a two-corner-end rather than a corner.

In order to reap the benefits of both structure and scale hierarchies, it is important to differentiate between them as well as combine them into a single representation. One way to integrate both hierarchies into a single descriptive representation is to define two orthogonal dimensions along which the hierarchies operate, as in Figure 3.4. Along one dimension whole models are decomposed into their component sub-parts. The

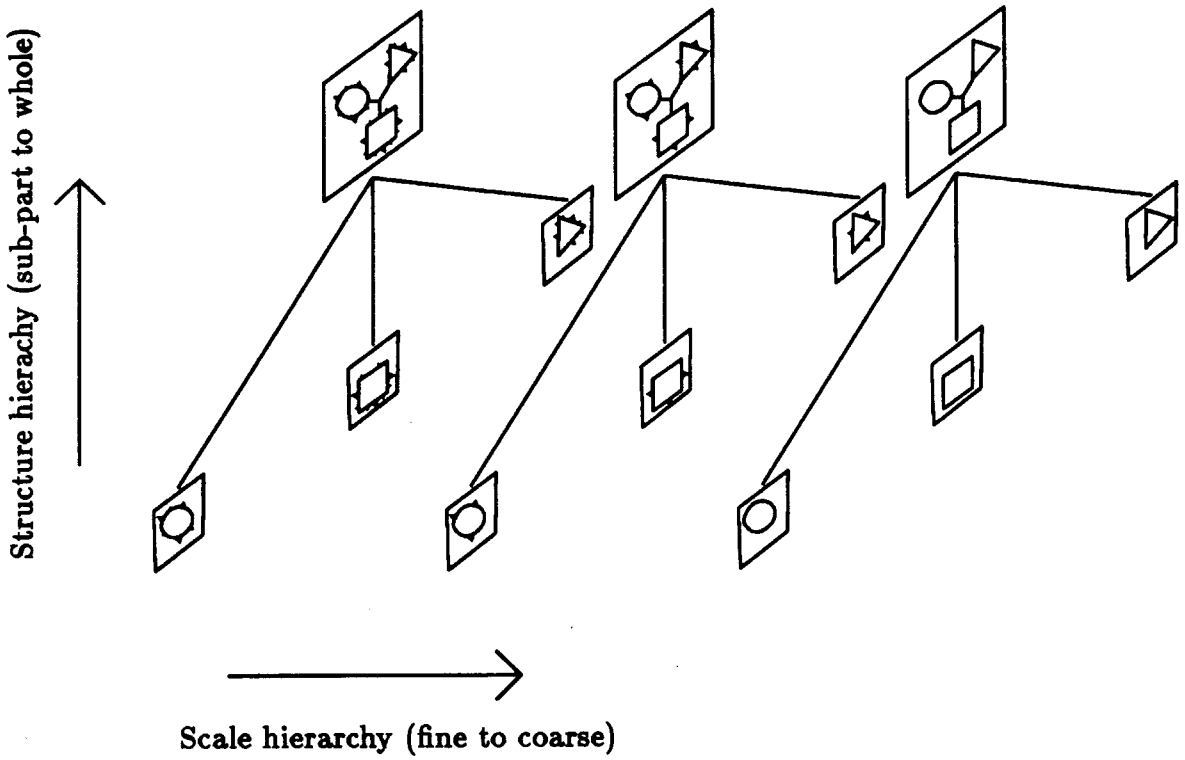


Figure 3.4: Structure and scale hierarchies may be thought of defining orthogonal axes of the representation. The model object shown consists of three geometric shapes connected together, with their edges perturbed at fine scales.

other dimension consists of a fine to coarse description of these components. With this paradigm, the system can explore either one of the object decomposition methods by processing the representation along one of the axes, or both hierarchical methods by stepping through the representation in both horizontal and vertical steps.

The model libraries consist of collections of these two-dimensional representations of objects. Another type of hierarchy can be developed within the library to define object classes. These classes are specified by the degree of sharing of their component sub-parts. Figure 3.5 shows an example of a hammer class that is defined by connecting a common (possibly scaled) handle sub-part by two concave corners to a hammer head sub-part. The different head sub-parts, while differentiating the hammers, all have at least one striking surface sub-sub-part in common. Such a class hierarchy allows a system to compare two objects and reason about their similarity, but does not define canonical types—single models that represent all the objects in the class. In order to derive representative types for a class, the object components and their relationships must be

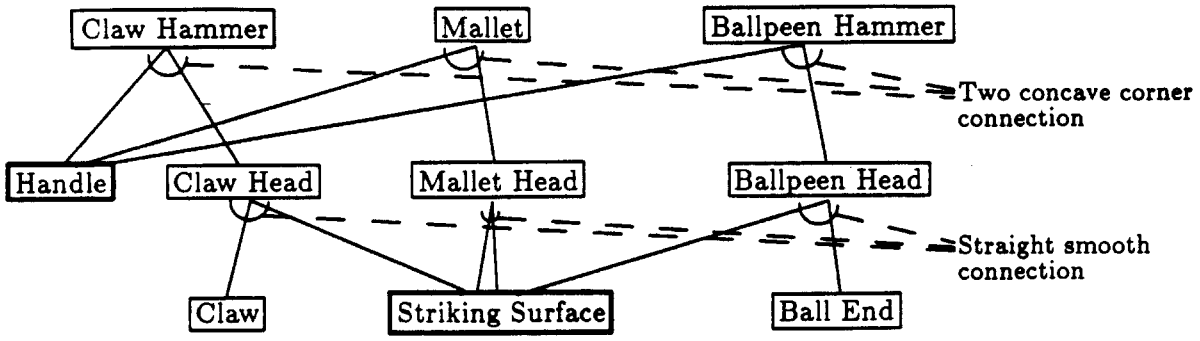


Figure 3.5: **The sharing of components in the structure hierarchy of a hammer class.**

analyzed in order to determine which characteristics determine class memberships and which features can be derived from all members of the class. These criteria are usually determined by analyzing functionality issues since objects are often grouped together by virtue of their ability to perform the same task.

Since the approach in this thesis is to classify objects solely based on shape, functional issues are not used in determining canonical object models. Knowledge of function requires much more information than just object shape. This knowledge must be provided by extensive user interaction or by learning from the experience of using the objects. These types of representation pose interesting and challenging problems in the field of recognition which should certainly be pursued in future work. A simple shape-based method of determining representative models without user intervention may be to average the matched features of model components that are determined to be similar and then use the averaged component to represent the whole class of components. This averaging process may consist of averaging relative sub-part transformations (scaling factors, rotations, translations) and feature properties (feature orientations and locations, corner angles). The problem with this approach, though, is that the averaging process does not necessarily abstract the important features that unite objects in a class. Instead the canonical object, in the context of this work, is defined by the interaction of the scale and structure hierarchies. The shared components in a class define the commonality among its members and the coarse features of these components abstract their important shape information.

This type of class hierarchy may be used in a recognition engine to achieve a focusing general-to-specific recognition behavior. In order to index into a class of objects, the system attempts to identify its common sub-parts and connectivity relations in the viewed scene. The recognition of these elements is performed by first using their coarsest features to ensure that an approximate match exists and then using the finer features to verify and refine their interpretation. Once an accurate identification and localization of these indexing components is performed, the unique member of the object class may be identified by attempting to match sub-parts that continually narrow the sub-class under consideration. Since the configuration of some of the sub-parts is already known, the configuration of components that differentiate between class members may be predicted and quickly verified by the system.

3.6 Inherent Tradeoffs

The correctness, efficiency, and robustness goals of a recognition system place many constraints on its design and performance, such as the representation that may be used or the number of scenes that may be recognized. In order to effectively evaluate a recognition system, it is necessary to outline how the constraints interact with each other in order that we may analyze which ones are stressed and which ones are compromised in reaching the desired type of performance. As the constraints on the system are varied, certain system performance parameters vary too. These parameters define a multi-dimensional space of recognition behavior, where along each dimension a parameter defines a performance tradeoff. This section describes some of these dimensions in terms of the recognition behavior that is achieved with variations in the tradeoffs.

3.6.1 Model-Driven versus Scene-Driven

One dimension along which the recognition behavior can vary is the direction the recognition is driven. A top-down, or model-driven, approach uses the information contained in the model library to direct the search for appropriate model matches in the scene. A bottom-up, or scene-driven, approach constructs the interpretation by using features found in the scene to index into the library and hypothesize matches.

A totally model-driven system would attempt to match each model in the library against features in the scene and then return the best match (or matches) as its interpretation. The best match may be based on the number of features that are matched

or the fraction of the model that is supported in the scene (e.g. the normalized length of the object contour that is found in the scene). Since this search is exhaustive, it guarantees finding the best interpretation, but by virtue of being exhaustive, it is also very inefficient. One of the goals in the development of object libraries is to index into a small subset of models and not explore every model/scene match.

A scene-driven system is much more efficient at the cost of possibly missing the best interpretation. Such systems attempt to hypothesize (or make educated guesses of) probable model configurations by (1) grouping together some scene features into larger primitives that can index a subset of object models or by (2) executing a fast preprocessor over the library models to find the configurations that are most compatible with the scene features. These hypotheses are then verified by actually matching the features of the hypothesized model configuration with the scene features. Since completely exact matches are unlikely due to occlusion and noise, the system returns any complete interpretations that are found to exceed a preset match threshold. The search continues until all scene features are accounted for or no hypotheses remain. Since this scheme does not explore the complete search space of model/scene matches, it may miss interpretations that are better than the ones that it derived. It therefore strongly relies on accurate hypothesizing capabilities in order to include the best final configuration(s) among its initial candidates.

The model-driven approaches and scene-driven approaches may be combined in order to achieve a hybrid system. For example, some hypothesizing capabilities can be added to the model-driven scheme so that only a few of the possible configurations are searched. Alternatively, the system can begin by matching all models in a model-driven fashion, but after matching only small parts of the models determine the best matches and hypothesize the complete configurations. The main problems in such a system are deciding how many parameters of the interpretation to actually hypothesize and at what point to perform the hypothesis step.

The risk of making bad hypotheses can be lowered by reducing the number of parameters that are hypothesized. For example, we can just hypothesize the identity of the model object and then perform the recognition for that object in a model-driven manner. Increasing the amount of information that is hypothesized reduces the amount of verification work remaining to be done, but if one of the hypothesized parameters is wrong, we might miss a good interpretation, even if all the other parameters are correct. For instance, if we hypothesize both the identity and the transformation of a

model object, but the hypothesized scaling factor is wrong, we might mistakenly conclude that the object is not in the scene. Based on the nature of the hypothesis step, several requirements are imposed on its performance:

- **Conservative behavior.**
  The list of hypotheses must include the seeds for the correct interpretations—the parameters that are hypothesized should be accurate. The hypothesizing process must therefore avoid inadvertently pruning any parts of the search space that lead to the correct matches.

- **Good foresight.**
  While being conservative, the hypotheses should also not include very many wrong predictions that the system would needlessly attempt to verify. The hypothesis module should attach preferences to its predictions so that the recognizer can know which ones to try first. This ranking is desirable so that the recognition system can find the correct interpretations early and remove the scene features which contributed to that configuration, therefore simplifying the remaining process.

- **Efficiency.**
  Since it is just a preprocessor, the hypothesis step should be very efficient. Its running time should be negligible compared to the time the rest of the recognition process takes. Since this module will usually scan all the objects in the library, its time complexity will be linear in the number of objects. Even though the time complexity of the whole recognition system is designed to be sub-linear, the execution time of the hypothesis module should not overtake the time of the recognition module for any reasonable size library.

The general goal of the hypothesis module is thus to efficiently generate rough ideas of the scene interpretations. The information in the scene is used to focus the search since we do not want to blindly start matching library models in the scene.

Hypothesis generation can be performed at many steps during the recognition process, leading to a distinct tradeoff. Clemens [Clemens 86] provides an in-depth discussion of this tradeoff. If we hypothesize too much too early, we risk misidentification of the best configuration. If we wait to hypothesize until we are almost sure of the best configuration, we reduce the efficiency of the system. The optimal point would thus seem to be where we can explore as little of the search space as possible without sacrificing our confidence in the derived interpretations.

3.6.2 Feature Complexity versus Matching Complexity

The primitives that form the basis for the representation of a model object must be chosen judiciously since all interpretations are based on their degree of matching. The purpose of these features is to make the important shape information explicit in the representation so that feature matches can lead directly to object recognition. This goal tends to require complex primitives that demand much computational processing of the image and as a result may be difficult to construct accurately. Simpler primitives are easier to detect, but since they contain less information, they are more difficult to match against the object models. Thus, an inherent tradeoff exists between the complexity of the representation, corresponding to the amount of processing needed to derive the primitives, and the size of the search space, corresponding to the amount of processing needed to derive scene interpretations.

Some points along this tradeoff continuum are presented in Section 2.1, which describes some of the types of representations used in current recognition systems. At one extreme is the use of actual contour patches as the representational primitives. These features are simply derived by breaking the bounding contour into variable length segments and do not require much processing. Matching of these features, though, is costly since they are matched on a point-by-point basis of pixels of model features against contour pixels in the scene. Since this representation does not abstract much shape information, it also does not extend well to scale variations and sub-part parameterization. As features abstract more shape information, the representation adapts better to more robust recognition. A higher level representation can compensate for scaling of the object, variations in its sub-part relationships, or mirror image reversal by understanding how these changes modify the lower level primitives. The matching combinatorics with more complex features is also reduced since the representation is more compact. The tradeoff is that now more processing is needed to derive the features and to decide whether two features actually match. The more complex features are usually defined by many parameters and since exact matching is not usually possible, rules must be developed that define the tolerance levels for variations in feature parameters. As more flexibility is added to the system, it becomes more complex, possibly leading to less accuracy and more noise sensitivity.

In order to better judge the advantages and disadvantages of the feature complexity tradeoff, the following criteria for shape features are outlined.

- **Large domain.**

The features used for the representation should be able to represent a large class of objects. Some features, such as polyhedra, are limited by their inability to represent curved surfaces well and are only suitable for planar regions. Other features do not extend well to concave objects since concavities are used to segment objects. A general representation should not be limited by the types of shapes that it can represent. Of course, restricted domains of objects may often be recognized with a small set of simple features, simplifying the problem greatly.

- **Stable and sensitive.**
  This criterion, defined by Marr and Nishihara [Marr 78], suggests that feature specifications should not fluctuate greatly in the presence of sensor noise or distortions, variations in viewing conditions, or small variations in shape geometry. Features should be generated reliably so that instances in the model and the scene can be matched accurately. At the same time, though, the features should be able to differentiate between similar shapes. A scale hierarchy adapts well to the separation of the stable features from detailed sensitive features. The coarse features remain stable in the face of small fluctuations of the shape which will only be noticeable at the finer levels. The derivation of a hierarchical representation adds some complexity to the feature generation step, but simplifies the matching process by converging on the correct interpretations.

- **Local support.**
  The complexity of the features is limited by their size. Features should be based on local shape cues in order to handle occlusion. If features are used to represent large sections of the shape, they may be more difficult to recognize if that part of the shape in the scene is occluded or perturbed. Features such as object area, for example, that are based on global shape information cannot be extracted accurately in the presence of occlusion.

- **Information preserving.**
  The model representation should be information preserving—the original shape, and only that shape, can be reconstructed from the representational features. The reconstruction should be accurate enough so that the recognition system cannot distinguish between the original shape and the reconstructed one. This criterion guarantees that the representation is unique relative to the recognition system in which it is used. If features abstract too much information they yield ambiguous

matches and thus multiple interpretations can result. Global features, such as object area, tend to exhibit this form of ambiguity.

- **Multiple feature types.**
  The use of several feature types achieves a combinatoric benefit, as shown in Section 3.2. These distinctive features are easier to match as long as the feature stability criterion is met.

3.6.3 Sub-part Size versus Number of Sub-parts

Since objects can be decomposed into a variable number of sub-parts of various sizes, it is worthwhile to analyze the effect of this tradeoff. At one extreme, the hierarchy can contain just one level—the whole object. At the other extreme, the object is broken down into many small components, the smallest of which contain a single feature. The object can be broken down directly into these small components, or recursively into smaller and smaller components, resulting in many levels in the hierarchy. By virtue of the benefits of the structure hierarchy outlined in Section 3.4, we would like at least two levels in the hierarchy. The smallest sub-parts should also consist of at least a few features since recognition of single feature sub-parts is meaningless. The recognition of one feature sub-parts for the purpose of recognizing their parent component consists of the same procedure as the recognition of that component in terms of its constituent features.

The smallest components in the structure hierarchy should be large enough (contain enough features) that their identification indexes some proper subset of the model objects, i.e. the components are not contained by all the models. Many individual features will generally be contained by many, if not all, model objects, and as a result their identification does not limit the number of models still to be considered. It is also desirable, though, to use sub-parts that are not unique and as a result may be shared by several objects. If all the components are salient in the library, their identification quickly leads to the identification of the model object, but the recognition process consists of a linear search through these components until one is identified. While it is possible to use various hypothesis techniques to attempt to find probable components, the use of sharable components would increase the probability of recognition of these components. Once a component is identified we can begin focusing on the complete interpretation by considering only parent components that contain the identified one and

by matching finer features to refine the constructed configurations. Since the focusing process cannot begin until at least one sub-part is identified, we would like to identify that component as fast as possible. Thus a high degree of sharing is preferable to no sharing at all since, even though identification of a highly shared component does not greatly reduce the number of models that need to be considered, it does determine the localization of each of these models [Ullman 86], thus constraining the remaining search. An optimal degree of sub-part sharing among model objects is difficult to define since that task would require the ability to choose any part of the object to be a sub-part. We would instead prefer the sub-parts to be semantically meaningful so that a more robust recognition can be performed—the identification of sub-part parameters. Thus we are not really free to break objects into sub-parts at those points that would result in some optimal measure of sub-part sharing.

The recognition process will usually commence with the identification of the components on the bottom level of the structure hierarchy and progress into higher levels until a complete object interpretation is derived. The components on the lowest level must be matched in order to conclude a match for components higher in the hierarchy. As a result, the lowest level is the one for which the size, number, and shareability criteria apply. Since any intermediate sub-part levels must be combined to yield a complete object interpretation using the smallest components, we may just think of the object being divided directly into those small sub-parts. Using this type of recognition scheme, the number of levels in the hierarchy will generally not affect the recognition process since the system will need to explore matching all the components on the lowest level regardless of the number of levels between the lowest level and the top level. The intermediate levels would only be useful in providing names for components composed of subsets of sub-parts and for providing more effective parameterization of these components. If an intermediate component, such as the hammer head in Figure 3.5, is found to be rotated relative to other components, we only need one parameter to describe that variation rather than a parameter for each of the smaller sub-parts.

3.6.4 Binary Matches versus Qualitative Matches

Different methods may be used to indicate similarity of features, sub-parts, or objects. Since matching of these elements will usually not be exact, due to quantization variations and occlusion, measures of similarity must be defined to take into account the possibility of various forms of noise. One way to define similarity is to use a binary

measure where elements are either concluded to match or not to match [Grimson 84]. A threshold for the maximum amount of noise can be defined for the measurement of feature parameters and if (and only if) the difference in the values of a parameter in the model and scene falls within that threshold, the features are matched. For example, in verifying the location of matched model and scene features, a tolerance sphere can be defined around each scene feature within which the model feature may be located. Similarly, a predefined cone of error can be used to allow some orientation variation between matched features. The goal of this scheme is to allow enough flexibility in the system to match inexact components without introducing spurious interpretations. The method is appealing due to its simplicity. The system does not depend on the particular setting of tolerance values. Rather, the system's performance will gradually vary from recognition of exact instances of models to more inexact recognition as the tolerance levels increase. This behavior can occur as long as the representation is based on descriptive features, and the recognition engine exploits constraints between these features so that invalid interpretations are not derived even with large tolerance levels.

An alternative method of defining similarity is to define a quality measure of the goodness of match [Bhanu 84(a)] [Turney 85] [Ayache 86]. A quality factor, such as a number in the range [0, 1], can be assigned to indicate the closeness of the match of two features. These factors can then be combined to yield quality factors of sub-part matches and object matches. If these object match measures exceed a certain threshold, then an object match is concluded. The problems with this scheme relate to the problems with inexact reasoning in other areas of artificial intelligence. While it is desirable to assign a number to measure the goodness of fit, the source of this number tends to be arbitrary. For example, when measuring the difference in orientations of features, should the match value decrease linearly, polynomially, or exponentially with increasing orientation difference? Another problem is the manner in which these quality factors are combined to reach a conclusion that is based on several measurements. Various schemes, such as fuzzy reasoning [Zadeh 65] or Dempster-Shafer probabilistic reasoning [Shafer 76], have been proposed to solve this evidential reasoning problem. While these mechanisms exhibit varying kinds of behavior, it is unclear how to select the desired manner in which the measures of uncertainty should be combined.

While quality measures provide an easy way to evaluate matches, they suffer from a lack of expressiveness. These measures abstract away all the information incorporated in their formation and as a result that information cannot be considered when the measures

are later combined [Bagley 85]. Since the match values are propagated through larger and larger interpretation, errors in the setting of these values may expand and lead to incorrect interpretations. These errors are hard to measure, though, since it is difficult to decide on what the *correct* value of the quality measures should be. As a result, the performance would be sensitive to variations in the setting of these measures. In general, recognition systems would be more dependent on the setting of the quality measures than the setting of the tolerance levels for the binary matching scheme.

The tradeoffs described in this chapter, model-driven recognition versus image-driven recognition, feature complexity versus matching complexity, sub-part size versus number of sub-parts, and binary matches versus qualitative matches, have been explored experimentally in an implemented recognition system, as described in Chapters 4 and 6. An analysis of these tradeoffs is then given in Section 6.3.

Chapter 4

The SAPPHIRE Recognition System

A complete library-driven recognition system combining scale and structure hierarchies was developed in order to demonstrate and test its advantages, evaluate its performance criteria, and define its limitations. The resultant system, named SAPPHIRE (**S**ub-**p**art **A**nalysis **P**rocedure for **P**arameterized **HI**erarchical **RE**cognition), is described in this chapter. SAPPHIRE is designed to recognize two-dimensional objects using visual camera input to derive their shape, based on the goals specified in Section 1.1. The system was completely implemented on Symbolics 3600 series Lisp Machines.

Figure 4.1 shows a description of the system. Each of the main modules: representation processor, sub-part/library processor, and recognition engine are presented below.

4.1 Representation Processor

The representation processor is used to process both models and scenes in order to ensure that the same representation is derived for a model and its instance in a scene. It receives as input a two-dimensional array of grey values that are furnished by a TV camera. Its output consists of a scale hierarchy of features that represent the viewed object(s).

The input intensity pixels are processed by an edge detector [Canny 86] in order to find step changes in intensity. The output of the edge detector consists of the location and orientation of the edge points. These points are then strung together by a boundary

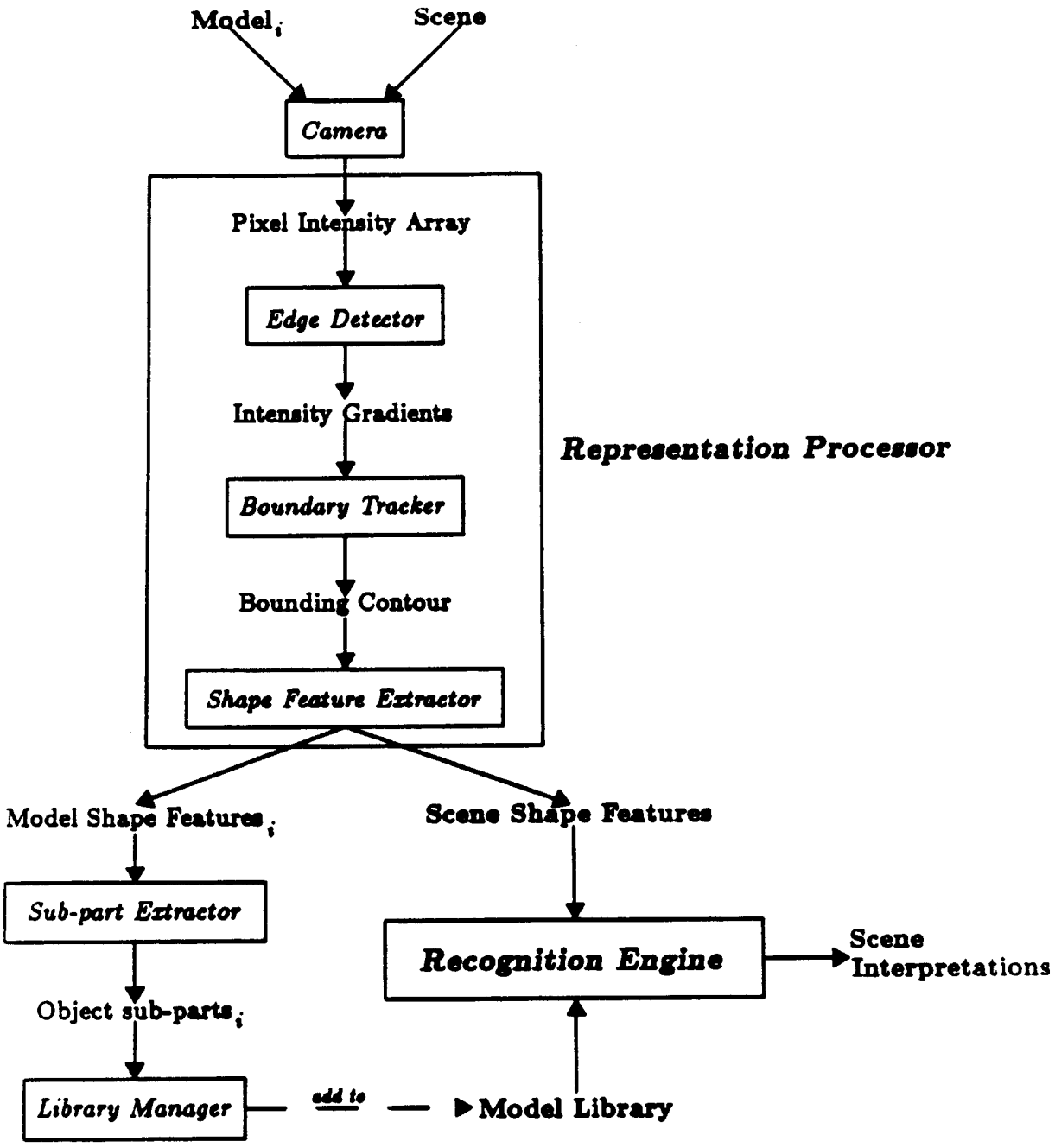


Figure 4.1: A top level description of the SAPPHIRE recognition system.

tracker that follows the edge points based on their orientation, possibly jumping across small gaps that the edge detector was not able to identify accurately. Short contours are removed at this point since the system will not be able to identify features on them. The resulting contours define the shapes in the image.

These contours are then processed by a feature extractor in order to derive the actual representation. The representation is based on the Curvature Primal Sketch (CPS), developed by Asada and Brady [Asada 86]. It is in turn based on the primal sketch representation advocated by Marr [Marr 82] who used this representation to describe significant intensity changes. Marr used symbolic primitives as the lowest level features—edges, bars, blobs, and terminations of discontinuities. The Curvature Primal Sketch extends this idea to finding significant changes in the curvature along the bounding contour of a planar shape. These curvature changes correspond to knot points of the contour, such as corners or points of inflection. A similar representation is the codon [Richards 84] [Richards 85], which identifies maxima, minima, and zeros of curvature as well as constraints between them. The CPS representation is used for the SAPPHIRE system since it has the added advantage of a scale hierarchy.

In order to derive the CPS representation, a scale-space filtering approach is used to identify and localize discontinuities in contour orientation and curvature, resulting in a multi-scale interpretation of the contour. The contour is convolved with several Gaussian filters with varying standard deviations, σ. In this implementation, four different filters are used, usually with $\sigma \in \{34, 24, 17, 12\}$ (measured in pixels). Starting with the coarsest scale, the contour is examined for step changes in orientation and curvature by locating zero crossings in the second derivative of that property. When such a discontinuity is encountered, the scale at which it is found is used to define the coarseness of each feature. It is then tracked through the finer scales in order to better identify its type and to better localize it.

The features used to represent the significant changes in curvature are shown in Figure 4.2. The simple primitives consist of a corner, signifying a step change in orientation, and a smooth-join, signifying a step change in curvature (but not in orientation). An inflection is a smooth-join with a change in the sign of the curvature. Based on these simple primitives, three compound features are defined for configurations of closely spaced occurrences of simple features. These consist of an end, two corners of the same sign; a crank, two corners of opposite sign; and a bump, three corners of alternating signs. These compound primitives are usually identified by the way they break up

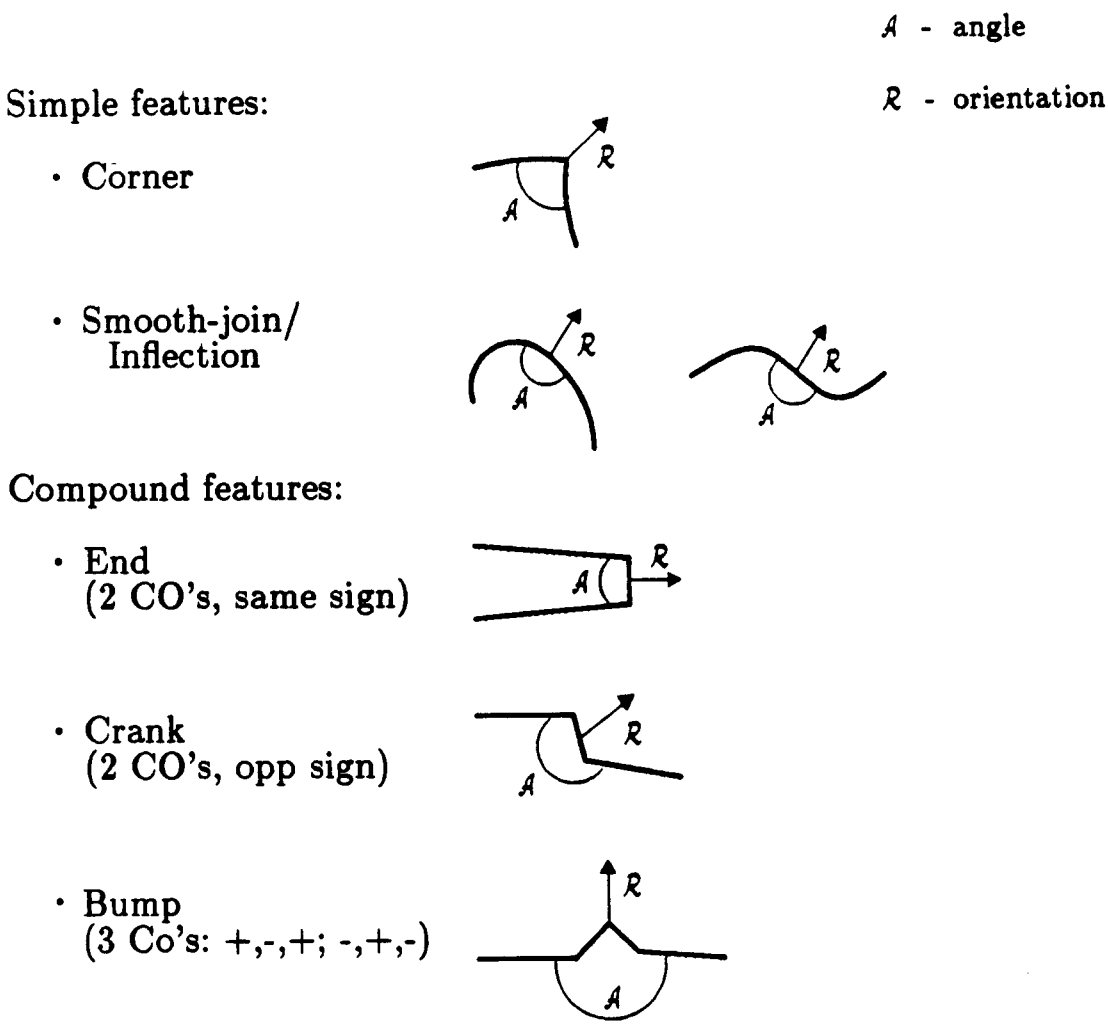


Figure 4.2: The CPS representational primitives showing the definition of the feature angle and orientation.

into their sub-primitives at finer scales. Even though the sub-primitives are not always identifiable at the finer scales, the compound features can still be identified by their discontinuity pattern at the coarser scales.

Some of the representational primitives are more reliably extracted and contain stronger shape cues than others. As a result, they are divided into two groups based on their significance. Type I primitives, the major set, consists of the end, crank, and corner. Type II primitives, the minor set, consists of the smooth-join, inflection, and bump. The smooth-join and inflection are more difficult to derive since they require an additional derivative (curvature is computed as the derivative of the orientation of the contour) to identify. They are also difficult to localize since it is often unclear exactly

where the curvature changes along a curve. Bumps are less significant since by their definition they are small features defining only fine details of the shape. Within each group, features are further ordered by their significance. Compound primitives tend to be more prominent than simple features. Primitives also become less significant as their angles approach 180°. This ranking of features is used in driving the recognition process since the matching of the significant primitives leads to more accurate configurations.

In order to adapt the CPS representation for a recognition system, the following properties are defined for each feature: type, scale, location, orientation, and angle. Compound primitives are also described by their sub-primitives and the distance between their outer primitives. These properties allow the system to compare features and match compatible ones. The concept of feature edge vector is also described since it is used in the definition of angle and orientation. The feature properties are summarized below and demonstrated in Figure 4.3.

- **Type** — The type of feature is one of: corner, end, crank, smooth-join, inflection, or bump. These types are abbreviated as: CO, E, CR, SJ, I, and B, respectively, in the recognition examples.

- **Scale** — The scale field is the coarsest scale in which the feature is noticeable. Processing commences with the coarsest scale and progresses to the finer ones. The first scale in which a feature is identified by the CPS processor defines the coarseness of the feature.

- **Location** — The feature location is defined as the center point of the feature, which is the actual feature point for simple primitives, and the midpoint of the chord connecting the two sub-primitives for compound primitives.

- **Feature edge vectors** — Simple primitives are defined at points where different contour segments meet. Define the direction of the edge vector to be the direction of the tangent to each segment at the feature point. Each simple primitive has two unit edge vectors emanating from it. The edges of compound primitives are the outer edges of its sub-primitives.

- **Orientation** — The feature orientation is defined as the outward pointing normal to the feature. For simple primitives the orientation is the angle of the vector bisecting the edge vector, but aligned to point from the object outward. For

compound primitives, the orientation is defined as the (unweighted) average of the orientations of its sub-primitives.

- **Angle** — The feature angle is defined as the interior angle between its outer edge vectors, as shown in Figures 4.2 and 4.3.

- **Sub-primitives** — Compound primitives are composed of finer features that may be distinguishable at lower scales if the range of scales used is fine enough to identify them. The sub-primitives are defined using the parameters above.

- **Displacement** — For compound primitives the displacement value is the distance between the two sub-primitives that compose it. This value is only defined if these sub-primitives are identifiable.

The output of the representation processor consists of a list of object features, organized by type, scale, and contour identification (the contour from which they were extracted). An example of this representation is given for the model object shown in Figure 4.4. The model is an image of a bike viewed on a *bikes allowed* traffic sign. The edges found by the edge finder are shown in Figure 4.5. Some noisy edges are found, but most of these are filtered by the boundary tracker and CPS processor. The derived representation is shown in Figures 4.6 through 4.9, the coarsest through finest sets of features. Each figure shows the features that are first found at that scale. The features are indicated by their symbols: E – end, CO – corner, SJ – smooth-join. The shown contour is the shape generated directly from the representation. Only contours on which features were found are shown since featureless contours, such as smooth circles, are not easily recognizable by this system. The featureless contours do not contain distinctive points and as a result are difficult to localize.

It is worthwhile to note a few characteristics of the representation. While perfectly smooth straight or circular segments do not contain any features, the CPS processor does locate a few features on noisy or distorted such curves. As a result, a smooth-join is found on one of the outer circles of the figure and three corners are found on the wheels of the bike (see Figure 4.6). The recognition is not greatly effected by these ambiguous features since it is designed to tolerate some noise. These features are also classified as less significant features since their angles are close to 180°. Note also that while some of the fine features reflect details that are not visible at the coarser scales, others are the sub-primitives of ends at coarser scales. For example, the coarse scale

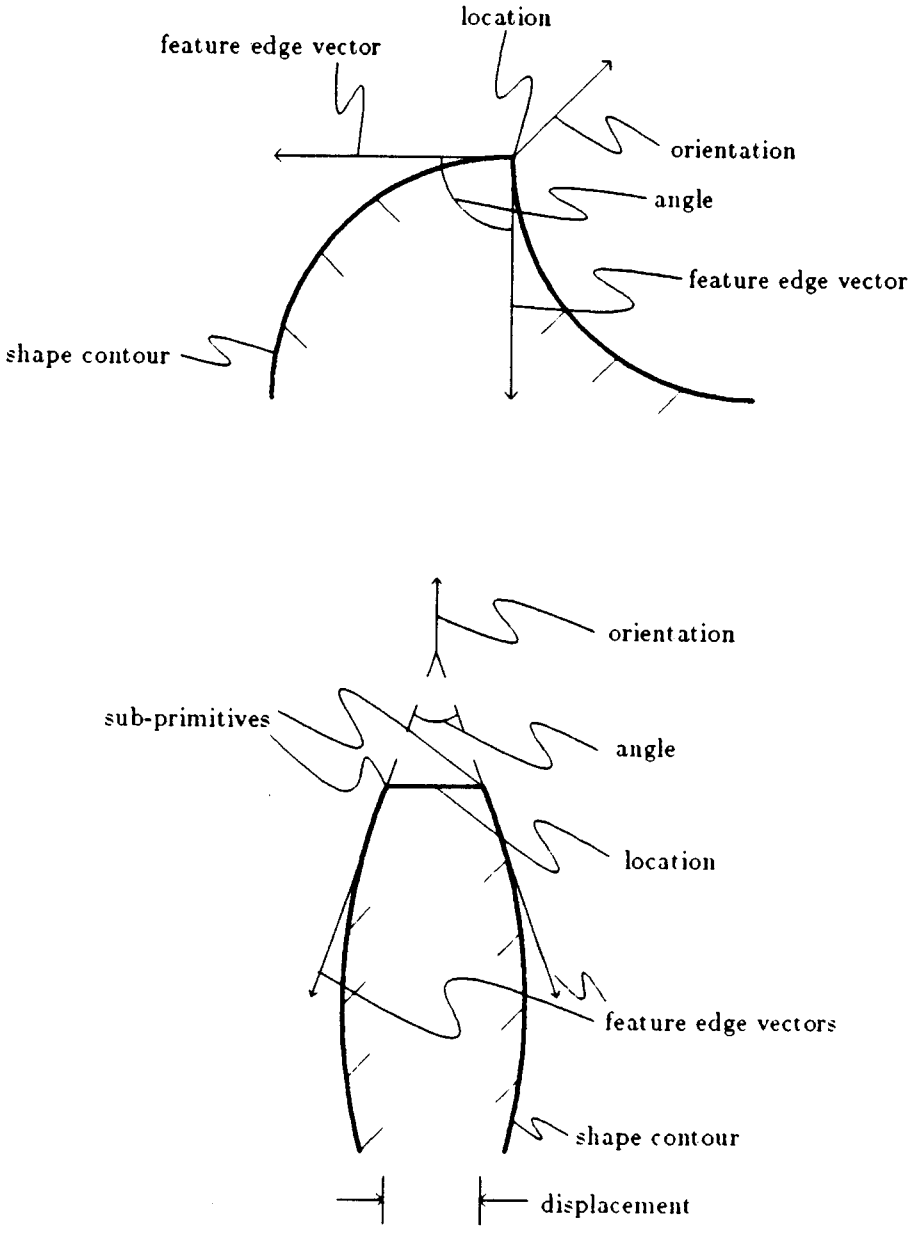


Figure 4.3: The definition of feature properties shown for a corner (top) and a two-corner end (bottom).

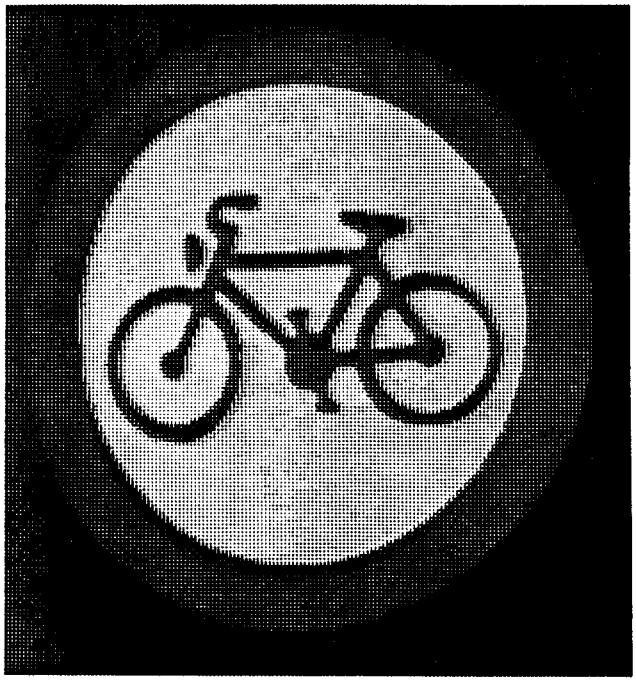


Figure 4.4: **Sample model object—a bike on a traffic sign.**

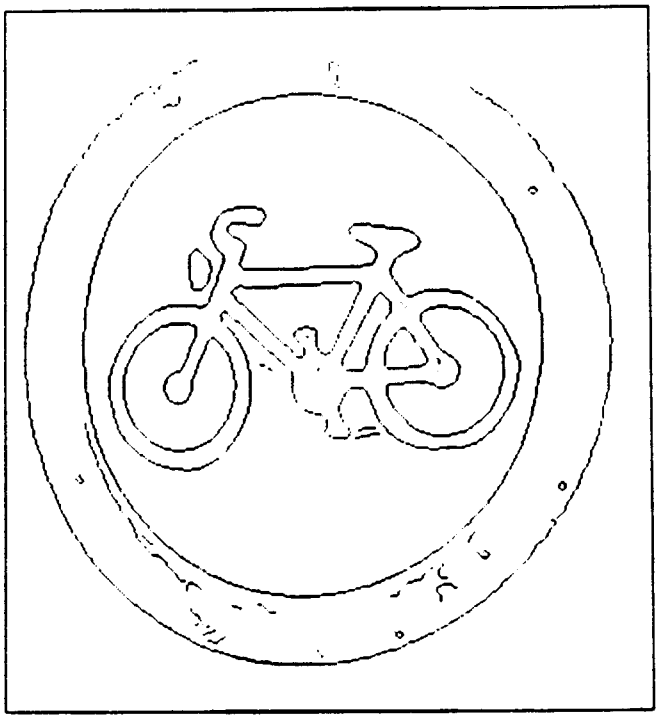


Figure 4.5: **The edges of the bike object found by the edge finder.**

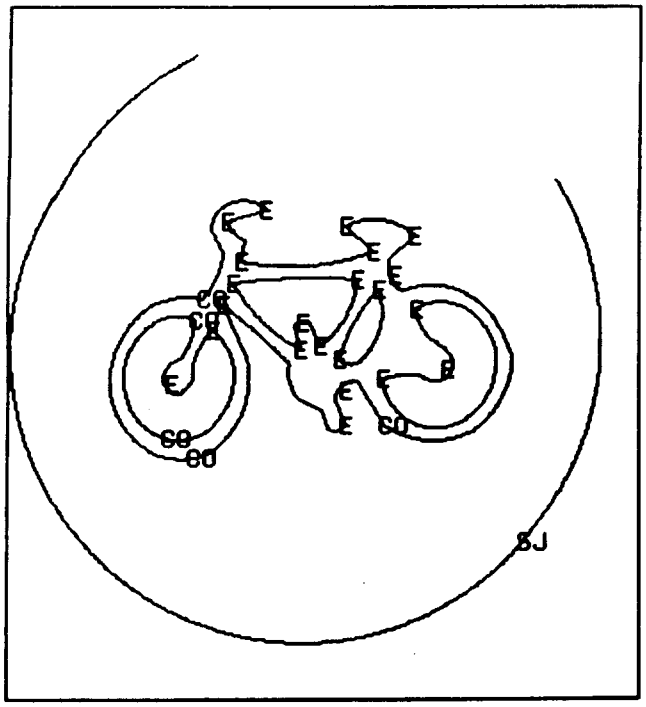


Figure 4.6: **The features found at the coarsest scale of the representation.**

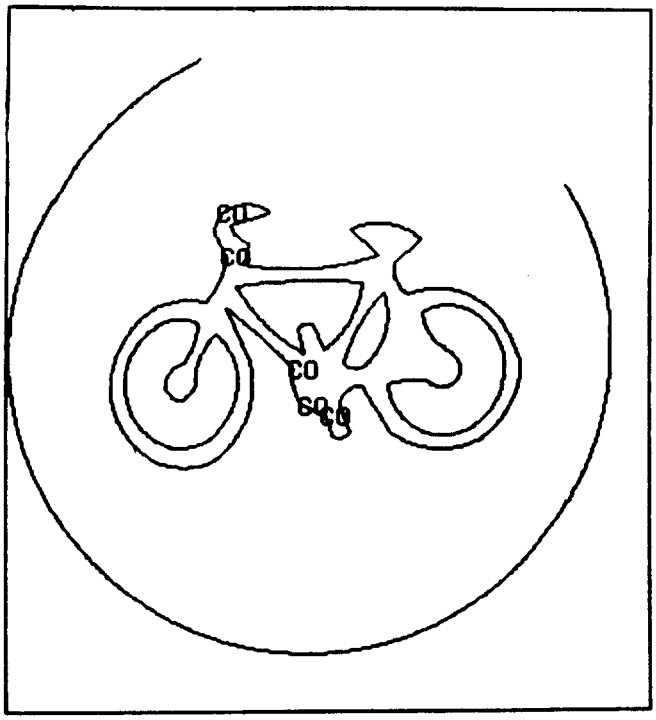


Figure 4.7: **The features found at the second scale of the representation.**

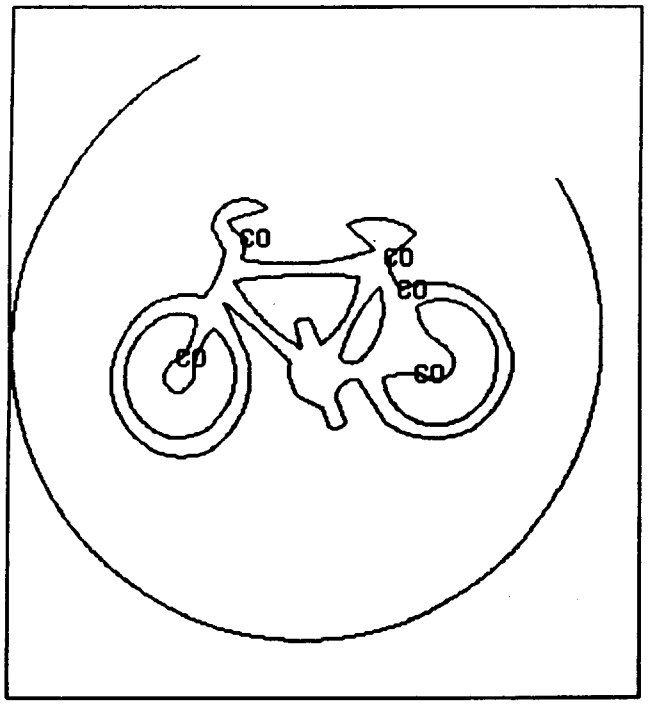


Figure 4.8: **The features found at the third scale of the representation.**

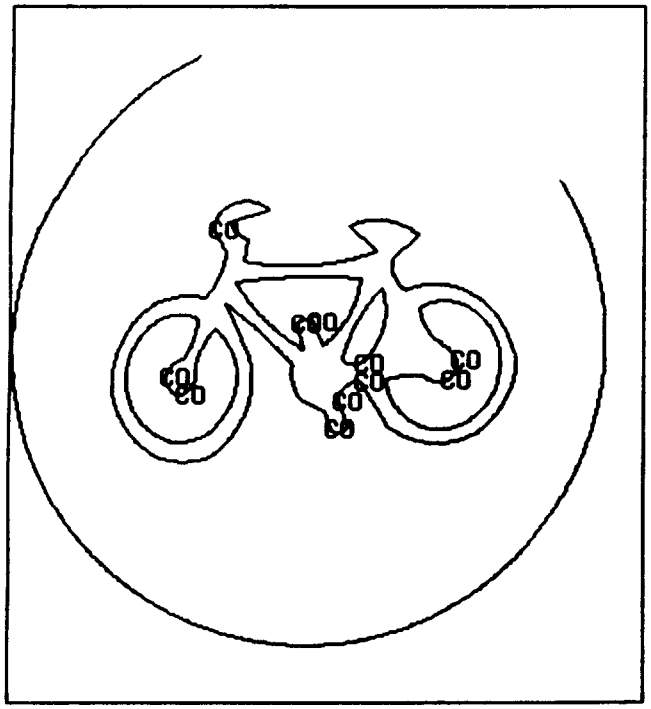


Figure 4.9: **The features found at the fourth, or finest, scale of the representation.**

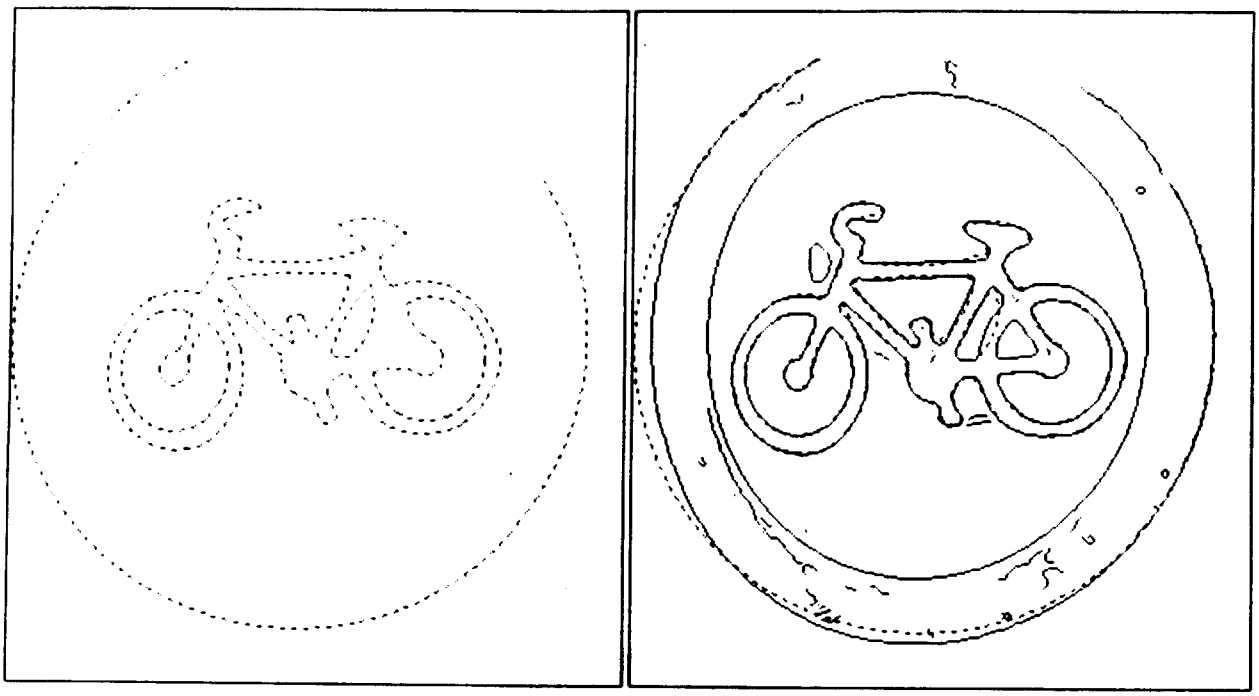


Figure 4.10: **The reconstructed edges of the bike object (left) and an overlay of the reconstructed object and the original object (right). The original contours are solid while the reconstructed ones are dashed.**

ends at the top pedal, and both wheel axles in Figure 4.6 are decomposed into pairs of corners at the finest scale (Figure 4.9).

The contour reconstruction is performed using piecewise linear and circular approximations of the contour segments connecting the feature knot points. These contour segment approximations are computed during the derivation of the representation, but are not currently used as part of the actual object representation. Asada and Brady show that this reconstruction process leads to a close approximation of the original object. Figure 4.10 shows the superimposition of the dashed approximate contours on top of the original bounding contours. This example demonstrates the information preserving quality of the representation.

This representation thus provides a rich vocabulary to describe the shape of an object which allows SAPPHIRE to achieve constraint and consistency in the recognition process. The descriptive nature of the representation allows feature matches to constrain derived interpretations—i.e. feature matches are not ambiguous. In addition, the hierarchical nature of the representation leads to reliably extracted features that may be matched consistently.

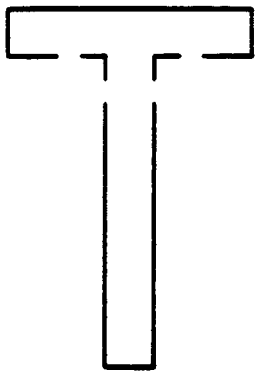


Figure 4.11: **The decomposition of an object at an indicative corner pair.**

Figure 4.12: **The decomposition of an object at an indicative crank pair.**

4.2 Sub-part Extractor and Library Manager

Once the representation is generated, the next step in processing model objects is to decompose them into their component sub-parts. This process, applied only to model objects and not to viewed scenes, is implemented based on the ideas developed in Section 3.4. The SAPPHIRE system performs this task automatically by attempting to identify points along the contour where sub-part variations may occur. In extracting the sub-parts the system ensures that the resulting sub-parts are not too small. The size measure used is the number of coarse features. Since the sub-parts are used to index into the library, we would like to use sub-parts that contain more than one feature—individual features are often located in many places on many objects and thus do not constrain the selection of candidate models. We would like the sub-parts to contain more than two features, though, in order to allow recognition of these components in the presence of occlusion. As a result, distinctive sub-parts are defined to have at least three coarse features in order to support stable sub-part matches. The following rules are used to generate the sub-parts.

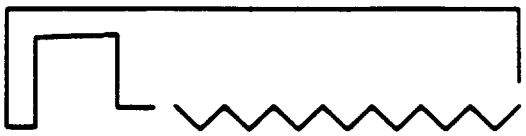


Figure 4.13: **The decomposition of an object due to close spatial proximity of successive features.**

1. **Contour connectivity.**
   Each connected contour of the model constitutes a sub-part. If none of these resultant sub-parts are distinctive (as defined above) then all contours together are considered one component.

2. **Corner concavities.**
   Each contour is analyzed for indicative pairs of concave corners at the coarsest scale. An indicative corner pair consists of two concave corners lying on the same boundary in close spatial proximity to each other, but which are separated from each other by at least one feature in each direction along the contour. The sum of the angles should be approximately 540° ($2 \times 270°$) and the difference in their orientations should be approximately 90°. This definition describes points of intersection that occur often in many objects. Figure 4.11 shows one example of this type of sub-part decomposition that can be used to separate the head of a hammer from its handle. The corner pair becomes part of neither sub-part and instead becomes a *connecting component.* The connection can then be verified separately since it will exhibit variations when the two sub-parts are rotated relative to each other in the scene.

3. **Crank concavities.**
   Each contour is also analyzed for indicative pairs of cranks at the coarsest scale. An indicative crank pair consists of two cranks lying on the same boundary in close spatial proximity to each other, but which are separated from each other by at least one feature in each direction along the contour. In addition, the corresponding concave corners of the cranks should meet the criteria for indicative corner concavities above. This type of sub-part decomposition signifies the type of part intersection that is shown in Figure 4.12. In this example, components

corresponding to the handle and shaft of a screw-driver become the sub-parts. The connecting pair of cranks becomes a separate *connecting component.*

4. **Close spatial proximity.**
   The last method for extracting sub-parts consists of locating segments of the shape containing strings of sequential features that are in close proximity to each other relative to their distances from the remaining features of the model. The average feature-to-feature distance for the model is computed and then each contour is analyzed for a minimum length string (currently 6) of sequential features where the distance between each pair of successive features is less than some fraction of the average distance (currently 0.67). An example of this type of sub-part is the set of teeth of a saw, as shown in Figure 4.13.

It is worth noting that the parameters that control the sub-part extraction process can be easily modified in order to fine tune the process. The parameters used for the system allow enough variability in order to extract most distinctive sub-parts. The identification of additional sub-parts does not in general hinder the recognition capabilities of the system.

The sub-part extraction process consists of one application of rule #1 above, followed by recursive applications of rules #2 through #4 until no more sub-parts can be generated. In this manner all intermediate sub-parts are further decomposed into their smallest components and the intermediate sub-parts do not become part of the final representation. The output representation consists of a graph of sub-parts in which the nodes are the feature representation of the sub-parts and the arcs specify how the sub-parts are connected. The connection information consists of the connecting primitives, if any, and the relative transformation of the sub-parts. The sub-part transformation is composed of the translation, rotation, and scaling factors that the sub-parts may have relative to each other. Currently, global parameters are used to define transformation tolerance levels for all sub-part relations, but local ones can easily be defined. These parameters specify the amount of variability to allow between instances of the sub-parts. Variability may be required due to noise or allowances for actual variation in the sub-parts.

The sub-parts generated for the bike model shown in Figure 4.4 are shown in Figures 4.14 and 4.15. The connection primitives connecting the front and back of the bike as well as the ones separating the top pedal from the frame are shown with both of their

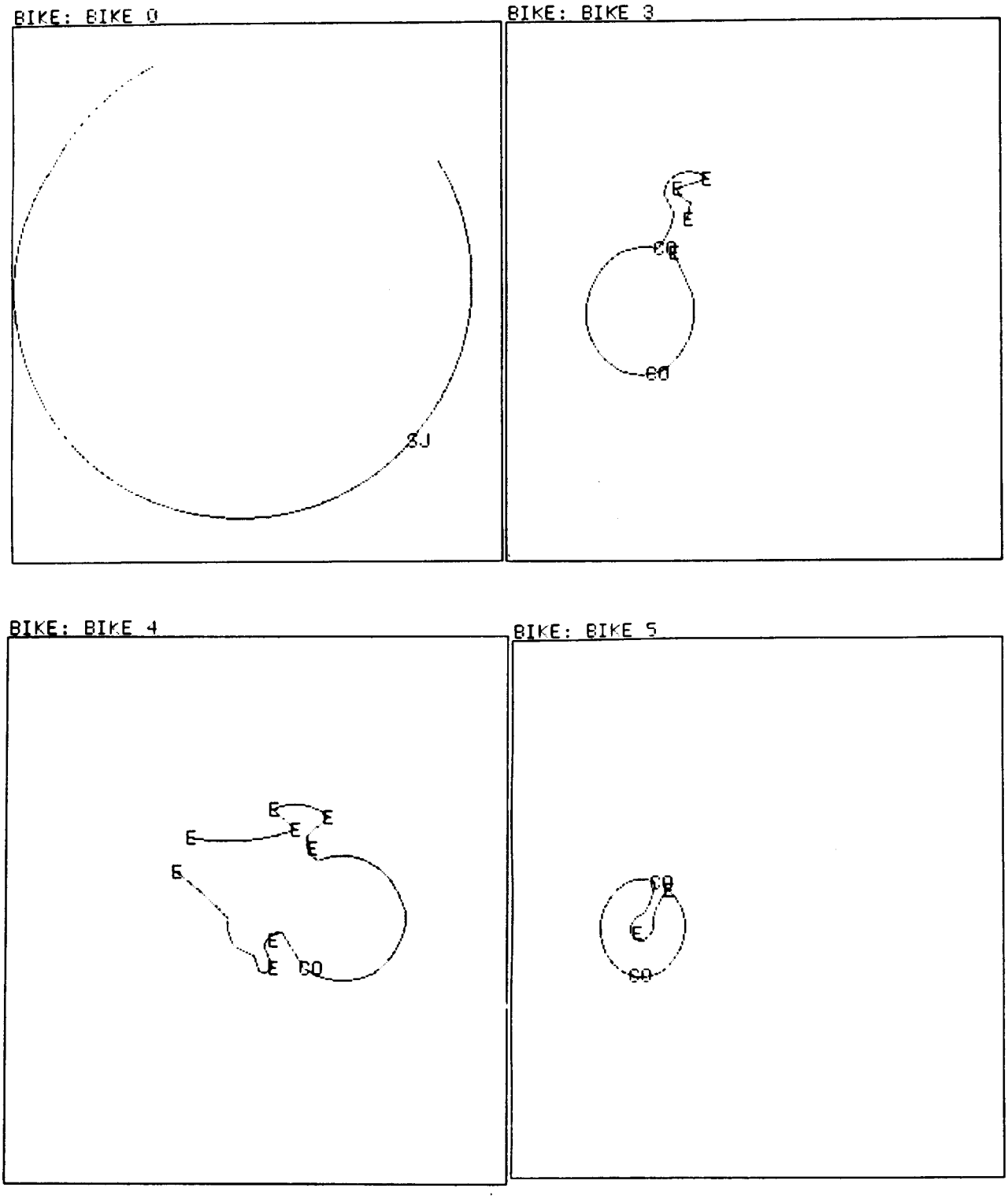


Figure 4.14: **First of two figures showing the sub-parts generated for the bike model.**

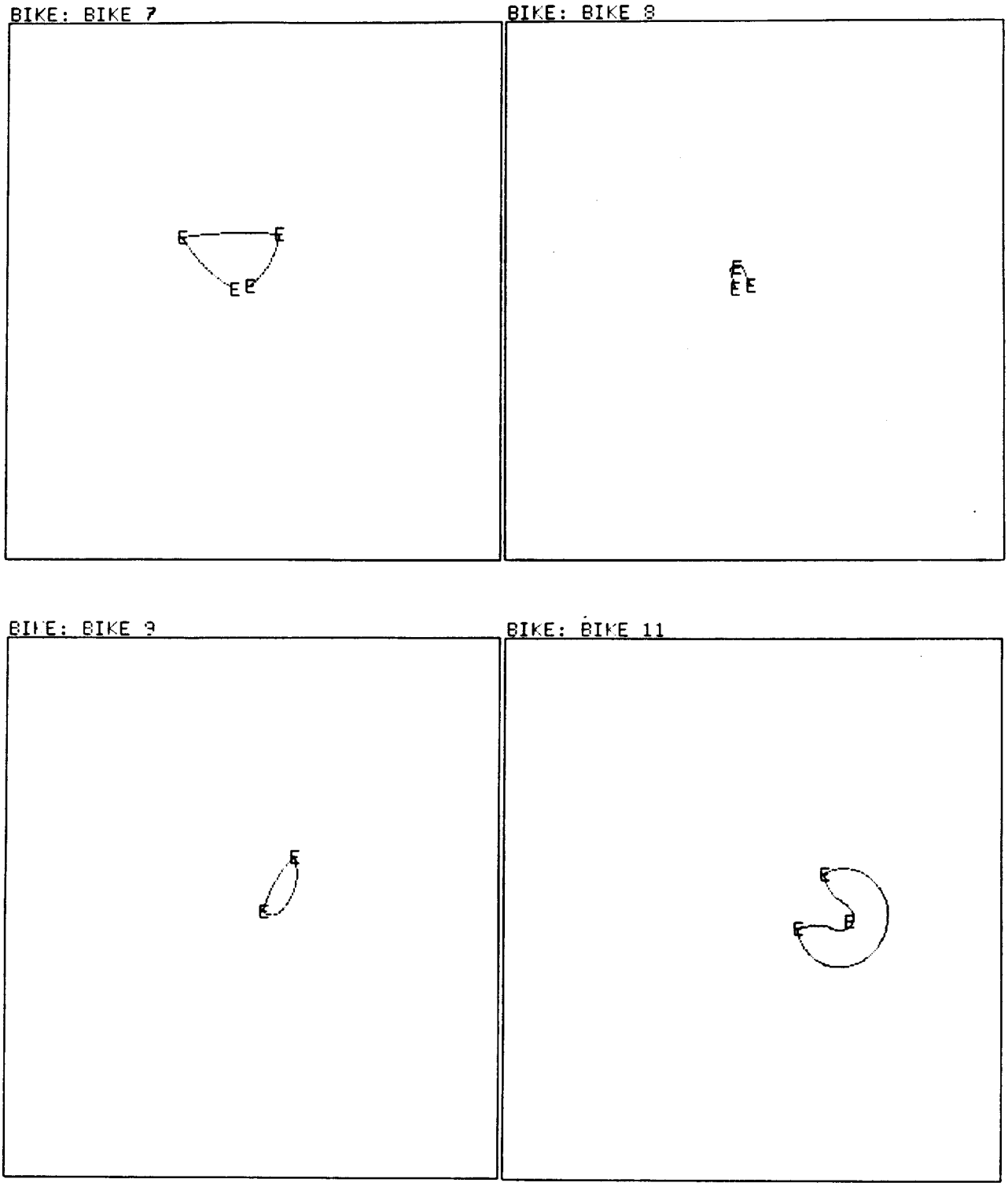


Figure 4.15: Second of two figures showing the sub-parts generated for the bike model.

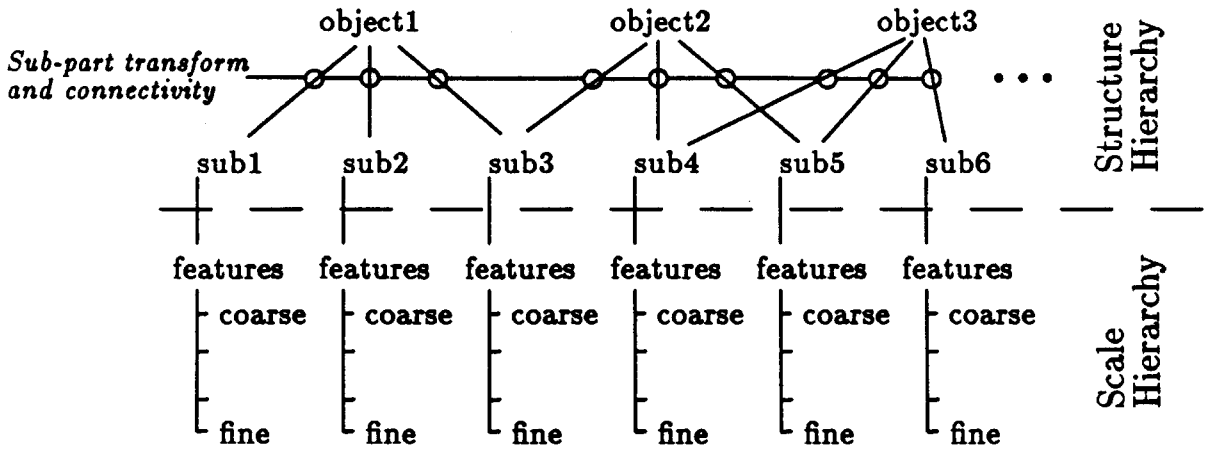


Figure 4.16: **Organization of the model library. The library is indexed through the sub-part level.**

respective sub-parts even though they are actually considered as separate components. The bottom pedal was not extracted since the corners defining it were not significant enough. Overall, the sub-part decomposition achieves an appealing and intuitive subdivision of the object.

Once the model sub-parts are generated, the model is added to the library using the sub-parts as indexing keys. The system attempts to match each of the new sub-parts to existing sub-parts as well as to each other. Any sub-parts that are found to match are consolidated by sharing a single instance of it. As noted earlier (Section 3.5) the shape of shared sub-parts is not averaged in any way due to a lack of enough information to perform such abstraction correctly. Instead, tight parameters are used to compare the library sub-parts to each other in order to ensure that the sub-parts have adequate similarities. By using tighter parameters for updating the library than for recognizing components in the scene, we guarantee that the sub-parts are not generalized too much for recognition tasks that might require discrimination between two components that have been consolidated. The algorithm used to match sub-parts is actually the same one used in the recognition engine, as described in the following section (4.3).

An example of the library organization is shown in Figure 4.16. The pointers between the objects and their sub-parts are bidirectional in order to allow indexing through the sub-part level. Once a sub-part is recognized the system may find the objects that contain that sub-part and then go down the hierarchy to predict other sub-parts needed

to confirm an interpretation. The recognition of the individual components uses the scale hierarchy in order to take advantage of the coarse to fine nature of the description. The two hierarchies are again organized along two separate dimensions.

4.3 Recognition Engine

The goal of the recognition engine is to take full advantage of the rich description of the representation in order to achieve correct and efficient interpretation of scenes based on the knowledge contained in a large model library. The recognition engine takes as input the model library and the feature level description of a scene and outputs all consistent scene interpretations—configurations of object identity and transformation. The SAPPHIRE recognition engine combines facets of the hypothesis-prediction-verification scheme with the constrained search technique. Hypotheses and predictions are made at the level of sub-parts, while the actual recognition as well as the verification steps are accomplished using the scale hierarchy of the sub-part features. This combination of scene-driven and model-driven modules is supported in Section 3.6.1.

The description of the recognition engine is shown in Figure 4.17. Each of the main modules are described below. The Hough transform (Section 4.3.1) is used to generate an ordered set of likely sub-parts based on the features in the scene and the library. The system then attempts to recognize these sub-parts in the scene (Section 4.3.2). Once one or more sub-parts are identified and localized, the system is able to predict configurations of other sub-parts in order to constrain the list of possible model objects and to verify the configurations of identified objects (Section 4.3.6).

4.3.1 Hypothesis Step

The hypothesis step forms a crucial module of the recognition engine since it generates the seeds from which the final interpretations are derived. It acts as a preprocessor that *directs* the recognition process and is therefore not intended to perform the actual recognition. It is the only module of the system, though, that gets a chance to survey the complete knowledge base in the library since after this module, the recognition focuses only on the hypothesized components.

The Hough transform is used in this module since it can achieve the following goals of the hypothesis module (defined in Section 3.6.1): conservative behavior, good fore-

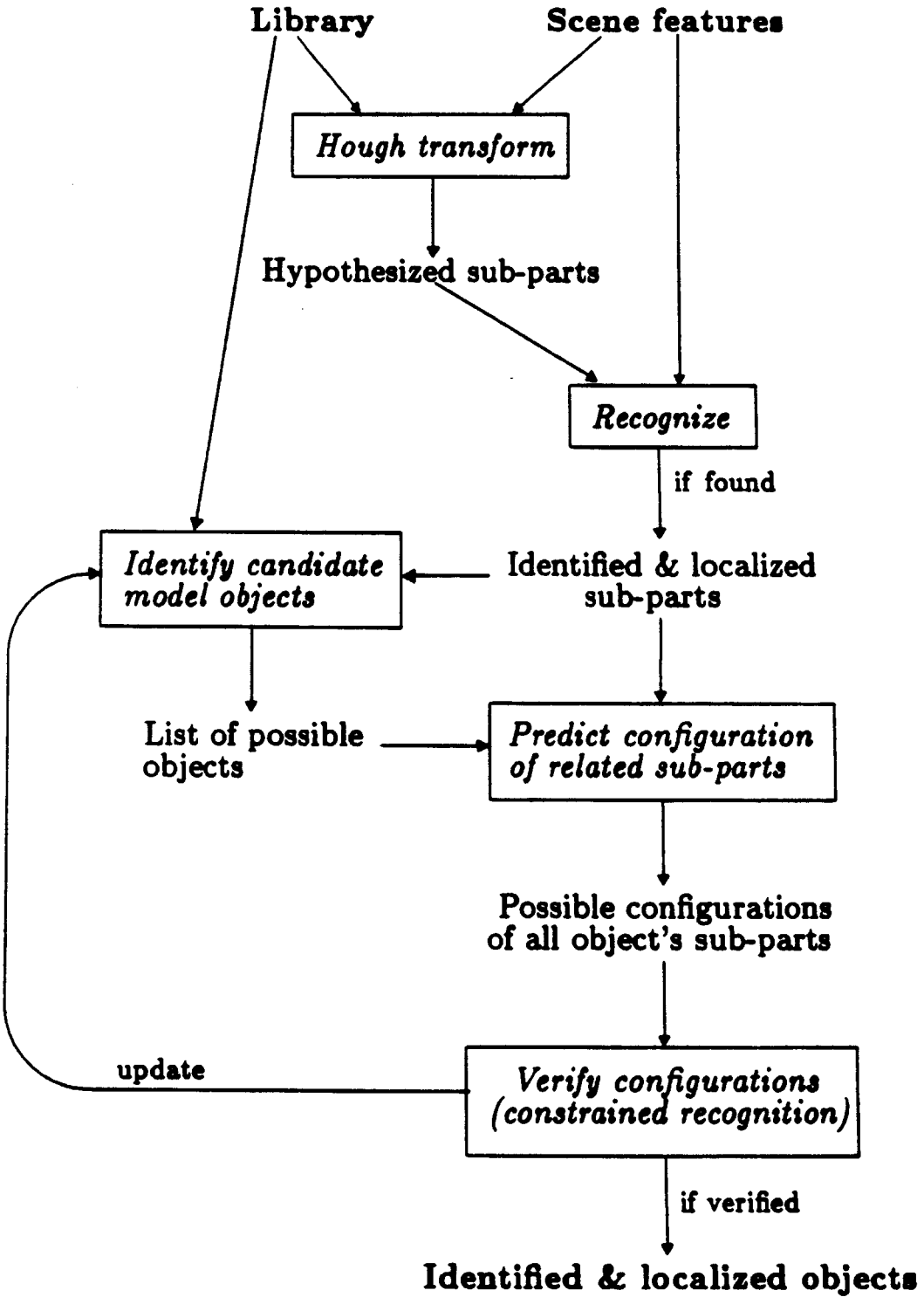


Figure 4.17: Control structure of the recognition engine.

sight, and efficiency. The method used to implement the Hough transform in order to accomplish these goals is described below.

In the SAPPHIRE system, the Hough transform is used to only predict the identity of probable sub-parts. It is intended to be a coarse filter since most of the benefits of the system are derived from the recognition algorithm. SAPPHIRE is therefore not dependent on the Hough making very accurate hypotheses. The Hough transform is not used to predict the transformation of probable interpretations as is done in many applications of the Hough. The transformation is not hypothesized since it consists of too many parameters to be predicted accurately: scaling and mirror image in addition to rotation and translation. Most schemes which do hypothesize the transformation assume known scaling factors and do not consider mirror images. These assumptions greatly simplify the hypothesizing of rotation and translation values.

The Hough transform is performed by counting votes for each model sub-part in the library based on compatible features found in the scene. Each scene feature is checked against each feature of each sub-part. If the model and scene features are found to be compatible, then the model sub-part receives a vote. Only the coarsest type I features (corners, ends, and cranks) for both scene and sub-parts are used. The compatibility criteria used are feature type and angle. This measure of compatibility is the same one used for the recognition algorithm when performing the actual matching. The votes are normalized by dividing the vote total by the total number of coarse type I features on each sub-part. An important effect of the Hough is to remove invalid model components from consideration—if no compatible scene features are found for many features of a model sub-part, there is no need to attempt to recognize that sub-part in the scene. Such sub-parts are not hypothesized since they receive a low vote score.

The sub-parts with high normalized vote scores are further processed by some heuristics in order to better identify the candidates to be recognized. The sub-parts selected must have received votes for enough model features to meet the matching threshold requirements of the recognition algorithm. This measure can be improved by employing symmetry analysis to account for similar features on the same sub-part and reach better normalization of votes.

Since we have high confidence that the sub-parts with the higher vote scores will be recognized, we would like to favor those high scoring sub-parts that are contained in very few objects. Recognition of these sub-parts will thus substantially reduce the number of candidate model objects to be considered. Therefore, for sub-parts receiving

high vote scores, their degree of shareability is used to reorder them.

Larger sub-parts (in terms of the number of features) are also favored over smaller ones in order to pre-arbitrate between matching conflicts. Such a potential conflict may occur if two sub-parts may be recognizable in a scene using some of the same scene features. In that circumstance we must decide which of the two sub-parts should match those features since scene features generally belong to only one object. The larger sub-part is preferred since it conveys more information and it is more likely that the scene feature configuration matching the smaller sub-part was accidental rather than the one matching the larger sub-part. This decision is made at the hypothesis step since we would also like the recognizer to match as many scene features as possible early in the recognition process in order to remove them from consideration if they contribute to an object match. The arbitration could be performed later if we do not remove from consideration any scene features that contributed to object matches, and run a post-processing step to arbitrate among any conflicting object configurations. Such conflicting object configurations, which share some of the same scene features, are found to occur infrequently, and when they do, the larger object tends to be the correct interpretation. Therefore, SAPPHIRE is implemented with the efficiency benefit of removing scene features matched in model object configurations, and with ranking larger sub-parts higher than smaller ones for components that received close vote scores.

This preprocessing method is sufficient to generate a rough ranking of candidate sub-parts, which is the required task of the hypothesis step. It also guarantees that correct hypotheses are not pruned from its list of sub-parts by only hypothesizing the sub-part identity (and not its transformation) and by conservative vote gathering. Since it does not perform much computation, the Hough transform is efficient—proportional to $\beta\gamma$, where β is the number of scene features and γ is the total number of features for all the objects in the library. This number is small compared to the exponential size of the search spaces encountered for the recognition step. Since the number of sub-parts generated is small, the time of the sorting processing applied to them is negligible.

Some other interesting heuristics can also be used to aid in hypotheses generation and should be explored. One idea is to histogram the sub-part rotation when features match and to prioritize sub-parts that receive many compatible feature votes as well as clusters of similar rotations. The rotation parameter of the transformation is computable since features are described by their orientation. Translation and scaling factors cannot be computed easily from single feature matches. The rotation values can then be used

to constrain the recognition of the selected sub-parts. Complications arise, though, in robust recognition cases. Mirror image instances of objects in the scene complicate the rotation voting scheme since feature matches will not contribute to the same rotation any more. Another drawback is that the rotation vote is also susceptible to noise. Noise sensitivity becomes an even more problematic issue if we attempt to hypothesize more parameters, such as translation and scaling factors. As the number of parameters grows, noise tends to smear out the histogram since the parameters are dependent on each other. As a result, it becomes difficult to differentiate between probable and improbable hypotheses.

Another hypothesis heuristic is to cluster several high scoring sub-parts into a single hypothesis. If we can find several sub-parts that receive a high vote total and that belong to the same model object, the likelihood of them all matching is high. We can then modify the recognition engine to recognize several sub-parts in the scene, rather than just one, before making predictions as to the configuration of related sub-parts.

4.3.2 Recognition Algorithm

Given the ordered list of sub-parts, the system will attempt to recognize each of them in the scene. This task consists of matching scene to model[1] features (or vice versa) in such a manner that the object identity, rotation, translation, scaling factor (the viewed scene can contain smaller or larger instances of the models), and object reversal (mirror image) flag can be determined. The recognition should be performed hierarchically in a coarse to fine manner taking advantage of the scale hierarchy of the representation. Furthermore, the rich descriptions of the primitives as well as their spatial layout should be exploited in deriving the interpretations.

One possible recognition algorithm is *graph matching.* A model component can be represented as a graph consisting of its features at the nodes and the spatial relationships of the features along the arcs. We can then attempt to match sub-graphs of the model graph to sub-graphs of a suitable graph representation of the scene. A drawback of this approach is the problem of inexact matching [Binford 82]. Due to occlusion, noise, and variations in the object the graphs to be matched will contain many differences. Some model features will be missing in the scene and the scene will introduce additional features not present in the model graph, both due to occlusion. Since the graph is

[1]The models taken by the recognition algorithm as input are the individual sub-parts.

generally syntactic, it is an awkward mechanism for dealing with these variations. The scene graph would need many additional arcs to specify sub-part relations in order to compensate for model sub-parts being occluded. Spurious components in the scene graph would also have to be tolerated by increasing the connectivity of the model graph. As a result, the problem becomes needlessly complicated. And the sub-graph isomorphism problem is complex to begin with; the general problem is NP-complete.

Another problem with graph matching is that it requires the model and scene graphs to be equivalently expressed, as in [Connell 85]. Thus, in order to recognize complete model objects, model graphs of their component sub-parts must be matched against appropriate scene graphs. This requirement prevents graphs from being used for the recognition task since the scene would need to be segmented into its component sub-parts prior to matching. We would like to avoid this segmentation problem since it is often difficult as well as inefficient to segment the scene correctly and any segmentation errors (even small ones) can lead to recognition errors. The system would become too dependent on this segmentation process which sometimes cannot be performed completely due to occlusion and noise. The recognition would fail, for example, if the concavities used to decompose the sub-parts were occluded in the scene.

A more natural way to structure the recognition algorithm is based on the constrained search scheme used by Grimson and Lozano-Pérez to match edge segments and surface patches [Grimson 84] [Grimson 85]. The SAPPHIRE system adapts this recognition algorithm to be used with the CPS based representation. The CPS representation improves the combinatorics since it enforces a one-to-one matching that is not used by Grimson and Lozano-Pérez (see Section 3.2.1).

The recognition problem is structured as a search through an interpretation tree consisting of all possible matching configurations of scene to model component features. The first level of the tree has nodes corresponding to the assignment of the first model feature to each scene feature. The second level expands the first level nodes by enumerating all possibilities for the assignment of the second model feature to scene features given the assignment in the parent node.[2] In order to allow for occlusion, model features may also be assigned to the null-features, a matching that indicates a configuration where that feature cannot be identified. This scheme increases the branching factor of

[2]The interpretation tree can also be structured by assigning model features to a scene feature at each level (as done by Grimson and Lozano-Pérez), but since the scene will usually have more features than the model, the model-guided approach is preferred in order to reduce the size of the tree.

the interpretation tree by one. The expansion of the tree continues until all model primitives have been assigned.

4.3.3 Geometric Consistency

In order to prune inconsistent configurations, and avoid generating whole sub-trees, geometric constraints are applied to the configurations. The system compares relationships between pairs of scene features with the relationships between the assigned pair of model features. If the scene and model values do not match for any pair of features, that configuration can be pruned and none of its descending nodes need to be explored. The advantage of using pairwise constraints is that they are coordinate-frame independent and thus measurements in model coordinates can be directly compared to measurements taken from the scene. The constraints used here are also simple and therefore do not require any complex constraint management system. The constraints used by SAPPHIRE consist of the following three measures between pairs of features:

- **Orientation difference** — the angle formed by the orientation vectors of the two features.
- **Distance** — The distance from one feature to the other.
- **Direction** — The angle of the vector drawn from one feature to the other. The angle is formed by that vector and the feature orientation vector.

In the 2D domain these parameters completely constrain the attitude of one feature relative to another since three parameters are needed to specify the three degrees of freedom (two translational and one rotational). The parameters can be transformed into other forms, but with the same constraint abilities—such as the orientation difference and the two components of the vector from one feature to the other. In the 3D domain, the constraints yield four parameters: orientation difference, distance, and angle of direction vector (which requires two parameters to specify in 3D). Therefore, the constraints do not completely constrain the attitude of features relative to each other in 3D, which requires five parameters. Grimson shows that these constraints (in both 2D and 3D), though, are complete in the sense that no additional geometric constraints are possible [Grimson 86].

The constraints between all model features can be computed off-line and used for all recognition runs. Therefore, SAPPHIRE only computes the relationships among

scene features that are being matched for comparison to the model constraints. The recognition process starts off by attempting to match the coarsest level features of the given sub-parts to the scene features. At the end of this processing, the system will be left with at least one matching configuration if the sub-part is actually in the scene. Subsequent levels of processing match the next three levels of the scale hierarchy in order to discriminate between the remaining configurations as well as localize them better by matching the finer features. All the final configurations are returned as interpretations of the sub-part in the scene.

The final interpretations must match a minimum percentage of the model component's features in order to be declared a valid match. The threshold value typically used is 50% of the sub-part's type I features. This value can be modified depending on the type of recognition desired. The system can even be designed to set the threshold itself by analyzing the similarity of the sub-parts of the library. If the sub-parts are very similar to each other, the threshold value can be increased in order to differentiate between them; as the similarity of the library sub-parts decreases, the threshold can be decreased.

The feature match is performed on a binary basis, as described in Section 3.6.4. Tolerance values are defined for the different constraints in order to account for variations such as sensor noise and distortion, scaling effects, discretization errors, or occlusion. Therefore, if model features M1 and M2 are matched to scene features S1 and S2, respectively, the system would ensure that in order for the matches to be consistent, the following inequalities hold (ω = orientation difference tolerance, χ = distance tolerance, and ϕ = direction tolerance):

- $|$ Orientation-difference(S1,S2) - Orientation-difference(M1,M2) $| \leq \omega$
- $|$ Distance(S1,S2) - Distance(M1,M2) $| \leq \chi$
- $|$ Direction(S1,S2) - Direction(M1,M2) $| \leq \phi$

Due to the nature of the recognition process, all the correct matches[3] are guaranteed (within the bounds of the allowed noise and spatial variations) to be included in the final set. This behavior is due to the consideration of the whole search space in the interpretation tree and the pruning of all invalid configurations, leaving behind only the

[3] *Correct* matches are defined as the best configurations in terms of the percent of type I model primitives matched in the scene.

valid ones. This behavior is desired since the low level features usually do not contain enough shape information to make *educated hypotheses*. As a result, we like to have this low level completeness guarantee. The hypotheses are relegated to the more descriptive sub-part level where the hypotheses can be supported by much more information.

In performing the matching process, two heuristics are used to improve its efficiency: lookahead pruning and identification of congruent configurations. Lookahead pruning is used to remove configurations from further considerations if it can be determined early on that they will not yield valid interpretations. The final configurations must have a minimum percentage of the sub-part's features, as specified above. Therefore, we can compute ahead of time the maximum number of null-feature matches that a configuration may have. Any time that this maximum is exceeded we can stop expanding that branch of the interpretation tree. Using this heuristic, many of the invalid configurations are pruned early rather than filtered at the end.

The identification of congruent configurations arises from the use of breadth-first search in expanding the interpretation tree. A common result after a few levels of processing is that many of the consistent configurations are actually derived from the same interpretation. For example, two configurations could have all the same primitive assignments except one which assigns a model features to the null-feature. Since these configurations actually correspond to the same interpretations, we would only like to keep one of them. The way congruent configurations are identified by the system is by computing the sub-part/scene transformation for each configuration. As shown below in Section 4.3.5, this computation is efficient by virtue of the relatively small number of features in a sub-part and by restricting the number of consistent configurations. By comparing these transformations, the congruent configurations are identified and only the better ones, in terms of the number of type I features matched, are kept. Tight tolerances are used to compare the transformations so that only configurations that are very similar to each other become candidates for pruning.

4.3.4 Feature Compatibility

Since the features are described by different types, the system does not actually explore the assignment of all scene features to model features. Only features that are judged to be compatible are matched. The system thus achieves two forms of constraint: feature compatibility and geometric consistency. Compatibility is defined by a set of rules that compare the scale, type, and angle of the features. One rule incorporates the

Feature 1	Feature 2	Tolerance (degrees)
End	End	50
Crank	Crank	40
Corner	Corner	30
Bump	Bump	40
Smooth Join	Smooth Join	40
Inflection	Inflection	40
End	Corner	45
Corner	Crank	20
Corner	Smooth Join	20
Crank	Inflection	20
Smooth Join	Inflection	20

Table 4.1: **Feature matching rules. Only features of the given types and with angles differences falling within the given tolerance values are judged to be compatible.**

scale hierarchy of the representation. The recognition already exhibits a hierarchical nature by only matching features of the same relative coarseness levels. In order to handle possible ambiguity in the assignment of scale to the feature, some cross-scale matching is incorporated. After the processing of each level of the scale hierarchy, the system attempts to match any remaining unmatched model features in all consistent configurations to scene features in the immediately coarser and finer levels. Unmatched scene features are similarly processed. As a result, features that are on the borderline between two scales, and may be classified in one scale in the model and in the other scale in the scene, may still be matched. But since these features are more ambiguous than features for which the scale is better determined, they are not used to drive the recognition.

The remaining compatibility rules compare the class and angle properties of features. Since exact matches are not likely, the rules define the allowed variability in the feature types and feature angles. The rules allow variations in the primitive specification to be gracefully tolerated. The compatible primitive types are shown in Table 4.1. The associated tolerance value indicates the difference in the angles of the model and scene features that is deemed acceptable for compatibility.

The tolerance values are higher for primitives of the same type since they have the same appearance and more fluctuation in their angles can be tolerated. Features of different types look somewhat different locally, so their edges should be more aligned to compensate for the difference in appearance. Angles of corners are computed relatively accurately, resulting in a lower tolerance for corner-corner matches than for other intra-class matches. The tolerance values for end matching are higher than for other features in order to allow for scaling variability. The angle of a two-corner end tends to vary somewhat as it is viewed more coarsely. It can appear to be a sharp end at coarse scales and as a result the angle at the end point will be somewhat different than the angle of the finely-viewed flat two-corner end. These effects can also manifest themselves for scaled instances of the same object and so the tolerance values for end-end and end-corner matches are increased. Due to this problem, the tolerance value (given in the table) for matches consisting of one compound feature and one simple feature is multiplied by an additional factor of 1.25. In addition to these rules, bumps are allowed to match ends and corners if both primitives in the matched pair are convex or concave. Actual angles are not used for these matches since the edges of the bump do not correspond to the edges of the matched end or corner.

Other parameters can also be used for the compatible matching rules. One such parameter is the displacement value of compound primitives. Matching displacements, though, requires the ability to parameterize them by the scaling factor involved in transforming the model to its instance in the scene. The problem is that in order to decide if the displacements are compatible, the scaling factor is needed, but the final scaling factor is derived from the configuration of all the matched primitives. The scaling factor is therefore not available when features are still being tested for compatibility. As a result, displacement values can be checked only at the completion of the matching phase, when they can be scaled appropriately and the displacements can be checked for consistency.

4.3.5 Computing Model/Scene Transformations

For each interpretation, a transformation is defined that transforms the model sub-part from model coordinates to scene coordinates. The transformation vector consists of mirror image flag (to indicate if the object in the scene is the mirror image of the model), rotation, scaling factor, and translation. Since these variables depend on each other, they are defined to apply in the order given. Thus the model is first flipped, if

necessary, then rotated to the orientation in the scene, rescaled, and finally, translated to its position in the scene. The mirror image flag and scaling factor are derived from the application of the geometric constraints. The orientation difference and distance constraints do not change for mirror images, but the sign of the angle of the direction vector reverses. Thus, if the direction constraint for a configuration fails, the system can check if the mirror image is consistent. Similarly, the scaling factor is derived from the application of the distance constraint. Instead of comparing the pairwise feature distances in the scene directly to the model, a scaling factor is used to parameterize those distance measurements. In order to be consistent, all the inter-feature scene distances in a configuration must be consistent with a scaling factor times the distances between the assigned model features.

The rotation and translation components are computed by comparing absolute measurements of the model and scene features. The rotation is computed by the average difference in the orientations of assigned model/scene features. After the model is transformed by the mirror image flag, rotation, and scaling factor, the translation is computed by averaging the translating vectors required to relocate each model feature to the location of its matched feature in the scene.

4.3.6 Prediction and Verification

Once a sub-part has been identified and localized in the scene, that knowledge can be used to simplify the recognition of related sub-parts. By using the library, all the objects that contain the identified sub-part can be selected as possible candidate model objects to explore. The system can then predict the identity and transformation of related sub-parts that will reduce the list of model object candidates. As these predictions are verified, the list of candidates is consolidated. The system continues predicting and verifying instances of the model components in an effort to recognize all parts of each remaining candidate object. If enough components of a model object have been recognized, the resultant component configuration is returned as a scene interpretation. The minimum number of components needed to match is specified as a percentage of the *distinguishable components* of the models. Distinguishable sub-parts are defined as having at least three coarse features, as defined in Section 4.2. The percentage of sub-parts needed to be recognized is usually set at more than half of the distinguishable sub-parts (at least 51%). With high minimum percentages, we achieve a tight inspection type of recognition behavior. With lower minimum percentages, we achieve a higher

degree of inexact matching.

Various heuristics can be used to select the sub-parts to be predicted and verified. Since the hypothesized sub-parts are selected to have a low degree of shareability in the library, the number of candidate objects will usually be low. The straightforward way to process them is just to attempt to recognize each of their components separately and then select the best overall component configuration if any of them satisfy the minimum number of identified components criterion. The best model object is judged to be the one that matched the most sub-parts. Other parameters can certainly be used in evaluating the best model match, such as the number of features matched or the measure of the match of model contour segments to scene segments. The selection of sub-parts to predict can also be determined by finding common sub-parts with common component relationships among the candidate objects. These sub-parts can then be predicted first since, if they are verified, they will reduce the size of the candidates list, converging on the best model match.

In predicting sub-part configurations, the relative sub-part relationships specified in the model library are used to compute the predicted transformation. Tolerance parameters are then added around these transformation values in order to define a range of rotations, scaling factors, and translations to use for verification. The mirror image flag for the recognition of all the sub-parts of the object is determined by the mirror image flag of the recognized hypothesized sub-part. The same recognition algorithm, described in Section 4.3.2, is used for verification as is used in performing the original sub-part recognition. For verification, though, the transformation is greatly constrained so only configuration nodes whose transformations fall within the specified ranges are expanded. The verification step is therefore much faster than an un-informed recognition of the sub-part. If a predicted component is verified, its exact transformation is also determined as a direct result of the constrained recognition processing. In addition to verification of the transformation, the connection of the sub-parts is also checked. The location of the connection should not be substantially modified, i.e. only rotation about the connection is allowed. An attempt is also made to match the model connection primitives in the scene in order to further justify a component configuration. The relative rotation of the two sub-parts is used to modify the angles of the connection corners or cranks.

As predicted sub-parts are verified, a component configuration of the model object is developed. A lookahead pruning heuristic, similar to one used in the recognition

algorithm, is used to remove candidate models from consideration. If enough sub-parts have failed to be recognized so that the minimum number of identified components criterion cannot be met, that model object is pruned.

The current verification scheme is based solely on feature matching. Once an interpretation is generated, additional types of verification may be used to check if the resultant configuration is actually valid. Since we are performing inexact matches and are allowing for noise, we cannot discount the possibility that interpretations may be wrong. Some other forms of verification that may be incorporated into the system include:

- Check the consistency of the location of the actual model contour segments rather than just the features. The contour approximations derived in the CPS processing can be used to ensure that enough of the model contour is accounted for in the scene.

- If the system knows the color of the background, it can verify that no such background points exist where the model object is interpreted to be.

Chapter 5

Examples of Recognition Performance

The SAPPHIRE system was extensively tested using a library of model objects found on traffic signs. This domain was selected since the objects are two-dimensional in nature and they cover a broad range of complexity. Some objects are relatively simple while others contain many detailed features; some objects are very different from each other while others contain many similarities. The tests were carried out to show the following aspects of the recognition system:

1. Automatic construction of a model library based on the models' sub-parts.
2. Recognition of objects in scenes of varying complexity. The scenes may have any of the following properties:
    - Occlusion of objects caused by overlapping objects.
    - Variation of noise in the scene introduced by sensor distortion.
    - Variations in feature specifications introduced by using non-exact instances of the model objects in the scene.
    - Variations in sub-part relationships such as rotation, translation, or scaling.
    - Variations of global parameters such as scaling of the whole object or reversing it in the scene (recognizing the object's mirror image).
3. Efficient recognition in terms of the number of objects in the library and the possible number of configurations that may be explored for any individual object match.

The first two aspects of the recognition system are examined in this chapter. An extensive analysis of the recognition efficiency of the system then follows in Section 6.1.

5.1 Generation of Library

Thirteen model objects were presented to the system for the purpose of constructing the library. These models are shown in Figures 5.1 through 5.3. These figures show the edges extracted from the images of the models. Some noisy edges are present in these images, but, as will be shown in Section 5.2, they do not reduce the recognition abilities of the system. The size (in pixels) of the images shown in these figures as well as in the recognition examples range from about 250x250 to 350x350. The resolution of all images shown in this chapter is about 100 pixels per inch. Due to the nature of the representation, the system cannot recognize objects much smaller than the letters shown for the *parking* word object (Figure 5.3). Since the CPS processor smoothes the contours, small features are not identifiable at the coarse level. As a result, these features are not used to drive the recognition, but to refine interpreted configurations. This behavior is desirable since the larger coarse features should drive the recognition. With increased resolution, though, the system will be able to use the smaller features as coarse primitives and therefore recognize smaller and smaller objects.

The thirteen model objects were processed by the representation processor, sub-part extractor, and library manager to yield 47 sub-parts. The sub-part decomposition of each model object is shown in Appendix A. Several of the sub-parts were found to be common among several objects even though they are of various sizes. These include the arrow head of the *right turn only*, *no left turn*, *straight arrow*, and *U-turn* signs, the outer triangles of the *bend in road* and *junction ahead* signs, the letter *E* in the words *REDUCE* and *SPEED*, and the letter *N* in the words *PARKING* and *NOW*. The only common sub-part that was not identified was the second *E* in *SPEED* which, due the tight library updating tolerance values, was not found to be similar enough to the other *E*. In examining these two letters closely (in their reconstructed form in Figure A.14) it can be seen that some of the features do have different parameters.

An examination of the sub-parts reveals the compactness of the representation as well as its information preserving properties—the sub-parts are shown with their coarse features and the contour approximations that are derived from the knot points of the representation. The representation is thus capable of describing a wide array of complex

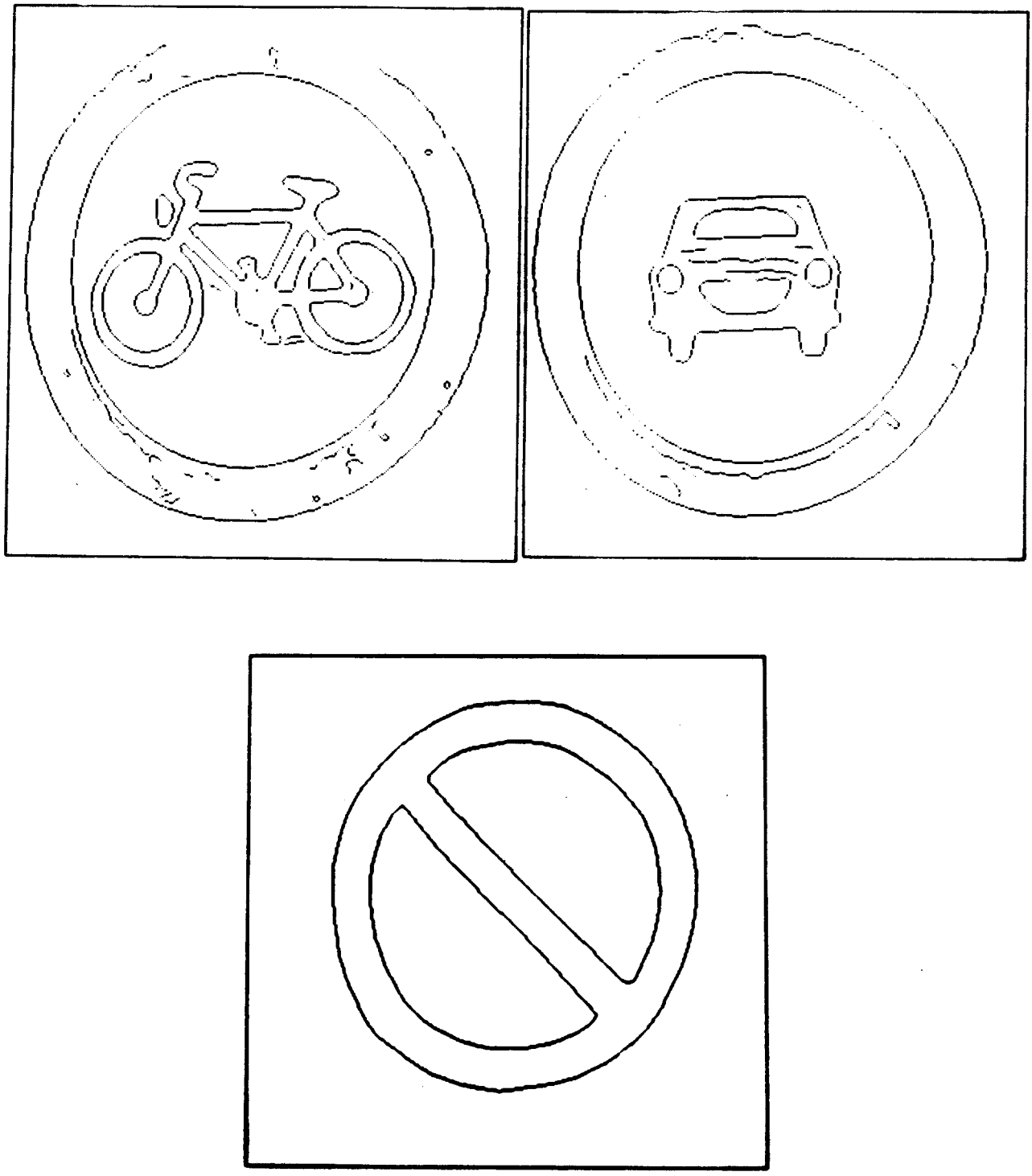


Figure 5.1: **First of three figures showing the model objects in the sample library. These objects consist of: a *bike*, a *car*, and a *disallow* symbol. Output of the edge finder is shown.**

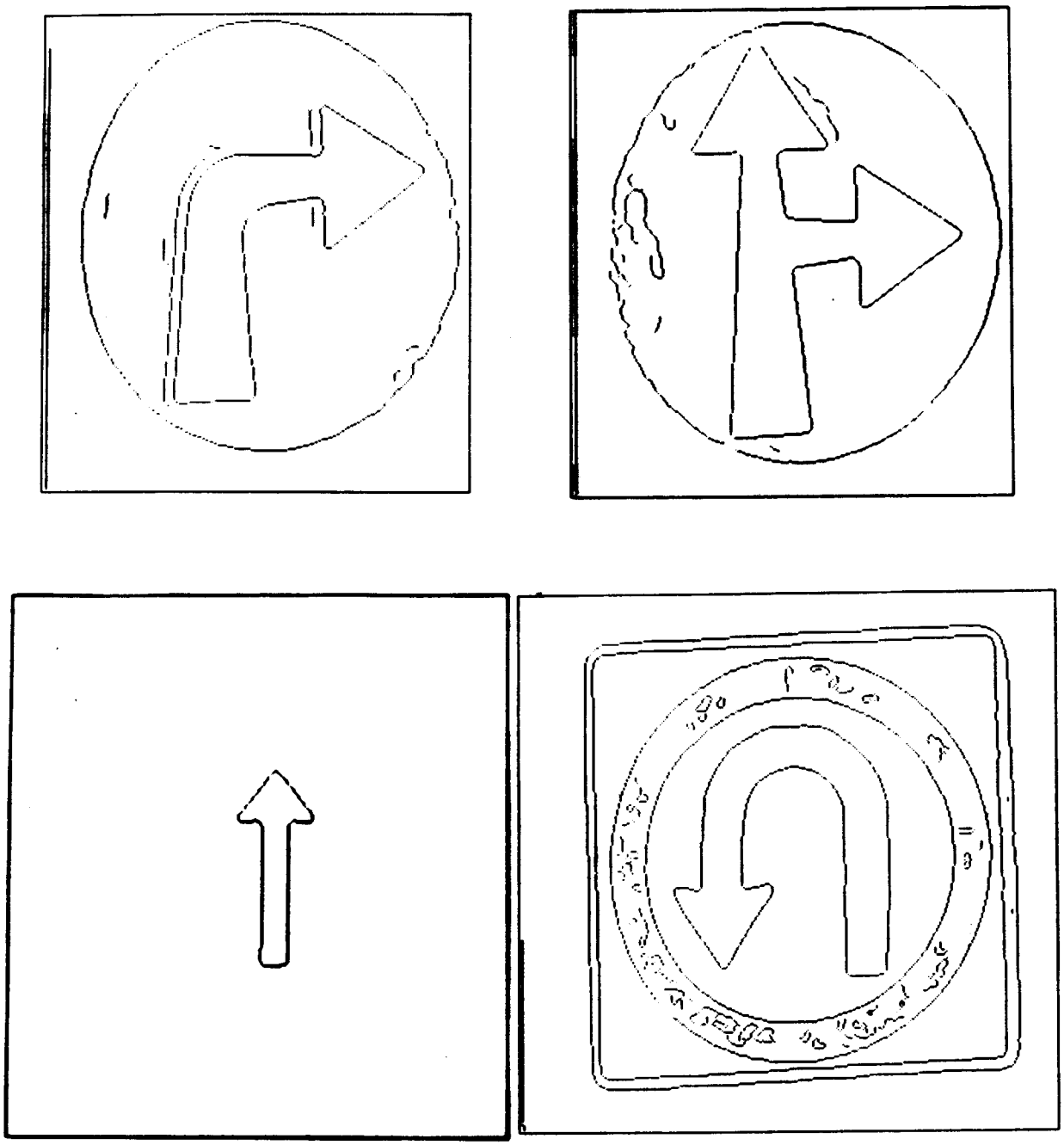


Figure 5.2: **Second of three figures showing the model objects in the sample library. These objects consist of: *right turn only*, *no left turn*, *straight arrow*, and *U-turn*. They all share an *arrow* sub-part. Output of the edge finder is shown.**

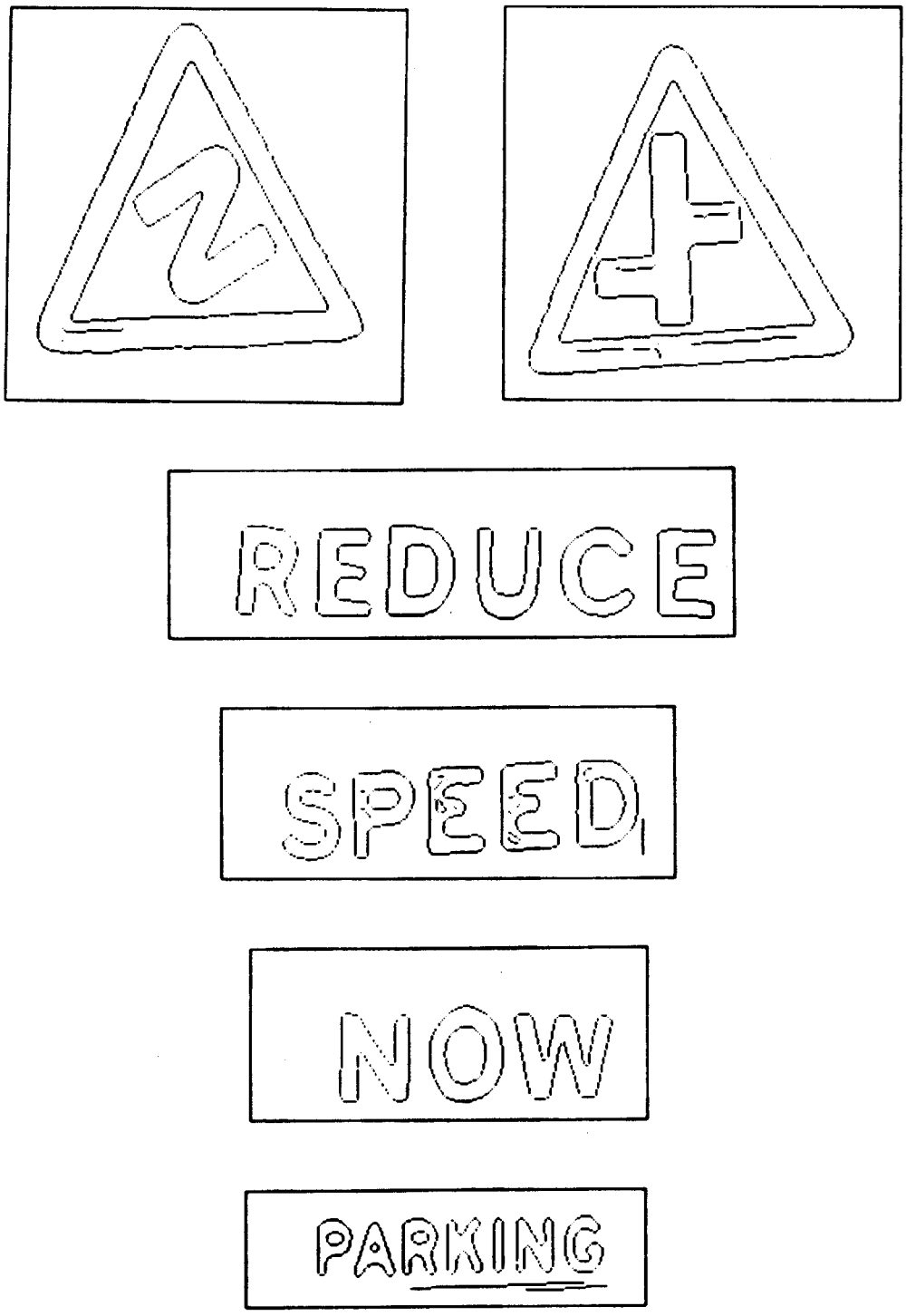


Figure 5.3: **Third of three figures showing the model objects in the sample library. These objects consist of:** *bend in road*, *junction ahead*, **and four word objects (each word is a model). Output of the edge finder is shown.**

objects.

5.2 Some Recognition Examples

Ten different scenes were tested with the traffic sign library. For each one, the component configurations of the identified objects, selected by the system from the complete library, are shown. These figures show the features whose match led to the interpretation of the sub-parts shown in solid lines. The dashed sub-parts are the ones that did not receive enough supporting evidence to be identified, but their location is predicted by averaging the transformation vectors of the identified sub-parts. This average transformation is also used to define a transformation for the whole object. The transformed object is then superimposed on top of the original scene. In the superimposed figures, the contours of model object are shown as dashed lines while the edges of the objects in the scenes are shown as solid lines. The superimposition of the model on the scene is usually not completely overlapped since, in most of the examples, the viewed objects are variations of the models.

The recognition of a *no left turn* sign, which is different than the one in the library, is shown in Figures 1.1 through 1.5 of the Introduction. This example shows the power of using parameterized sub-parts. The spatial layout of the two arrow heads and arrow base is different in the model and scene. But by allowing some translation and scaling variability between the sub-parts, SAPPHIRE is able to identify all three sub-parts in the scene.

Figure 5.4 shows a sample scene of a *no bikes* sign. The bike in that image is not the same bike as the one in the library. As a result, several of its sub-parts are displaced from their positions in the model object. Figure 5.5 shows that SAPPHIRE was still able to recognize three of the four distinguished sub-parts (ones with at least three coarse features) as well as one of the smaller sub-parts. The component configuration shows that even though some of the identified sub-parts are displaced from their position in the model, they are still identified. When the whole model object is superimposed on the scene, Figure 5.6, a good match is achieved. The contours of the two bikes do not overlap exactly due to the differences in the bikes' specifications. The *disallow* symbol was also identified in the scene as shown in Figures 5.7 and 5.8. By identifying these two symbols on the viewed traffic sign, the system can reason that bikes are not allowed

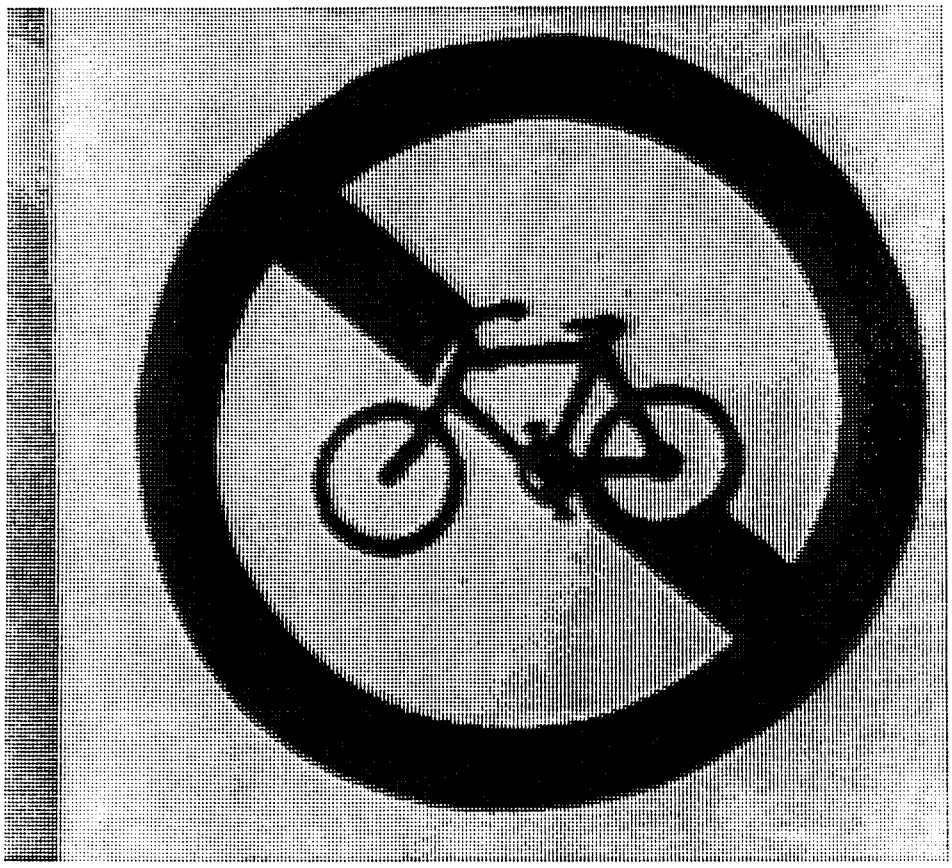


Figure 5.4: **The grey level test image of a *no bikes* sign.**

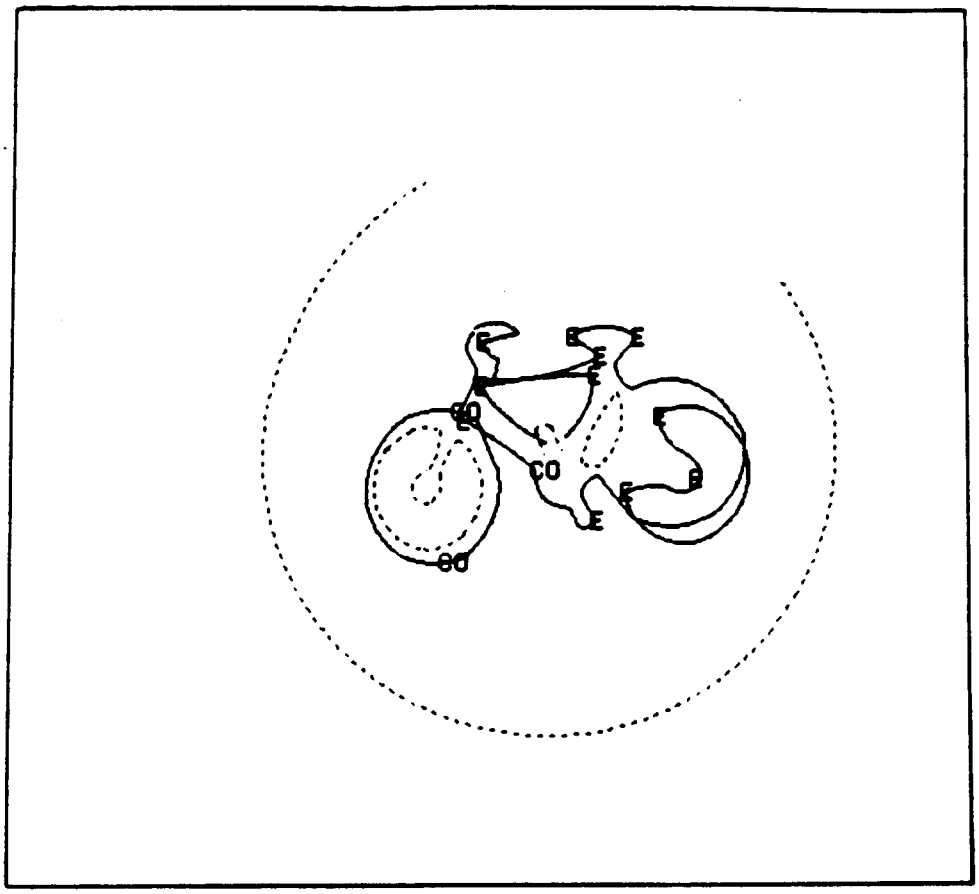


Figure 5.5: **The component configuration of the *bike* model object found in the *no bikes* sign.**

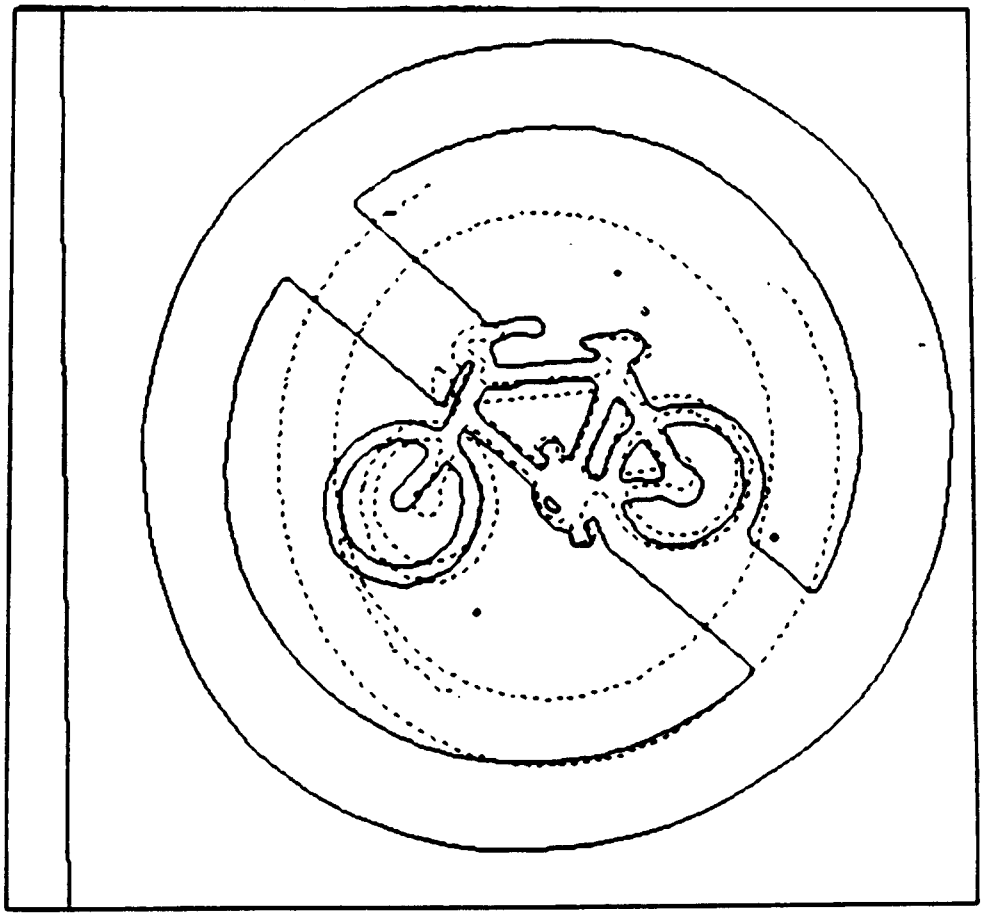


Figure 5.6: **Superimposition of the *bike* model object on the *no bikes* sign.**

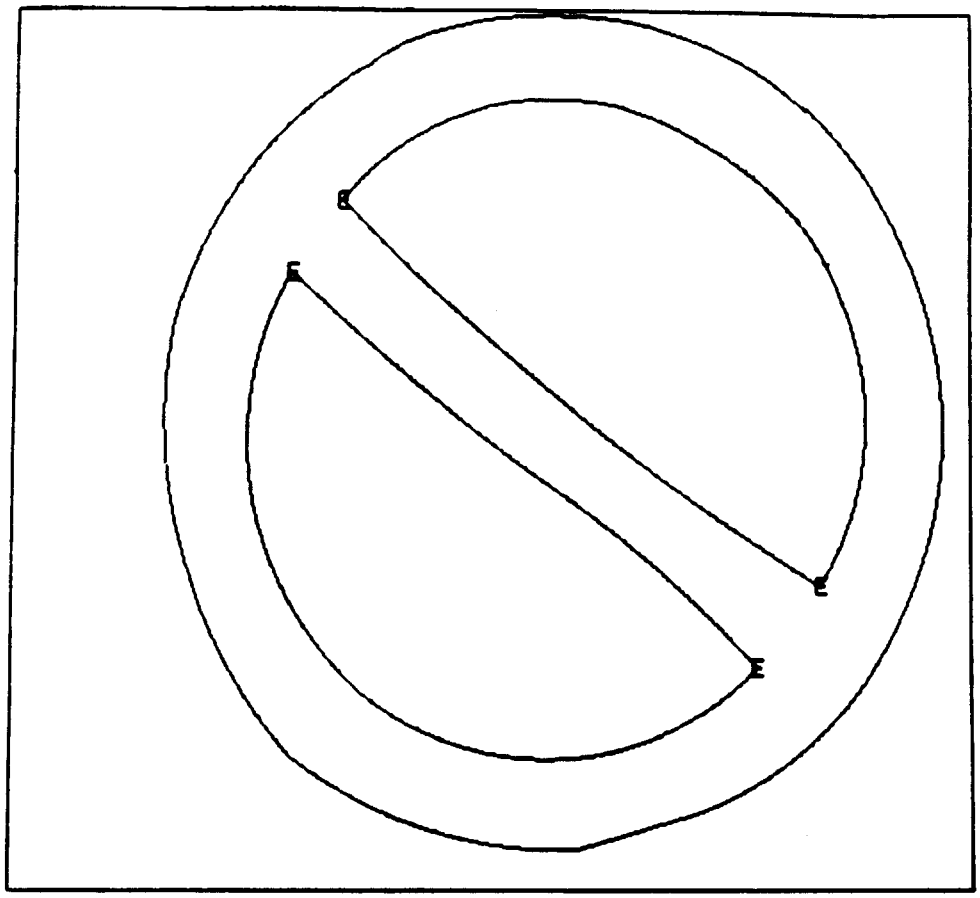


Figure 5.7: **The component configuration of the *disallow* model object found in the *no bikes* sign.**

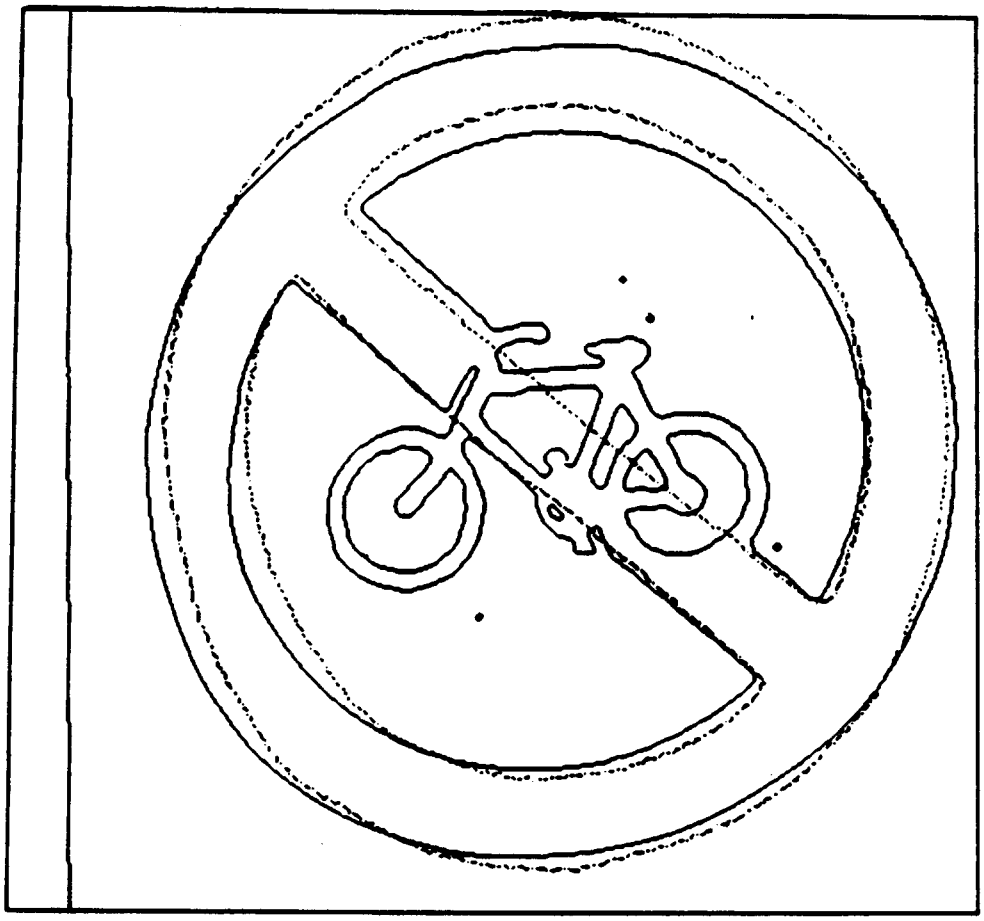


Figure 5.8: **Superimposition of the *disallow* model object on the *no bikes* sign.**

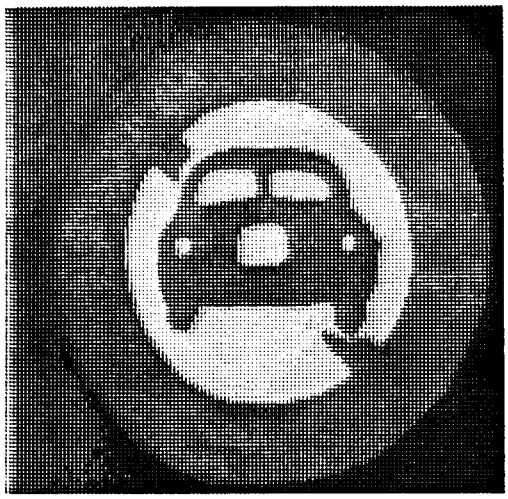


Figure 5.9: **The grey level test image of a *no cars* sign.**

on this road.[1] This example shows that the system works well for complex objects by exploiting the structure and scale hierarchies. By using this approach, the system was able to capture enough of the shape knowledge of bicycles in order to match different instances of that general shape.

Another example of inexact recognition is the *no cars* sign. The car in this image, Figure 5.9 is a different instantiation of the car object in the library. The representation, though, is again able to capture the abstract shape of the car in order to match the two instances. The recognition of the car is shown in Figures 5.10 and 5.11. All features except the left wheel were matched. The interior contours of the windshield, headlights, and the grille were not matched since they did not have enough identifiable features on them. This behavior is due to their small size which also results in these sub-parts being non-significant for the recognition process. The *disallow* symbol was recognized in the scene by virtue of the corners of the symbol being visible, even though most of the slash is occluded. The recognition of the *disallow* symbol is shown in Figures 5.12 and 5.13.

The recognition of the bend in the scene in Figure 5.14 shows the system's tolerance for sensor noise and distortion. Despite the the variations in the two object instances, SAPPHIRE was able to recognize the *bend* model object in the scene, as shown in Figures 5.16 and 5.17. As can be seen in the superimposition figure, the triangles are

[1] Now, all we have to do is teach robots how to ride bicycles.

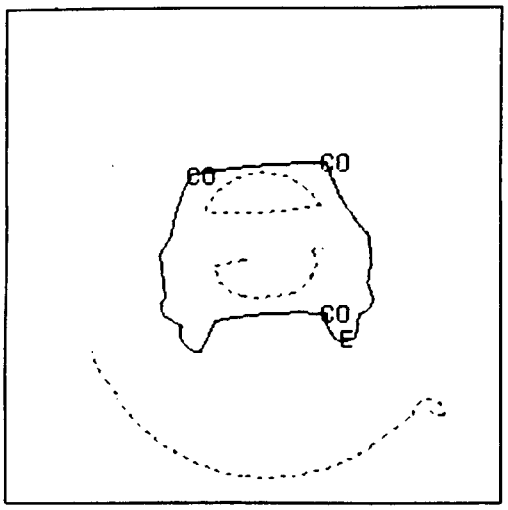


Figure 5.10: **The component configuration of the *car* model object found in the *no cars* sign.**

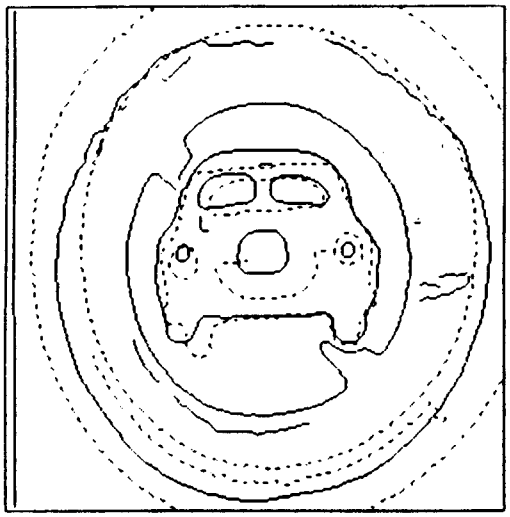


Figure 5.11: **Superimposition of the *car* model object on the *no cars* sign.**

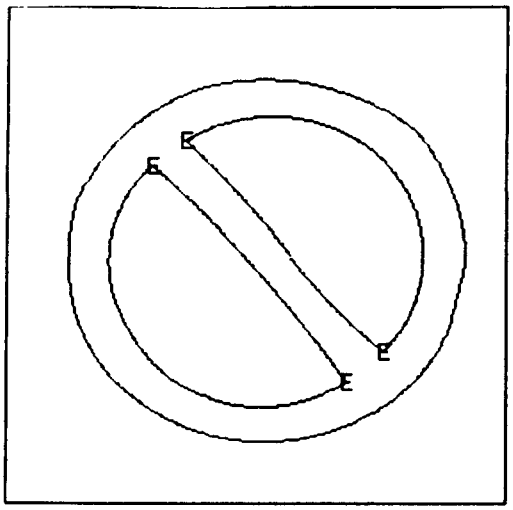


Figure 5.12: **The component configuration of the *disallow* model object found in the *no cars* sign.**

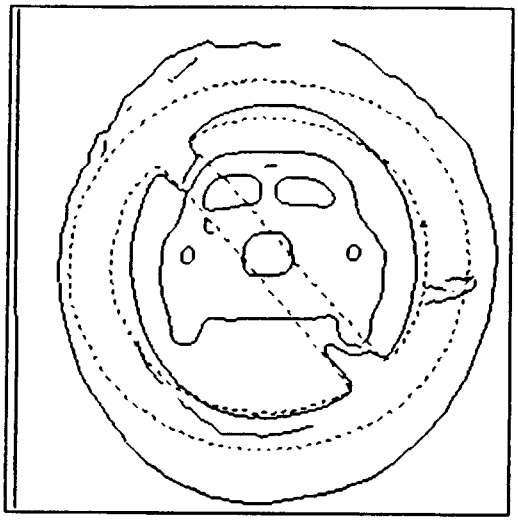


Figure 5.13: **Superimposition of the *disallow* model object on the *no cars* sign.**

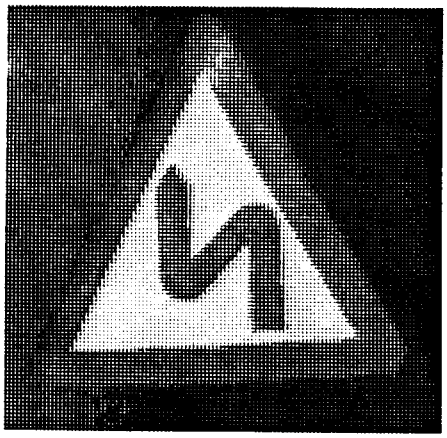


Figure 5.14: **The grey level test image of a *bend in road* sign that is similar to a rotated mirror image of one of the model objects.**

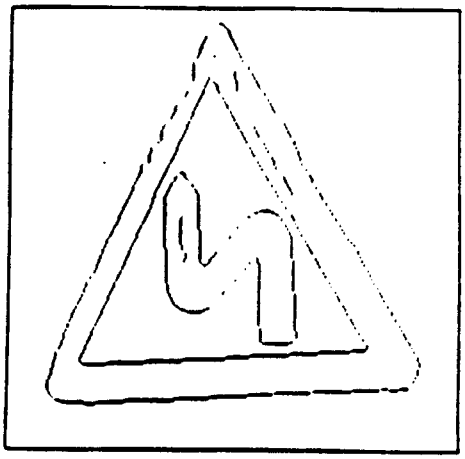


Figure 5.15: **The output of the edge finder processing the *bend in road* sign. Note the noisy contours between the two triangles that result in the orientation of the two triangles pointing in the same direction.**

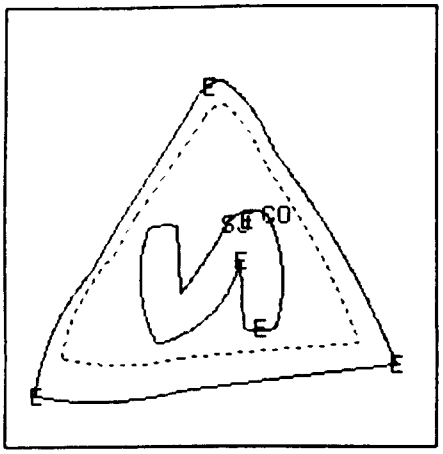


Figure 5.16: **The component configuration of the *bend in road* model object found in the *bend in road* sign.**

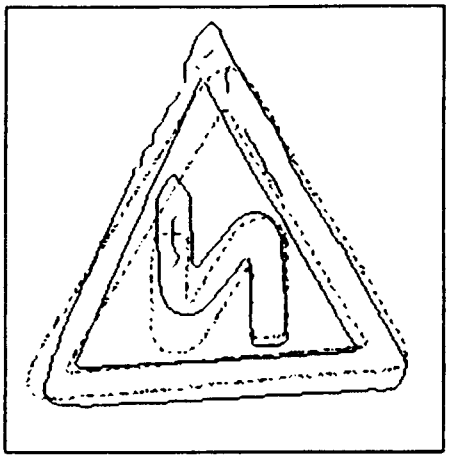


Figure 5.17: **Superimposition of the *bend in road* model object on the *bend in road* sign.**

Figure 5.18: **The grey level test image of a *parking* sign.**

no longer equilateral, which they were in the physical objects.[2] Despite the distortion, the system was able to recognize a rotated mirror image instance of the model object in the scene, even though after superimposition the two instances were distorted in different dimensions. Due to noise, the edge finder assigned the same (parallel) orientations to the two triangles. The noise can be seen as additional contour lines between the two triangles in the edge finder output of the scene, Figure 5.15. Since concentric contours alternate pointing in and out, the two triangles point in the same direction. As a result of this noise and distortion, the upper corner of the outer model triangle matched the inner triangle corner of the scene. The inner model triangle was therefore not found and is not predicted accurately due to the scaling and translation of the identified sub-parts. Despite these problems, though, SAPPHIRE still generated a good interpretation.

While the system was not specifically designed for recognizing words, it can certainly be used for that domain since letters are automatically generated as sub-parts of words. Due to the nature of the recognition, SAPPHIRE actually uses the context of letters in order to recognize words. This behavior is achieved since once some letters are identified, only words containing those letters in the identified positions are considered for verification. One example is the *parking* sign in Figure 5.18. In this test, both the parking word object, Figures 5.19 and 5.20, and the *straight arrow*, Figures 5.21 and 5.22, were identified. The large *P* letter was not matched due to the large scale

[2]The TV camera stretched the vertical dimension relative to the horizontal dimension.

Figure 5.19: **The component configuration of the *parking* word model object found in the *parking* sign.**

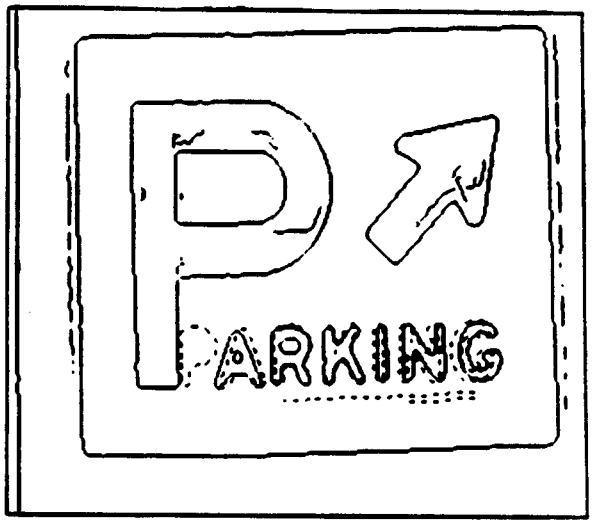


Figure 5.20: **Superimposition of the *parking* word model object on the *parking* sign.**

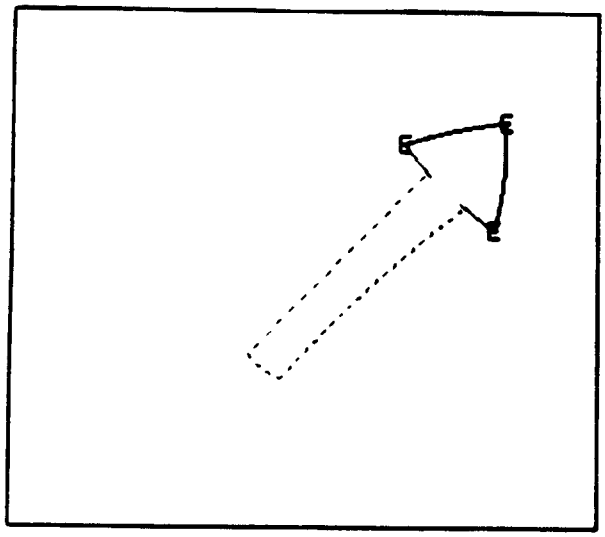


Figure 5.21: **The component configuration of the *straight arrow* model object found in the *parking* sign.**

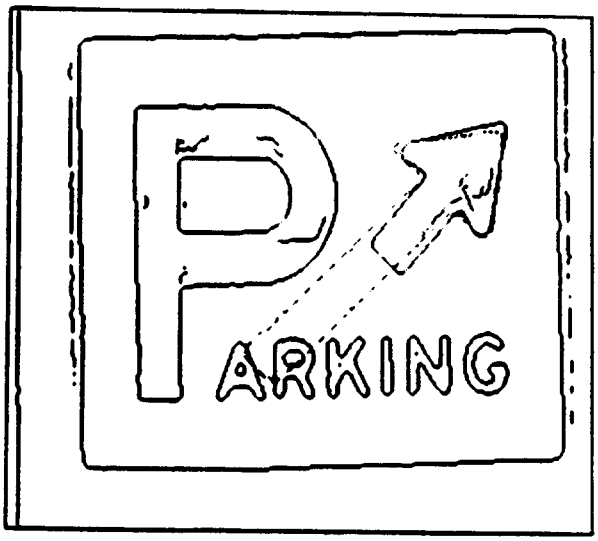


Figure 5.22: **Superimposition of the *straight arrow* model object on the *parking* sign.**

difference. Even though the system can allow for such large five-fold relative sub-part scaling variations[3], the model P is not matched to the scene P since the size of the small model letter prevents the system from deriving the same features as for the same large letter. Clearly, the features for the model P in Figure A.12 cannot match the ones for the character in the scene since the representation for the model letter only consists of two ends, appropriately parameterized, at the bottom and top of the letter. With higher resolution, such as in the scene, the letter is described much more accurately—it does not have an end in the middle of the top of the letter. The reason for this phenomenon is that, as stated earlier, the size of the letters in the *parking* model object is at the boundary of the resolution of the system.

These examples demonstrate some of the important facets of SAPPHIRE's recognition performance: identification and localization of the correct objects from a model library, stability in the presence of variations of global and sub-part parameters, tolerance for sensor noise, and wide range of objects that can be recognized. The remaining five examples using the traffic library are shown in Appendix B. These examples consist of signs for the following: *passing allowed*, *railroad crossing*, *no U-turn*, *junction ahead*, and *reduce speed now*. They demonstrate similar performance and provide more data for analyzing the system's capabilities. An example with a different set of objects is shown in Appendix C. This example shows recognition in the presence of much occlusion, a situation that was not common in the traffic sign examples.

[3]The system is usually run with a relative sub-part scaling factor of 1.5.

Chapter 6

Evaluation of SAPPHIRE System Performance

By analyzing the tests run on the SAPPHIRE system, as described in Chapter 5 and the appendices, we can evaluate the advantages and limitations of this particular system as well as of the overall scale/structure hierarchy approach. This analysis is performed in the context of the efficiency and performance goals set out for the system, as outlined in Section 1.1, and the tradeoffs that arise in the implementation of this approach, as described in Section 3.6.

6.1 Efficiency Analysis

The efficiency specifications for the system consisted of two goals: efficient derivation of scene interpretations and efficient indexing of candidate models from an object library. Due to the hierarchical nature of the recognition process, it is difficult to separate these two tasks, but various measures will show how they both lead to effective overall recognition. We are mainly concerned with the efficiency of the recognition process rather than the efficiency of the library construction phase. The latter can be done off-line and does not impact the recognition execution times. It is actually advantageous to perform as much computation as possible off-line in order to minimize the computations to be done during recognition [Goad 86]. The SAPPHIRE library construction phase consists of decomposing the objects into sub-parts, computing the sub-part relationships, computing inter-sub-part feature relationships, and comparing all the sub-parts to each other for addition to the library. Construction of the traffic sign library (not including

the representation processor times) took about five minutes on a Symbolics 3640.

The measures for success given for recognition efficiency consist of:

1. Reduction in the size of the search space with use of a structure hierarchy,
2. Small ratio of configurations explored to total number of possible configurations,
3. Reduction in the ratio of configurations explored to total number of possible configurations with increased number of objects in the library, and
4. Sub-linear increase in the number of configurations explored with increased number of objects in the library.

In order to measure these characteristics, efficiency statistics were accumulated for the ten sample scenes in the traffic sign domain. These measures are shown in Table 6.1. The table shows the number of configurations explored as well as the number of total possible configurations for each test run. Two test runs are shown for each scene: one with the whole library and one with a library composed of only the objects similar to those in the scene. The number of total possible configurations consists of the sum of the sizes of the search trees matching all scene features to all features of each model sub-part. The table also provides processing times divided among representation time, hypothesis time, and recognition time. The representation times shown consist of only the feature extraction times. Most of this time is spent performing the convolutions for the Curvature Primal Sketch. This time is directly proportional to the length of the contours in the image. It can easily be speeded up by performing the convolutions in parallel using convolution boards or highly parallel machines. The representation time does not include the edge detection time which takes about five minutes on the Symbolics computer. Since it is also performing many convolutions it can be similarly speeded up. The hypothesis times shown are generally short. The recognition time is directly proportional to the number of configurations explored since the verification of consistent sub-part relationships is very quick.

One measure that can be derived from these statistics is the fraction of the search space that is explored. The average percentage of explored configurations is 0.58% for the whole library runs and 2.4% for the partial libraries. We therefore see that as more objects are added to the library, a smaller fraction of the total search space is explored.

In order to verify the combinatoric advantage of using sub-parts two tests were performed on model objects with and without sub-parts. The results of these recognition

Scene	time (seconds)			configurations explored	total possible configurations
	represent.	hough	recognition		
No bikes					
whole library	275	8	56	1875	627,038
2 objects		1	35	1164	94,160
No cars					
whole library	114	3	16	793	164,894
2 objects		0	2	96	17,206
Passing allowed					
whole library	125	3	22	756	164,894
1 object		0	13	247	17,206
Railroad crossing					
whole library	137	6	58	2235	418,694
1 object		0	9	296	44,246
No U-turn					
whole library	331	5	44	1437	388,058
5 objects		0	23	689	32,960
No left turn					
whole library	339	5	42	1797	288,770
5 objects		1	19	690	19,856
Junction					
whole library	75	4	22	867	68,678
2 objects		0	6	211	4226
Bend					
whole library	80	3	21	868	68,678
2 objects		0	8	342	4226
Parking					
whole library	166	15	117	3738	877,430
2 objects		1	38	856	77,720
Reduce speed now					
whole library	378	16	195	4135	4,194,030
3 objects		2	150	2929	2,466,360

Table 6.1: **Efficiency statistics from ten samples scenes run with the traffic sign library.**

Scene	Model	Configurations explored	Total possible configurations
No bikes	Bike: 8 sub-parts	711	86,126
	Bike: 1 whole object	8170	65,968,012,086
No cars	Car: 4 sub-parts	96	17,206
	Car: 1 whole object	456	408,214

Table 6.2: **Comparison of recognition of objects with and without sub-parts.**

tests are shown in Table 6.2. For the recognition of the bike object in the *no bikes allowed* scene, the size of the search space increased by a factor of 770,000 when the object was treated as a single part instead of as eight sub-parts! Fortunately, the recognition time does not increase by the same factor—the number of configurations explored only increases by a factor of 11. For the recognition of the car object in the *no cars allowed* scene, the search space increases by a factor of 24 and the number of configurations explored increases by a factor of 5 when the object is not decomposed into its sub-parts. Since the bike object is decomposed into more sub-parts and the features are distributed more evenly among these sub-parts, it displays a greater reduction in the size of the search space with use of sub-part decomposition. The results of the recognition with and without sub-parts were very similar. Figures 6.1 and 6.2 show the recognition of the bike in the *no bikes allowed* scene when the bike is represented as a single whole object. As can be seen by comparing these figures with Figure 5.5 and 5.6 for the recognition of the bike when decomposed into eight sub-parts, the recognition results are very similar. The efficiency benefits of sub-part decomposition, though, are significant. In addition, the variations between the bike sub-parts are explicitly accounted for in the decomposition example while they are treated as noise in the single part object recognition.

The small fraction of the search space that is explored in these recognition examples shows that a descriptive representation, a scale hierarchy, and a set of simple geometric constraints can combine to effectively reduce the amount of the search space that is explored. This reduction becomes more apparent for the very large search spaces. For example, only 0.00001% of the search space was explored for the recognition of the

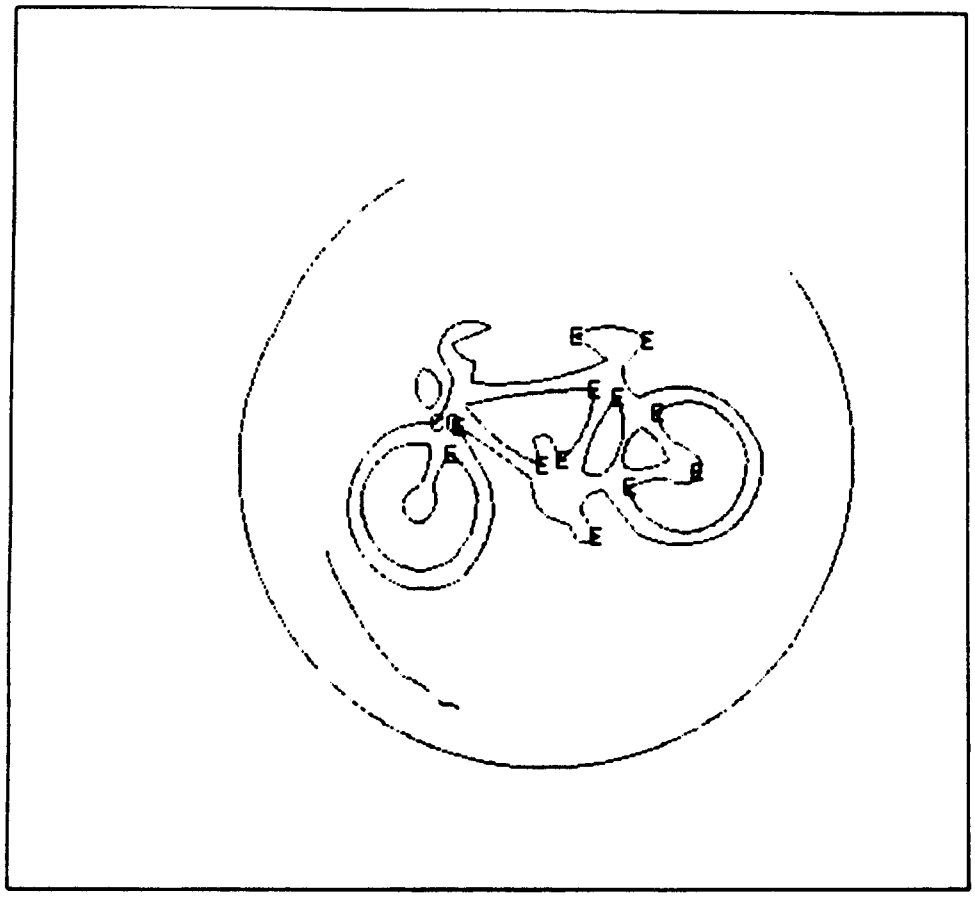


Figure 6.1: **The configuration of the identified *bike* object in the *no bikes allowed* scene when the model is represented as a single whole object.**

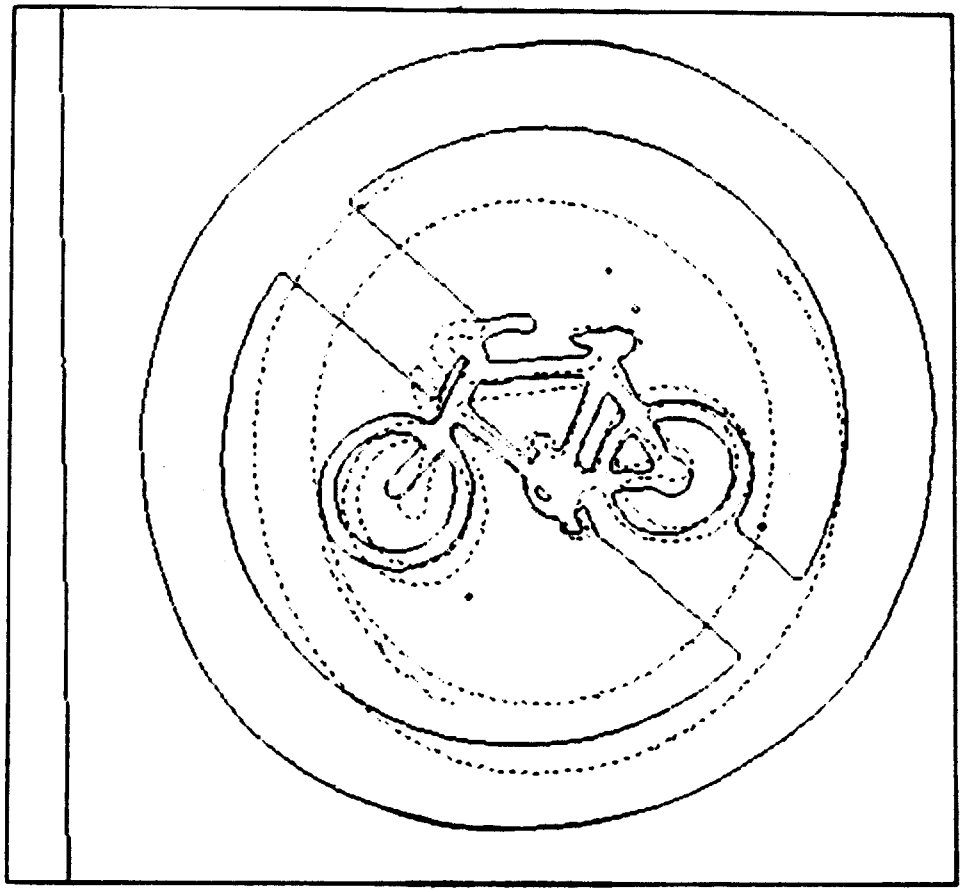


Figure 6.2: **The superimposition of the identified *bike* object on the *no bikes allowed* scene when the model is represented as single whole object.**

single part *bike* object in the *no bikes allowed* scene.

In order to check the relationship of recognition time to number of objects in the library, several test runs were made with different size libraries. Each of the ten traffic sign scenes was run with library sizes of: 1, 4, 5, 8, 9, 11, 13, 17, 19, 21, 23, 26, 30, 34, 38, 42, and 47 sub-parts. These libraries were composed of different arrangements of the model objects. The library size is measured in sub-parts instead of objects since the sub-parts tend to have a more standard size. The results of these tests are shown in Figure 6.3. This graph shows the number of configurations explored versus the number of sub-parts in the library for each of the different scenes. By following each symbol, we can see that the curve tends to flatten out with increased number of sub-parts, thus indicating a sub-linear growth in the recognition time. The solid curve on the graph plots the average number of configurations explored for all runs. The curve exhibits a definite sub-linear behavior.

In order to further show this sub-linear behavior, lines are fit to different portions of the *configurations explored versus library size* curves. For each scene we fit the best line, using a least-squares approximation, to the curve segment signifying the ten smallest library sizes (1 – 21 sub-parts) and the curve segment signifying the ten largest library sizes (17 – 47 sub-parts). The slopes of these lines are shown in Table 6.3. This data shows that the slope decreases with more sub-parts in nine out of the ten scenes. The data shows some fluctuation since it is dependent on the identity of the objects in the library. If no library objects are in the scene, the system can quickly determine that no matches exist if the library objects are different than the ones in the scene. It can also process the objects much longer if they are similar to the viewed objects, but not similar enough to be declared matches. On average, though, the slope of lines decreases by more than a factor of 2 for the larger libraries. The average slope for the ten smallest libraries was 59.1, while the average slope for the ten largest libraries was 24.8.

6.2 Performance Analysis

The examples in Chapter 5 and the appendices show that the SAPPHIRE system was able to meet its performance goals—the system was able to reach accurate interpretations for a variety of scenes. The performance goals that we set out to reach consisted of: correct identification and localization, tolerance for occlusion and noise, stability in the presence of variations of global and sub-part parameters, and a large domain of

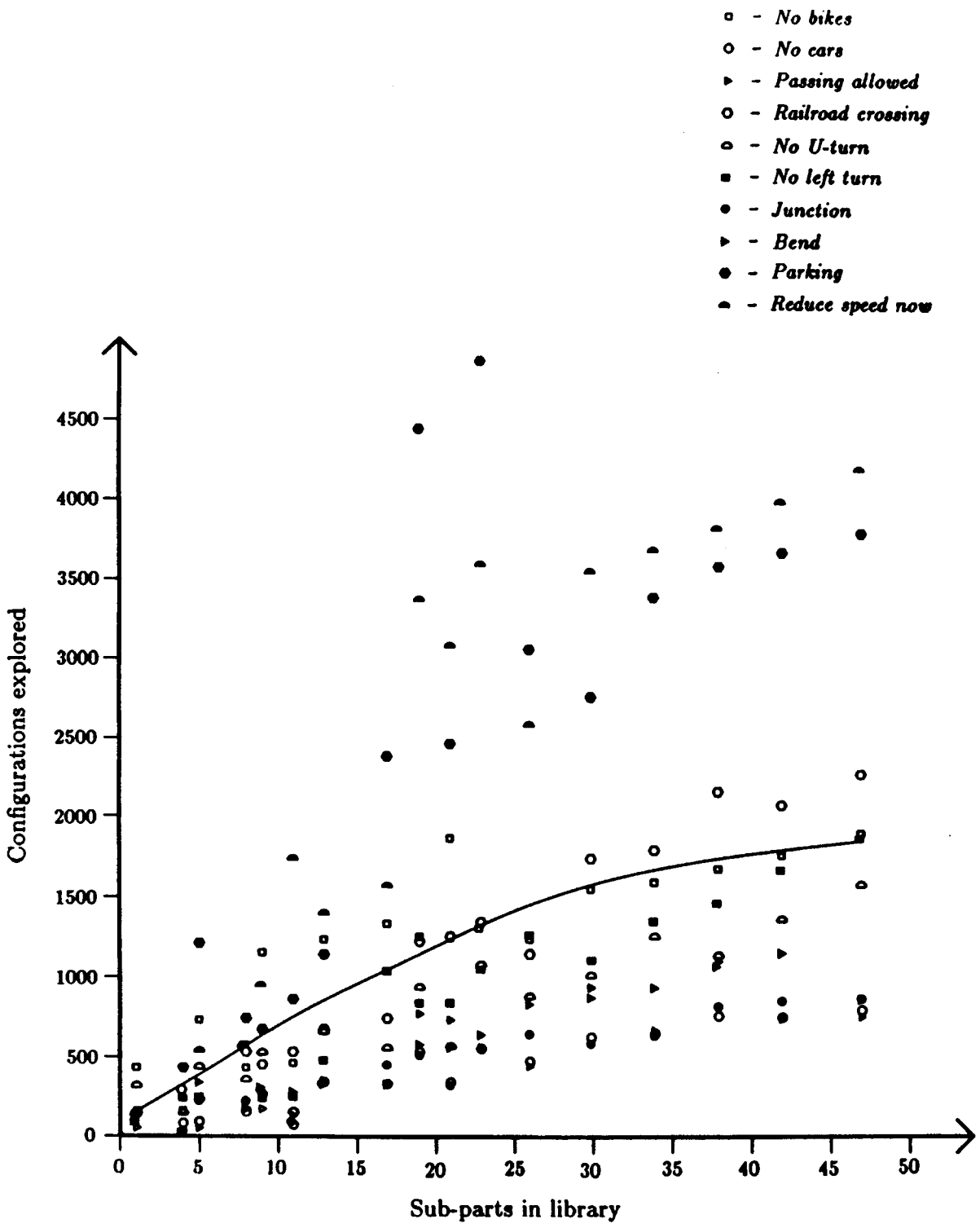


Figure 6.3: **The number of configurations explored versus library size for ten traffic sign scenes. The average number of configurations explored for all scenes is plotted by the solid curve.**

	No bikes	*No cars*	*Passing allowed*	*Railroad crossing*	*No U-turn*	*No left turn*	*Junction*	*Bend*	*Parking*	*Reduce speed now*
1 to 21 sub-parts	68.2	17.0	33.3	54.8	25.3	41.9	16.7	22.3	158.8	152.9
17 to 47 sub-parts	16.1	14.8	10.3	45.7	27.1	30.9	16.0	20.3	11.9	54.8

Table 6.3: **Slopes of lines fit to data of *configurations explored versus sub-parts in library* for ten traffic sign scenes.**

applicability. These performance criteria have been demonstrated by the examples and are summarized below.

In the test run on the system, SAPPHIRE was able to correctly identify the objects in the scenes by selecting the best objects from the model library. The localization of these objects could not always be determined exactly since the objects in the scene and the models were not exact duplicates. The system therefore generated the best localization based on the location and orientation of the features. The interpretations could be better localized by using the contour boundaries as additional verification tokens.

SAPPHIRE was able to recognize objects in the presence of noise and occlusion. The *bend in road* scene demonstrated correct recognition despite noise, which resulted in misleading edges, and distortion, which modified the shape of the object. Some noisy edges were present in many of the models and scenes, but did not hinder the recognition. Many of the scenes also exhibited some forms of occlusion (especially the cardboard object scene in Appendix C), but effective recognition was possible with only parts of the objects visible. The system benefited from the descriptive representation of local features and the scale hierarchy in performing this type of recognition.

The system was able to correctly identify objects even though they were parameterized differently than the models. Mirror image identification and global scaling were performed in several of the examples (*passing allowed*, *bend in road*, and cardboard object scenes). Fluctuations in the spatial layout of the object's sub-parts were also incorporated into the recognition process (*no bikes* and *no left turn* scenes).

SAPPHIRE is able to recognize a broad range of objects. The complexity of the

objects tested ranged from a straight arrow to a (side view of a) bicycle. By employing the structure hierarchy, the system enlarged its domain of applicability by allowing variations in sub-part relationships. In addition, the system can function without outside intervention since it constructs the model library automatically and does not require much user input about the viewed scenes, such as scaling factors or presence of mirror images.

6.3 Resolving the Tradeoffs

Several tradeoffs that arise in the development of this type of hierarchical recognition system are discussed in Section 3.6. These tradeoffs consist of: model-driven recognition versus scene-driven recognition, feature complexity versus matching complexity, sub-part size versus number of sub-parts, and binary matches versus qualitative matches. This section describes how the SAPPHIRE system resolves these tradeoffs.

As described in the Section 4.3, SAPPHIRE's recognition engine incorporate both model-driven and scene-driven approaches. Hypotheses based on data in the scene are made at a high level where enough information is available to be reasonably certain about the correctness of the hypotheses. The system can be certain of these predictions since it does not attempt all at once to hypothesize many parameters of the scene interpretation. The hypothesis step provides a focus for the recognition process which then proceeds with a model-driven approach in order to derive the remaining parameters. The use of the structure and scale hierarchies proves beneficial in deciding where and how much to hypothesize. The identity of viewed sub-parts can be hypothesized based on their coarse features. The fine features are then be used to verify those hypotheses. In this manner, we avoid hypothesizing either the identity of the whole object or its transformation from model to scene coordinates, but use the abstract features of one sub-part (or few of them) to build up the hypotheses slowly.

Several criteria are outlined in Section 3.6.2 for analyzing the tradeoff between feature complexity, matching complexity, and recognition robustness. SAPPHIRE's representation scheme satisfies these criteria as follows:

- **Large domain.**
  The CPS-based representation is capable of representing objects with both straight and curved edges as well as objects with concavities. The features are general enough to represent many non-trivial contour shapes.

- **Stable and sensitive.**
  The scale hierarchy nature of the representation provides for stability at the coarse levels and sensitivity at the fine levels.

- **Local support.**
  The primitives require only local analysis of the contour for their derivation. The specification of the knot points is not susceptible to global variations.

- **Information preserving.**
  The representation allows for direct reconstruction of the original object. Even though the representation is compact, it does not abstract too much shape information—the original shape cues are still available.

- **Multiple feature types.**
  The representation incorporates parameterized types that may be easily matched to each other. The system thus achieves a combinatoric advantage by this compatibility constraint.

In order to resolve the sub-part size tradeoff, SAPPHIRE attempts to decompose the object into semantically meaningful sub-parts while ensuring that the decomposition contains at least one distinctive sub-part. A distinctive sub-part contains at least three features in order to be accurately identifiable. Only two structure hierarchy levels are used since intermediate levels are not directly useful for recognition. Thus the model object is directly broken up into its smallest component sub-parts. In order to separate the sub-parts at points where scaling, rotation, or translation may occur, SAPPHIRE finds distinctive points of concavity. These points correspond to pairs of concave corners or cranks. In order to further decompose objects, SAPPHIRE employs a spatial locality criterion to identify sub-parts that contain features located close to each other relative to features in the rest of the object. This sub-part extraction process is usually able to decompose an object into several components in order to reap the efficiency benefits of the structure hierarchy and allow for sub-part parameterization.

SAPPHIRE uses parameterized binary matches instead of qualitative matches for measuring feature similarities. This scheme is chosen since it is easier to define the amount of allowed variability in feature characteristics than to define the quality measures for similarity. It also much easier to combine binary matches into an interpretation than to combine qualitative measures. As a result, the effect of modifications to the

binary matching scheme are easier to anticipate since their effects are much more direct. As outlined in Section 3.6.4, a system is easier to tune with binary measures since the meaning of a feature comparison is much clearer if it is defined as either matching or not matching than if many intermediate measure are used as well. SAPPHIRE therefore uses tolerance variables to define allowed matching disparities. These parameters have generally been derived from experimentation with the system even though their values do have intuitive justification, as shown in Chapter 4. If the tolerance level of the parameters is too low, the system starts to miss interpretations of the scene since small variations, which invariably occur, are not tolerated. If the tolerance levels of the parameters is too high, the system starts to generate spurious interpretations since too much variation is allowed in feature specification. By varying the parameters, we achieve a continuum of behavior bounded by these two extremes. The values chosen for the recognition parameters tend to approximate the middle of that range. An interesting avenue of future research is the analysis of both the derivation and effects of these parameters in more analytical terms.

6.4 Summary of Benefits

The SAPPHIRE system has been shown to achieve robust library-driven recognition by using scale and structure hierarchies of representational features based on the Curvature Primal Sketch and a constrained search recognition engine. SAPPHIRE is able to automatically construct a model library that it then uses to efficiently recognize multiple objects in the scene. The recognition results consist of both identification and localization of model objects that are not necessarily exact replicas of the ones in the scene. The system incorporates inexact matching by allowing for mirror images, variations in scaling, and sub-part rotation and translation. Recognizable objects cover a broad range of complexity, and can be recognized in the presence of noise, discretization errors, and distortion introduced by the sensor (a TV camera).

The use of scale and structure hierarchies have contributed to these robust recognition capabilities. The hierarchies allow the system to converge on the correct interpretation and therefore reduce the combinatorics without sacrificing accuracy. By explicitly allowing for sub-part variations instead of treating the whole object as a primitive, a wider class of objects can be recognized. If the object is represented as a single part, any variations are treated as noise and as a result only small perturbations in scaling or

other sub-part transformations can be tolerated. The hierarchies also lead to efficient recognition by reducing the size of the search space as well as the fraction of the search space that is explored.

The scale and structure hierarchies form the basis for the model library organization. In one dimension the objects are decomposed into their component sub-parts that may be shared among several objects. In another dimension, a coarse to fine description is defined for these components. The resultant library is indexed at the coarse sub-part level. A hypothesis module, based on the Hough transform, generates the identity of probable sub-parts by comparing scene features to model sub-part features in the library. The recognition engine then attempts to recognize these sub-parts in the scene by using the coarse to fine object representation and employing geometric constraints and feature compatibility to prune invalid matches and to derive the best configurations. These best interpretations are then expanded to include related components by using the structure hierarchy to predict and verify the identity and location of sub-parts contained by parent models. The use of this library structure leads to efficient recognition that grows sub-linearly with increased number of sub-parts in the library.

6.5 Limitations

The domain of applicability for the SAPPHIRE system covers a broad range of two-dimensional objects. The system works well for complex objects since they can derive the most benefits from the scale/structure hierarchy approach. In order to take advantage of these hierarchies, objects should have enough features so that they can be decomposed into meaningful sub-parts, and contain enough detailed features to benefit from a coarse to fine description. Therefore, the benefits of SAPPHIRE's recognition abilities are better displayed by the recognition of the complex bicycle object (Figures 5.4 through 5.8) than for the recognition of the simpler *no U-turn* sign (Figures B.11 through B.17). The constraint power of the system is reduced by the recognition of small objects that contain only a few features. The system also does not perform any rotational symmetry analysis and as a result could have trouble localizing objects that are almost symmetric. Therefore, small symmetric objects pose the most difficulty for the system.

Another limitation is the failure to recognize objects in certain particular scene alignments. If a large fraction of each of an object's sub-parts is occluded in the scene, the system may fail to identify these components. Enough features may be visible, though,

so that the object should still be identifiable. SAPPHIRE will miss this interpretation since it currently requires sub-part recognition to lead to whole object recognition. While these cases are remote, the capability to recognize such occluded objects may be added by incorporating some global processing of features as a final pass where the structure hierarchy is not used.

An example where symmetry and occlusion resulted in the system localizing the object incorrectly is shown in Figures C.7 through C.9. In that example, the mirror image of the object (corresponding to the object rotated 180°) was identified. This interpretation was given since a slightly better match was concluded for this configuration. The triangle sub-part was occluded such that it could not be identified to resolve the symmetry ambiguity. One vertex of the triangle, though, is visible so that if the system performed some global processing to resolve symmetric configurations, the correct localization could have been found.

Several of the recognition examples exhibited inexact localization of identified objects. While in some cases the inexact localization is due to variations in object parameters, in others (Figure C.9) it is due to not identifying enough features to recover all values of the transformation accurately. This problem is caused by the sole use of the point features in the derivation of the transformation. SAPPHIRE currently does not use the contour segments to verify interpretations. Since the CPS representation provides us with approximation to the contour segments, they can be used to improve the derivation of the transformation values as described in Section 4.3.6.

Chapter 5 also points out another limitation of the system—the minimum resolution required to identify features. Since the Curvature Primal Sketch smoothes the contour with Gaussian filters, it smoothes over very small features. As a result, small objects or ones with very detailed features are difficult to identify. This is not a serious problem, though, since we want to use the abstract features of the shape to drive the recognition. In order to better identify detailed features we can add more levels to the scale hierarchy such that the finest levels use very small Gaussian filters to smooth the contour. If we then want to use the fine features to drive the recognition process we can index into the object library at these fine levels rather than the coarsest level currently used. This processing can be incorporated as a post-recognition step so that only remaining features, not used in the normal coarse feature driven interpretations, are used to recognize small objects.

6.6 Future Research

The development of the SAPPHIRE system has revealed several interesting topics for future research. Many of these topics have been motivated in the discussion of the scale/structure hierarchy approach and the implemented system. These areas are summarized below.

- **Three-dimensional recognition.**
  In order for this system to be practical for general recognition, it should be extended to recognition of 3D objects. One way to accomplish this task is to extend the representation from a description of significant curvature changes to a description of significant surface changes. In order to apply the SAPPHIRE approach to this representation, the following requirements would need to be satisfied:

  - The representational primitives must be derived reliably from the scene using local support.
  - The representational primitives must be defined by a scale hierarchy.
  - Geometric properties, such as location, orientation, and size, must be defined for the object features.
  - The representational primitives must allow for extraction of sub-parts.
  - Rules defining primitive compatibility must be defined based on the feature characteristics.

  A possible representation that can satisfy these criteria is the Surface Primal Sketch [Brady 85] [Ponce 85]. This representation describes significant surface changes such as: steps (discontinuities in surface height), roofs (discontinuities in surface orientation), smooth-joins (discontinuities in curvature), shoulders (compound primitives consisting of two roofs), and thin bars (a step up followed by a step down). Ponce and Brady show methods for extracting these primitives from dense depth maps, such as laser range maps, using a scale-space filtering approach similar to the CPS processing. Since the 3D primitives are space curves, they are defined by more properties than their 2D counterparts and these properties vary from point to point. As a result, they are difficult to extract and match reliably. The number of feature types is also large, making it difficult to decide which features to extract. The solution to these problems constitutes a challenging research area, but will lead to an effective recognition system.

- **Contour segment approximations as additional features.**
  Since the CPS processor can generate approximations to segments of the bounding contour of the object, we can use these approximations as additional features. Linear and circular contour segment approximations can be described by their equations and then matched to each other. The benefit of using these additional features is in achieving better localization of derived interpretations.

- **Improved hypothesis generation.**
  The hypothesis generation module plays an important role in increasing the efficiency of the system and in quickly focusing on the correct interpretations. Methods of improving the power of this module will then have immediate positive impact on the performance of the system. Possible improvements include the incorporation of several hypotheses into a single very likely hypothesis or an increase in the number of parameters that are hypothesized.

- **Improved sub-part extraction.**
  The sub-part extraction process can be improved by using more information than just shape to separate components. Color and texture differences, for example, can provide additional cues for sub-part decomposition. The representation of part functionality can also aid in this task.

- **Symmetry analysis.**
  The identification of rotational symmetries in the model object can aid the recognition process by optimizing the search for feature matches. The system can derive the features that may be used to differentiate between nearly symmetric configurations and also reduce the search by realizing that several configurations may be combined into one due to their symmetry.

- **Increased resolution.**
  The system can benefit by increasing its resolution in order to recognize smaller, more detailed objects. The addition of finer levels to the scale hierarchy will allow these fine features to be labelled correctly. These features can then be used in the configuration refining process, but not to drive the recognition.

- **Global feature processing.**
  In order to avoid missing interpretations when individual sub-parts cannot be identified, the system should employ some form of global feature processing to

derive interpretations based on small amounts of information from the component sub-parts. The system could use individual features to distinguish between similar component configurations if the sub-parts that would distinguish between them cannot be identified.

- **Characterization of system parameters' effects.**
  An interesting avenue of research consists of defining the roles of the parameters that control the recognition process. Several parameters are used to define sub-part extraction, hypothesis generation, feature compatibility, geometric consistency, and configuration verification. In order to better understand the effects and significance of these parameters, we can carefully study how the recognition behavior varies with parameter modification and how the parameters affect each other. An analytical analysis of the parameters can derive equations for them and allow us to tune them more accurately for different modes of recognition.

- **Application of parallelism.**
  Several aspects of the SAPPHIRE system can be implemented in a parallel manner if a highly parallel machine is available. Since the convolutions performed by the edge finder and the CPS processor are local in nature, they can be distributed among many parallel processors. The exploration of the search tree can also be performed in parallel by distributing the search for all the sub-parts among groups of processors. Therefore, in addition to improving the efficiency, we can explore more of the search space and relax some of the hypothesis constraints.

- **Incorporation with other systems.**
  As mentioned above the incorporation of additional cues such as color and texture can aid in the sub-part extraction phase of the system. These additional cues can also help in the recognition phase by contributing additional features that may determine object context. The color of the object, for example, may automatically index a small class of objects. Additional information, such as functionality, can also control the setting of some of the system parameters. By understanding functionality, the system can set sub-part relationship variations that are specific to particular objects.

As we can see, much interesting work remains to be done in order to achieve a general vision system. The SAPPHIRE system, though, demonstrates the benefits

of incorporating scale and structure hierarchies into a robust recognition system that contains a large model object library. These benefits include an appealing recognition behavior of focusing on the best scene interpretations and an efficient derivation of these interpretations in terms of the number of objects in the library.

Appendix A

Sub-parts of Traffic Sign Models

This appendix shows the sub-part decomposition of the thirteen model objects in the traffic sign library described in Chapter 5 and analyzed in Chapter 6. For each model object, the identity, position, and coarse features of each sub-part are shown. Each sub-part is labelled with the following documentation: <*object-name*>: <*sub-part name*>. Common sub-parts will have the same sub-part name. Most objects generate some noisy sub-parts. Most of these are discarded since they do not have enough coarse features. The few remaining noisy sub-parts are included as part of the model representation, but do no affect the recognition since they usually do not contain enough features to be considered distinctive relative to the actual object. The word objects are generally decomposed into sub-parts consisting of individual letters. Some letters like *O* and *D* are broken up into two sub-parts since they have two closed contours. This decomposition also does not greatly affect the recognition since both parts of the letter would need to be identified for a valid match (currently more than half of the sub-parts must be matched). The system is designed to be flexible and is not dependent on any specific sub-part decomposition.

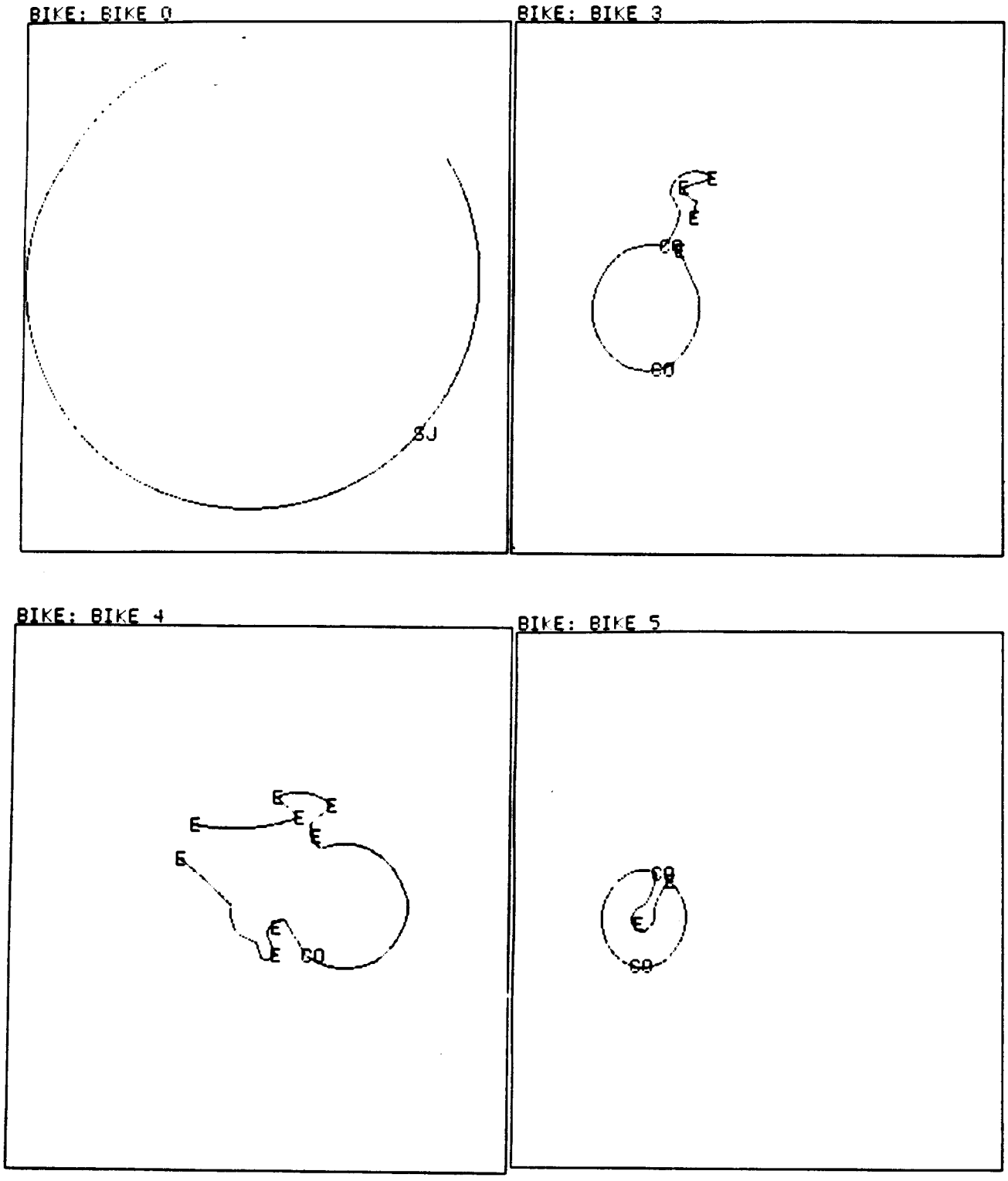


Figure A.1: The first four sub-parts of the *bike* model object (eight sub-parts total).

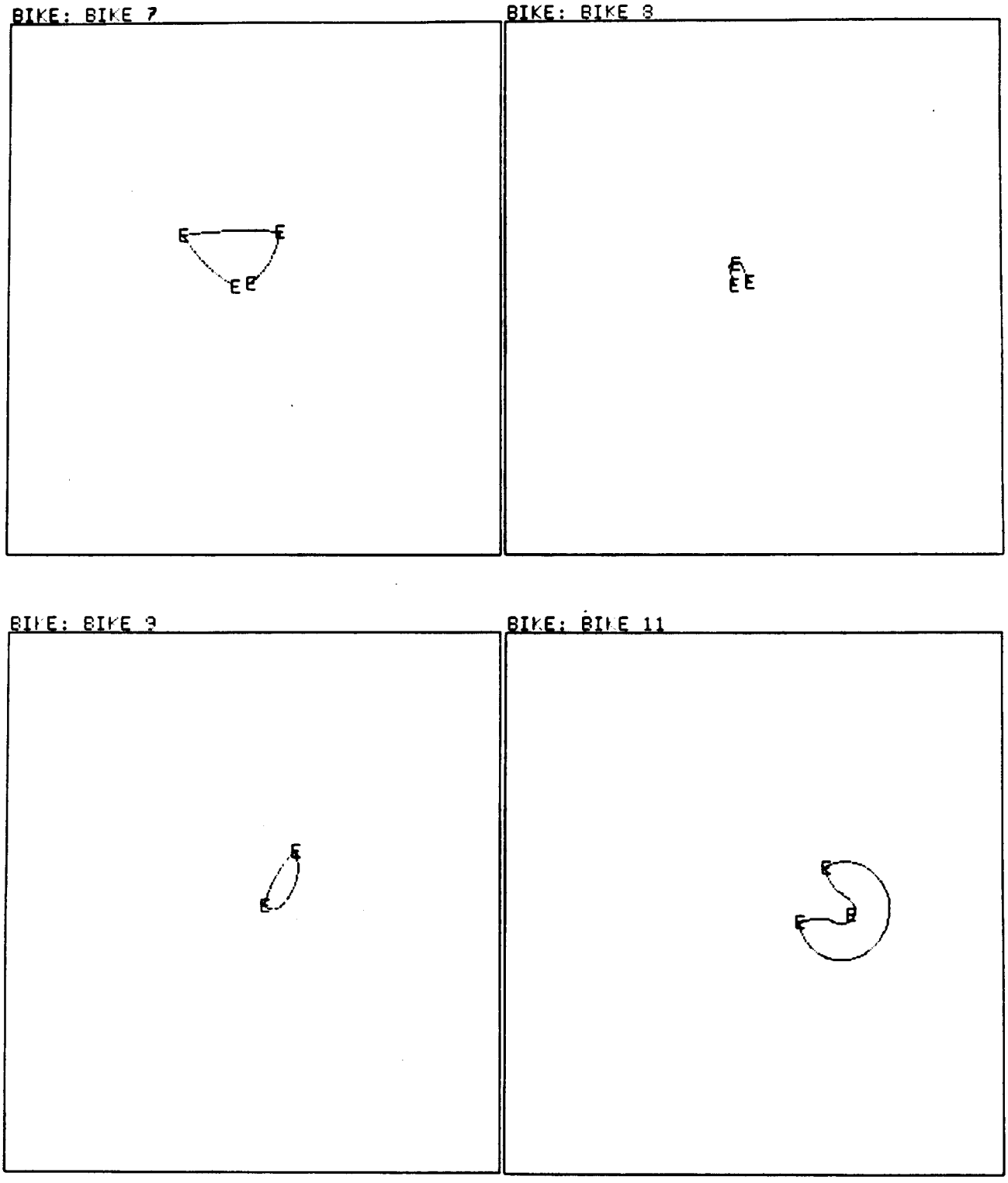


Figure A.2: **The last four sub-parts of the *bike* model object (eight sub-parts total).**

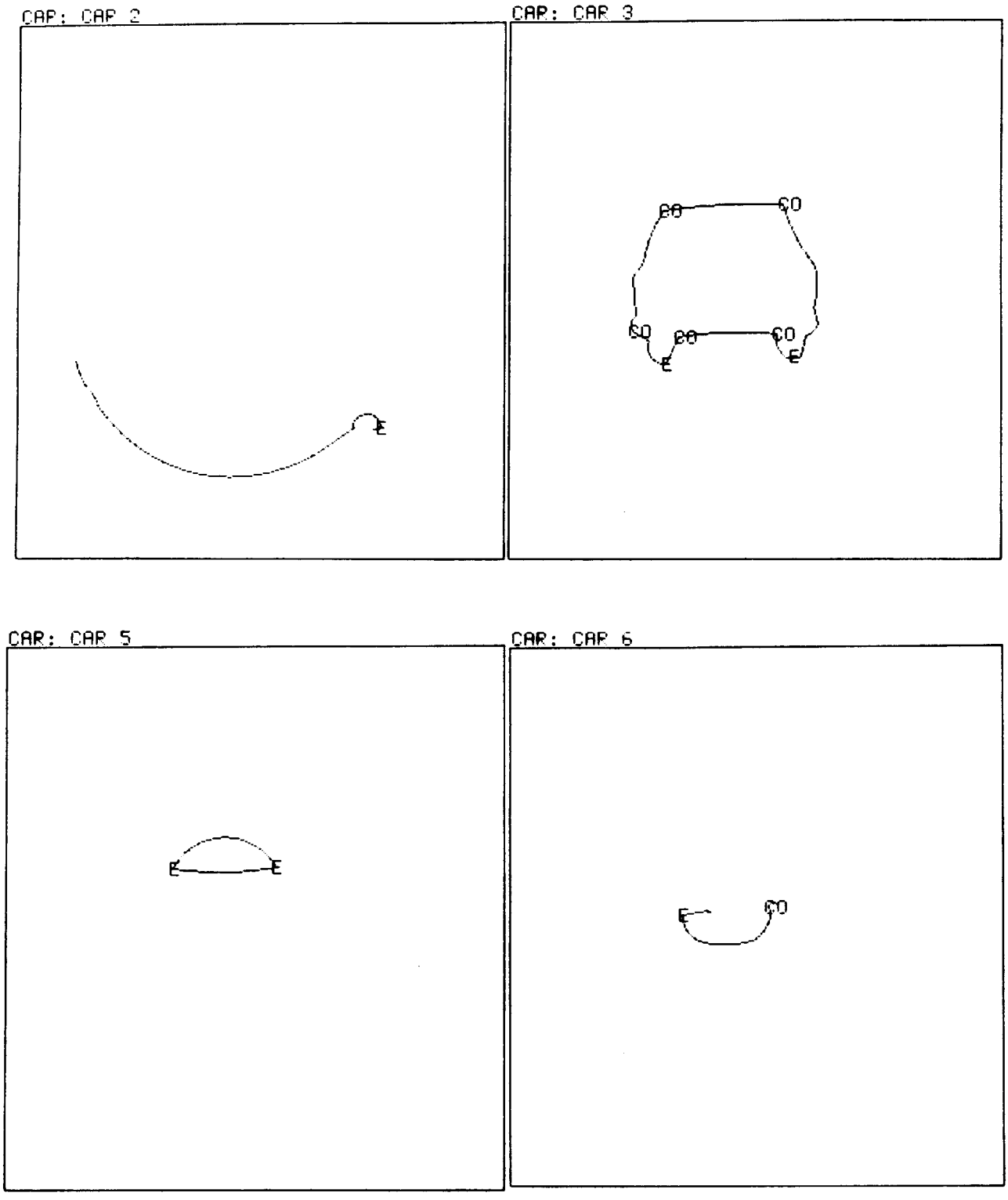


Figure A.3: **The four sub-parts of the *car* model object.**

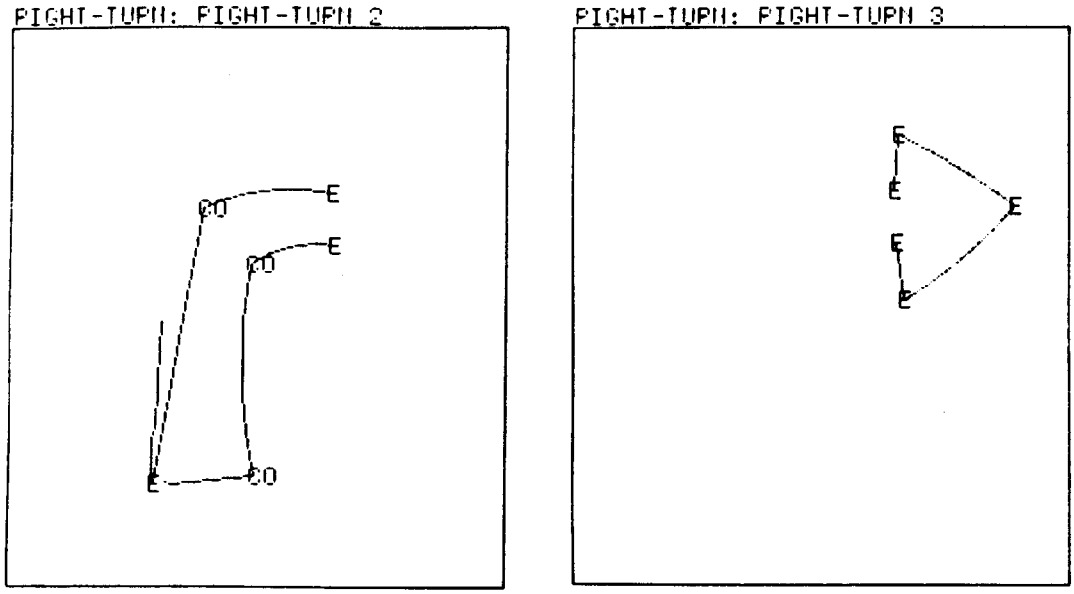


Figure A.4: The two sub-parts of the *right turn only* model object.

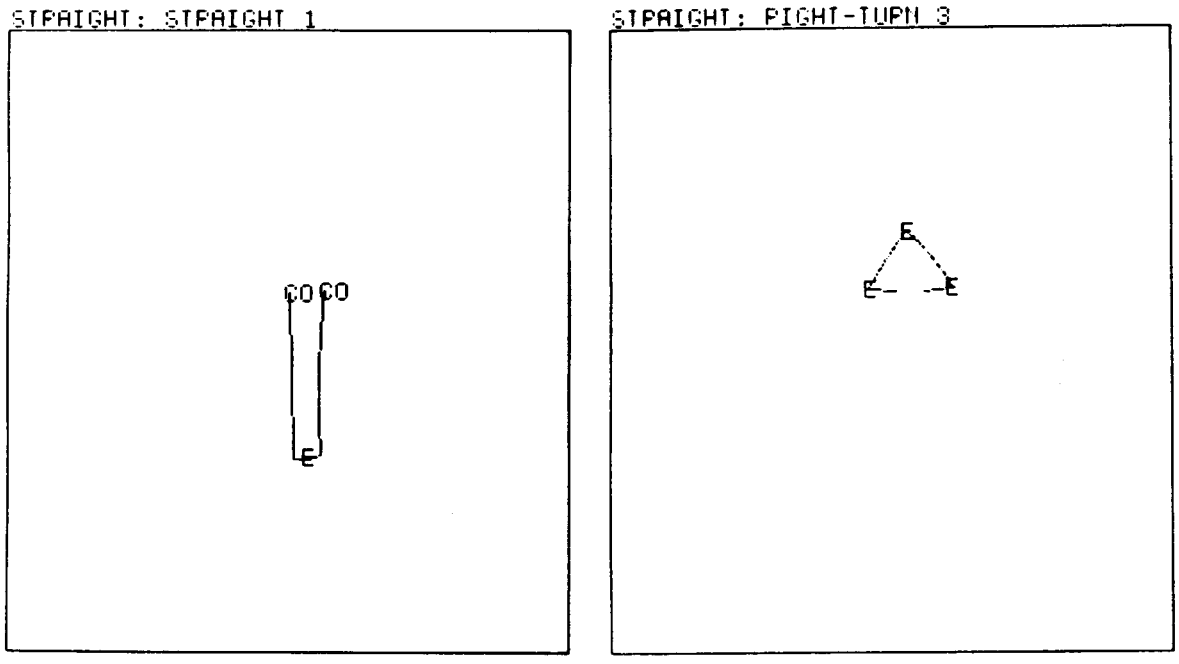


Figure A.5: The two sub-parts of the *straight arrow* model object.

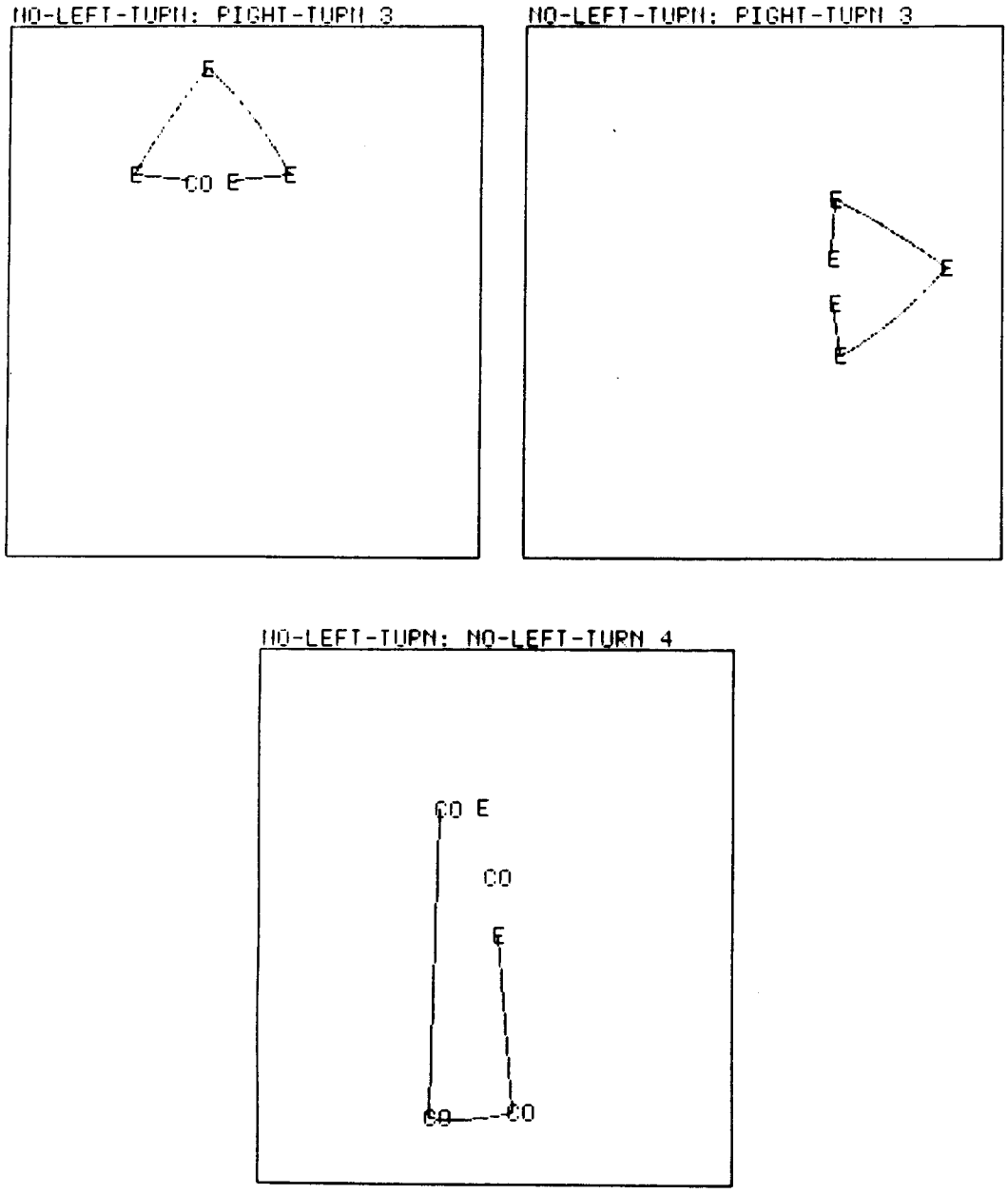


Figure A.6: **The three sub-parts of the *no left turn* model object.**

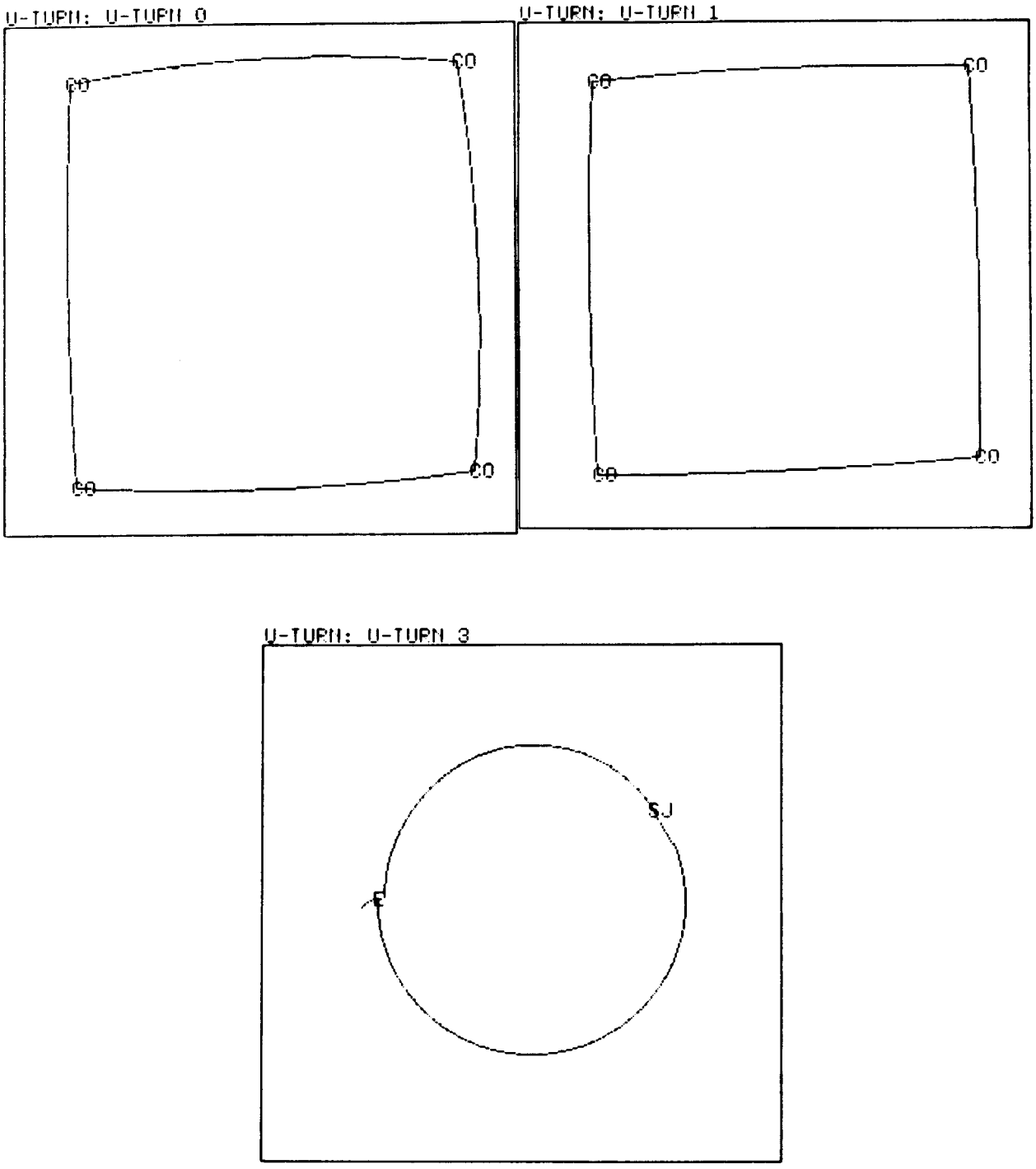
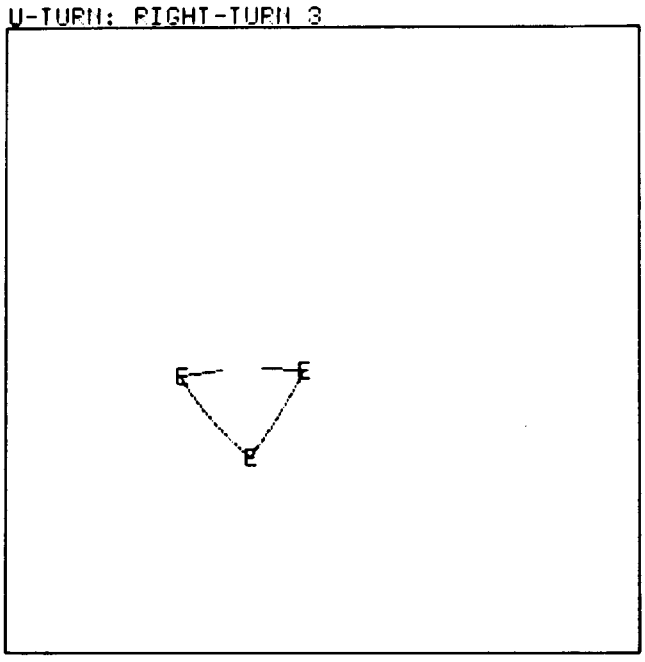


Figure A.7: **The first three sub-parts of the *U-turn* model object (five sub-parts total).**



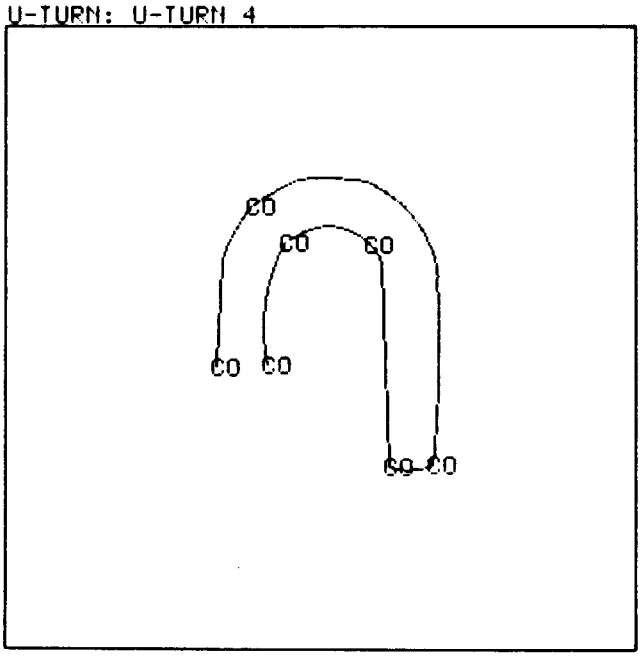


Figure A.8: **The last two sub-parts of the *U-turn* model object (five sub-parts total).**

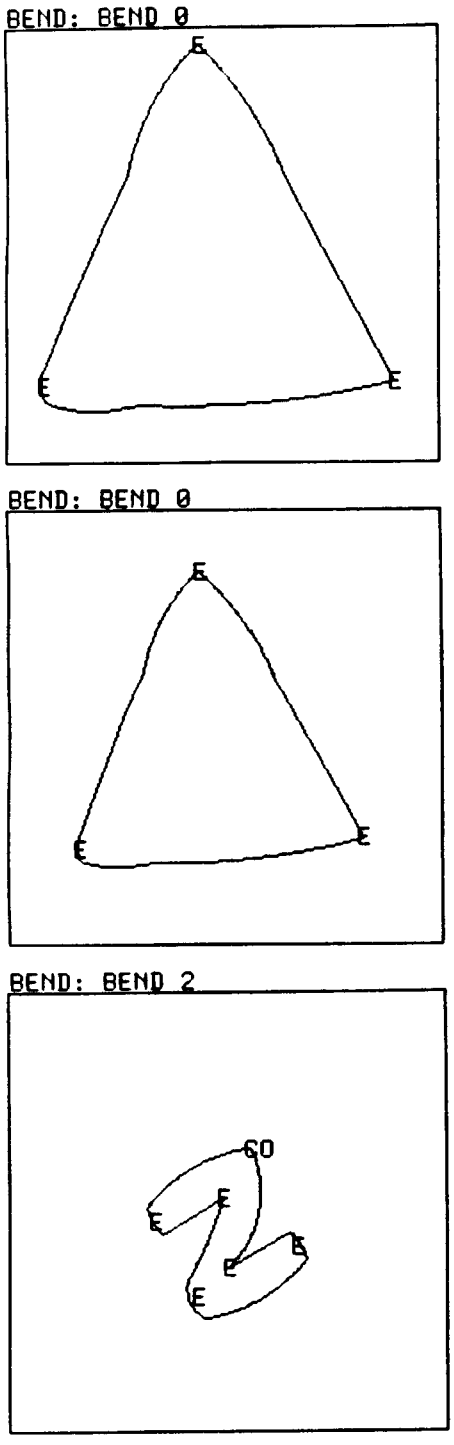


Figure A.9: **The three sub-parts of the *bend in road* model object.**

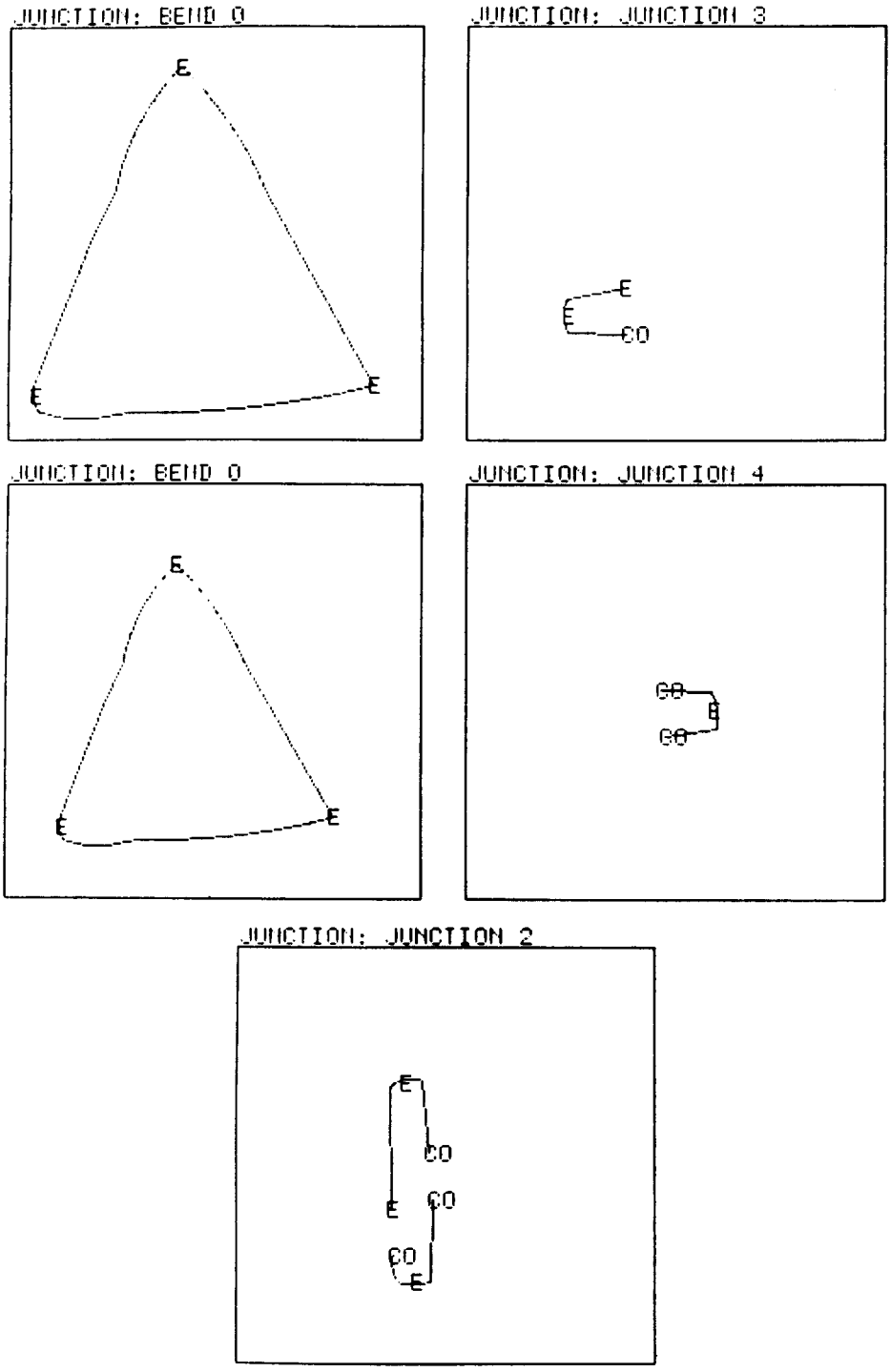


Figure A.10: **The five sub-parts of the *junction ahead* model object.**

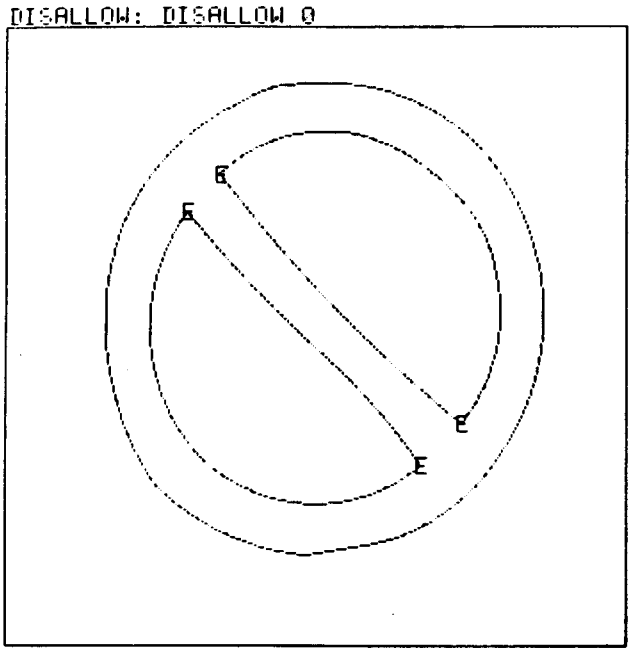


Figure A.11: The single sub-part of the *disallow* model object.

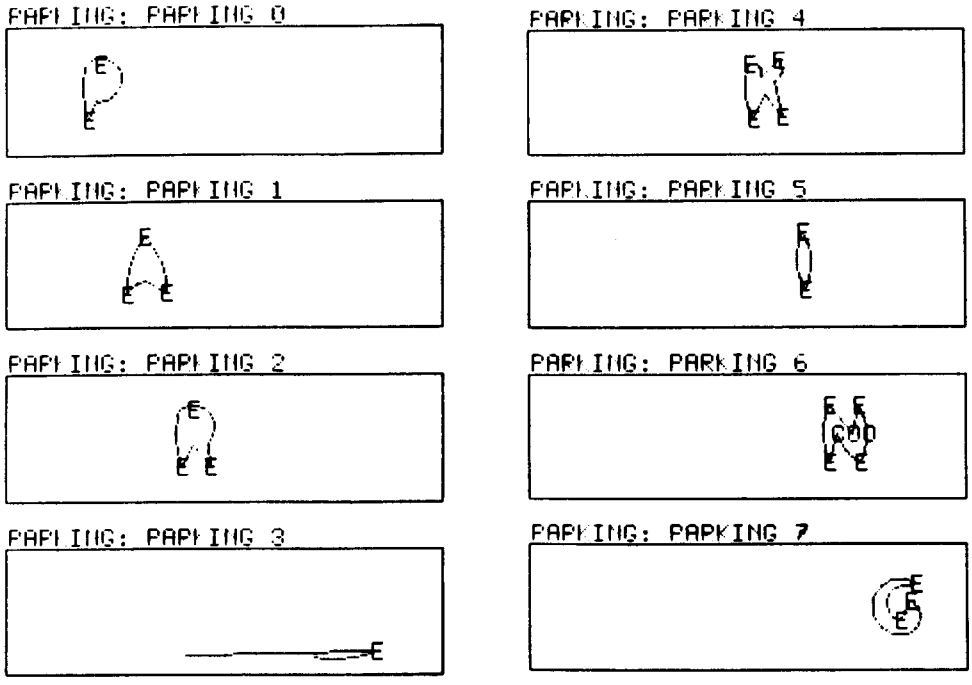


Figure A.12: The eight sub-parts of the *parking* word model object.

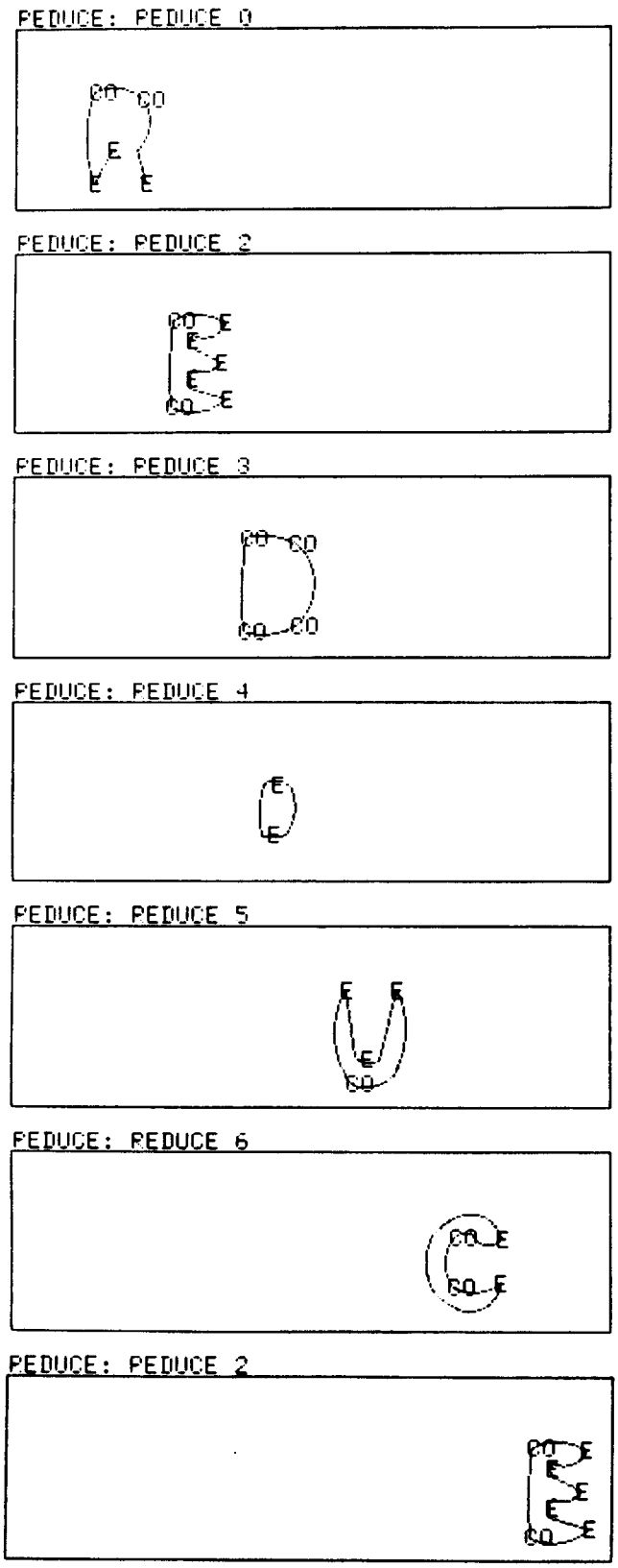


Figure A.13: **The seven sub-parts of the *reduce* word model object.**

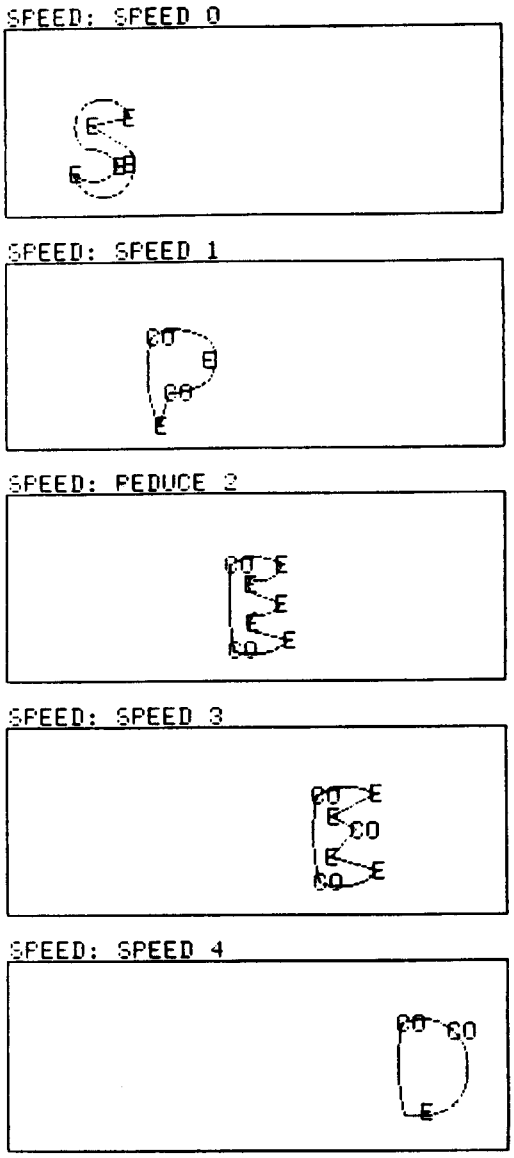


Figure A.14: **The five sub-parts of the *speed* word model object.**

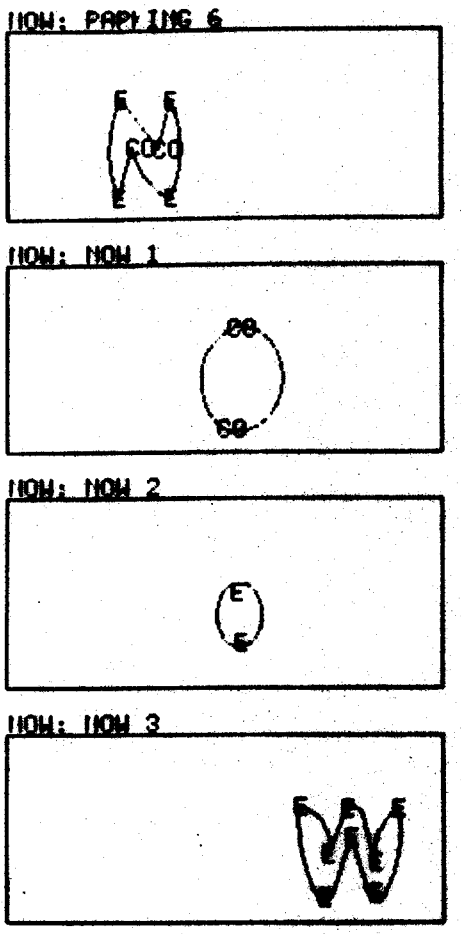


Figure A.15: The four sub-parts of the *now* word model object.

Appendix B

More Examples of Traffic Sign Recognition

This appendix contains the remaining five recognition examples of the SAPPHIRE system using the traffic sign library, shown in Chapter 5 and Appendix A. These tests, in conjunction with the ones in Chapters 1 and 5, are analyzed in Chapter 6. Associated with each example are figures showing:

- The grey level image of the scene.
- The edges found in the scene.
- All component configurations of library objects identified in the scene. Matched sub-parts are shown by solid lines along with the features that were matched.
- Superimposition of the transformed model objects on the scene. The transformation is derived by averaging the transformation vectors of all the matched sub-parts. The contours of the model objects are shown with dashed lines, while the scene edges are shown with solid lines.

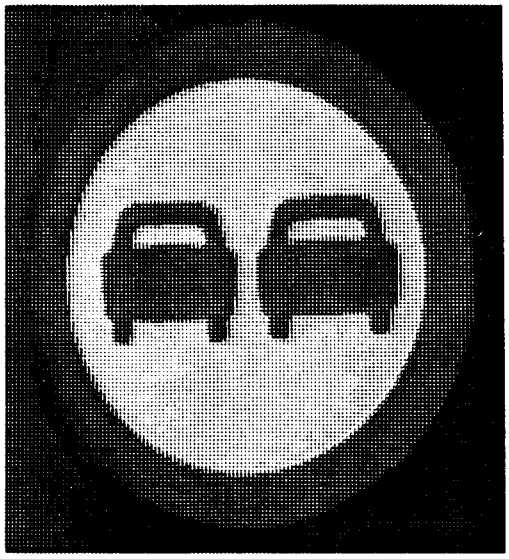


Figure B.1: **The grey level test image of a *passing allowed* sign.**

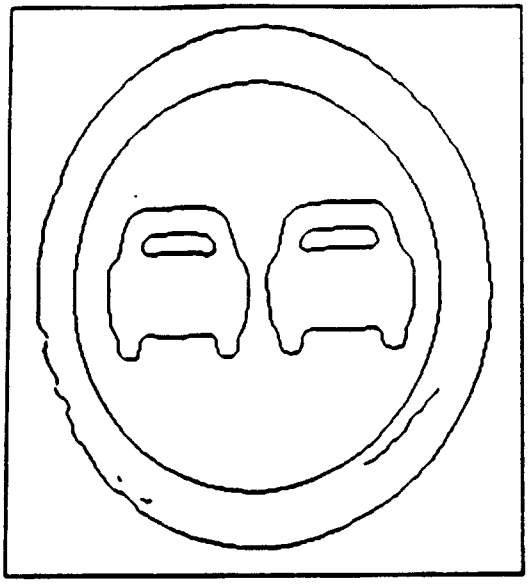


Figure B.2: **The edges found in the *passing allowed* sign.**

The recognition of a *passing allowed* sign is shown in Figures B.1 through B.6. Both cars in the scene are identified even though the features of the viewed cars vary from the features of the model car. Due to this variation, only one wheel of the left car is identified and as a result the transformation of that car is slightly tilted.

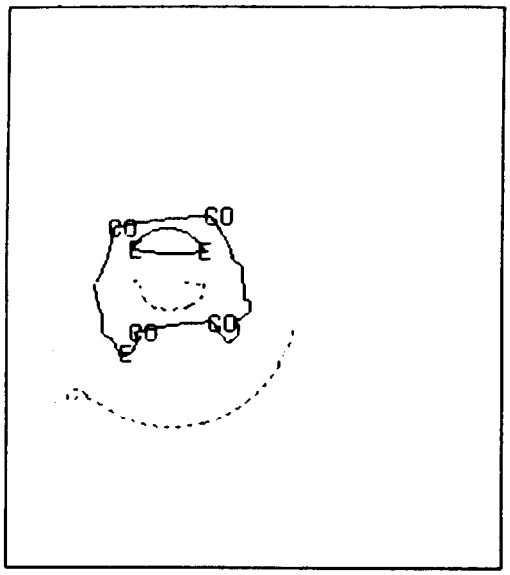


Figure B.3: **The component configuration of the first *car* model object found in the *passing allowed* sign.**

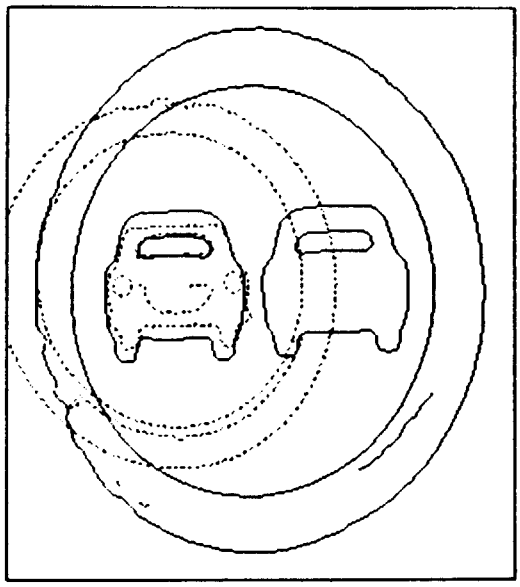


Figure B.4: **Superimposition of the first *car* model object on the *passing allowed* sign.**

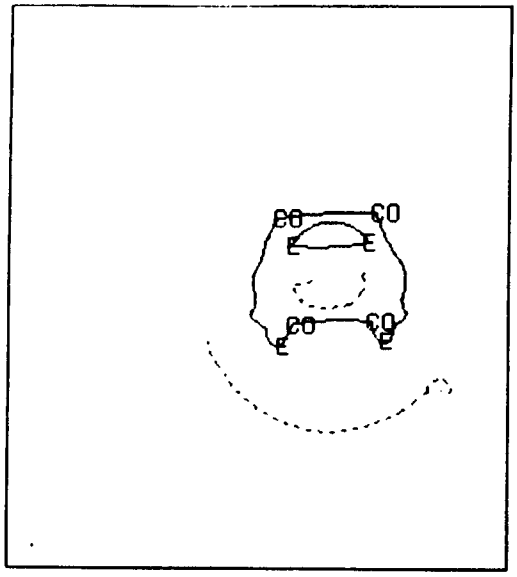


Figure B.5: **The component configuration of the second *car* model object found in the *passing allowed* sign.**

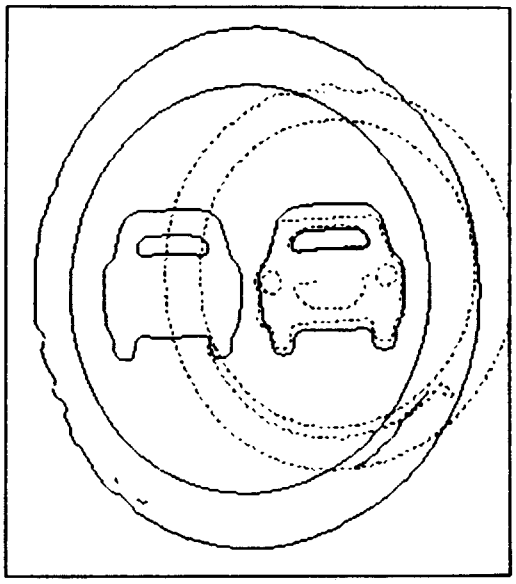


Figure B.6: **Superimposition of the second *car* model object on the *passing allowed* sign.**

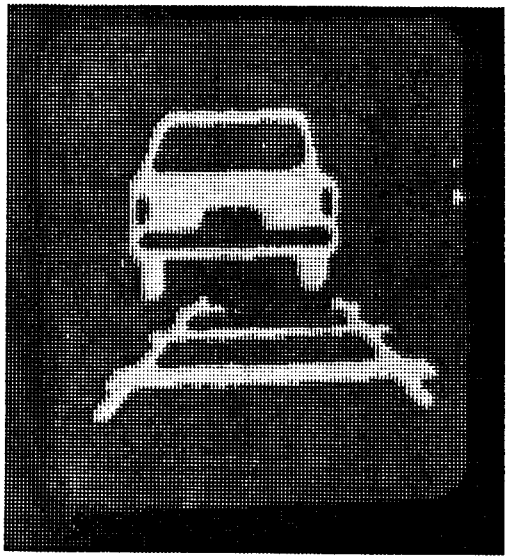


Figure B.7: **The grey level test image of a *railroad crossing* sign.**

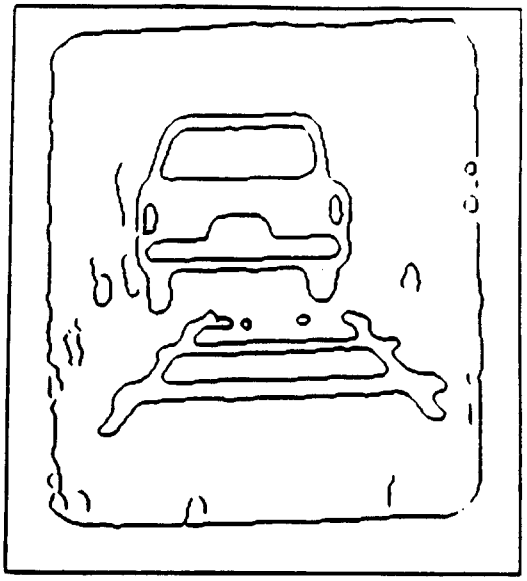


Figure B.8: **The edges found in the *railroad crossing* sign.**

The *railroad crossing* scene in Figure B.7 is processed as another example. The recognition results are shown in Figures B.9 and B.10. This test demonstrates the recognition of yet another type of car, showing the system's ability to extract the important shape features.

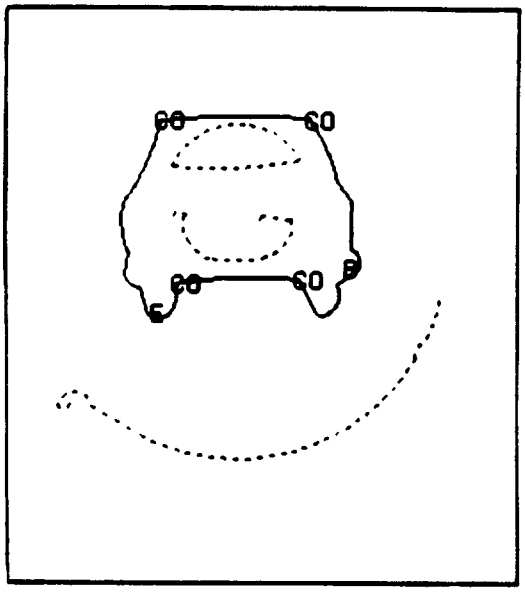


Figure B.9: **The component configuration of the *car* model object found in the *railroad crossing* sign.**

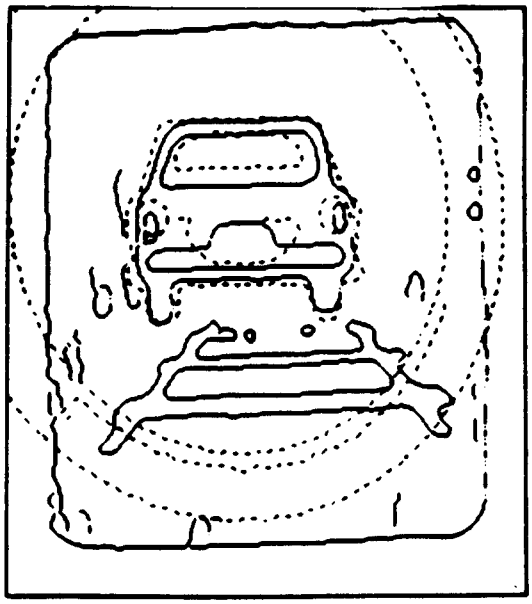


Figure B.10: **Superimposition of the *car* model object on the *railroad crossing* sign.**

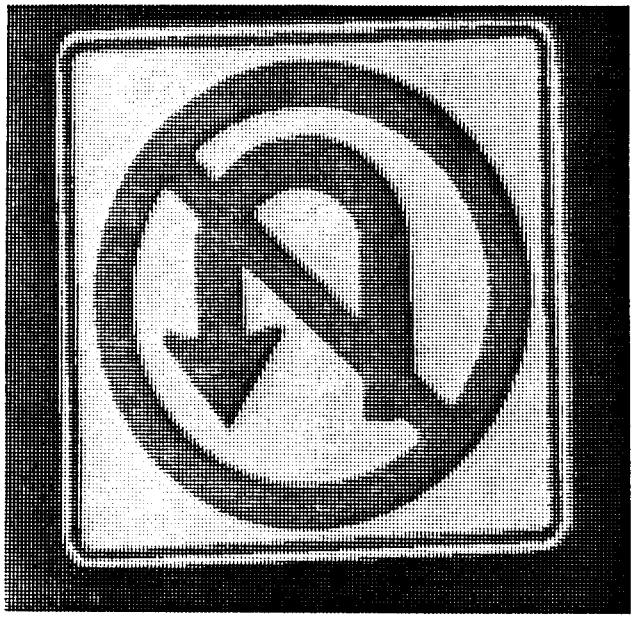


Figure B.11: **The grey level test image of a *no U-turn* sign.**

The recognition of a *no U-turn* sign, Figure B.11, shows the system's ability to identify the correct object when four objects in the library share the viewed *arrow head* sub-part. The system's interpretation is shown in Figures B.14 through B.17. SAPPHIRE is able to correctly identify the *U-turn* object even though its tail is occluded. The correct behavior is achieved by identifying the square sub-parts on the sign. The *disallow* object is not localized correctly since the system could not match the upper end points of the symbol correctly. As can be seen in Figure B.13, which shows the coarse features found in the scene, the end points to be matched to the upper end points of the *disallow* object are specified differently due to occlusion. SAPPHIRE was able to find a scaled version of the disallow symbol by matching a corner inside the *U*.

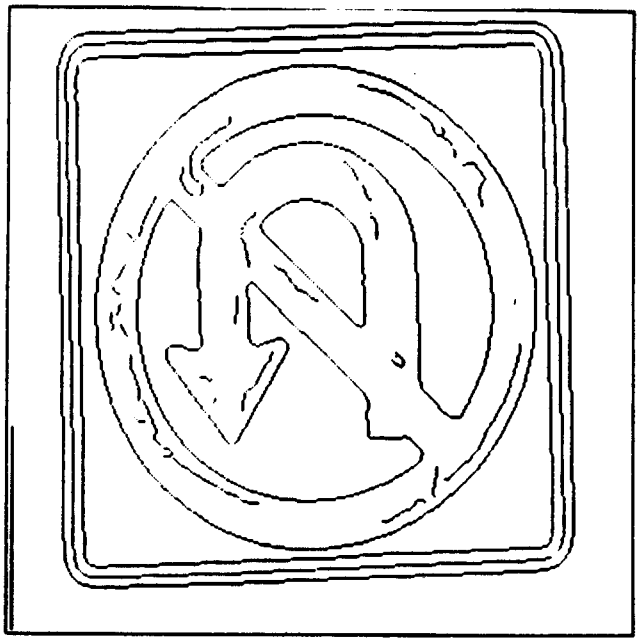


Figure B.12: **The edges found in the *no U-turn* sign.**

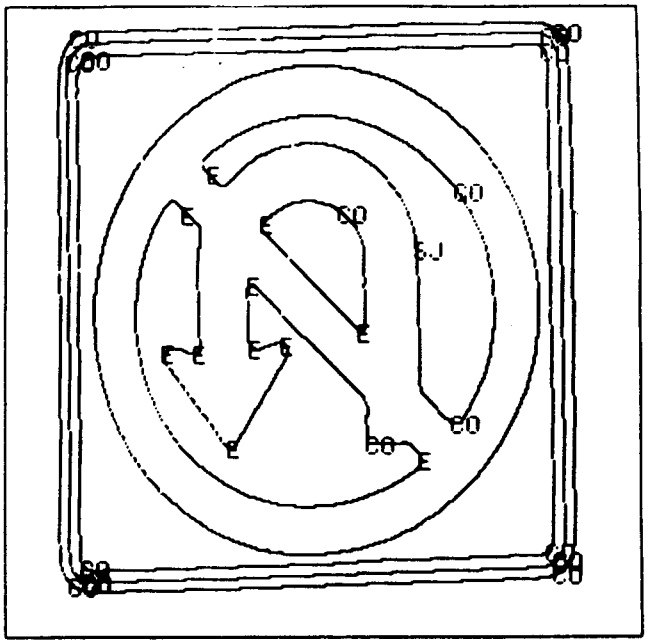


Figure B.13: **The coarse features found in the *no U-turn* sign.**

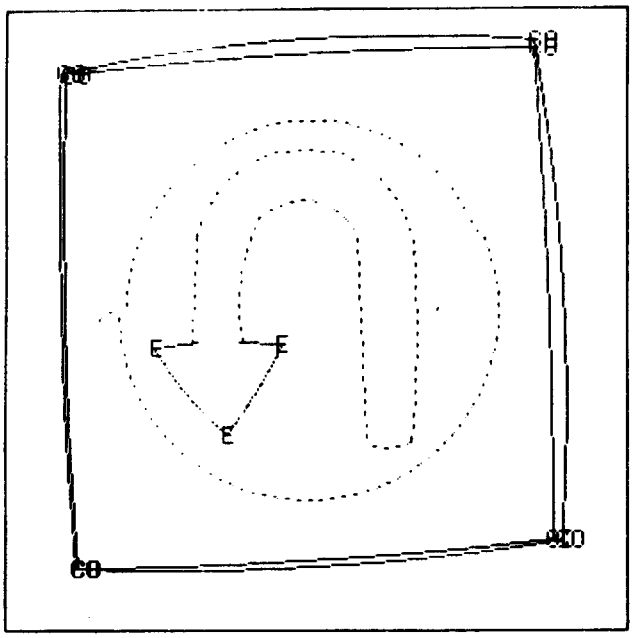


Figure B.14: **The component configuration of the *U-turn* model object found in the *no U-turn* sign.**

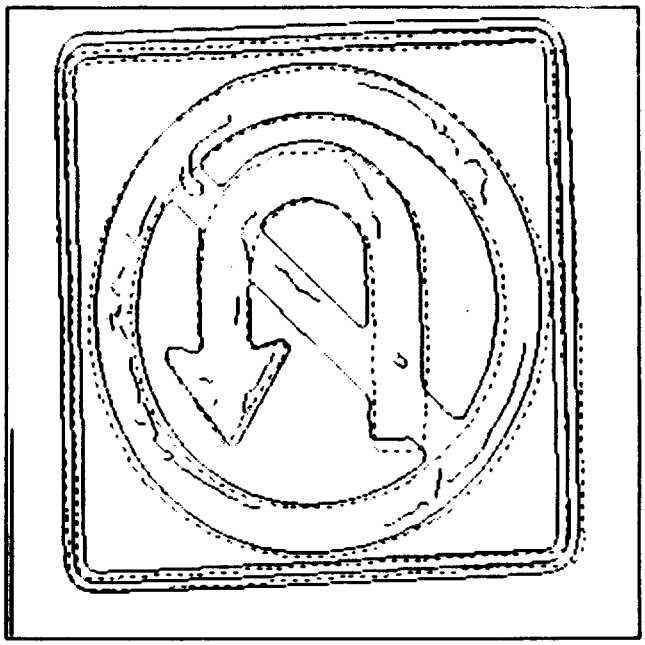


Figure B.15: **Superimposition of the *U-turn* model object on the *no U-turn* sign.**

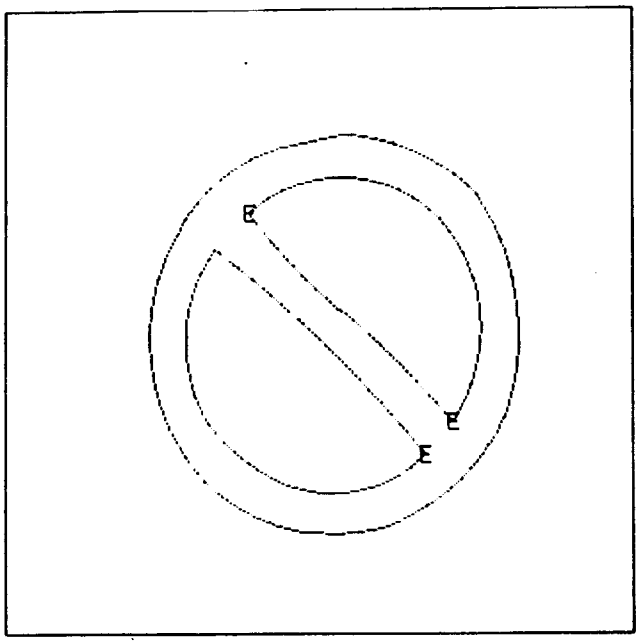


Figure B.16: **The component configuration of the *disallow* model object found in the *no U-turn* sign.**

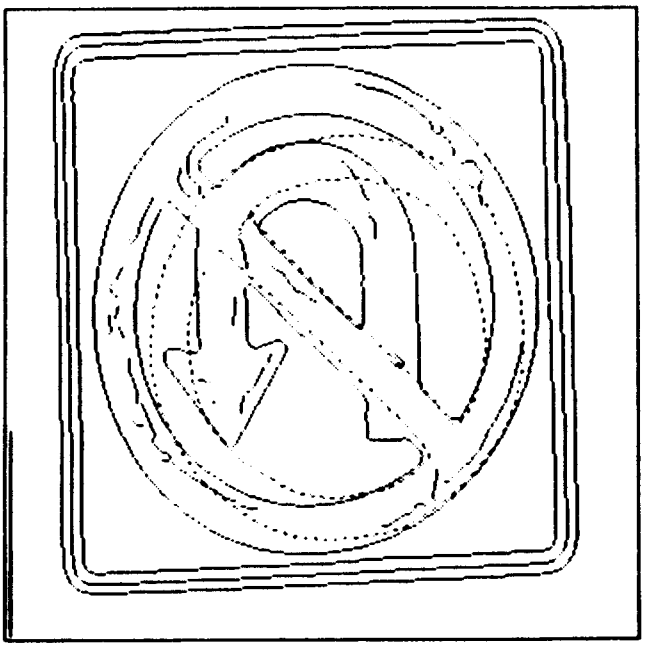


Figure B.17: **Superimposition of the *disallow* model object on the *no U-turn* sign.**

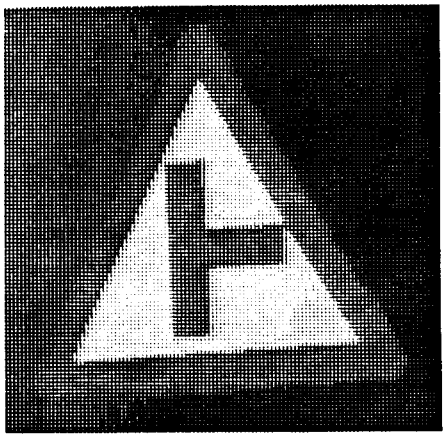


Figure B.18: **The grey level test image of a *junction ahead* sign.**

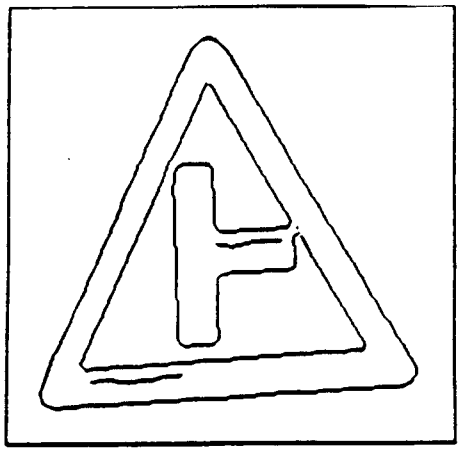


Figure B.19: **The edges found in the *junction ahead* sign.**

The recognition of a *junction ahead* sign, shown in Figure B.18 presents another inexact matching problem. As shown in Figures B.20 and B.21, SAPPHIRE was able to match features to conclude a match. The end of the right branch of the junction was not matched due to noise in the edge specification, as demonstrated in Figure B.19.

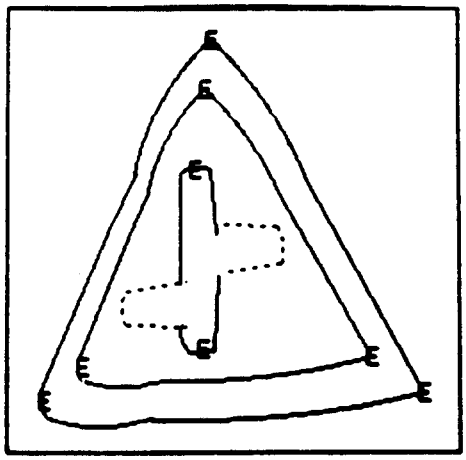


Figure B.20: **The component configuration of the *junction ahead* model object found in the *junction ahead* sign.**

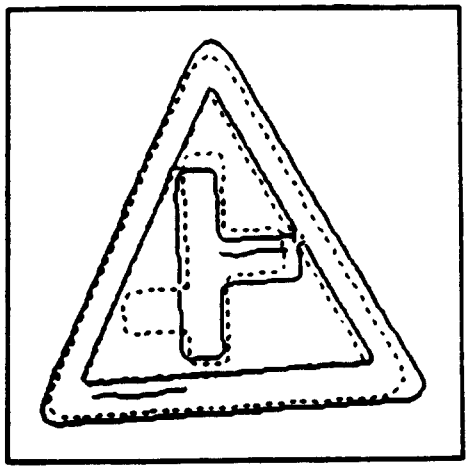


Figure B.21: **Superimposition of the *junction ahead* model object on the *junction ahead* sign.**

Figure B.22: **The grey level test image of a *reduce speed now* sign.**

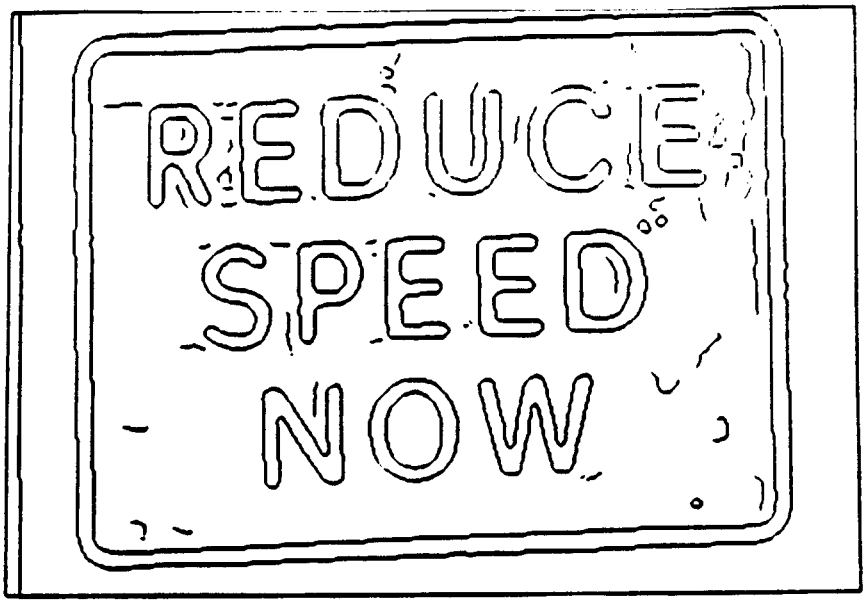


Figure B.23: **The edges found in the *reduce speed now* sign.**

The *reduce speed now* scene, Figure B.22, shows another example of recognizing words. In order to avoid needless search, the system is run in a restricted mode of not allowing mirror images or scale variations from the model since words do not usually show these variations. SAPPHIRE is able to recognize all the word objects in the scene, as shown in Figures B.24 though B.29, even though letters with circular segments present recognition difficulty since they do not have enough distinctive features. For these letters, the use of actual segments as additional features would seem beneficial.

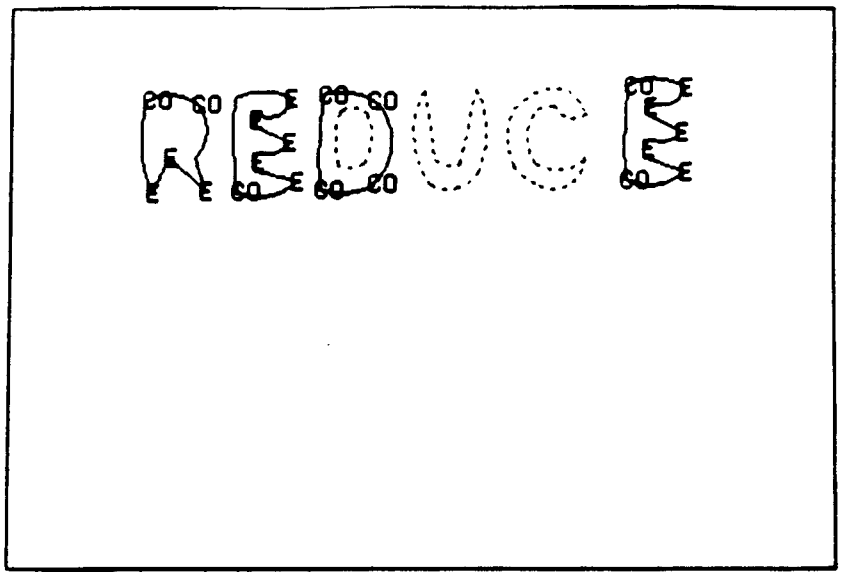


Figure B.24: **The component configuration of the *reduce* word model object found in the *reduce speed now* sign.**

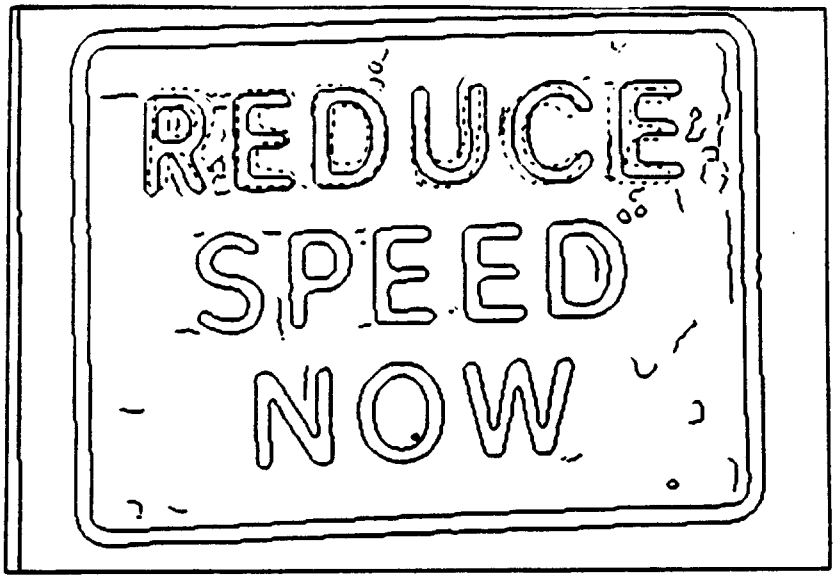


Figure B.25: **Superimposition of the *reduce* word model object on the *reduce speed now* sign.**

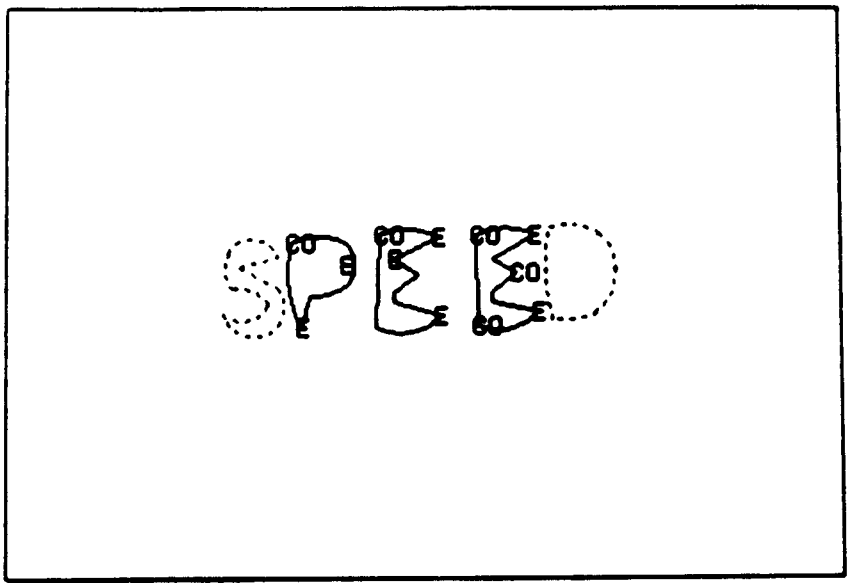


Figure B.26: **The component configuration of the *speed* word model object found in the *reduce speed now* sign.**

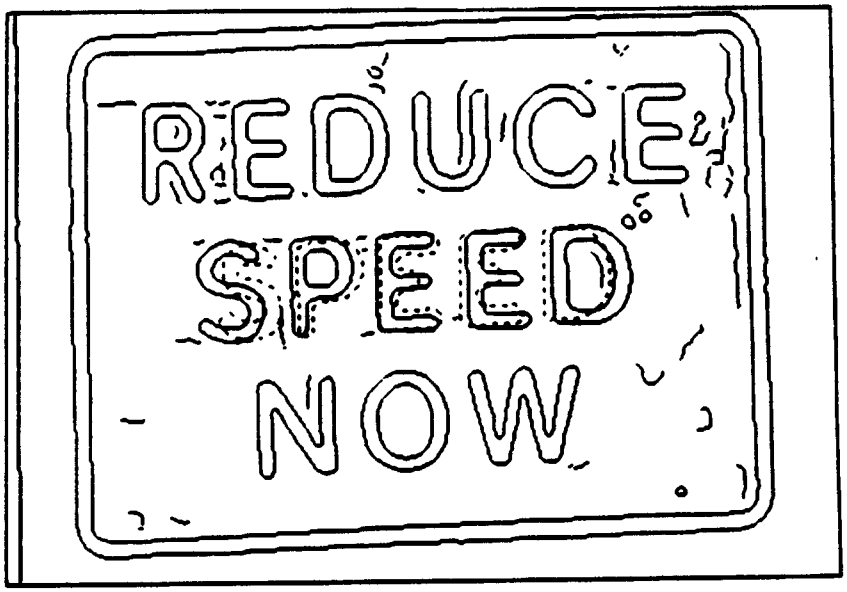


Figure B.27: **Superimposition of the *speed* word model object on the *reduce speed now* sign.**

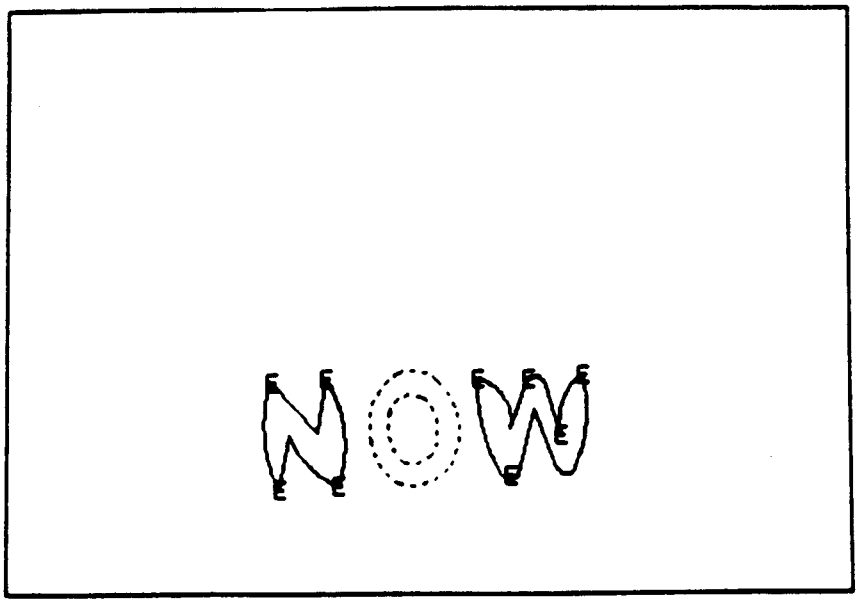


Figure B.28: **The component configuration of the *now* word model object found in the *reduce speed now* sign.**

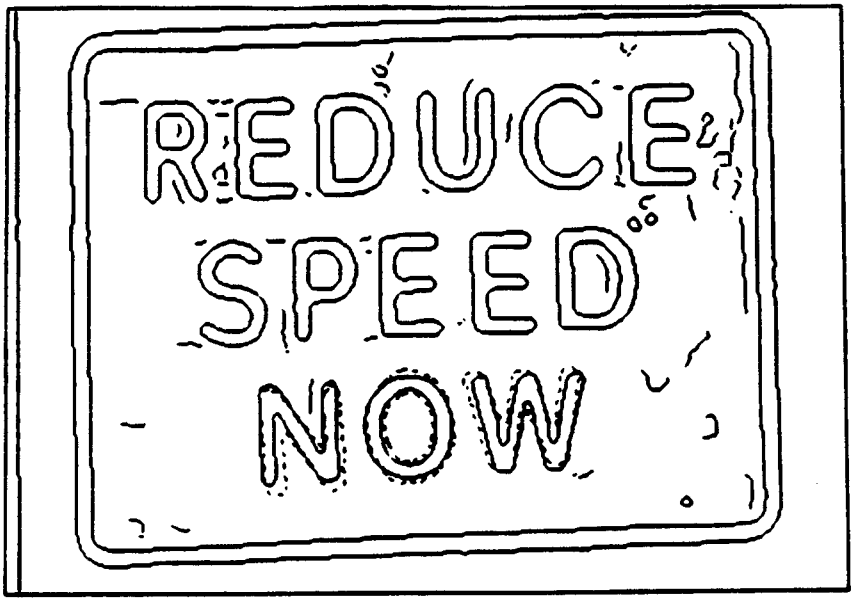


Figure B.29: **Superimposition of the *now* word model object on the *reduce speed now* sign.**

Appendix C

Another Example Set

The SAPPHIRE system was tested with another set of parts in order to show the system's robustness as well as its limitations. This example was run with the set of three cardboard shapes shown in Figures C.1 and C.2. These shapes were used due to the similarity in their teeth structure—many good matches may occur between the sets of teeth. By occluding some of these features, the recognition task is made very difficult.

A library was created with the three objects. These models generated eight sub-parts. The sub-part decomposition of the three objects is shown in Figures C.3 through C.6. The teeth structures were designated sub-parts by virtue of the close spatial proximity of their features. One sub-part was shared by the *mask* and *maskmod* objects, while the symmetric teeth structures of the *chip* object were found to be instances of the same sub-part. One instance is the mirror image of the other. The teeth structures also point out the smallest features that the system can identify. This resolution issue is raised in Chapter 5. In examining the sub-part consisting of the set of variable sized teeth (MASK 1), we can notice that the descriptiveness of the coarse features shown gradually degrades as the size of the teeth decreases. As the tooth size shrinks, they are identified as two-corner ends, sharp ends, and then not identified at all (at the smallest one). As a result, the smaller teeth will not be used in driving the recognition.

A sample scene is shown in Figure C.7. All three objects are present in this scene, but with substantial areas of the objects overlapped. Since this example is testing the ability to recognize objects in the presence of occlusion, the threshold for the number of sub-part features to match was decreased to 35% (from 50%). The two objects that SAPPHIRE recognized are shown in Figures C.8 through C.11. The *chip* object was

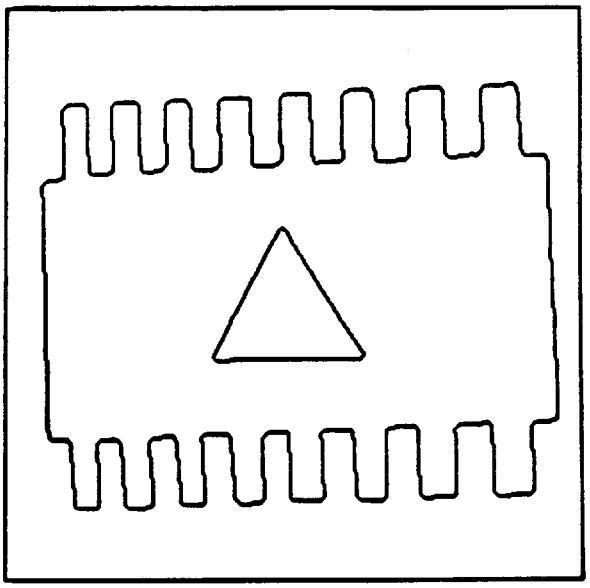


Figure C.1: **The *chip* model object. Bounding contours are shown.**

not localized accurately since, due to the occlusion, the positions of some of the features are displaced. In addition, since the triangle could not be identified, the system could not resolve the symmetry ambiguity. The *mask* object was localized better, but also not perfectly since not all the features were identified due to the occlusion. The use of the actual contour segments can be used to refine the localization of these interpretations. The third object, the *maskmod* model, was not identified since not enough of its coarse features were identified. Since the small teeth are not very distinguishable by the system they are not used to drive the recognition. If the recognition parameters are relaxed, a rough interpretation of that object can be obtained.

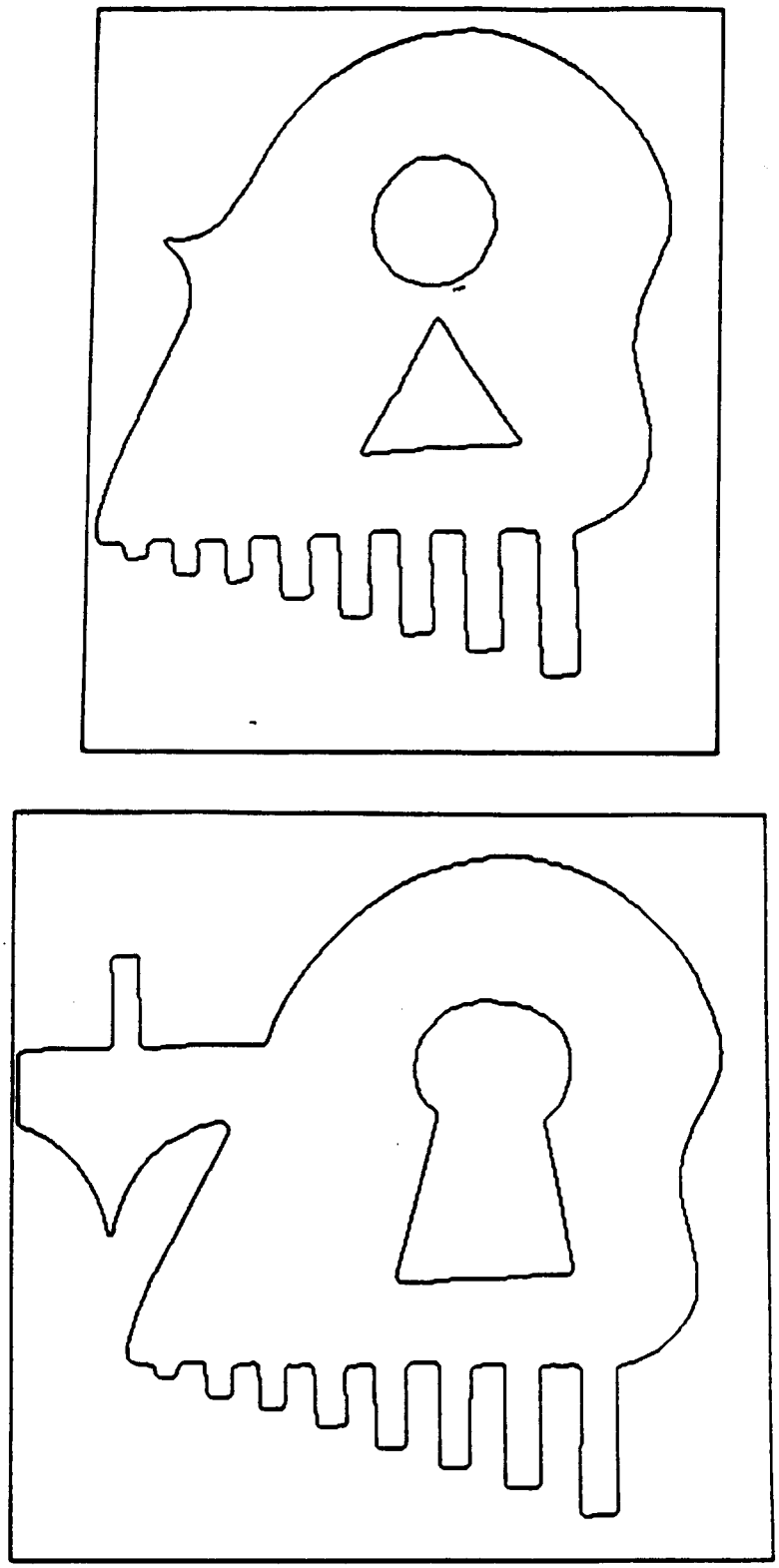


Figure C.2: **The *mask* (top) and *maskmod* (bottom) model objects. Bounding contours are shown.**

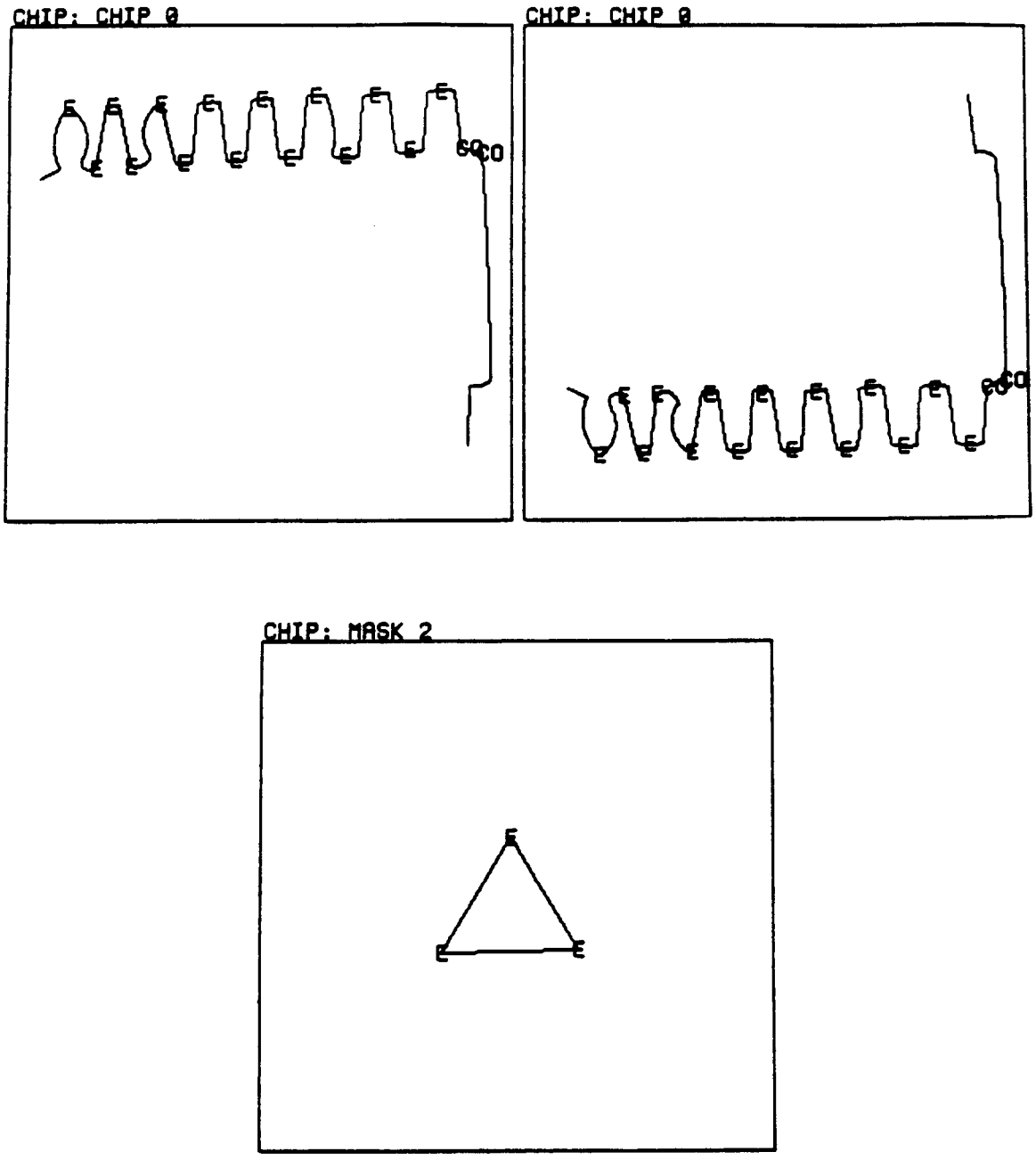


Figure C.3: **Sub-part decomposition of the *chip* model object.**

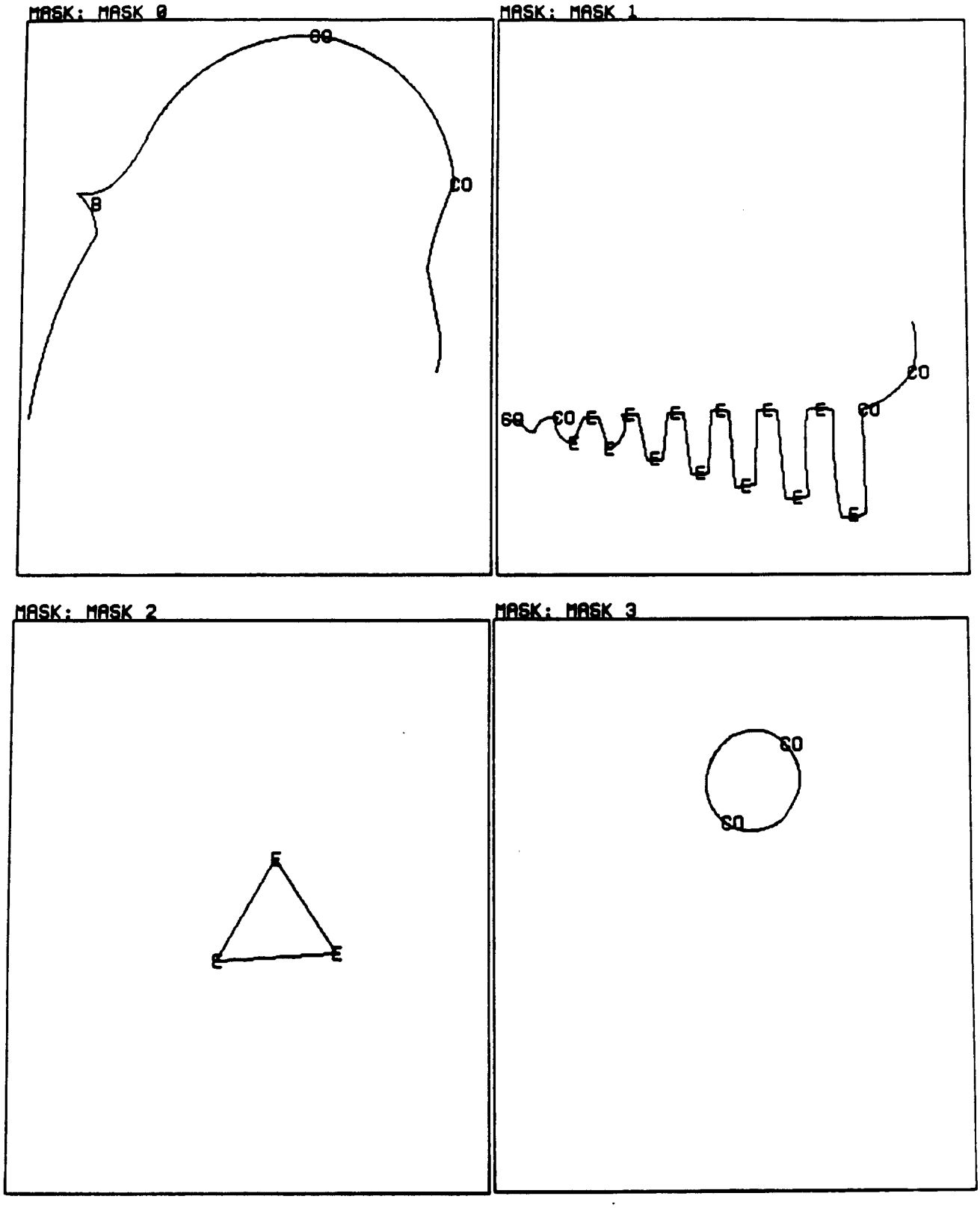


Figure C.4: **Sub-part decomposition of the *mask* model object.**

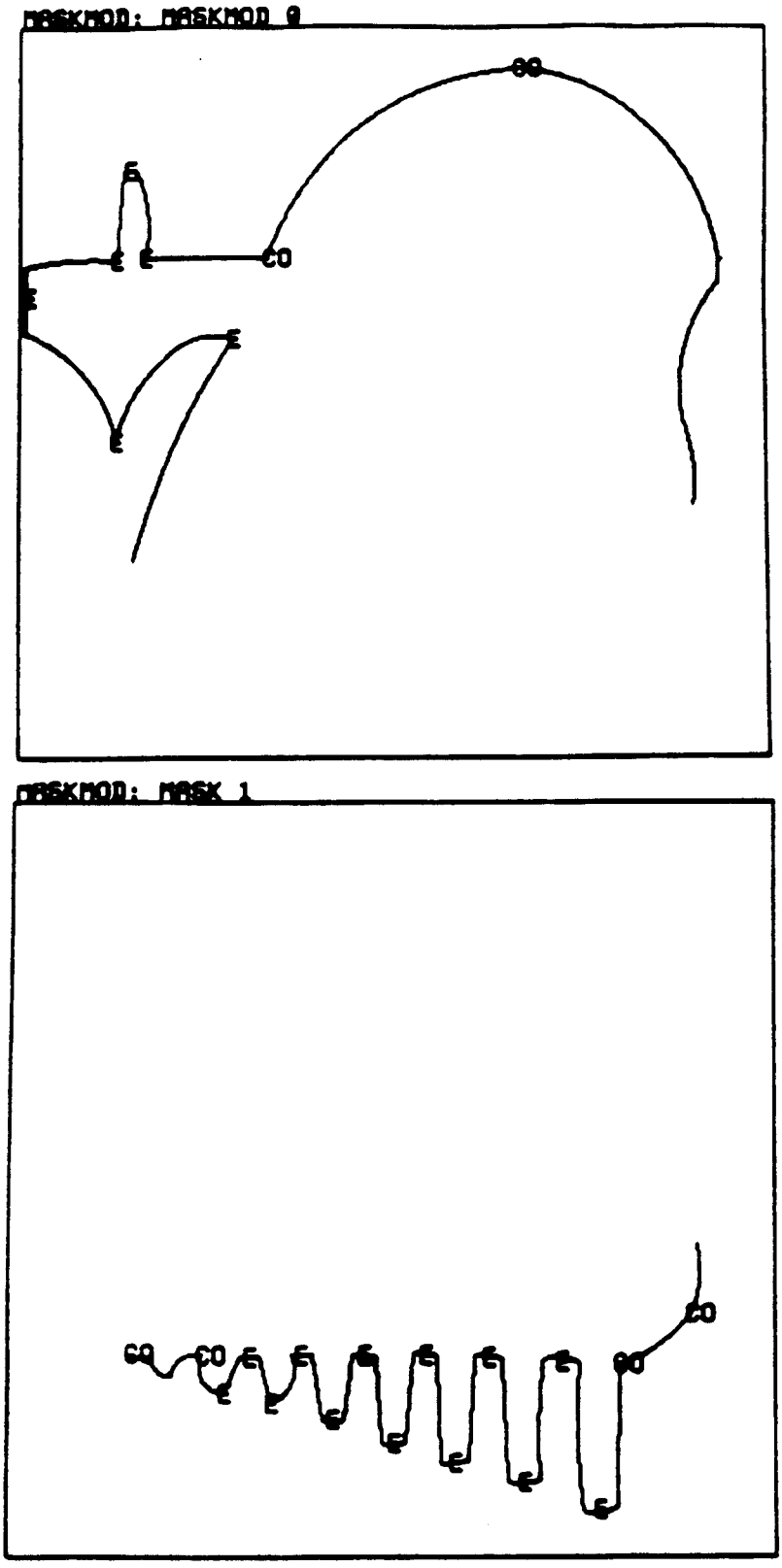


Figure C.5: **Two of the three sub-parts of the *maskmod* model object.**

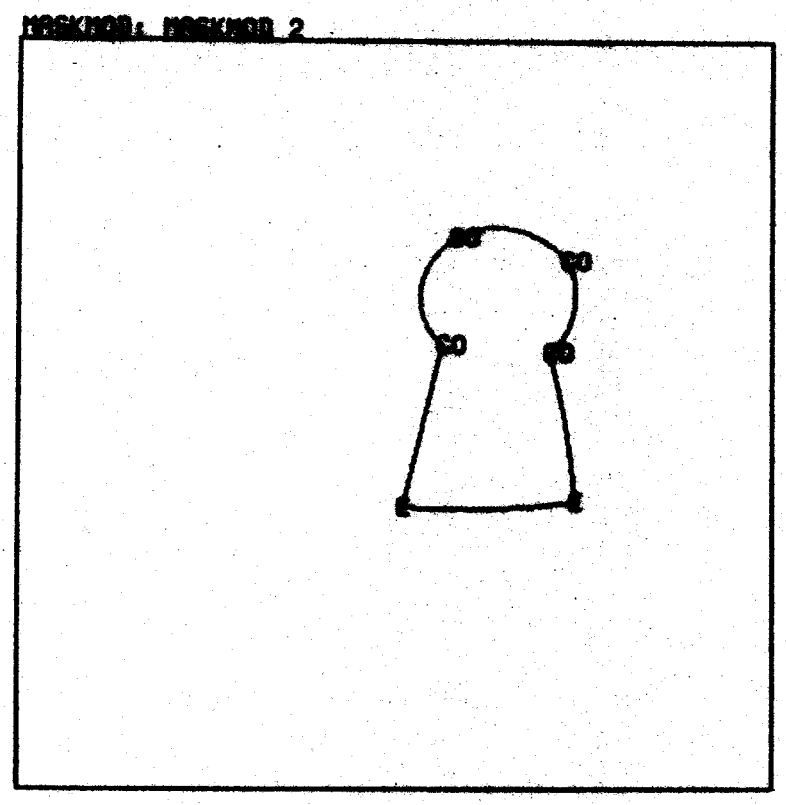


Figure C.6: The third sub-part of the *mashmod* model object.

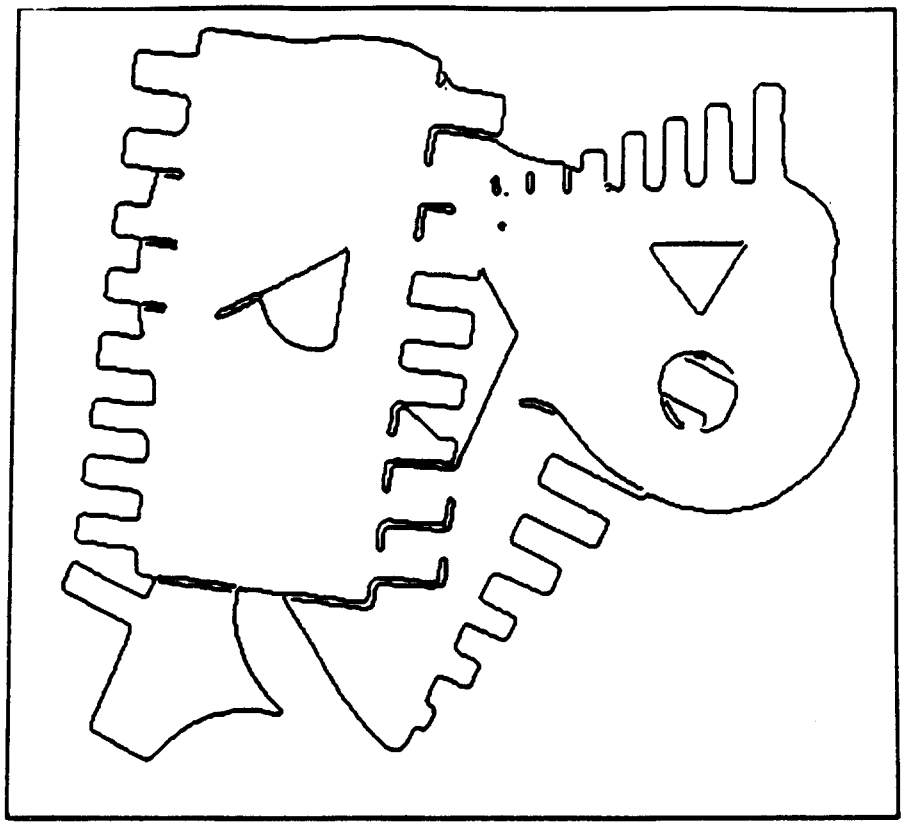


Figure C.7: **A sample scene of a pile of objects. The output of the edge finder is shown.**

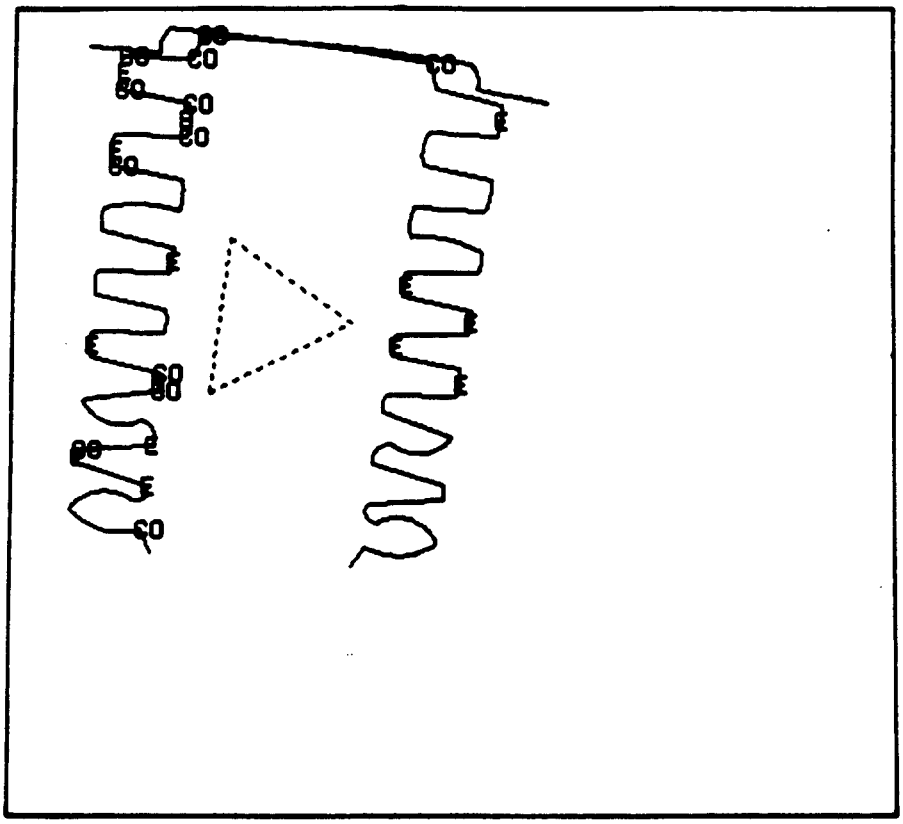


Figure C.8: **The component configuration of the *chip* model object found in the scene.**

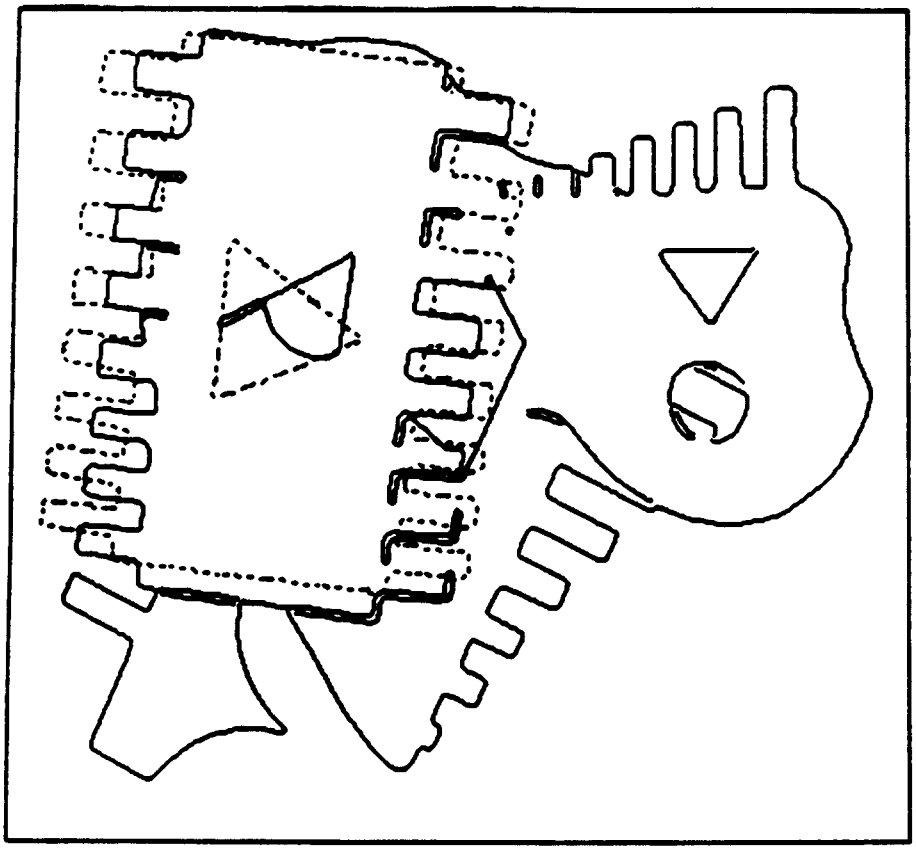


Figure C.9: **Superimposition of the *chip* model object on the scene.**

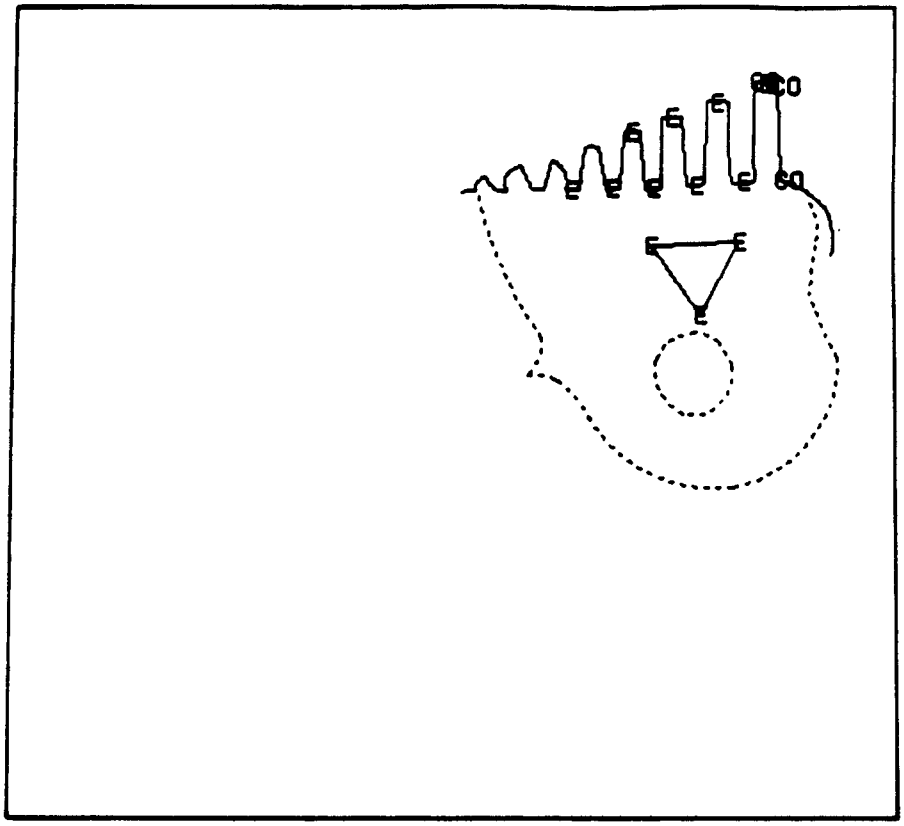


Figure C.10: **The component configuration of the *mask* model object found in the scene.**

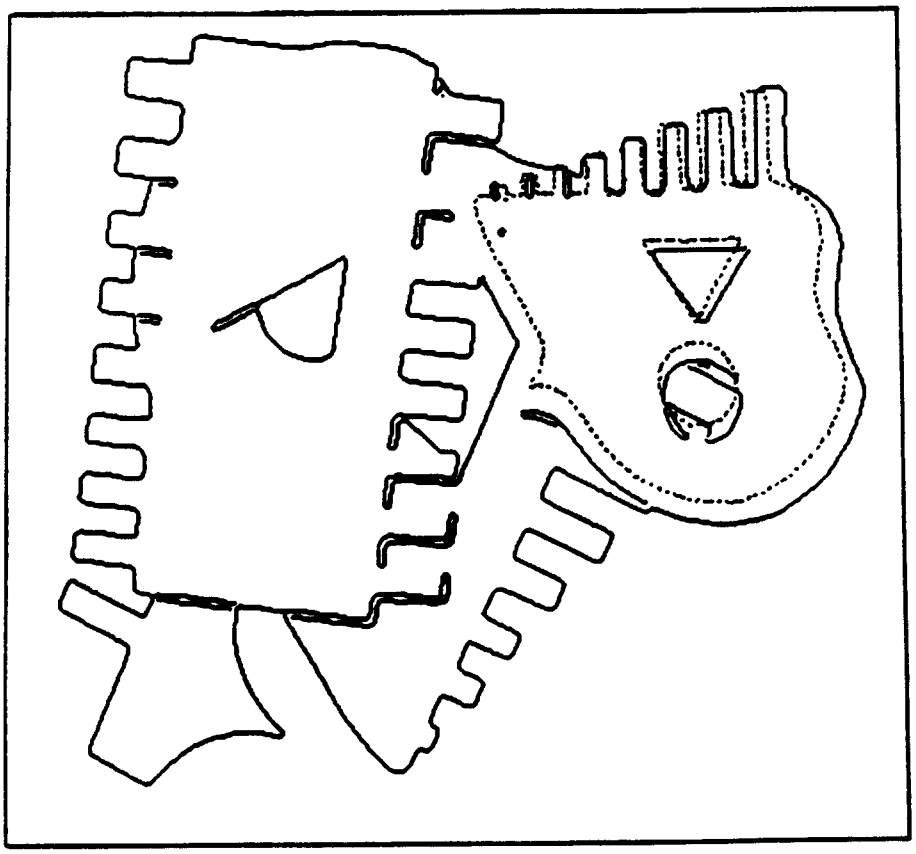


Figure C.11: **Superimposition of the *mask* model object on the scene.**

References

[Asada 86] Haruo Asada and Michael Brady. The Curvature Primal Sketch. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-8(1), 1986.

[Ayache 86] Nicholas Ayache and Olivier D. Faugeras. HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-8(1), 1986.

[Bagley 85] Steve C. Bagley. *Using Models and Axes of Symmetry to Describe Two-Dimensional Polygonal Shapes.* Master's thesis, Massachusetts Institute Technology Dept. of Electrical Engineering and Computer Science, 1985.

[Bajcsy 87] R. Bajcsy and F. Solina. Three-Dimensional Object Representation Revisited. 1987. To appear in Proc. First International Conference on Computer Vision, London, England.

[Ballard 82] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[Barr 81] A. Barr. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*, 1, 1981.

[Barr 84] A. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18(3), 1984.

[Besl 85] Paul J. Besl and Ramesh C. Jain. Three-Dimensional Object Recognition. *Computing Surveys*, 17(1), 1985.

[Bhanu 84(a)] Bir Bhanu and Olivier D. Faugeras. Shape Matching of Two-Dimensional Objects. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-6(2), 1984.

[Bhanu 84(b)] Bir Bhanu. Representation and Shape Matching of 3-D Objects. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-6(3), 1984.

[Binford 82] Thomas O. Binford. Survey of Model-Based Image Analysis Systems. *International Journal of Robotics Research*, 1(1), 1982.

[Bolle 84] Ruud M. Bolle and David B. Cooper. Bayesian Recognition of Local 3-D Shape by Approximating Image Intensity Functions With Quadric Polynomials. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-6(4), 1984.

[Bolles 82] Robert C. Bolles and Ronald A. Cain. Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method. *International Journal of Robotics Research*, 1(3), 1982.

[Bolles 86] Robert C. Bolles and Patrice Horaud. 3DPO: A Three-Dimensional Part Orientation System. *International Journal of Robotics Research*, 5(3), 1986.

[Brady 84] Michael Brady and Haruo Asada. Smoothed Local Symmetries and Their Implementation. *International Journal of Robotics Research*, 3(3), 1984.

[Brady 85] Michael Brady, Jean Ponce, Alan Yuille, and Haruo Asada. *Describing Surfaces.* Memo AIM-822, Massachusetts Institute Technology, 1985.

[Brooks 81] Rodney A. Brooks. Symbolic Reasoning Among 3-D Models and 2-D Images. *Artificial Intelligence*, 17, 1981.

[Brooks 83] Rodney A. Brooks. Model-Based Three-Dimensional Interpretations of Two-Dimensional Images. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-5(2), 1983.

[Brou 84] Philippe Brou. Using the Gaussian Image to Find the Orientation of Objects. *International Journal of Robotics Research*, 3(4), 1984.

[Canny 86] John Francis Canny. A Computational Approach to Edge Detection. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-8(6), 1986. Also in *Finding Edges and Lines in Images*, MIT-AI-TR-720, 1983.

[Clemens 86] David T. Clemens. *The Recognition of Two-Dimensional Modeled Objects in Images.* Master's thesis, Massachusetts Institute Technology Dept. of Electrical Engineering and Computer Science, 1986.

[Connell 85] Jonathan H. Connell. *Learning Shape Descriptions: Generating and Generalizing Models of Visual Objects.* Technical Report AI-TR-853, Massachusetts Institute Technology, 1985.

[Crowley 84] James L. Crowley and Alice C. Parker. A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-6(2), 1984.

[Ettinger 86] Gil Ettinger. Hierarchical Constrained Search Recognition Based on the Curvature Primal Sketch. 1986. Master's thesis proposal, Massachusetts Institute Technology Dept. of Electrical Engineering and Computer Science.

[Faugeras 84] Olivier D. Faugeras, M. Hebert, E. Pauchon, and J. Ponce. Object Representation, Identification, and Positioning from Range Data. In Michael Brady and Richard Paul, editors, *The First International Symposium on Robotics Research*, MIT Press, Cambridge, Massachusetts, 1984.

[Goad 86] Chris Goad. Fast 3D Model-Based Vision. In Alex P. Pentland, editor, *From Pixels to Predicates*, Ablex Publishing Corporation, Norwood, New Jersey, 1986.

[Grimson 84] W. Eric L. Grimson and Tomás Lozano-Pérez. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *International Journal of Robotics Research*, 3(3), 1984.

[Grimson 85] W. Eric L. Grimson and Tomás Lozano-Pérez. *Recognition and Localization of Overlapping Parts From Sparse Data.* Memo AIM-841, Massachusetts Institute Technology, 1985. Also to appear in IEEE PAMI, 1987.

[Grimson 86] W. Eric L. Grimson. The Combinatorics of Local Constraints in Model-Based Recognition and Localization From Sparse Data. *Journal of the ACM*, 33(4), 1986.

[Hebert 84] M. Hebert and T. Kanade. *The 3D-Profile Method for Object Recognition.* Research Note, Carnegie-Mellon University, 1984.

[Heide 84] Scott S. Heide. *A Hierarchical Representation of Shape from Smoothed Local Symmetries.* Master's thesis, Massachusetts Institute Technology Dept. of Electrical Engineering and Computer Science, 1984.

[Hoffman 86] Donald D. Hoffman and Whitman Richards. Parts of Recognition. In Alex P. Pentland, editor, *From Pixels to Predicates*, Ablex Publishing Corporation, Norwood, New Jersey, 1986.

[Hollerbach 75] John M. Hollerbach. *Hierarchical Shape Description of Objects by Selection and Modification of Prototypes.* Technical Report AI-TR-346, Massachusetts Institute Technology, 1975.

[Horn 84] Berthold K. P. Horn. Extended Gaussian Images. *Proceedings of the IEEE*, 22(12), 1984.

[Horn 86] Berthold K. P. Horn. *Robot Vision.* McGraw Hill, New York, 1986.

[Huttenlocher 87] Daniel P. Huttenlocher and Shimon Ullman. *Recognizing Rigid Objects by Aligning Them With an Image.* Memo AIM-937, Massachusetts Institute Technology, 1987. Also to appear in Proc. First International Conference on Computer Vision, London, England, 1987.

[Kalvin 86] Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir. Two Dimensional Model Based Boundary Matching Using Footprints. *International Journal of Robotics Research*, 5(4), 1986.

[Knoll 85] Thomas F. Knoll and Ramesh Jain. *Recognizing Partially Visible Objects Using Feature Indexed Hypotheses.* Technical Report RSD-TR-10-85, University of Michigan, Robot Systems Division, Center for Research on Integrated Manufacturing, 1985.

[Little 85] James Little. Determining Object Attitude From Extended Gaussian Images. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, 1985.

[Lowe 85] David Lowe. *Perceptual Organization and Visual Recognition.* Kluwer, Boston, 1985.

[Marr 78] D. Marr and H. K. Nishihara. Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. *Proceedings of the Royal Society of London*, B 200, 1978.

[Marr 82] David Marr. *Vision.* W. H. Freeman, New York, 1982.

[McKeown 85] David M. McKeown, Jr., Wilson A. Harvey, Jr., and John McDermott. Rule-Based Interpretation of Aerial Imagery. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-7(5), 1985.

[Nagao 79] Makoto Nagao, Takashi Matsuyama, and Hisayuki Mori. Structural Analysis of Complex Aerial Photographs. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, 1979.

[Nevatia 77] Ramakant Nevatia and Thomas O. Binford. Description and Recognition of Curved Objects. *Artificial Intelligence*, 8, 1977.

[Pentland 85] Alex P. Pentland. *Perceptual Organization and the Representation of Natural Form.* Technical Note 357, SRI, 1985.

[Ponce 85] Jean Ponce and Michael Brady. Toward a Surface Primal Sketch. In Takeo Kanade, editor, *Three Dimensional Vision*, Academic Press, New York, 1985.

[Ponce 87(a)] Jean Ponce and David Chelberg. Finding the Limbs and Cusps of Generalized Cylinders. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, 1987.

[Ponce 87(b)] Jean Ponce and David Chelberg. Localized Intersections Computation for Solid Modelling With Straight Homogeneous Generalized Cylinders. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, 1987.

[Richards 84] Whitman Richards and Donald D. Hoffman. *Codon Constraints on Closed 2D Shapes.* Memo AIM-769, Massachusetts Institute Technology, 1984.

[Richards 85] Whitman Richards, Jan J. Koenderink, and Donald D. Hoffman. *Inferring 3D Shapes From 2D Codons.* Memo AIM-840, Massachusetts Institute Technology, 1985.

[Schwartz 85] Jacob T. Schwartz and Micha Sharir. *Identification of Partially Obscured Objects in Two and Three Dimensions by Matching of Noisy Characteristic Curves.* Research Note 46, New York University, Courant Institute of Mathematical Sciences, Robotics Activity, 1985.

[Shafer 76] Glenn A. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, New Jersey, 1976.

[Shapiro 79] Linda G. Shapiro and Robert M. Haralick. Decomposition of Two-Dimensional Shapes by Graph-Theoretic Clustering. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-1(1), 1979.

[Shapiro 82] Linda G. Shapiro and Robert M. Haralick. Organization of Relational Models for Scene Analysis. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-4(6), 1982.

[Tomita 85] Fumiaki Tomita and Takeo Kanade. A 3D Vision System: Generating and Matching Shape Descriptions in Range Images. In Hideo Hanafusa and Hirochika Inoue, editors, *The Second International Symposium on Robotics Research*, MIT Press, Cambridge, Massachusetts, 1985.

[Turney 85] Jerry L. Turney, Trevor N. Mudge, and Richard A. Volz. Recognizing Partially Occluded Parts. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-7(4), 1985.

[Ullman 86] Shimon Ullman. *An Approach to Object Recognition: Aligning Pictorial Descriptions.* Memo AIM-931, Massachusetts Institute Technology, 1986.

[Witkin 86] Andrew Witkin. Scale Space Filtering. In Alex P. Pentland, editor, *From Pixels to Predicates*, Ablex Publishing Corporation, Norwood, New Jersey, 1986.

[Zadeh 65] Lofti A. Zadeh. Fuzzy Sets. *Information and Control*, 8, 1965.

CS-TR Scanning Project
Document Control Form

Date : 10/5/95

Report # AI-TR-963

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 172(179-IMAGES
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form (2) ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number): FOLLOW TITLE & ACK PAGES

Photographs/Tonal Material (by page number): 13, 61, 91, 96, 99, 101, 138, 141, 143, 147, 149

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-172) UN# TITLE, BLK, ACK, BLK PAGES,	3-170
(173-179) SCAN CONTROL COVER, DOD(2), TRGT'S (3)	

Scanning Agent Signoff:

Date Received: 10/5/95 Date Scanned: 10/7/95 Date Returned: 10/12/95

Scanning Agent Signature: Michael W. Cook

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR-963	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-A187476
4. TITLE (and Subtitle) Hierarchical Object Recognition Using Libraries of Parameterized Model Sub-parts		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gil J. Ettinger		8. CONTRACT OR GRANT NUMBER(s) N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE June 1987 13. NUMBER OF PAGES 174
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Machine Vision	Scale Hierarchy
Object Recognition	Parameterized Objects
Model Libraries	Curvature Primal Sketch
Structure Hierarchy	Constrained Search

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis describes the development of a model-based vision system that exploits hierarchies of both object structure and object scale. The focus of the research is to use these hierarchies to achieve robust recognition based on effective organization and indexing schemes for model libraries. The goal of the system is to recognize parameterized instances of non-rigid model objects contained in a large knowledge

(over)

20. base despite the presence of noise and occlusion. Robustness is achieved by developing a system that can recognize viewed objects that are scaled or mirror-image instances of the known models or that contain component sub-parts with different relative scaling, rotation, or translation than in the models. The approach taken in this thesis is to develop an object shape representation that incorporates a component sub-part hierarchy--to allow for efficient and correct indexing into an automatically generated model library as well as for relative parameterization among sub-parts, and a scale hierarchy--to allow for a general to specific recognition procedure. After analysis of the issues and inherent trade-offs in the recognition process, a system is implemented using a representation based on significant contour curvature changes and a recognition engine based on geometric constraints of feature properties. Examples of the system's performance are given, followed by an analysis of the results. In conclusion, the system's benefits and limitations are presented.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**