Frameworks for the Flexible Utilization of Local Features in Image Analysis

Yuichi NAKAMURA

Frameworks for the Flexible Utilization of Local Features in Image Analysis

Yuichi NAKAMURA
Dept. of Electrical Engineering, Kyoto University

Abstract

A great number of Image Processing and Recognition Methods, software programs and systems have been proposed or developed all over the world. Some of them deal with image processing itself, whereas others focus on the recognition or further understanding. Image Recognition and Understanding, however, has remained a quite difficult field to be tackled with.

The research in this thesis places much emphasis on constructing a flexible image analysis system which utilizes local features efficiently. A framework for flexible feature description, a framework for performing image processing and further recognition tasks according to description of features, and a framework for flexible control of these processes are investigated in this research. Parallelism of execution is also an important factor in considering image analysis, since image analysis has a considerable degree of potential parallelism in many forms (spatial parallelism, etc.). Therefore, the realization of parallel execution of image analysis is also focused on.

The research presented in this thesis includes two systems for Image Processing and Recognition. In the first part of this thesis, a system for recognizing overlapping two-dimensional objects (RTS) is described which proposes a methodology for recognizing scenes where already known objects are partially occluded. A method for recognizing the objects by integrating locally residing features that were detected from an image is described in this part. In the second part of this thesis, a system (PAFE) for feature extraction which provides a flexible platform for defining structural features as well as primitive features is proposed for the realization of parallel execution of the feature extraction

processes. A method for the integration of locally residing features is also discussed in this part. The aim of this research is not to select general features that can be applied to all kinds of applications, but to construct a general platform which allows any kinds of features to be used without provoking serious conflicts. In both of the approaches, the integrating of local features, which are fragments, into some kind of meaningful structure such as a part of an object, a complete object or a scene description, is discussed.

In Chapter 2, a system RTS for the recognition of overlapping two dimensional objects is described. The features that are useful even for the partially occluded objects are introduced. The method to synthesize them according to model description and the method to infer object models from combined features are also shown in this Chapter. Moreover, a method to match them to object models is presented.

From Chapter 3 to Chapter 5, a system PAFE is presented. The framework for feature definition, the system's basic structure with the multi agent model and the system's mechanism for parallel execution are described. In Chapter 3, the framework for defining features is shown. In Chapter 4 the organization of the system with the multi agent model is presented. In Chapter 5, a methodology for the extraction of structural features in parallel processing is presented. This method includes parallel feature extraction of bottom-up ways, top-down ways and their combinations. Some experiments are also shown in Chapter 6.

Acknowledgement

First of all, the author wishes to express his deep appreciation to Professor Makoto Nagao for supervising this research and for his continuous encouragement and guidance.

The author also wishes to express his gratitude to Professor Yuji Matsumoto and Professor Satoshi Sato for their stimulating discussions and valuable advice.

The author is grateful to his colleagues in Prof. Nagao's research group for their helpful discussions and co-operation. Especially, the author thanks Mr. Toshiyuki Kamiya and Mr. Kozo Satoda for the development of the application systems based on this research. The feature descriptions and the image processing routines in the experiments with aerial photographs were mainly developed by Mr. Kamiya.

The author is grateful to Mr. Michael Hild for his careful proof-reading of the manuscript of this thesis. He gave the author a number of valuable comments on the contents as well as grammatical corrections.

Last but not least, the author wishes to thank his wife Yukiko for her constant encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A great number of Image Processing and Recognition Methods, software programs and systems have been proposed or developed all over the world. Some of them deal with image processing itself, whereas others focus on the recognition or further understanding. Image Recognition and Understanding, however, has remained a quite difficult field to be tackled with.

One of the major research topics in image recognition is the problem of object modeling, feature definition and feature extraction. A number of research efforts were devoted to defining the features, objects and scenes in the images irrespective of the dimensionality (1D, 2D, 3D, ...). Since it is extremely difficult to find a general description that covers all the kinds of objects and scenes, many specialized methodologies were independently proposed for dealing with many specific targets. Some of them work fairly well for the specific targets to which they are directed. Few of them, however, work for a wide variety of target due to the extreme difficulty of the construction of a general and complete set of features and models. This is the cause for the tremendous amount of work needed for developing methods, programs, and systems, for each target, although the many methodologies, features and feature extraction control structures have many common properties.

Therefore, modularity and portability of the features and objects descrip-

tions should be investigated, even though generality of description can not be achieved at present. A flexible framework is required which can utilize many kinds of object descriptions. For instance, a system is required which can be customized for various types of targets, which may be partially occluded objects defined by arbitrary structural features and varying with time.

At the same time, a framework is required which provides automatic extraction of features according to the feature definitions provided by the user or other systems. This implies the need for feature extraction planning and execution control.

In this thesis, two systems for Image Processing and Recognition are presented. In the first part, a system for recognizing overlapping two-dimensional objects (RTS) is described which proposes a methodology for recognizing scenes where already known objects are partially occluded. A method for recognizing the objects by integrating locally residing features that were detected from an image is described in this part. In the second part, a system (PAFE) for feature extraction which provides a flexible platform for defining structural features as well as primitive features is proposed for the realization of parallel execution of the feature extraction processes. A method for the integration of locally residing features is also discussed in this part. The aim of this research is not to select general features that can be applied to all kinds of applications, but to construct a general platform which allows any kinds of features to be used without provoking serious conflicts. In both of the approaches, the integrating of local features, which are fragments, into some kind of meaningful structure such as a part of an object, a complete object or a scene description, is discussed.

1.1 Recognition of Overlapping Objects

Recognizing overlapped and occluded objects is one of the most important topics in automation and robot vision systems. Not only occlusion but also the incompleteness of image processing techniques lead to the same situations

in which the features of objects are not completely extracted. Therefore, the recognition of partially occluded objects is quite important and widely related to other tasks in the image recognition field.

When occlusion occurs, recognition becomes difficult due to two reasons. One is the problem of *segmentation* and the other one is *partial matching*. Most of the traditional pattern recognition approaches are too weak to cope with these problems, since they require that the unknown objects be extracted before their feature values can be measured. For instance, topological features, central moment, area, or perimeter used in such approaches are not preserved when the objects are partially occluded.

Recently some methods were proposed for tackling these problems [Tro81, Hae82, AF86]. Certain sets of local features are used to define objects. In most cases, they are straight lines and circles (sometimes ellipses). These methods try to match a set of local features in the image with a set of local features of a model. Though they work fairly well for the problem of partial matching, they have no proper means for effective selection of candidate models. In other words, no means for inferring models from features extracted from the image are provided. Therefore, the cost of processing increases exponentially with the number of models.

A new method for the recognition of partially occluded 2-D objects is proposed in this thesis. The research focuses on the problem of inferring models from a set of local features. This method considers a set of corners, parallel lines and so on as typical features of objects. Possible candidate models are estimated from these features, and structural matching is performed between these models and the features obtained from a picture by constructing larger structures that are the combinations of various features. Even if the whole structure is not obtained due to partial occlusion, the system can infer an object if some unique features of the object are obtained. With this method, recognition of 2-D objects becomes easier because the number of exact matchings performed in a recognition process becomes smaller.

1.2 Parallel Feature Extraction System (PAFE)

Many systems have been proposed for image recognition, analysis and understanding. There are systems that interprets aerial photographs with blackboard model by Nagao and Matsuyama [NM80], ACRONYM [BGB79] for the interpretation of 3D scenes by Brooks, and a system for the understanding of outdoor scenes by Ohta [Oht87]. Recently, a number of more sophisticated experimental systems have been constructed for different application domains, such as the interpretation of high-altitude aerial photographs [NM80, FP81, DMM85, MH90] including airport scenes; and for outdoor scenes [Oht87, HR78, Dea89, AK87].

Some of them are based on the production system or the blackboard model [Nii86a, Nii86b]. In these models, data, i.e. images and features extracted from them, are stored in shared memory which can be accessed by many other modules. The rules or procedures, which are pieces of knowledge about the target objects and knowledge about the operations to be performed in the systems, process and modify the data in the shared memory. These frameworks provide quite flexible platforms for image interpretation, since various kinds of data and knowledge can be handled in the same framework. While success has been demonstrated to various degrees, developing a domain-independent and systematic framework for constructing knowledge-based image interpretation systems is still an open problem.

One serious problem left open by the above researches is modularity of object modeling and feature definition. In other words, knowledge about objects (features, models etc .), procedures for extracting features and control strategies for the recognition process are often mixed in both implicit and explicit ways in the programs, rules and pieces of knowledge. A large number of very complicated programs and rules were developed for each specific task, which cannot be applied to other domains. Therefore a framework for defining features, objects, scenes, feature extraction and object recognition is still

needed.

Another serious problem is the maintenance of consistency and parallelism of execution. Many algorithms and systems, including special hardware, have been proposed for low level image processing, since the computations at low level are well suited for parallel processing. On the contrary, problems of in connection with intermediate and high level image analysis such as the extraction and recognition of structural features have not been well thoroughly investigated, although some systems have been reported [Dea89, Oht91]. In this sense, parallel processing has not received enough attention in the image interpretation systems mentioned above. Since a large number of rules and procedures have to be integrated in each system in various forms, it is quite difficult to maintain consistency during execution. One rule may rewrite the data into A, while another rule may rewrite the same data into B causing complicated conflicts. At the stage of high level recognition, the data including intermediate results are mutually depending on each other. Therefore, it it extraordinarily difficult to maintain consistency with approaches such as the blackboard model.

Nevertheless, there are many areas of application for parallel processing in intermediate level processes, i.e. feature extraction or feature synthesis. The features can be hypothesized, detected and discriminated with considerable amount of parallelism. A great amount of effort should be invested into this research topic.

In this thesis, a system called PAFE, which provides a flexible platform for feature extraction, is presented.

There are three objectives in this system:

- A flexible framework for feature definition
- Coarse grained parallelism
- Flexible control of feature extraction with multiple agents

To achieve these objectives, modularity of feature definition, feature extraction and control is an essential factor. In this system, features are defined

in a structural and hierarchical way. Features and their extraction methods are organized in a feature extraction network. In this network, features can be defined and extracted in multiple ways. Various kinds of features can be defined with the framework proposed for the PAFE system.

This system is constructed based on the multi agent model, in which many agents work co-operatively and concurrently. In this sense, the structural definition of the features plays an important role also for the realization of parallel execution. The modularity of the hierarchical definition of features makes it possible that the extraction of higher level features can be divided into subproblems of extraction of lower level features. One of the objectives of this system, coarse grained parallelism, can be realized by the parallel execution of multiple extraction methods for one kind of feature in parallel or by the extraction of multiple kinds of features in parallel. Also another objective of flexible feature extraction is achievable through the combinations of top-down extraction and bottom-up extraction performed by many agents.

The efficiency of the proposed framework was verified in some experiments in which this system was applied to some 2-dimensional objects.

1.3 Outline of the thesis

In Chapter 2, a system RTS for the recognition of overlapping two-dimensional objects is described. Features that are useful even in the presence of partially occluded objects are introduced. A method for synthesizing such features according to model descriptions and a method for inferring object models from combined features are also shown in this Chapter. Moreover, a method for matching these features to object models is presented.

From Chapter 3 to Chapter 5, the PAFE system is presented. The framework for feature definition, the system's basic structure together with the multi agent model and the system's mechanism for parallel execution are described. In Chapter 3, the framework for defining features is shown. In Chapter 4 the organization of the system on the basis of the multi agent model is pre-

sented. In Chapter 5, a methodology for the extraction of structural features suitable for parallel processing is presented. This method includes parallel feature extraction in bottom-up mode, top-down mode and their combinations. Experimental results and the evaluation are given in Chapter 6.

Chapter 2

Recognition of Overlapping Objects

In this chapter, a system (RTS) for the recognition of overlapping objects is presented. The objectives of this system are the selection of features that are useful in such situations and to develop an efficient method for the recognition of partially occluded objects.

2.1 Introduction

Recognizing overlapped and occluded objects is a very important problem in automation and robot vision systems. In many situations, objects touch, overlap and occlude each other. For example, the task of bin-picking requires the recognition of scenes where work-pieces are scattered in a bin. In such situations, recognition is difficult for two reasons. One is the problem of (1)*segmentation* and the other one is the problem of (2)*partial matching*. Many of the traditional pattern recognition approaches are too weak to handle these problems, since they require complete objects to be extracted in order to measure their feature values. For instance, topological features, central moment, area and perimeter used by these methods are not preserved when the objects are

partially occluded.

Recently some methods were proposed for tackling these problems. They work generally well for partial matching of a partly occluded object with a model [Hae82, Tro81, AF86]. Several research reports [SH86, RB84, Rum86] describe how the recognition process can be implemented by tree search. They focus on one feature (straight line, circle, etc.) and check whether it possibly matches a model feature. If it does, the focus shifts to extracting the next features to be compared again with the model features. Finally the best matching is determined. However, these approaches have no proper means for efficient selection of candidate models. In other words, these methods have no efficient means for selecting models from features extracted from an image without matching. For instance, the researches reported in [Hae82, Tro81, AF86] check every possible correspondence between a line in the image and a line in a model until a valid one is found. Therefore the larger the number of models is, the higher the cost of processing becomes (exponential growth in the worst case).

To solve this problem, we propose a new method of inferring models of an object from its parts. In this method, several kinds of local features and relationships between them are utilized to express an object as well as a set of primitive features (that is, straight lines). The uniqueness of the candidate model which matches the recognition object is verified by building larger structures thorough combinations of those local features. More concretely, a 2-D object is considered as a set of local topological features such as corners and parallel lines. Partial models are inferred from these features. Then by combining these features according to local topological relations of lines, the number of the candidate models is reduced to a small number. When the set of candidate models is reduced to a single model, detailed matching between the model and lines extracted from the image is performed.

The outline of our system is shown in Fig. 2.1. The system has two kinds of data of the recognition objects. The first kind is data of individual models consisting of lines and features of the model, and the second kind is data of the *feature-to-model-table* (an inverted table derived from the first data set) which

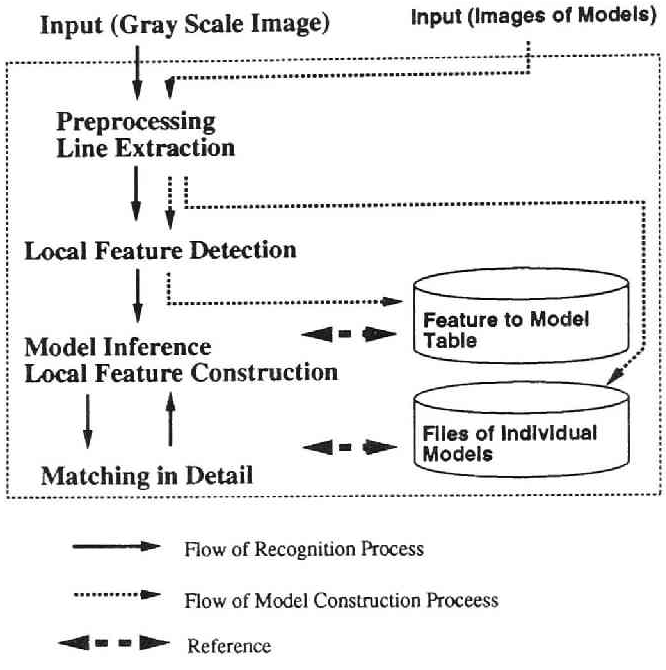


Figure 2.1: Overview of the Recognition System

contains all features gathered from all models. The data in the feature-to-model-table is referred to in the model inference step, while the data containing the individual models is referred to in the detailed matching step. The recognition process consists of three steps and is performed as shown in Fig. 2.1. The structural model description and the modeling process are semi-automatic as is indicated by the dotted arrows in Fig. 2.1.

With this system, we can manage many models efficiently. Flexible recognition can be achieved, because the most probable candidates are checked first. The efficiency of this method was tested by recognition experiments with 2-D objects such as pliers, screwdrivers and so on.

2.2 Features and Model Description

The primitive feature used in this system is the straight line, since extraction of straight lines is relatively easy compared with other features such as curvature, regions or textures. The local features utilized in this system are composed of straight lines. Curves are approximated as a set of straight lines.

In this research, the camera is assumed to be right above the objects, but the camera-to-object distance is unknown. The essential data of a straight line is a pair of starting point and ending point positions. This data is too weak for inferring corresponding candidate models, since rotation and scale changes do not preserve these values. More complicated structures than the straight line are needed for this model inference problem; for example, corners, parallel lines or even more complicated structure must be utilized.

Moreover, only relative characteristics of model features should be used for dealing with scale changes. In other words, objects should be considered as a set of relative characteristics of line combinations.

Features

Because of the constrained camera location, only rotation and scale changes must be considered. This implies that four degrees of freedom are to be handled. Since a corner and a parallel line pair consist of two lines (which means four points [1]), they are theoretically sufficient for inferring a model. When occlusion occurs, however, false end points may be detected. In this situation, even a corner or a parallel line pair are not sufficient.

On the other hand, inner-angles of corners are preserved under rotation, scale changes and occlusion, although they may not be sufficient to uniquely determine candidate models. Many of them remain unchanged even when objects are partially occluded. The aspect ratio of the parallel lines is preserved

[1] In reality, a corner has three points because the corner point should be treated as one point.

under rotation and scale changes, but it is not preserved in the case of occlusion. However, approximate correspondences may be found, because the aspect ratio never increases in any case. In other words, models with larger aspect ratio than the parallel lines in the image can be considered as candidate models.

Therefore, corners and parallel lines are used as significant features in our system. As for parallel lines, if more than three lines are parallel to each other, they are treated as a third kind of feature (multiple parallel lines).

The characteristics that represent the features used in the system can be summarized as follows.

Corner: The characteristic used to represent a corner is the inner angle.

Parallel lines: The characteristic is the aspect-ratio of the region which is bounded by the parallel lines. The aspect-ratio is calculated as follows:

$$\text{aspect-ratio} = 2d/(l_1 + l_2)$$

where d is the distance between the two lines, l_1 and l_2 are the lengths of the two lines.

Only these features with their characteristics are used to infer the local features of models.

Relations

Many relationships which are preserved because of the particular camera arrangement are possible, since a combination of two or more than two lines has more than four degrees of freedom. From these relationships, we chose four relations which commonly appear between lines in the objects. They are (1) *connected*, (2) *parallel*, (3) *collinear* and (4) *adjacent* (shown in Fig. 2.2). They appear in an image even when objects are partially occluded. The lines and features which satisfy these relations have a high probability of belonging to the same object. The relations (1) and (2) are used for detecting local features (corner and parallel lines, respectively). The two other relations are used to combine features in the model inference step.

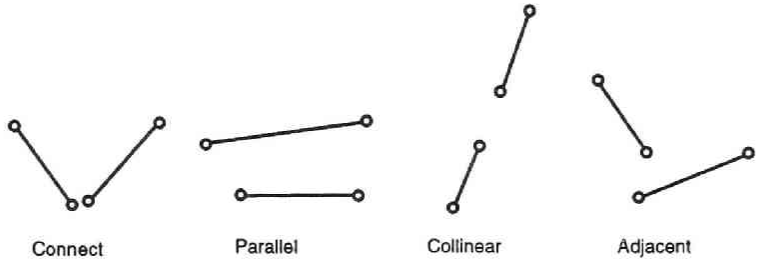


Figure 2.2: Relations of Lines

Model Description

The structural description of models is composed of two parts as mentioned before (shown in Fig. 2.3).

1. A set of lines : positional data of component lines
2. A set of features : corners, parallel lines and multiple parallel lines.

The first part is the positional data of the component lines. Each model has its own coordinate system, with respect to which the locations of lines are determined. It is not necessary that the lines are identical with object contours (that is, object boundaries) and that they are connected to each other end-to-end. The second part consists of the features of the models, i.e. corners or parallel lines.

Feature-to-model-table

When a feature is extracted from an image during the recognition process, the objective of the recognition process is to find those models that possibly have this feature. It is necessary to search for models whose features' characteristics are consistent with the features that were obtained from the image. For this purpose we prepared the *feature-to-model-table*. In this table, all features of

```
[MODEL DESCRIPTION]

<MODEL>::=(<LINE DATA>,<FEATURE DATA>)
<LINE DATA> ::= LIST OF LINE'S ATTRIBUTES (POSITION DATA)
<FEATURE DATA>::=(<CORNER LIST>,<PARALLEL-LINES LIST>,
                        <MULTI-PARALLEL-LINES LIST>)

[EXAMPLE]

NAME: MODEL-2
```

```
;; <<LINE-DATA>>
;; (1) LINE-ATTRIBUTE-LIST:
;;      <line>::=(<name>,<x1>,<y1>,<x2>,<y2>,<length>,<orientation>)

        ((LINE1 -99.375 -11.875 -128.375 47.125 65.74192 116.17529)
         (LINE2 153.625 -3.875 -128.375 46.125 286.39832 169.94566)
         (LINE3 72.625 -31.875 154.625 -2.875 86.977005 19.476574)
         (LINE4 72.625 -31.875 -97.375 -10.875 171.29214 172.95795))

;; <<FEATURE2-3DATA>>
;;(2) CORNER-LIST:
;;     <corner>::=(<angle>,<model-name>,<line1>,<line2>,<type>)

        ((29.530884 MODEL-2 (LINE2) (LINE3) 2)
         (153.48137 MODEL-2 (LINE3) (LINE4) 1)
         (123.21735 MODEL-2 (LINE4) (LINE1) 3)
         (53.770386 MODEL-2 (LINE1) (LINE2) 4))

;;(3) PARALLEL-LINES-LIST:
;;     <parallel-lines>::=(<aspect-ratio>,<model-name>,<line1>,<line2>)

        ((4.9520884 MODEL-2 (LINE2) (LINE4)))
;;(4) MULTI-PARALLEL-LINES-LIST:
;;     <mp-group>::=(<number-of-lines>,<parameters>,<model-name>,<lines>)

       nil
```

Figure 2.3: Model Description

all models are sorted by their characteristics. For example, Fig. 2.4 illustrates a feature-to-model-table for corners. This table is modified when the set of models is changed (addition, deletion or modification of models).

2.3 Preprocessing, Line Extraction and Grouping

The flow of the model inference process is shown in Fig. 2.5. This process consists of two parts. The first one is (A) feature extraction and grouping of lines, and the second one is (B) model inference by local feature construction. The former part is described in this section and the latter part is described in the next section.

Preprocessing

Pictures of the recognition objects are taken by TV-camera and are fed into the image processing unit. Subsequently the following operations are performed sequentially: (1) smoothing by averaging filtering, (2) differentiation by Sobel operator, (3) binarization by thresholding and (4) noise elimination by region growing and region reduction. For example, the binary image shown in Fig. 2.7 was computed from the original image shown in Fig. 2.6.

Line Extraction

After preprocessing, lines are extracted from the binary image by the *variable size slit method*. At first, a large slit is applied to the whole picture and projection curves are obtained. If there are some sharp peaks in the projection curves which exceed a certain threshold, lines may exist in this part of the image along the slit. Then, several narrow slits as shown in Fig. 2.8 are applied to these parts to investigate whether lines exist or not. By repeating these operations

```
[DESCRIPTION]

<feature-to-model-table>::=(<model list>,<data list>)
<model list> : list of models
<data list>  : list of features extracted from models
<feature>::=(<property(single value)>,
             <model-name>,<component-lines>)

[EXAMPLE]

FEATURE-TO-MODEL-TABLE FOR CORNER

;; <<MODEL-LIST>>

(KNIFE HEXAGON WIRE2 WIRE YATTOKO CAP PEN-CAP PLIERS CLIP RIBBON)

;; <<DATA-LIST>>
;; <corner>::=(<angle>,<model-name>,<line1>,<line2>)

        ((1.1864014 YATTOKO (LINE10) (LINE1))
         ..............................
         (84.41774 CAP (LINE8) (LINE1))
         (84.79385 PLIERS (LINE11) (LINE10))
         (86.84451 HEXAGON (LINE3) (LINE7))
         (88.45166 KNIFE (LINE6) (LINE7))
         (89.71416 KNIFE (LINE7) (LINE1))
         (89.97446 RIBBON (LINE11) (LINE10))
         (90.80713 CLIP (LINE2) (LINE3))
         (90.86807 RIBBON (LINE10) (LINE9))
         ..............................
         (332.1392 YATTOKO (LINE8) (LINE9))
         (335.8559 WIRE (LINE9) (LINE10)))
```

Figure 2.4: Feature-to-Model Table

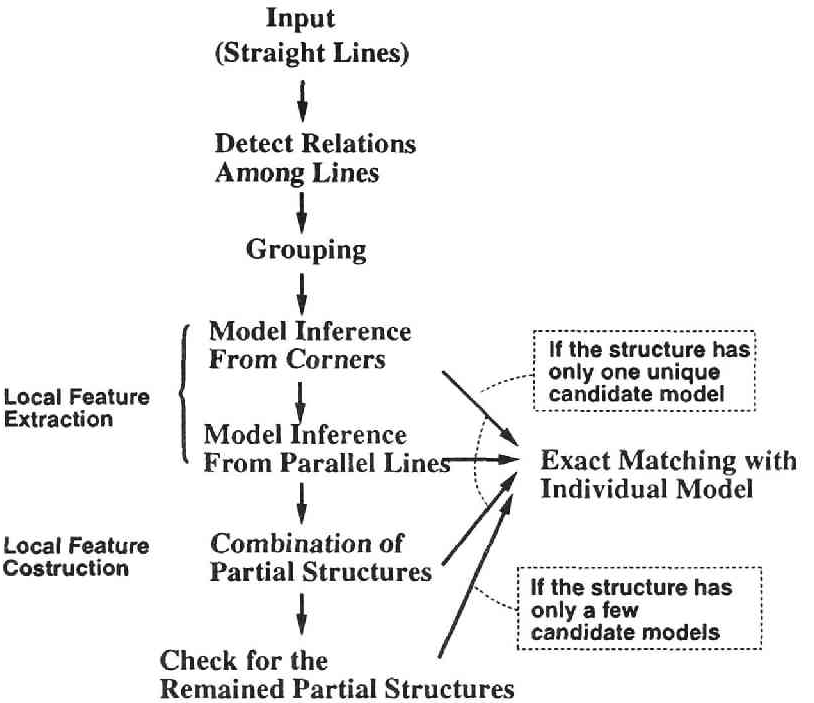


Figure 2.5: Flow of Recognition Process

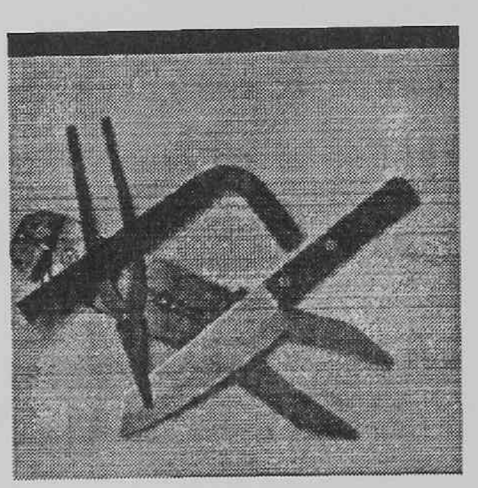


Figure 2.6: Original Image

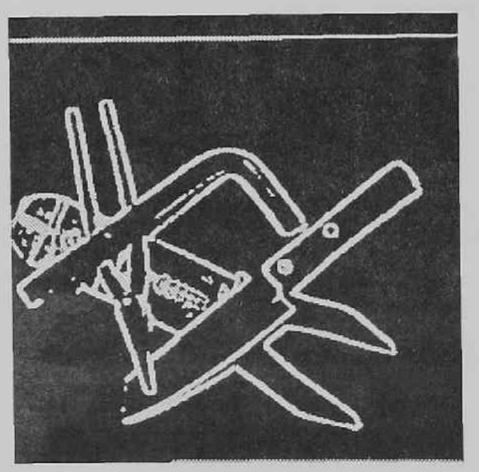


Figure 2.7: Differentiated Image

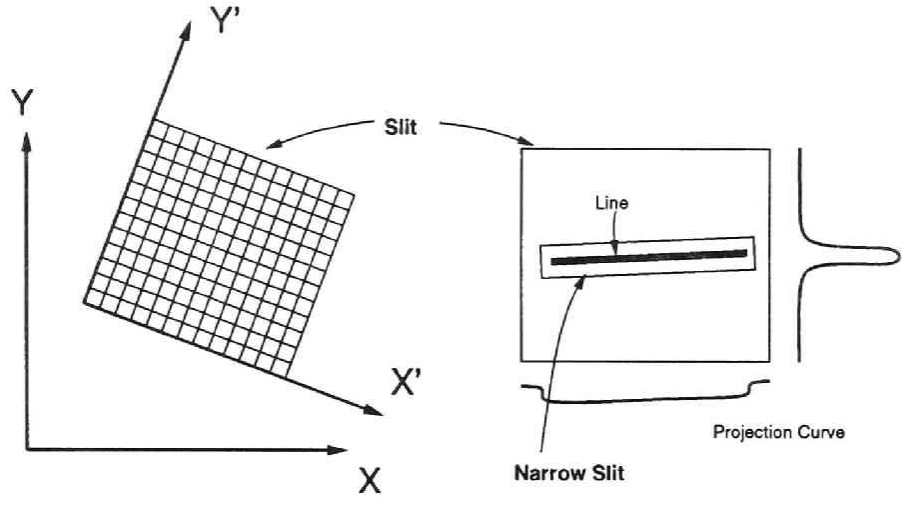


Figure 2.8: Line Extraction by Slit Method

with changed directions of the initial large slit, lines in every direction and of a certain minimum length can be extracted.

The result of line extraction applied to Fig. 2.7 is shown in Fig. 2.9.

Relation Detection and Grouping of Lines

Line relations are detected by checking whether a combination of two extracted lines satisfies any one of the four relation criteria defined in Section 2.2.

Next, those lines which are identified as being related are grouped. As shown in Fig. 2.10, lines are properly divided into groups, if reliable contours of objects are extracted. However, it is often difficult in a real image to divide them into proper groups, owing to occlusion, shadows and the imperfectness of the line extraction process. Since lines, which are grouped together, may be component lines of a single object, they are not further split up.

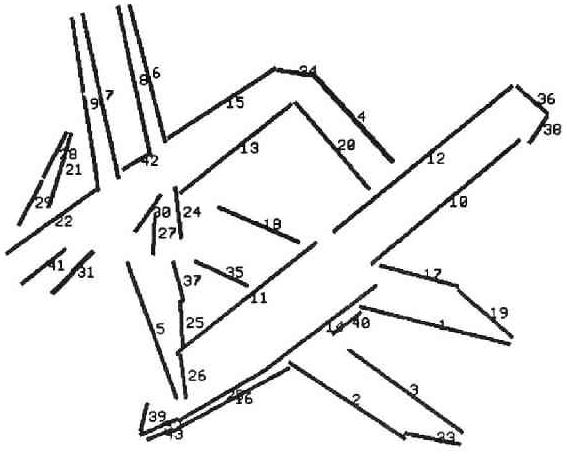
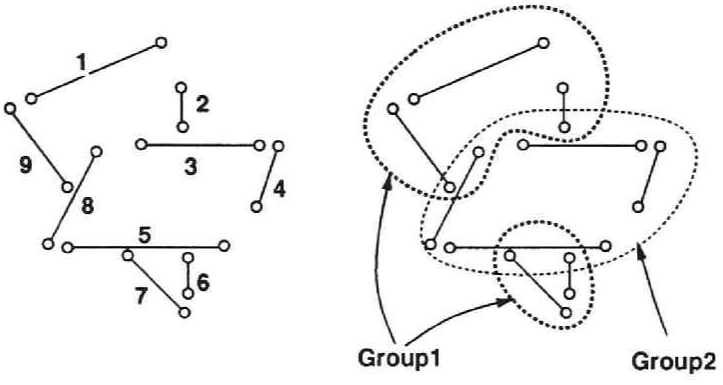


Figure 2.9: Result of Line Extraction

2.4 Model Inference by Local Feature Construction

For each group formed in the previous step, candidate models are inferred according to the steps indicated in Fig. 2.5. Since small structures (local features) are too weak for inferring candidate models uniquely, larger structures are generated by repeatedly combining those local features during the recognition process. The following terms are used in this section (illustrated in Fig. 2.11).

Partial structure A set of features composed of lines which are extracted from an image. Corners and parallel lines are the smallest partial structures. A combination (union) of partial structures is also a partial structure.




```
Relation between lines:

 connected  : (1 9), (3 4), (5 8), (6 7)
   parallel : (3 5), (4 8)
  collinear : (2 6), (7 9)

Result of Grouping:

  Group1 = { 1, 2, 6, 7, 9 }
  Group2 = { 3, 4, 5, 8 }
```

Figure 2.10: Example of Grouping Procedure

Partial model A set of model features. Corners and parallel lines in models are the smallest partial models.

Hypothesis A hypothesis about a correspondence between a partial structure and a partial model. Hypotheses are attached to each partial structure. This means that candidate models of a partial structure are held by the hypotheses attached to this partial structure.

Construction An operation for combining partial structures into larger partial structures.

With the above notation, the purpose of the recognition step is expressed as the construction of larger *partial structures* whose candidate *partial models* amount to a very small number. Construction is performed based on relations defined in the preceding section.

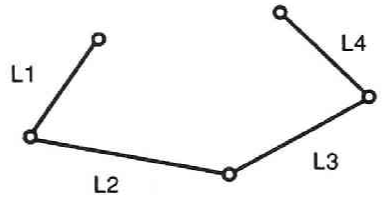
First, partial models (i.e. corners and parallel lines in models) are inferred for each feature. Since the features used in this research do not uniquely distinguish models but are common to many models, there are often many partial models consistent with the features. This implies that the correct model of an object cannot be derived from a single feature in most cases. Therefore a combinations of various kinds of features is necessary for deriving the correct model. Second, in our method, the consistency of partial models suggested by the features is checked. If they are consistent, the features are combined into one compound feature (this step is called construction). By doing this, the candidate models can be reduced to a set containing only the most probable ones. These combinations of features are tested in the order given by the probabilities that the lines belong to the same object (described in Section 2.2). When the number of candidate models is reduced to one, detailed matching is performed directly between lines extracted from the image and lines in that model.

```
[Partial Structure]

<partial structure> ::= (<SL>, <HYP>)
<SL> ::= A set of lines extracted from an image
<HYP>::= A set of hypotheses of correspondence
        between <SL> and <partial model>s

<partial model> ::= A set of lines in models

[Example]
```




```
SL  = {L1, L2, L3, L4}
HYP = {PM1, PM2, ...., PMn}

where
        PM1 = {M11, M12, M13, M14}
        PM2 = {....}
        ............

"M*" indicates a line in models,
"PM*" indicates a partial model.
```

Figure 2.11: Example of a Partial Structure

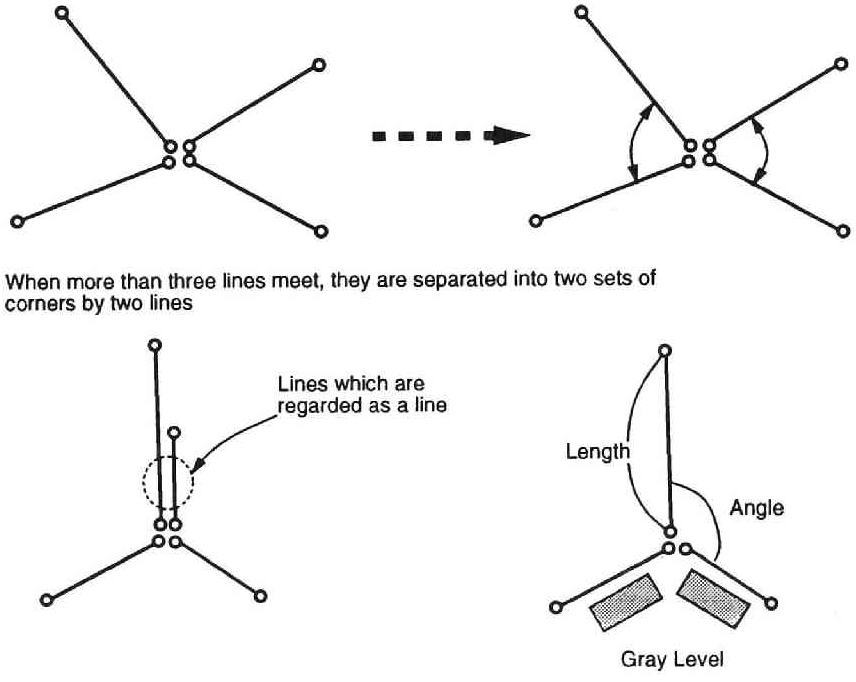


Figure 2.12: Division of Junction

2.4.1 Model Inference from Corners

Partial models are inferred from corners extracted from an image as follows. At first, the corners are generated as the smallest *partial structures*. For each partial structure, a set of hypotheses (i .e. hypothetical correspondences between lines included in the partial structure and candidate partial models) is given.

- Complex junctions are divided into sets of line pairs on the basis of judgements about the length of the lines, inner angles of the pairs and average gray-levels of certain regions (as shown in Fig. 2.12). By doing this, complicated junctions are modified into a set of simple corners for which

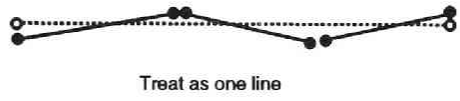


Figure 2.13: A sequence of lines are approximatedby a straight line

candidate corners in models can be inferred.

- Lines which form an obtuse corner (angle close to 180°) are re-labeled as a single line as shown in Fig. 2.13. This makes it easy to match sets of lines which approximate smooth curves (as shown in Fig. 2.14, if the scale changes the lines are extracted differently.).

- The *inner side* of each sequence of connected lines including corners is determined by judgements about the average gray-levels of small regions along these lines (as shown in Fig. 2.15) and the sum of the inner angle for each side. The side whose gray-level is approximately the same as the gray-level of the background is regarded as the *outer* side. If the gray-scales of both sides are not close to the level of the background, the side which has the bigger sum of inner angles is regarded as the outer side.

- Finally, partial structures are created based on corners (one partial structure for each corner). Corresponding partial models are suggested by referring to the feature-to-model-table. These partial models are the corners of models which have inner angles within some tolerance of the inner angle of the corner in the image. The correspondence between the partial structure and the partial models is attached as a hypothesis to each partial structure.

These operations are sufficient for the planar objects used in this research, which consist of straight lines and do not contain many lines in their inner regions.

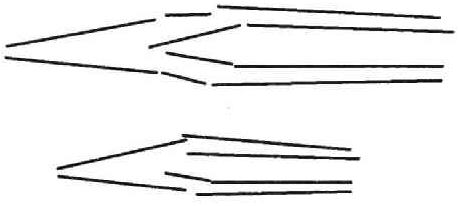


Figure 2.14: Example Line Extraction Result

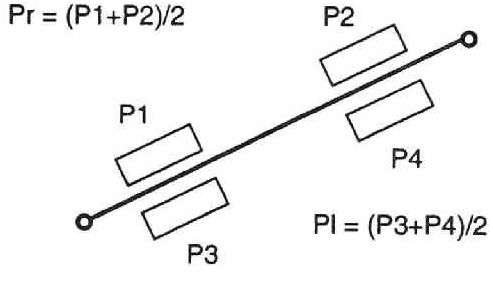


Figure 2.15: Gray level at both side of a line

2.4.2 Construction of Corners

Combinations of corners are considered in this step. If more than two corners exist in neighboring positions (this means that two corners share a line), the corners are combined, if the candidate partial models corresponding to the image-space corners are consistent.

For example, assume that in the image $corner_1$ (consisting of L_1 and L_2) has a corresponding set of candidate models called $a\_set\_of\_model\_corners_1$ and that $corner_2$ (consisting of L_2 and L_3) has $a\_set\_of\_model\_corners_2$. When we consider the combination of $corner_1$ and $corner_2$, the combined partial structure will consist of L_1, L_2 and L_3. In this case, the pair of partial models which share the same line can be a candidate for the combination of $corner_1$ and $corner_2$. Now suppose that a partial model $model\_corner_1$ consists of M_1 and M_2, another partial model $model\_corner_2$ consists of M_2 and M_3. The combination of those partial models is consistent because of the common model line M_2, so it can become a new partial model consisting of M_1, M_2 and M_3.

If there are such models, a new partial structure for the combined corners is created (as shown in Fig. 2.11). In this operation, if a set of lines in one partial structure implies a set of lines in another partial structure, the implied partial structure is deleted (only the biggest one is kept). Even if some partial structures are inconsistent, they can co-exist. The reason for this is that the evidence from the partial structures may not be strong enough to determine which of the structures is correct.

2.4.3 Model Inference from Parallel Lines

Partial structures are generated from parallel lines in this step. Candidate partial models are also inferred and hypotheses are attached to each partial structure. Since lines may be shortened owing to occlusion, parallel lines in the model, which have larger aspect ratios than parallel lines in the image, are treated as candidate partial models. Therefore parallel lines which have small aspect ratio have many candidate models, and therefore only those parallel

lines, which have large a aspect ratios, are used in this step.

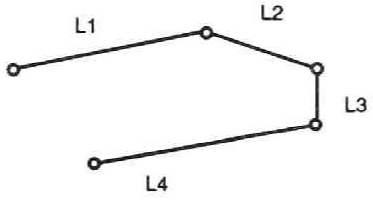
Partial structures are generated from multiple parallel lines in the same way. First, a partial structure is created for each multiple-parallel-lines, and then hypotheses are attached to it.

Moreover, if a partial structure exists which contains both lines of a parallel-lines, the consistency between candidate models of parallel lines and those of this partial structure is checked. Then, the number of hypotheses for this partial structure is reduced (as shown in Fig. 2.16).

2.4.4 Construction of Partial Structures

Since in most cases partial structures generated as described in the previous steps have many candidate models, further combinations of partial structures are attempted to reduce the number of their candidate models. The conditions for generating combinations are the relations between lines mentioned before (section 2) and the directional consistency of partial structures as shown in Fig. 2.17. For example, in the case of the *co-linear* relation, if end line segments of two partial structures are co-linear and the inner regions of the two partial structures (determined in the previous steps) are on the same side of the two lines as shown in Fig. 2.17(a), they are combined into a bigger partial structure. In this way, bigger partial structures, whose lines are union of the lines of their component partial structures and whose candidate models are the intersection of the lines of the candidate models of the components, are generated repeatedly.

The operations are as follows. At first, the relations extracted in the feature extraction step are selected in the order *co-linear*, *parallel* and *adjacent*. Partial structures including these selected relations are searched for each pair of lines which satisfy the relations. For two sets of partial structures obtained in this way, each combination of the partial structures included in different sets are checked. Then the consistency of the candidate partial models are checked and a new partial structure is generated from them. If two partial structures in the



1. Candidate partial models from the sequence of corners (sequence of lines $\{L1 - L4\}$):

$$SCM_1 = \{PM_{01}, PM_{02}, PM_{03}\}$$

$$\begin{aligned} PM_{01} &= \{M_1, M_2, M_3, M_4\} \\ PM_{02} &= \{M_5, M_6, M_7, M_8\} \\ PM_{03} &= \{M_9, M_{10}, M_{11}, M_{12}\} \end{aligned}$$

2. Candidate partial models from the parallel lines (parallel lines $\{L1, L4\}$):

$$SCM_2 = \{PM_{11}, PM_{12}, PM_{13}\}$$

$$\begin{aligned} PM_{11} &= \{M_1, M_4\} \\ PM_{12} &= \{M_5, M_6\} \\ PM_{13} &= \{M_9, M_{11}\} \end{aligned}$$

3. Candidate partial models after checking the both conditions

$$SCM_3 = \{PM_{01}\}$$

Figure 2.16: Candidate models after checking the both conditions

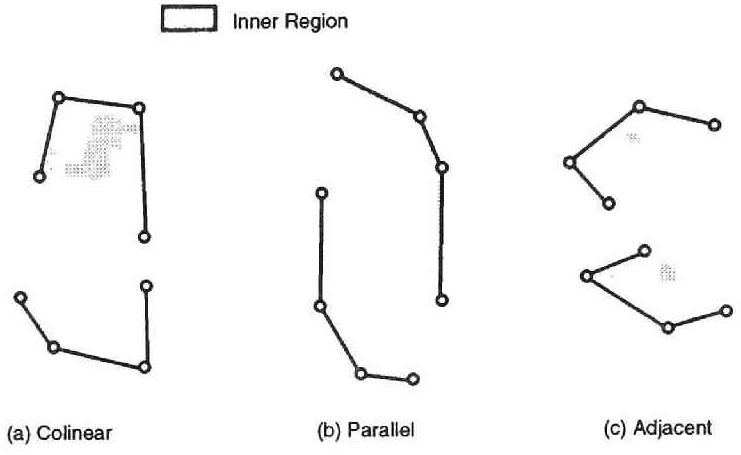


Figure 2.17: Combination of Partial Structures

combination are identical, only the check of candidate models is performed as described before in Fig.17.

2.4.5 Operation on Remaining Partial Structures

After all combinations are tested, there still may remain partial structures to which plural candidate models correspond. In such a case, detailed matching is performed for each candidate model, and the one with the highest score is regarded as the detailed matching result for this partial structure. If detailed matching was performed for partial structures which contain only a few lines but have many candidates, more objects could be found. The cost and efficiency of recognition, however, must be traded off Therefore detailed matching is performed for those partial structures which have a large [number-of-lines / number-of-candidate-models] ratio.

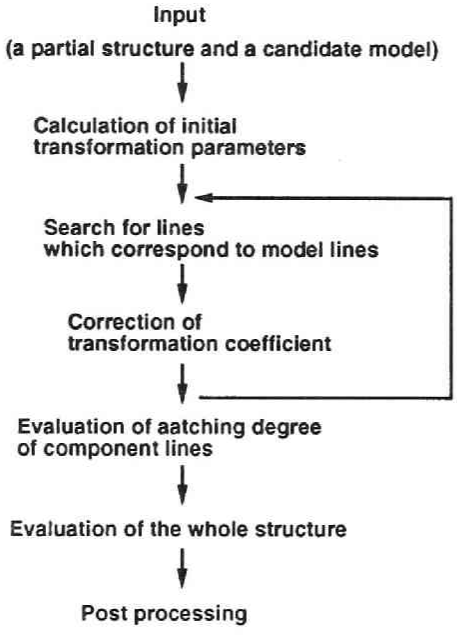


Figure 2.18: Flow of Matching Process

2.5 Detailed Matching

Since partial structures obtained from the model inference process contain only partial information about the model, matching must be performed globally to ascertain that the candidate model is the correct one. If plural candidates exist, a degree of matching (goodness of fit) must be evaluated for finding the best match. The matching process consists of two parts: (a) Searching for correspondences of lines and calculation of the translation parameters, and (b) evaluation of the matching degree.

2.5.1 Translation Parameters

As we assume that objects are 2-D shapes, the transformation of coordinates from model to image is linear (rotation, shift, scale change). The initial translation parameters are calculated from the correspondences of lines in a partial structure. Though translation parameters can be obtained from a corresponding pair of lines, i.e. two corresponding pairs of points, only end points of lines at which lines connect to other lines are used because of the possibility of occlusion. If plural corresponding pairs of points are acquired, the translation parameters are determined by computing the weighted mean value of the parameters for each individual line as follows:

$$P = \sum_i L_i \cdot P_i \,/\, \sum_i L_i \tag{2.1}$$

where L_i is the length of the i-th line (or distance between the terminal two points), P_i represents the value of parameters calculated for each line, and P is the resulting initial translation parameter.

The reason for computing the average is that longer lines are likely to be less affected by noise and line extraction errors. Though this method is less accurate than the minimization of the distance between lines by applying the least squares method as performed in [Low87], it is less expense and has proved to be sufficient in our experiments.

2.5.2 Searching for Lines

The following search methods are used:

(a) Search by relation : A line in the model, which does not yet correspond to a line in an image and which has at least one relation with lines already corresponded to image lines, is selected. Then the corresponding line in the image, which satisfies this relation, is searched for. Assume that *image-line-1* corresponds to *model-line-1* and that *model-line-1* and *model-line-2* satisfy *relation-1*. Then in order to find *image-line-2*

corresponding to *model-line-2*, image-lines which satisfy *relation-1* with *image-line-1* are searched for. This method is less affected by translation parameter errors, and the computation is much simpler, because the relations have already been extracted during the model inference step.

(b) Search by translation : The expected positions of lines in the image, which correspond to lines in models,are calculated from the translation parameters, and lines close to these positions are searched for. By this method, those lines which have no relations to other lines can be found. This method is very much influenced by translation parameter errors and the cost of searching is much higher than that of method (a).

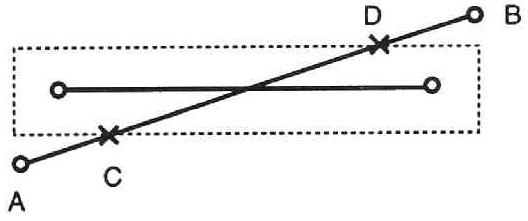
In our method, (a) precedes (b). When no lines can be found by (a), (b) is applied. If a corresponding line is found, the translation parameters are modified by this corresponding pair as follows.

2.5.3 Verification of Correspondence

The candidate lines found by searching are tested and the translation parameters are modified by equation 2.1. The individual translation parameters are calculated for the corresponding pairs, and the global translation parameters are modified by these values, if the new values exceed the old values by a certain amount.

2.5.4 Evaluation of Matching

Through the searching described above, the possible correspondences between image lines and model lines are obtained. Translation parameters are determined at the same time. In this step, these correspondences are evaluated based on the translation parameters, in the process of which the estimation values for the degree of global matching is calculated. It is determined by this estimation whether the correspondence between an object and a model



$$\begin{aligned} M &= L_{in}/L_{out} \\ &= CD/AB \end{aligned} \tag{2.3}$$

Figure 2.19: Allowance region for a line matching

is considered valid. In other words, the validity of the hypothesis of a partial structure is judged by this estimation.

The proximity of two lines is used as the measure for matching evaluation, because it can be estimated even in case where lines are partially occluded or when multiple correspondences exist for curves. For this purpose, we assume a window positioned on one of the lines as the region where matching is permitted. We define the matching degree to be the ratio of the length of the candidate line which reaches into the region to the length of the target line as shown in Fig. 2.19.

$$M = L_i^{in}/L_i \tag{2.2}$$

where M represents the degree of matching, L^{in} is the length of the candidate line's part included in the window, and L is the length of the target line.

The width and length of the window are determined to be large enough so that the errors of the line extraction process and the presence of noise will have no significant influence. In our experiments,this size is fixed to 10–15 pixels. this size is sufficient for lines that are longer than 30 pixels (50–80 pixels on the average), and the width is narrower than 5 pixels.

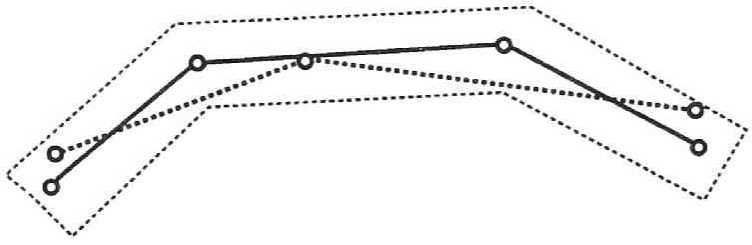


Figure 2.20: Allowance region for multiple approximated lines

To calculate the value of equation(2.2), lines in the image are transformed into model coordinates by using the translation parameters obtained before. Parameters for the *image-line to model-line* (M^{image}) and *model-line to image-line* (M^{model}) transformation are calculated. If lines have multiple correspondences, the value is calculated for each window of corresponding lines as shown in Fig. 2.20 and the values are averaged as follows:

$$M = \sum_i L_i^{in} / L_i \tag{2.4}$$

The estimation value for the whole structure is obtained by averaging the matching degrees for each line as follows:

$$M^{image} = \sum_i M_i^{image} \cdot L_i \cdot W_i \ / \ \sum_i M_i^{image} \cdot L_i \tag{2.5}$$

(M^{model} is also calculated by above equation)

$$M_{all} = B_1 \cdot M^{image} + B_2 \cdot M^{model} \tag{2.6}$$

where M_{all} represents the value for the overall matching degree, M_i is the degree of matching for each line, L_i as the length of the line, and W_i represents a weight assigned the line.

The weight W_i depends on whether a line is included in partial structures or not. It takes large values for lines that are included, because they are the

core of this matching for which candidate models are limited to a single one or a few. The weights for lines which are not included in any partial structures assume a small value.

Based on the value M_{all}, the match is judged to either have succeeded, or to have failed, or to be undetermined.

2.5.5 Operations after Matching

Image lines and partial structures are modified in compliance with the result of matching. This modification helps the recognition of other objects and can take the form deleting incorrect partial structures, modification of other partial structures, or the restoration of deleted partial structures. The operations are as follows.

(a) *succeeded* : Matched image lines are deleted. Partial structures which contain these lines are deleted. The number of partial structures decreases by this operation.

(b) *undetermined* : No operations are performed. The matching for the hypothesis inconsistent with this partial structure can be performed and the best scored one becomes the result.

(c) *failed*: This partial structure is deleted. Restoration of partial structures, which are deleted, is performed (They had been deleted because their lines are subset of the lines of the partial structure). No other operations are performed.

2.6 Automatic Construction of Models

Models used for recognition are built by off-line processing. The flow of this model construction process is shown in Fig. 2.21. For the most part of this step, processing is automatically performed similarly to the recognition process. New

models can be easily added to the recognition system by this model building program.

(1) *Extraction of lines* : Taking a picture of an object with a TV-camera. Preprocessing. Line extraction by the *Slit Method* [Nag84, Nag86, NN86]. These steps are described in detail in Section 2.3.

(2) *Manual modification* : Extracted lines are modified manually, since it is difficult to extract lines reliably at a single trial because of shadows, reflection and other kinds of noises. Lines are displayed on a terminal screen and the user can modify them interactively.

(3) *Extraction of features* : Extraction of features from these lines. Extracted features are stored in an individual model description.

(4) *Registration of features in the feature-to-model-table* : After all models have been built, all features are gathered, sorted and then registered in the feature-to-model-table.

By using the same analysis process as is used in the recognition step, it becomes easier to achieve a coincidence of an image feature and a model feature. Because if the user gives ideal features as models to the system, it is quite difficult to extract their components from the image.

2.7 Experimental Result

Recognition Objects

We applied our system to two sets of objects, which were (1) ten arbitrary 2-D shapes composed of straight lines, and (2) real tools such as pliers, screwdrivers, and hexagon-wrenches (10 models).

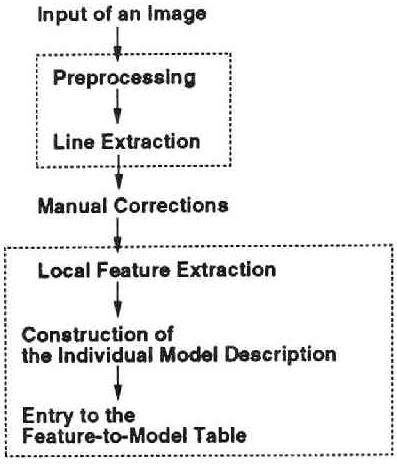


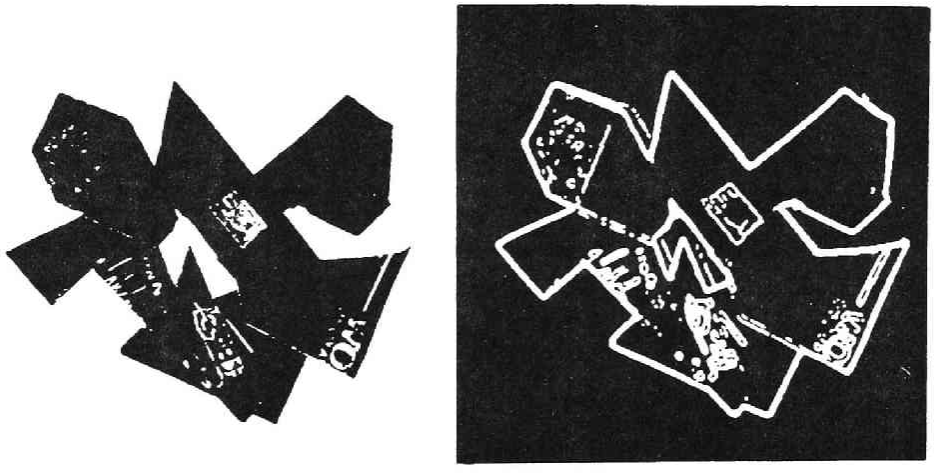
Figure 2.21: Model Construction Process

Result

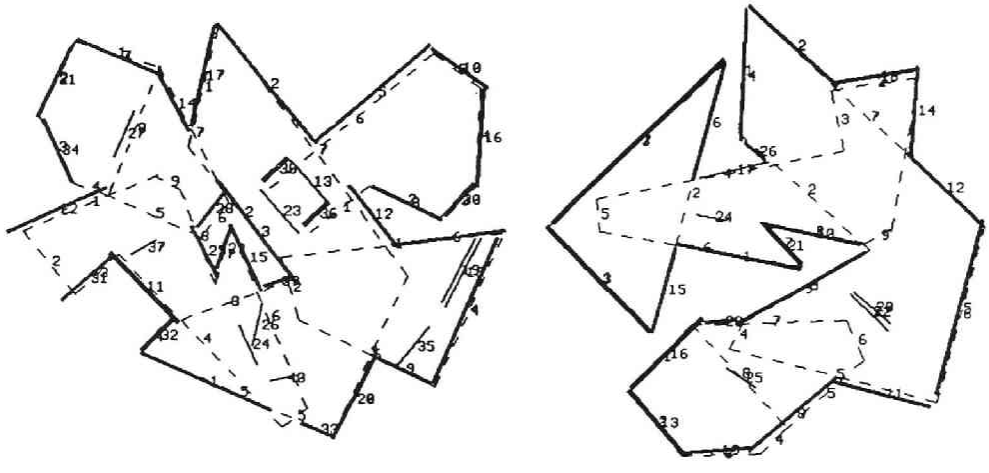
The recognition process applied to Fig. 2.6 resulted in Fig. 2.7, Fig. 2.9, through Fig. 2.22(a) to (f). The final result is shown in Fig. 2.22(f). In this recognition example, matching is performed 12 times and 4 objects are recognized correctly. The result applied to Fig. 2.23(a) is shown in Fig. 2.23(c) which also shows the complete success.

Efficiency

This method is especially efficient for model sets in which the objects are composed of straight lines. Many objects can be recognized regardless of the partial lack of objects' boundaries. As for the real tools, though the recognition sometimes suffers from the presence of curvature, shadows and lines in the inner region, it works well for most cases.



(a) Original Image (b) Differentiated Image



(c) Recognition Result (d) Another Experiment
Dotted lines show the positions of recognized shapes.

Figure 2.22: Recognition Result 1

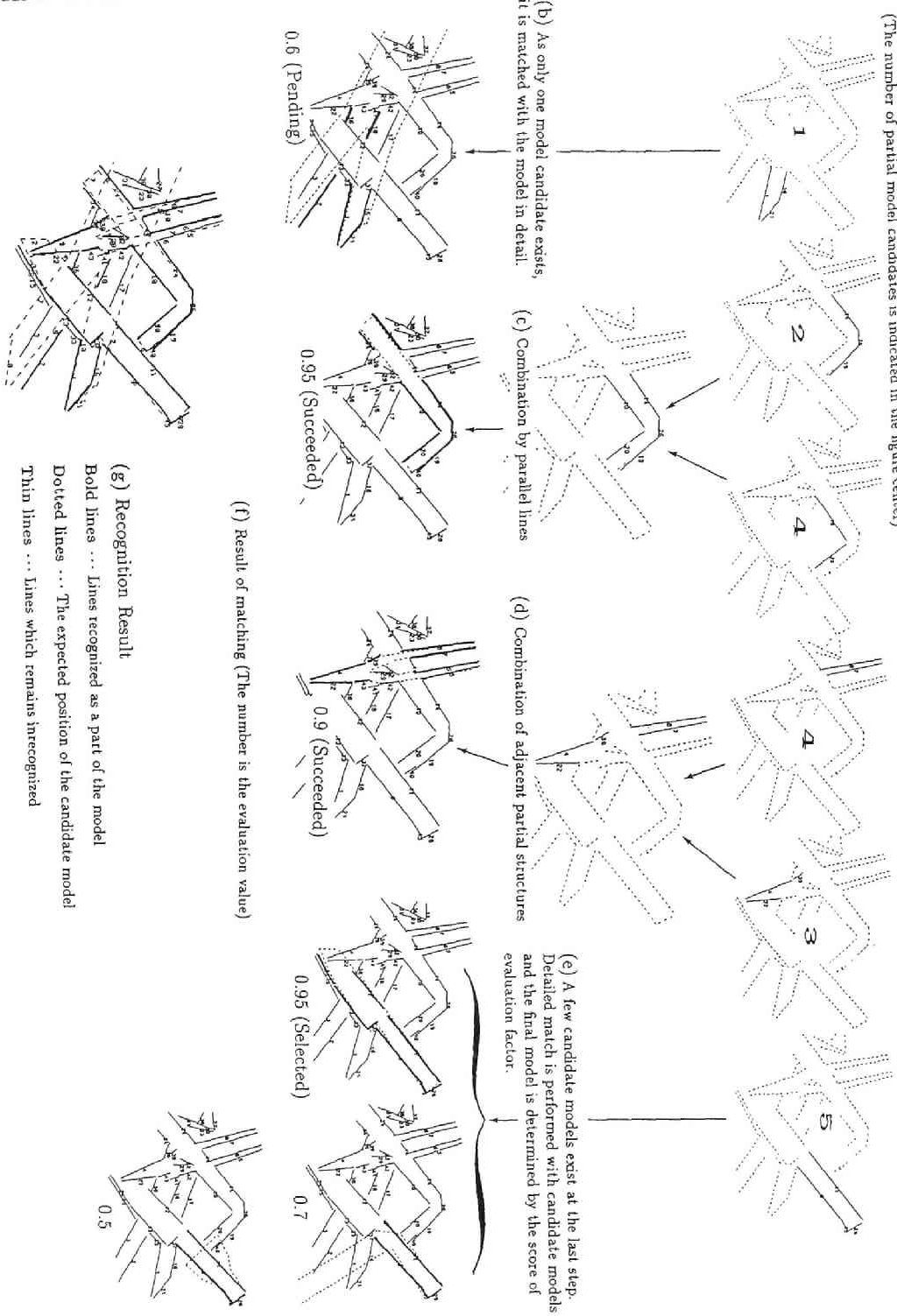


Figure 2.23: Recognition Result 2

The Number of Times of Matching

As shown in the result of Fig. 2.22, the matching was not performed as often as in other researches [Hae82, Tro81, AF86]. In the experiment for model set (2), the number of matching operations is generally 2 – 3 times of the number of the objects present in an image.

Efficiency of the Construction

Many objects are recognized during the step of matching corner patterns. It sometimes occurred for model set (1) that all objects in an image were recognized in this step. Although features of parallel lines appeared less often than corners and the number of candidate models suggested by their presence is sometimes large, the combinations formed from them are efficient and are generally correct for objects like (2) in which models have many parallel lines. The same can be said for the combination of aligned lines. By contrast, the combinations of adjacent lines generated many incorrect combinations. However, not a few objects were recognized through these combinations.

2.8 Conclusion

We have proposed a method for the recognition of overlapping 2-D objects, and discussed following points in this chapter:

- The selection of local features and relations which have proved to be sufficient for the recognition of overlapping objects.
- A method of inferring candidate models from local features.
- A method for reducing the number of candidate models by checking combinations of extracted features.

- A method for checking the consistency between lines in an image and lines in the candidate models which is applied to only very few such pairs.

The efficiency of our method was verified in experiments with a scene in which many pieces of papers of arbitrary shapes were stacked in such a way that occlusion of boundaries occurred, and in experiments with a scene in which real tools are scattered on a planar surface. Our system showed quite good performance both for inferring models and for matching shapes.

The following points are left for future work.

- The handling of various kinds of feature that are sufficient for the recognition of other kinds of objects. For example, region features based on texture are expected to be sufficient for objects whose shapes are not useful keys for recognition.

- The relaxation of camera position constraint. For example, allowing the camera to be tilted or slanted.

Chapter 3

Feature Definition in PAFE

In this chapter and the following chapters, a system with the name PAFE (PArallel Feature Extraction system) is presented. This chapter describes the framework for feature definition in PAFE. First, an overview of the system PAFE is given and then, the definition mechanism for features, the definition mechanism for feature extraction, and some experiments which demonstrate these definition in real recognition tasks are shown. The next chapter focuses on the configuration of this system. The organization of agents as basic mechanisms of feature extractions is presented. The control strategy is and various kinds of extraction operators realized with several kinds of basic operations are discussed in Chapter 5.

3.1 Introduction

Many methods proposed for image processing, recognition and understanding. Some of them work fairy well for a specific domain. In those systems, however, knowledge about recognition objects (features, models, etc.), methods for feature extraction and the control strategy for recognition process are mixed in complicated ways. Few of those systems have a mechanism for defining features (and models), which range from the low-level features to the high-level (com-

plex or compound) features within the same framework, and independently of the control strategy.

For this purpose, we introduce a system based on the multi agent model which provides a new framework for feature definition and feature extraction suitable for parallel processing. In this system, all features can be defined in a declarative way independent of extraction process, which can also be easily defined. The pieces of knowledge about features and their extraction processes are defined in the form of a network, in which they correspond to nodes and arcs respectively. The system performs flexible extraction of features by traversing the network.

We have constructed a system in which features can be defined hierarchically, and relations between features and their extraction process can be defined easily, especially for compound features.

3.1.1 Overview of PAFE

The PAFE system is composed of two groups of modules. One is needed for the definition of features and their extraction. This part is composed of the feature definition module and the module in which image processing and feature extraction routines are gathered. The other part is devoted to the execution, i.e. image analysis. Although the configuration of this part is similar to the blackboard model, its control mechanism differs from that of the blackboard model as was mentioned above.

The configuration of PAFE is as follows (illustrated in Fig. 3.1):

- 'Feature Definition Module' for feature (and relation) definition
- A Module in which primitive routines for image processing and feature extraction are gathered
- 'Feature Management Agents' for storing image data and features (MA)
- 'Feature Extraction Agents' for feature extraction (EA)

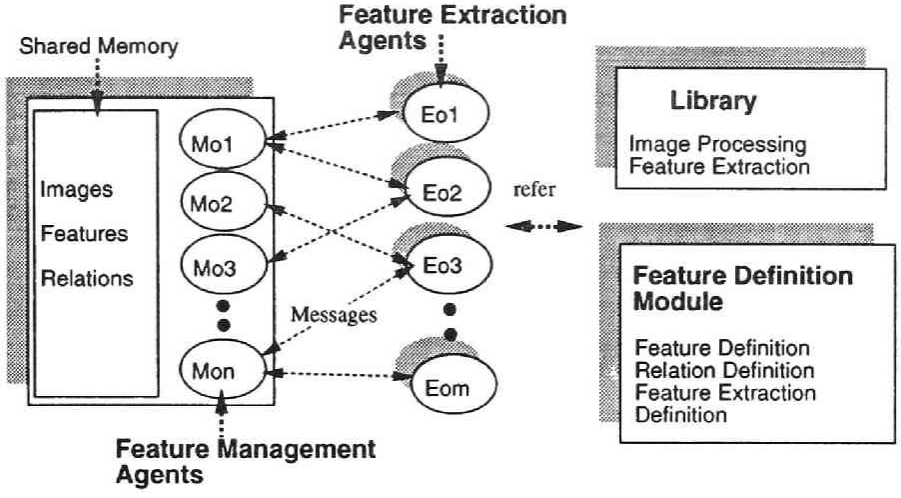


Figure 3.1: System Configuration

The feature definition module provides the framework for feature definition described in this chapter. Agents for feature extraction and agents for feature management are generated during execution according to feature definition. Details of the agents' organization and execution mechanism are described in the following chapters.

3.2 Feature and Relation Definitions

Features in images, which are used for image recognition and image understanding, take various forms. The methodology for their description and definition is one of the most important topics in image analysis. For instance, the comparisons between the procedural description and the declarative description are often discussed as well as the advantages and disadvantages of structural methods. A general framework into which these parts can be embedded, however, seems to have been too remote.

One objective of the PAFE system is to provide a framework for feature definition in which many kinds of features can be defined within the same system. At the same time, the mechanism for generating the feature extraction strategy from a feature definitions directly is also an objective of this system.

3.2.1 Several Kinds of Feature Definitions

First let us consider several classes of features: 1. images, 2. curves and regions, 3. corners, parallel lines and triangles, 4. complex features, objects and scenes. Note that this classification is not an absolute one, to which there would not be any alternatives.

An overview of the features is shown in Fig. 3.2. Class 1 is composed of features which have no components, and which have no part-of relations to other features, but they may have attribute values or feature values such as the range of gray levels of gray scale images. Class 2 consists of the features which have no structure or components, but they can have relations to other features; for example, adjacency to other features would be such a relation. The features in Class 3 have both components and relations. Their structures are so simple that alternative descriptions would not be found easily. The features classified as class 4 are the complex features, which have components, relations and various kinds of alternative forms of description.

"Attributes" is essentially required for describing features. The "Components" and "relations" to other features (relations among components at the same time) are also required except for the features of class 1. In class 4, an important aspect for feature definitions becomes clear. The uniqueness of feature description is extremely difficult to achieve in class 4. For instance, even a rectangle (which is a relatively simple structure) can be defined in more than three ways, where one is by the four lines, another one is by the four corners, etc. Therefore "multiple definitions of a feature", i.e. "redundant description of a feature", are required for an effective definition. In this way, a structural feature may be defined as a feature which has no components. A rectangle

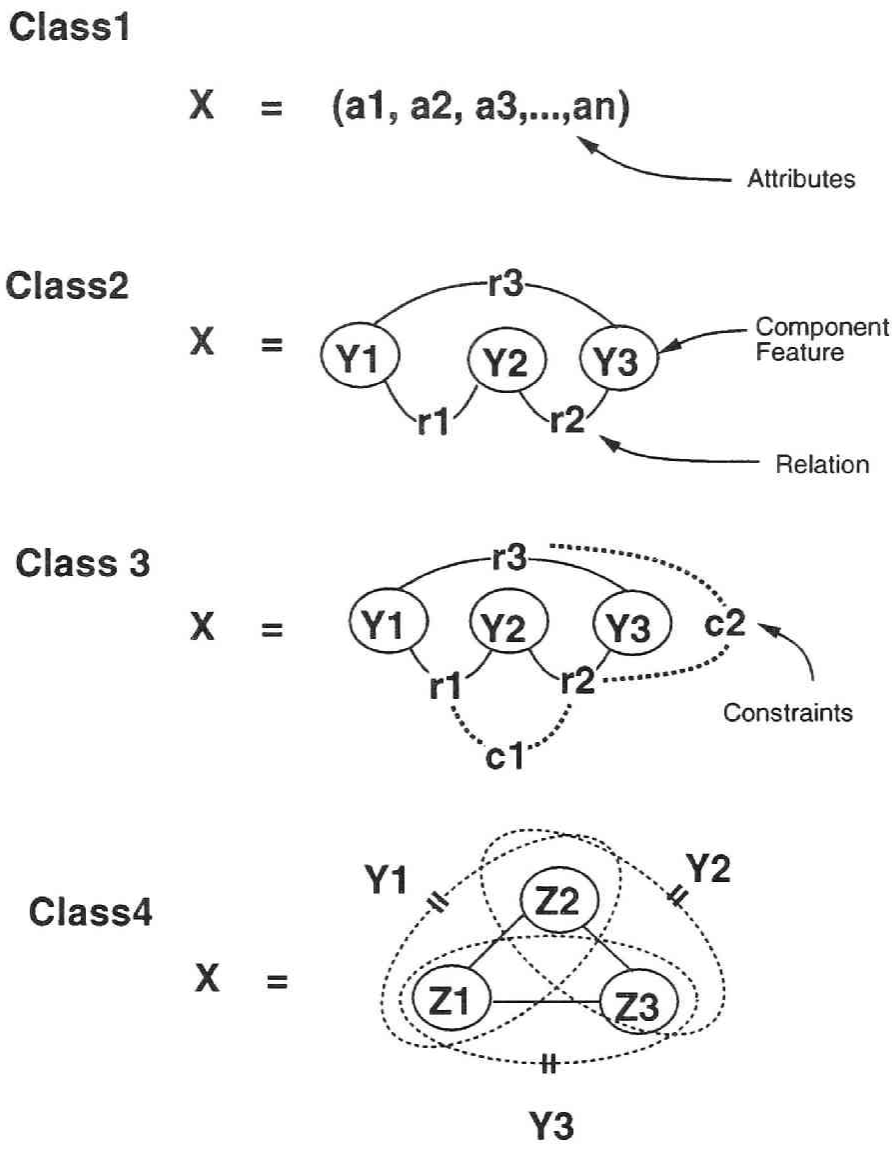


Figure 3.2: Several Types of Features

can be defined as a region segment which has rectangular shape. Allowing "multiple definitions of a feature", the definitions supplementary cover a wider variety of situations than the single definition does. If the recognition system has the ability of choosing the best description in a given context, the recognition task can become easier. Nevertheless, certain mechanisms for avoiding contradictory situations that multiple definitions may cause must be provided.

In this research, features are defined structurally and allow "redundant description of a feature". In the following sections, the definitions of features and relations are discussed.

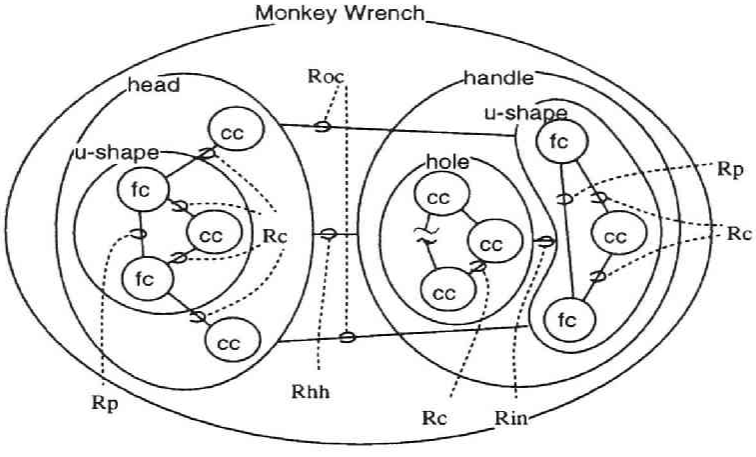
3.2.2 Feature Definition

In the PAFE system, features are defined in a hierarchical way. A structural feature is defined as a combination of component features related by defined relations. From another point of view, a feature is defined as a network in which the "nodes" are component features and the "arcs" are relations between component features, and both nodes and arcs can have attributes. Fig. 3.3 shows an example of a network defined for a monkey wrench used in the experiments (Section. 6.1).

By definition, a feature includes the following terms.

- Attributes and their calculation methods
- Component features
  - primitives (smallest components), $\{f_i\}$
  - compound features (which have f_i's as components), $\{fm_i\}$
- Labeling variations

  For identifying variations for component labeling– e.g. the two labelings L_1, L_2, L_3, L_4 and L_3, L_4, L_1, L_2 for the sides of a square are identical.



	mark	meaning
feature	cc	a curve which with high curvature
	fc	a curve which with slight curvature
relation	Rc	connect
	Rp	parallel
	Roc	overlap or connect
	Rin	include
	Rhh	both of end features have Roc

Figure 3.3: Feature Network for Monkey Wrench

- Relations among component features

The definition of features includes attributes and their calculation methods, component features, definitions for identifying variations of component labeling and relations between component features. An example is shown in Table. 3.1. In this way, both structural features and non-structural features can be easily defined. Redundant descriptions are allowed such as definitions using different sets of component features representing the same feature. For example, a rectangle can be simultaneously defined as a set of four straight lines and also as a set of four corners. This makes it possible to extract a rectangle from both straight lines and corners.

With this format, all feature types discussed in the previous section can be defined. For instance, an image can be defined as a feature, which has no components and relations to other features (or images). A complex object can be defined as combinations of many components which can share primitive features.

Inheritance is another important aspect of feature definition in our system. The features defined as subclasses of a parent feature can inherit attributes, component features, relations etc . from their parent features. 1D features such as 'sequence' and 'loop', for example, are defined as superclasses of other 1D features. Other 1D features can inherit attributes such as 'end-point' and 'length', and relations such as 'end-points-adjacent' and so on, if 'sequence' is specified as a superclass in their definition.

3.2.3 Feature Definition with Variable Components

Many types of features have components whose numbers or types can vary. For instance, a group of elements which have features in common is such a feature, since the number of the elements may vary from situation to situation. Additional mechanisms are provided for defining this kind of feature. We prepared two types of these 'variable-length' features, 'sequence' and 'group' as follows.

Table 3.1: Example of Feature Definition (Rectangle)

```
(deffeature rectangle
    (:attributes
       (center-point (:eval (calculate-center-point))
        area (:eval (calculate-area))
        ............))
    ............
    (:component-features
       (:primitive
          (line1 (:type straight-line :attributes (length %len1))
           ........................
           line4 (:type straight-line :attributes (length %len4))))
       (:compound
          (pl1 (:type parallel-lines :components (line1 line3))
           pl2 (:type parallel-lines :components (line2 line4))
           c1 (:type corner :components (line1 line2)
               :attribute (inner-angle (nearly-equal 90)))
           ................
           c4 (:type corner :components (line4 line1).......)))
       (:labeling-variations
           (:circular (line1 line2 line3 line4))))
    (:component-relations
       (parallel1 (:features (line1 line3) :type line-parallel))
        connect1 (:features (line1 line2) :type line-connect
                  :attribute (connect-type %ctype1
                              inner-angle (nearly-equal 90)))
        ..................
        ca1 (:features (corner1 corner2) :type corner-aligned)
        ..................)
    (:constraints
       (con1 (:parameters (%ctype1 %ctype2)
              :type :member (((1 3)(3 4))((2 4)(1 2))))
        ..................))))
```

Variable-length sequence: A complete order can be defined for a set of features.

Variable-length group: Defined as an unordered set.

A sequence of connected lines, for example, is defined as a *variable-length sequence*, whereas a group of parallel lines is defined as a *variable-length group*. The extraction methods for these variable-length features are different from features with fixed number of components, (they are described in section 3.3.5).

3.2.4 Several Kinds of Relations

As stated in Section. 3.2.1, relations between features play an important role in feature and object description. Before introducing our definitions of relations, let us consider the set of relations required for image recognition.

In image analysis, spatial relationships between two features (parallelism, for instance) are often used as relations. The relationships due to shared parts appears, if more than one structural features have the same feature as their component. Suppose that structural feature F1 has components (fa, fb, fc) and another structural feature F2 has (fc, fd, fe). Then F1 and F2 share fc as their component. This relationship is useful for defining connecting parts between structural features. A summary of the set of relations is shown in Fig. 3.4. All of these relationships can be classified as relationships between the attributes of two or more than two features. These attributes are the position, length, color or any other kind of feature values. Some relationships between structural features are considered as special cases where components are used instead of attributes.

Relationships often have their own attribute values such as inner angles for the connection of two line segments, or distance in the case of adjacency[1]. In

[1]If the relationships are classified and labelled precisely, these values sometimes become redundant. For instance, if the range of the distance is divided into precise classes (touch, close to, slightly apart from, apart from, etc.), the exact value of the distance is no longer required. However, this causes the required number of relationship types to increase drastically.

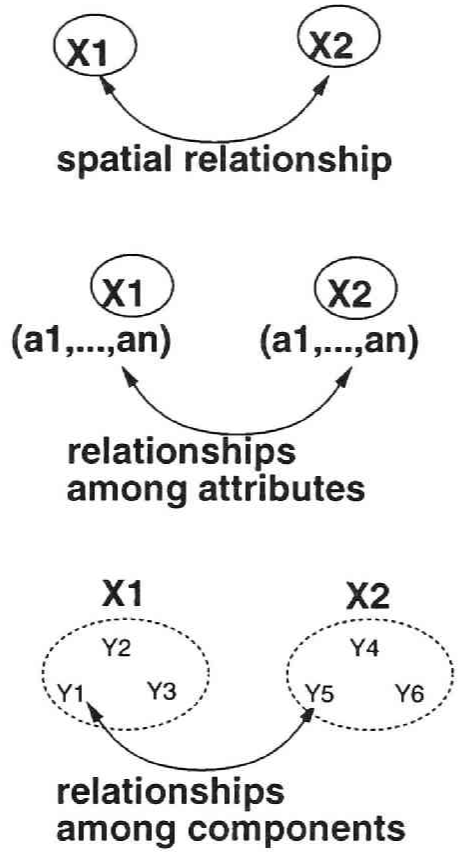


Figure 3.4: Relation types

this research, we designed relations to have attribute values in order to avoid a larger number of relation types. The formal definitions of relations are given in the following sections.

These relationships can be considered as a set of conditions that must be satisfied by the features. In other words, if a set of features satisfies the condition defined for a certain feature, the corresponding relation is treated as existent among the features. For instance, parallelism can be defined as the condition that the orientations of features are very close to each other. The conditions can be some procedural conditions, which can be checked only by calculation, a set of relations, or combinations of relations. The user can use relations as pre-conditions of a relation. Suppose that someone wants to define the "parallel-and-equal-length" relation. This relations can be defined using the already defined relations "parallel" and "equal-length". In this case, no additional conditions are needed.

3.2.5 Definitions of Relations

Based on the consideration in the preceding section, we developed the relation definition as follows. A relation is composed of attributes, component features, procedural conditions and/or other relations.

- Attributes and calculation method
- Component features (which have this relation)
- Condition
  - in relational form
  - in procedural form
- A scope for finding features

An example is shown in Table. 3.2. This format allows the definitions of relations as they were outlined in the previous section.

Table 3.2: Example of Relation Definition

```
(defrelation symmetric-parallel
  (:component-features
   (line1 (:type straight-line)
    line2 (:type straight-line)))
  (:condition
   (:relations
    (parallel (:features (line1 line2) :type straight-line-parallel))
   (:procedure
    (proc1 (:eval (straight-line-symmetric line1 line2 parameters))))
  (:search-area
   (:from line1 :to line2 :type :rectangle
    :parameter (:center (attribute center line1) :width .........))))
```

3.2.6 Relations between Structural Features

Relations between structural features are often necessary for recognition. The PAFE system provides a simple way for defining this kind of relation in that it allows its definition as a set of relations between component features. For example, the relation 'parallelism of two rectangles' can be defined as the condition that one side of one of the rectangles is parallel to one side of the other rectangle (Fig. 3.3 shows the definition). In addition, as there may exist various identical definitions, this system provides a simple way for including such definitions by using the 'labeling variations' defined Section 3.2.2. This system provides following the three types as variations.

:all ;every combination $({}^{\forall}f)$

:exists ;one combination $({}^{\exists}f)$

:sequence ;each pair along sequence $({}^{\forall}i f_i^1, f_j^2)$

If the above types are specified, the system tries every combination as defined above. In the case of parallel rectangles, the number of the combinations of sides, which may be parallel, is really 16, and these can be incorporated into one definition by specifying ':exists' as a variation.

Table 3.3: Example of Relation between Compound Features

```
(defrelation tow-rectangle-parallel
   ......................
 (:component-features
  (rectangle1 (:type rectangle)
   rectangle2 (:type rectangle)
   %line1-1 (:component-of rectangle1
             :name line1 :variation :exists)
   %line2-1 (:component-of rectangle2
             :name line1 :variation :exists))
   ......................))
```

3.3 Definition of Feature Extraction

In the previous section, the framework for feature definition and relation definition was introduced. The definitions of the feature extraction methods must be given in addition to that, because the feature and relation definitions by themselves do not provide any extraction methods. For this purpose, we provide a format for the definition of feature extraction methods. They are specified separately from feature definition, and are described in this section.

3.3.1 Several Types of Feature Extraction

In feature extraction, a number of situations arise. Some features are extracted by simply applying image processing, whereas others are extracted by utilizing combinations of features. First, types of feature extraction are discussed.

Several Types

In image analysis, feature extraction is always performed according to relationships. Let us consider semantic descriptions in Fig. 3.5. Note that these relationships are different from the 'relations' defined above, which are more abstract.

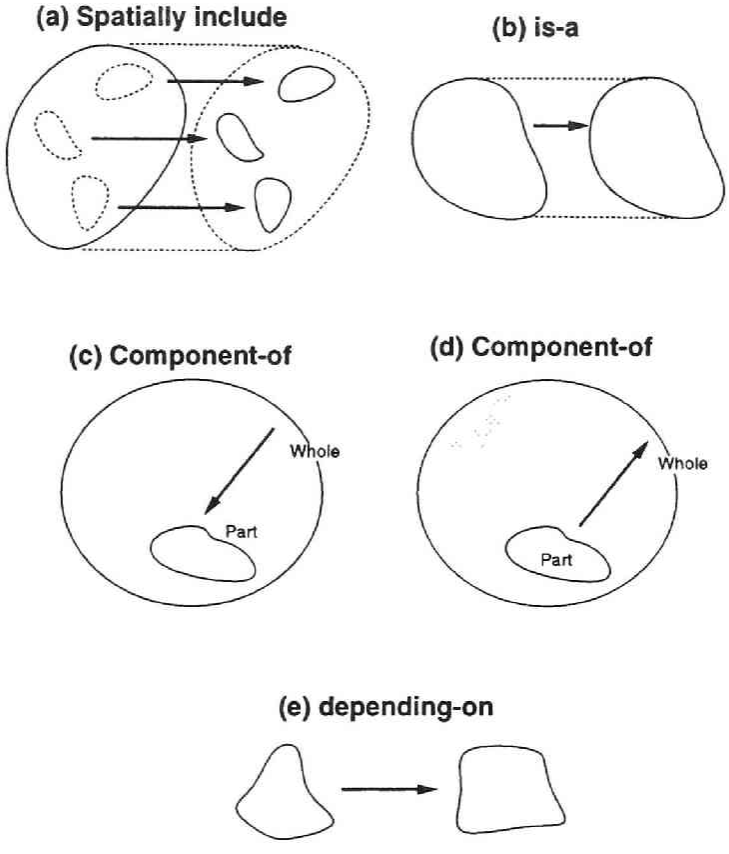


Figure 3.5: Feature Extraction at Semantic Level

spatial-inclusion: The output of feature extraction process (extracted features) have no structural relationships to the input of feature extraction process (input features), although extracted features are spatially included in the area of the input features.

component-of: The extracted features are the component features of the input features, or the input features are the components of the extracted features.

is-a: The extracted feature belongs to a subclass of an input feature.

depends-on: The existence or the characteristics of the extracted feature depend on the input features. To-occurrence is not always required. For instance, the input features can be considered as contexts (environment and so on).

According to these relationships, one can consider several types of feature extraction methods:

(a) Plural features are detected from a feature (or features) in which they are spatially included:

$$F_{output} \Leftarrow X(f^1_{input}, f^2_{input}, \ldots, f^m_{input})$$

(b) A feature is converted into another feature whose area of existence is spatially equivalent to or included in it:

$$f_{output} \Leftarrow X(f_{input})$$

(c) A feature is extracted as a combination of features according to the component-of relationships:

$$F_{whole} \Leftarrow F^1_{input} \otimes F^2_{input} \otimes \ldots \otimes F^n_{input}$$

(d) A feature is extracted as a component of a feature whose component is the extracted feature:

$$F_{component} \Leftarrow X(f_{input}, E(f_{whole}))$$

(e) A feature is extracted according to depends-on relationships.:

$$F_{depending} \Leftarrow X(f_{input}, E(f_{depended}))$$

where f is an individual input/output feature, F is a set of features, X is an extraction procedure, $\otimes$ expresses combination similar to direct-product in mathematics. and $E(x)$ can be any kind of useful information for the extraction process derived from x.

Several Types from Input and Output

According to the types of input and output features, four types of feature extraction methods are considered in this system: 'image processing', 'feature extraction from image', 'feature extraction from feature' and 'feature synthesis'.

Image processing: This operation applies ordinary image processing procedures to an input image for generating new images. The input to the operation is a fixed number of image, and the output is also a fixed number of image. This corresponds to (a) in the previous section.

Feature extraction from image: A defined operation is applied to the input images. The inputs to this extraction process are also a fixed number of images, but the number of the output features is unknown until the extraction process is completed. This corresponds to (b) and (d) in the previous section.

Feature extraction from features: A defined operation is applied to the input features. The number of input and output features is unknown. This corresponds to (d) and (e).

Feature synthesis: Feature synthesis is a syntactic method for extracting structural features. In this operation, a search is performed to gather the component features. This operation corresponds to (c).

These four types of feature extraction methods are utilized in the feature extraction definition in this system.

3.3.2 Definition of Feature Extraction Method

Considering the above types of feature extraction methods, definition of the feature extraction methods in this system is as follows.

- type of the extraction procedure
- procedures for the extraction
- input features (they can be component features for constructing structural features)
- relations among components
- calculation methods for extracting, finding the components (optional)

The extraction method may be one of the above four types 'image processing', 'feature extraction from images', 'feature extraction from features',or 'feature synthesis'. The components and input features can be any features, if they are required for extraction. In this context, it is required to specify the set of component features used for extraction, because redundant information is contained in this definition, such as the fm_i's mentioned in the example in Section 3.2.2. For example, the definition of syntactic type extraction requires (1) component features and (2) relations.

The calculation methods needed for extracting component features can be prepared, as in advance for later use. Assume that a parallel-line feature is

extracted directly from the image. If the routine for detecting the two component lines is necessary for further processing, then the routine is added to the definition of the parallel lines' extraction method.

The above definitions realize all the types of extraction methods discussed in Section 3.3.1, and allow to handle any type of input/output and any type of semantics.

With these feature definitions and feature extraction definitions, features are assumed to be organized as a network which shows how features are extracted from other features. Fig. 3.6 shows an example of such a network in which each extraction method is classified. The basic idea of this framework is similar to LLVE [Mat88]; however, structural features can not be defined in a syntactic way in LLVE.

As was mentioned above, more than one extraction method can be defined for a feature. For example, edges can be extracted both from a segmented image (label image) and from a differentiated image. This is another aspect of defining features in multiple ways as well as in the definition of features themselves. The redundant description of features is the key concept of this system as mentioned before.

3.3.3 Definition of the Synthesis of Features

As features can be defined in multiple ways, redundant entries appear in feature definitions. It is expensive and ineffective both in cost and robustness to use all of them for extraction at one time. In this system, the extraction method definitions of structural features are defined as set of component features and component relations to be utilized for extraction. In other words, only features and relations specified in this extraction definition are used for extraction.

The definition of synthesis type feature extraction is as follows:

- Features being used: $\{f_i, f_j, \ldots, fm_k, \ldots\}$
- Relations being used: $\{r_i, r_j, \ldots\}$

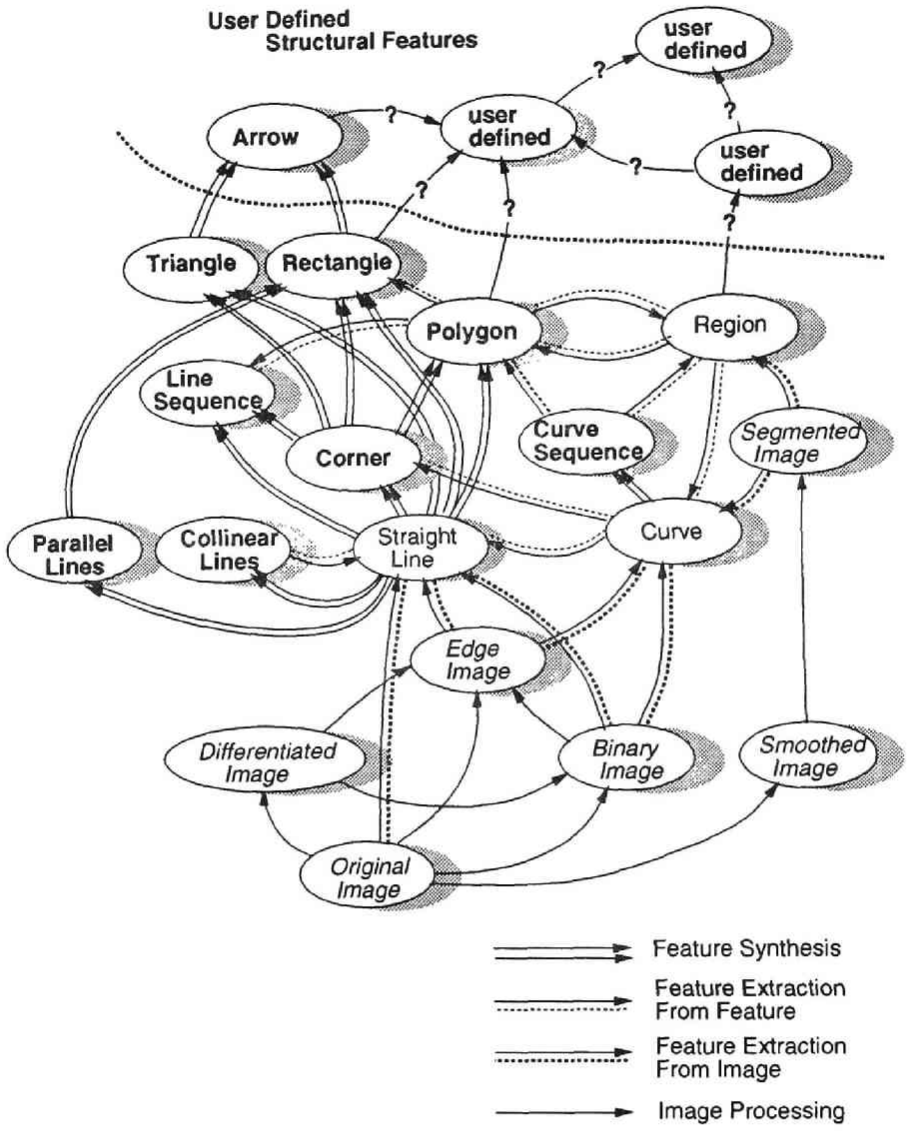


Figure 3.6: Network of Features and Their Extraction Methods
(Bold type face is used for structural features)

```
(deffeature-generation (rectangle-from-four-lines rectangle)
   (:type :syntactic :simple)
   (:component-features
      (line1 line2 line3 line4))
   (:component-relations
      (connect1 connect2 connect3 connect4)))
```

;; line1–line4, connect1–connect4 are defined in the feature definition (Fig. 3.1).

Figure 3.7: Definition of Feature Extraction

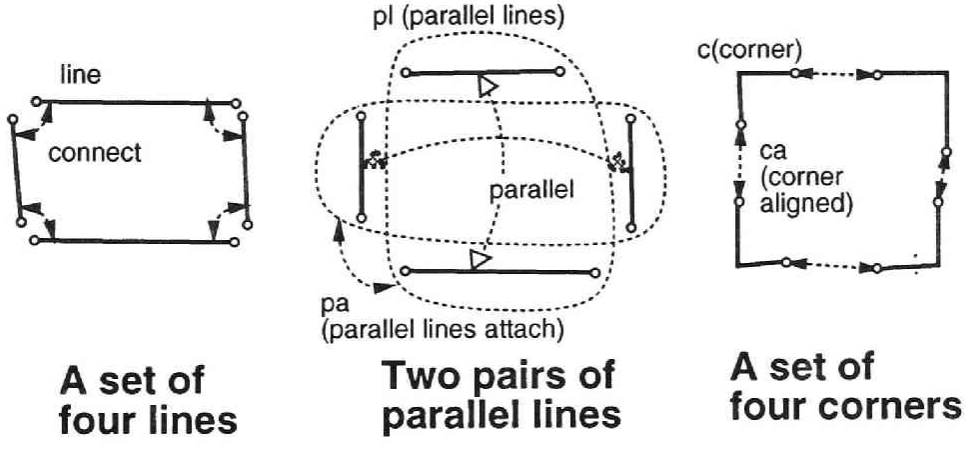


Figure 3.8: Definition for a Rectangle (three types)

Fig. 3.8 shows three examples of definitions for extracting rectangles. They use (1) $\{line1, \ldots, line4\}$ and $\{connect1, \ldots, connect4\}$, (2) $\{pl1, pl2\}$ and $\{pa1\}$, (3) $\{c1, \ldots, c4\}$ *and* $\{ca1, \ldots, ca4\}$, respectively (See Fig. 3.7).

3.3.4 Extraction Execution

For the extraction of non-structural features, the extraction operation is simply the application of defined operations to input images/features. On the other hand, extraction of structural features is complicated. It is performed by search; all component features that satisfy the defined conditions are searched. The algorithm is basically a *depth-first search*, in which the next feature defined in the feature definition is searched sequentially. In this search, however, it is not necessary that all component features are extracted before the search begins, because a hypothesized feature is created and assigned to a component feature that has not been extracted yet. This enables the asynchronous extraction of features as described in the following sections.

Let f_i^h be a hypothesized feature for a component feature, f_i^r be a candidate for f_i^h, $r^h(f_i^h, f_j^h)$ and $r^r(f_i^r, f_j^r)$ be the relation between the features, and let n be the number of component features. Then, the algorithm is as follows:

1. Sort features and relations
   $(f_1^h, \{\})$,
   $(f_2^h, \{r_{11}^h(f_1^h, f_2^h)\})$,
   $(f_3^h, \{r_{21}^h(f_1^h, f_3^h), r_{22}^h(f_2^h, f_3^h)\})$,
   .....

2. Find the first feature f_1^r in the area in which the area of this feature is expected to be include. Let $i = 2$.

3. Choose one feature f_i^r which has relation $r_{i1}^r(f_i^r, f_{i-1}^r)$
   (f_i^r is a candidate for f_i^h).
   If no f_i^r remains, then go to step 4. (if $i = 1$ terminate this search).

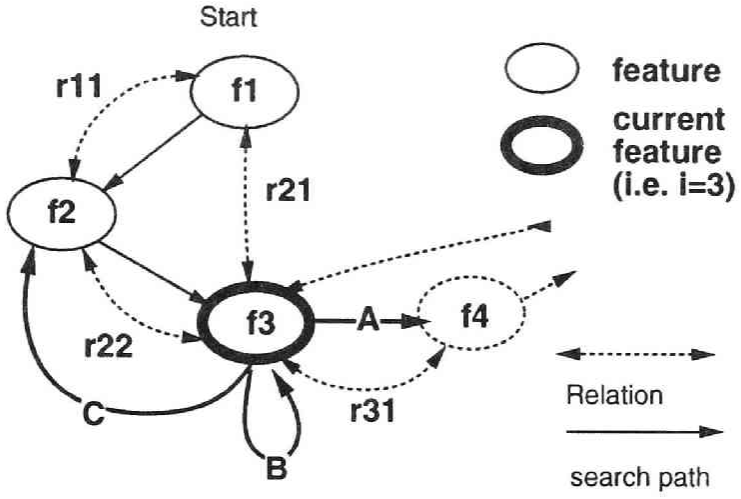


Figure 3.9: Operation of Search

Otherwise confirm that f_i^r satisfies the other relations (r_{ij}^r) and constraints specified in the definitions.
If confirmed, go to step 5 (This is shown as an arrow labeled with "A" in Fig. 3.9). Else try another f_i^r (as shown by the arrow labeled with "B")

4. If hypothesized feature $f_i^{h'}$ for f_i^r is not created yet, create a hypothesized feature $f_i^{h'}$ and let $i = i + 1$. Otherwise let $i = i - 1$.
   Go to step 3 (as shown by the arrow labeled with "C")

5. If $i = n$, then generate a new feature instance, else let $i = i + 1$. Go to step 3.

In case of the extraction of variable-length features, of component features (ex. f_i) of the same kind are searched until no more candidates are found. This operation is described in the following section.

```
(deffeature-generation (line-sequence-generation line-sequence)
  (:type :syntactic :variable-length)
  (:component-features
   (line1 (:type straight-line)
    $line2 (:type straight-line)) ;$ means recursive
  (:component-relations
   (connect1 (:type connect :features (line1 $line2)
              :attribute ((type %type1))
    connect2 (:type connect :features ($line2 $line2)
              :attribute ((type %type2)))
  (:constraints
    ..............................))
```

;;line1, $line2 must be defined precisely because they are not mentioned in the Feature Definition.

Figure 3.10: Definition of extraction method for a feature which has a variable number of components

3.3.5 Extraction of Variable-length Features

The extraction of variable-length features needs a special mechanism for synthesizing features recursively. For this purpose, a special component can be assigned for variable-length features, for which the system tries to extract features repeatedly until no more features are found.

A few options for this extraction are provided in this system.

Detect all or longest: Specify whether to detect all features or only the longest feature. Often, the longest feature is required. If 'longest' is specified, only the longest sequences (groups) are kept and other sequences (subsets of them) are abandoned.

Termination condition: If the termination condition is satisfied, the search stops. The features gathered up to the point where the search has stopped are used. This applies to cases where a set of feature, which satisfy certain conditions is required.

Loop condition: Specify the operation to be performed when a loop is detected (the sequence of features might become circular) while searching for variable-length sequences. (In some cases, it is effective to accept the loop as a new feature of a different type).

3.4 Feature Instance

In the preceding sections, the formalism for feature definition and feature extraction definition was discussed. The format for the data extracted from an image and the data which specify extraction request is also quite important for the image recognition system.

In this section, two types of data (instances) are presented: one is to express the feature extraction request and the hypothesis of existence of features (called 'hypothesized feature instance'), and the other is one to express the data extracted from an image by feature extraction methods (called 'real feature instance'). Each instance of these two types is generated as a structure[2].

3.4.1 Structures for Feature Instance

As was discussed in Section 3.2.2, features include several items of data. Considering this, the structure of an instance must be defined as shown in Table 3.4. The structure for relations is shown in Table 3.5. Both structures have slots for an identifier, type-of, attribute and so on. The slots for the component features and relations between components are defined only for the feature instance.

Real Feature Instances

A real feature instance is a structure for a feature which was detected during execution. An instance of a real feature (say f^r) is created, when a new feature is extracted by the feature extraction operation. Attributes for it are calculated and given to this instance according to the feature definition. In addition to

[2]In reality, this structure is an object of the object orient language FLAVOR.

Table 3.4: Structure for a feature instance

identifier	:	unique id is assigned to each feature instance
type-of	:	the type of a feature (straight line, etc.)
part-of	:	the parent feature in which a feature is included in a structure
component-features	:	component features of a feature
component-relations	:	relations between components
attribute	:	list of attribute values calculated according to a feature definition
relations-to-other-features	:	relations which a feature has with other features
position	:	data for a location

Table 3.5: Structure for a relation instance

identifier	:	unique id is assigned for each feature instance
type-of	:	the type of a relation (connect, etc.)
component-features	:	features which satisfy a relation
attribute	:	attribute values calculated according to a relation definition

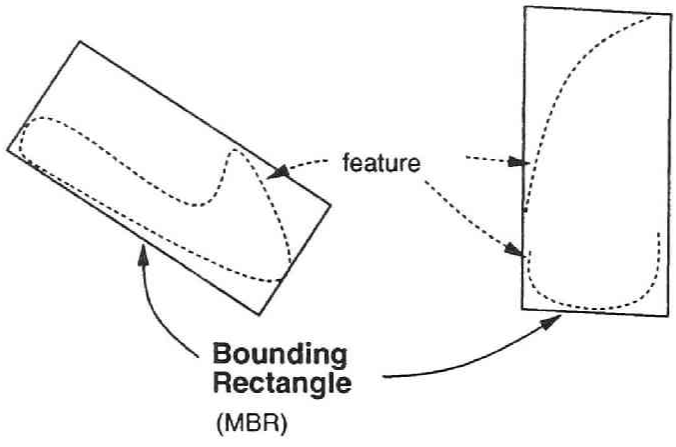


Figure 3.11: Existence area for real features

that, the existence area for each feature is given to each feature instance. This area is calculated from the occupancy of the feature in the image. Then it is expressed as a rectangle[3] for simplicity, which can take any size and orientation (this is shown in Fig. 3.11 in the following section).

As for relation instances, if a set of features satisfies all conditions, a relation instance is generated, attributes and component features are calculated, and then this relation instance is registered in features which have this relation.

Hypothesized Feature Instances

Another type of feature instance is utilized to express a hypothesis and a feature extraction request. This type of feature instance is called *hypothesized feature instance* (say f^h) whose structure is similar to that of a real feature instance except that all the attributes and component features do not have to be given. Given attributes are thought of as constraints which are used

[3]That is, a MBR (Minimum Bounding Rectangle) which encloses the feature (unless another calculation method is given in the feature definition).

for checking the candidates extracted for this hypothesized feature instance. Given components are thought as pre-extracted parts of the feature. Assume that f was defined to have three component features. Given a f^h with only one component feature, the system tries to extract the remaining two features and checks their combinations. In a sense, this is similar to the unification operation in logic programming. The area of hypothesized feature in the image is expressed with a rectangle (similar to the real feature instance case) in which candidate features for it are expected to be in.

In the case of a relation, a relation instance is generated when the relation is detected, and it is treated in a similar as the feature instance case. There are also two types of relation instances, which are the real relation instance (r^r) and the hypothesized relation instance (r^h).

Evaluation of Extracted Features

For a real feature instance, evaluation (scoring) is often necessary to judge whether it is acceptable as a result or not. The feature which receives score that is worse than threshold value is rejected and not registered as an extracted feature. Feature evaluation, however, is a quite difficult problem, so we decided to use a simple method for the time being and leave it for further investigation to establish a good evaluation method. In this system, the score of an extracted feature is a matching degree between the hypothesized feature instance (which is a request) and the real feature instance (which is the data). It is calculated based on the spatial arrangement of components and their scores as follows:

$$S_0 = \frac{\sum_i (Sp_i + S_i) \cdot As_i}{2 \sum_i A_i}$$

where, S_0 is the score itself, Sp is the spatial matching degree (for each component) between the expected position and real position, S_i is the score of the i-th component, and As_i is the square measured area of the i-th component.

The expected positions of components are calculated as follows:

$$Area_0^r = cal\_area(Area_1^r, ..., Area_n^r)$$

$$\begin{aligned} T_0 &= get\_transform(Area_0^r, Area_0^h) \\ Area_i^e &= transform(Area_i^h, T_0) \end{aligned}$$

where cal_area is a function for calculating the total area from the components' area, $get\_transform$ and $transform$ are functions which calculate transformation parameters between two areas and which transform an area with these transformation parameters respectively. $Area_i^h$ is an area defined in the feature definition.

The score for the spatial matching degree is based on those areas' overlapping degree, that is the overlapping area of $Area_i^r$ and $Area_i^e$. If these areas are completely identical, the score is 0. Therefore the lower the score, the more acceptable is the feature.

3.5 Examples of Feature Definitions

Examples of feature definitions for several kinds of targets are presented in this section. Two of the examples concern for the recognition of indoor objects and another one concerns the recognition of aerial photographs.

3.5.1 Indoor Objects

We applied this system to two sets of indoor objects, one of which is a set of blocks which are mostly composed of straight lines, the other one is a set of tools which are mostly composed of curves. In both experiments, features are based on contour lines, in other words, primitive features are composed of straight lines, curves and so on. Other features are defined as combinations of these.

Recognition objects: (1) Toy blocks, (2) Tools

Features, relations: Features and relations are defined as follows (shown in Fig. 3.12, 3.13). Features based on curve and relations between curve is illustrated in

features: superclass for images, original image, differentiated image, binary image, thinned image, straight line, curve, and compound features.

relation: connect, adjacent, collinear, parallel, etc.

For objects like blocks that are composed of straight lines, the intermediate features and objects can be quite well expressed in our framework. Although expressing feature descriptions for tools is more difficult than for blocks, they are well defined in this system.

3.5.2 Aerial Photographs

The following example shows how the PAFE system can be applied to aerial photographs. Regional features are suitable for the recognition of aerial photographs, since the recognition targets in aerial photographs may be considered as almost two dimensional regions whose detailed shapes are not important for recognition. For this reason, features in this experiment are mostly based on regional features. Therefore the example shows the ability of our system to handle the regional features.

Several types of regions are defined for the recognition of the semantic classes *road, house, residential area, crop field, plant area* and so on. The overview of the features is shown in Fig. 3.15. The attributes assigned to the *region* features are shown in Table. 3.6. The relations defined for the recognition task are the relationships between regional features. These are based on the relationships between two regions as shown in Fig. 3.16.

In this hierarchy, several feature detections are based on is-a relationships. For instance, a *large region* whose attributes satisfy the conditions for a *road region* may be detected as a road region. In addition to that, a special relationship *identical* is utilized for detection. This relation is used when a new feature can be defined based using more than two features on the condition that they share one physical structure in an image (or in the real world). For instance, the extraction of road regions from elongated regions and large regions implies

(1) Example for Toy Blocks (based on straight lines as primitives)

Feature	Input Features (Images)	Extraction Type	Method (Relations)
Differentiated Image	Original Image	Image Processing	Sobel operator
Binary Image	Differentiated Image	Image Processing	Fixed Threshold
Edge Image	Binary Image	Image Processing	Thinning
Straight line1	Edge Image	Feature Extraction1	Slit Method
Straight line2	Straight line1 × any number	Synthesis	Collinear, Connect
Corner	Straight line2 × 2	Synthesis	Connect
Line Sequence	Straight line2 × any number	Synthesis	Connect
Parallel lines	Straight line2 × 2	Synthesis	Parallel
Rectangle	Corner × 4	Synthesis	Corner-aligned
Rectangle	Parallel lines × 2	Synthesis	Parallel-lines-attach
Rectangle	Straight line2 × 4	Synthesis	Connect, Parallel
Arrow	Rectangle Triangle	Synthesis	Polygon-parallel

where,
Feature Extraction1: Feature Extraction from Images
Feature Extraction2: Feature Extraction from Features
Straight Line1: Extracted from an Image
Straight Line2: Collinear lines combined into one line

Figure 3.12: Definitions of Feature Extraction

(2) Example for Tools (based on curves as primitives)
In this example, an original image is binarized and then curves are extracted from this binary image.

Feature	Input Features (Images)	Type	Method
curve-prim	Edge Image	Feature Extraction1	Edge trace
curve	curve-prim	Synthesis	Curve-aligned
corner-curve	unified-curve	Feature Extraction2	Divide by curvature
flat-curve	corner-curve	Feature Extraction2	Divide by curvature
corner (beak)	any kind of curve	Synthesis	Curve-connect
hole	corner-curve × any number	Synthesis	Curve-connect
u-shape	flat-curve, corner-curve	Synthesis	Parallel, Connect
wrench-handle	u-shape, hole	Synthesis	Include
wrench-head	corner-curve	Synthesis	Curve-corner-connect
wrench	wrench-handle, wrench-head	Synthesis	Head-and-handle-attach
pliers	u-shape × 2, beak, hole	Synthesis	U-shape-parallel, etc

Figure 3.13: Definitions of Feature Extraction

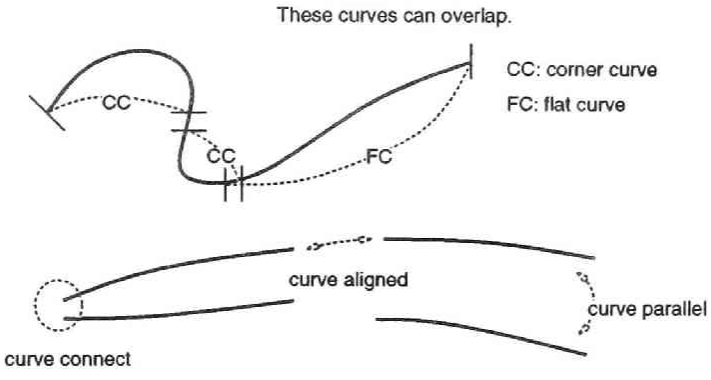


Figure 3.14: Curve Definition and their Relations

that "if a region is an elongated region and is a large region at the same time, and if it satisfies some additional conditions then it is a road region". There are, of course, detections of other types, for instance, a house region is detected by searching for a small region which is close to a road region.

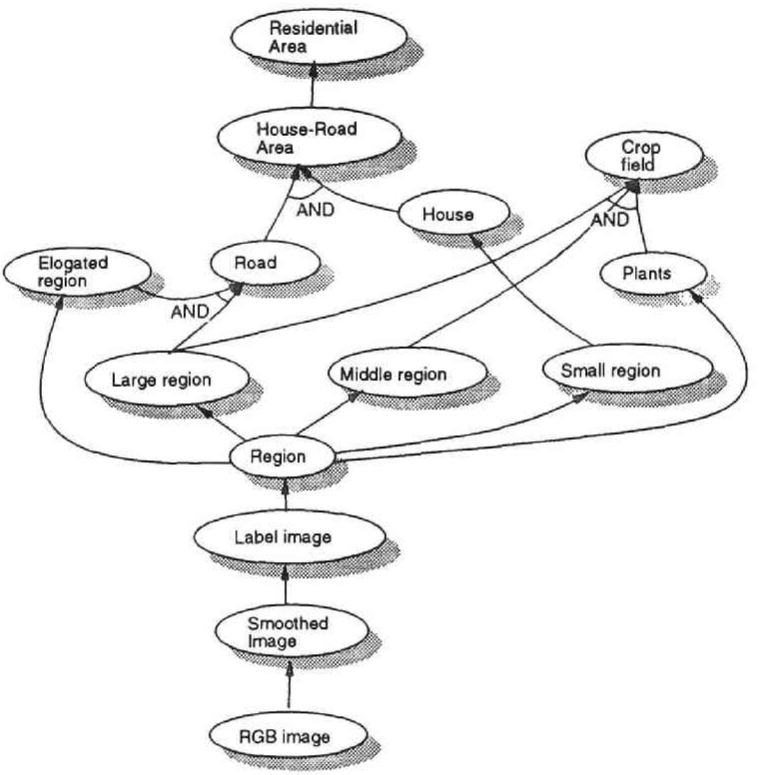


Figure 3.15: Features used for the recognition of Aerial Photographs

Table 3.6: Attributes for regional features

region-image: A pointer to the label image from which the region was extracted.

starting-point: An upper-left pixel in the region.

area: The area (number of pixels) of the region.

region-id: A unique id for the region.

color: The average level for the Red, Green and Blue components of the region pixels.

contour: The sequence of boundary points.

MBR: A Minimum Boundary Rectangle for the region.

direction: The direction of the longest side of the MBR.

eccentricity: The aspect ratio of the MBR.

centroid: The gravity center of the region.

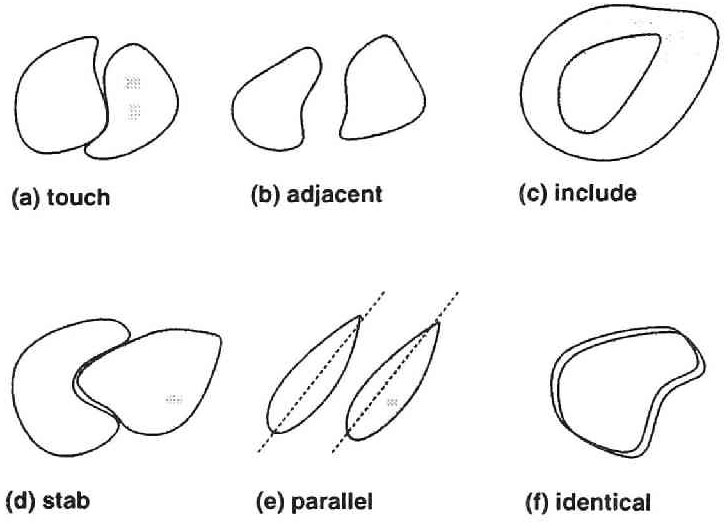


Figure 3.16: Relations for Regional Features

Chapter 4

Organization of Agents in PAFE

In the previous chapter, the framework of feature definition and feature extraction definition in PAFE was presented. This chapter describes the system configuration of PAFE is presented. The focus is on the advantages of the multi agent model for image analysis.

The PAFE system consists of agents (based on the multi agents model) as briefly mentioned in the previous chapter. Although the configuration of the part for execution is similar to the blackboard model, control is much different from the blackboard model. In this chapter, the blackboard model and its problems are introduced briefly first. Then the multi agent model and the detailed configuration of PAFE is described.

4.1 Blackboard Model

There are a system that interprets aerial photographs with blackboard model [Nii86a, Nii86b] by Nagao and Matsuyama [NM80], ACRONYM [BGB79] for the interpretation of 3D scenes by Brooks, a system for the understanding of outdoor scenes by Ohta [Oht87]. In these models, data, i.e. images and features extracted from them, are stored in shared memory which can be accessed by

many other modules. The rules or procedures, i.e. knowledge about the target objects and knowledge about the operations carried out in those systems, process and rewrite (add, delete or modify) the data in the shared memory. They offer a quite flexible framework for image interpretation, since various kinds of data and knowledge can be handled in the same framework. While success has been demonstrated to various degrees, developing a domain-independent and systematic framework for constructing knowledge-based image interpretation systems is still an open problem.

4.1.1 Outline of the Blackboard Model

The logical structure of the blackboard model is usually described as follows.

The Knowledge Sources (KS): The knowledge needed for solving the problem is partitioned into knowledge sources, which are kept separate and independent.

The blackboard data structure : The Problem-solving state data are kept in a global database, the blackboard. Knowledge sources produce changes to the blackboard that incrementally lead to a solution of the problem. Communication and interaction among the knowledge sources takes place solely through the blackboard.

Control : The knowledge sources respond opportunistically to changes in the blackboard.

The knowledge sources can be realized as procedures, sets of rules, or logic assertions. Most of the knowledge sources in all established systems have been represented as either procedures or as sets of rules. However, systems that deal with signal processing either make liberal use of procedures in their rules or use both rule sets and procedurally encoded knowledge sources. The knowledge sources modify only the blackboard or control data structures.

The blackboard holds computational and solution-state data needed by and produced by the knowledge sources. The knowledge sources use the blackboard

data to interact with each other indirectly. Sometimes the blackboard is hierarchically organized into levels of analysis.

For image analysis, the blackboard objects corresponds to the storage of images and features detected from images and knowledge sources correspond to procedures for image processing, feature extraction and reasoning.

The advantages of the Blackboard Model are as follows:

- It can handle a large solution space.
- A variety of input data can be integrated.
- Many independent or semi-independent pieces of knowledge can be organized.
- An incremental and opportunistic problem-solving approach is provided.

Several systems for image analysis realized with this model have focused on these advantages, especially on independence of the modules for feature extraction and reasoning.

4.1.2 Problems in Blackboard Model

As shown above, the blackboard model has characteristics which are suitable for image analysis. Several problems, however, remain unsolved:

- The ways in which objects and image features are defined.
- The ways of knowledge decomposition into (semi-)independent pieces.
- Maintenance of data consistency in spite of the fact that the data is modified by many knowledge sources.
- Parallelism of execution of independent knowledge sources.

In some systems, these points were not considered at all, whereas in others solutions are suggested which are not general enough. For instance, VISIONS proposed a hierarchical organization of the blackboard, which is decomposed into several levels according to the semantic levels of the data such as line, region, face, object and so on. The system by Nagao and Matsuyama used a structure for the scene at the semantic level. SPAM considered several levels for the description of an airport such as fragments, regions, functional area and airport. However, these systems do not propose a framework in which users can define their own data and their own knowledge sources.

The problem of knowledge decomposition and the problem of feature definition are linked by a tight relationship which is not to be neglected. Parallelism and knowledge decomposition are also tightly linked. Therefore, solutions for one problem is essentially relating to other problems. This means that solutions for each problem need to be considered at the same time.

4.2 Configuration of PAFE

Although the blackboard model has characteristics that are well suited for image analysis, a new framework for overcoming the weak points of the blackboard model is needed. The PAFE provides such a framework. This system is constructed on the basis of the multi agent model which is more flexible than the blackboard model.

4.2.1 Multiple Agent Model

Recently, frameworks of Multiple Agents have been proposed and incorporated into several information processing systems. In the robotics field, the research by Brooks [Bro91] is famous for its successful creatures which simulate natural insects. Although the research in this paper is unrelated to his proposition of intelligence without representation, the multi agents model itself is a quite efficient framework for handling complex problems.

The model can be defined as follows[1]:

- Agents have their own tasks as their specialties, i.e. methods for execution and purposes for their calculations.
- Agents act independently of each other.
- Communications are taken through certain pre-defined methods, for instance message passing.

The blackboard model can be considered as a restricted multi agent model in which communication is limited to referring to the data on the blackboard. Though the KS can be directly considered as an agent, the dependencies among the execution of KS's are still remained. Especially the parallelism can not be realized without modifications. Therefore, another framework called concurrent object model was incorporated into PAFE system.

4.2.2 Overview of PAFE

The PAFE system is composed of the following modules:

- The 'Feature Definition Module' for feature (and relation) definition
- A module in which primitive routines for image processing and feature extraction are gathered
- 'Feature Management Agents' for storing image data and features
- 'Feature Extraction Agents' for feature extraction

The agents for feature extraction and the agents for feature management are generated according to the feature definitions. Feature extraction is carried out through the activities of the agents and their interactions. The execution in this system can be described roughly as follows.

[1]Since many researches proposed the agent model independently, the common standard has not been established yet.

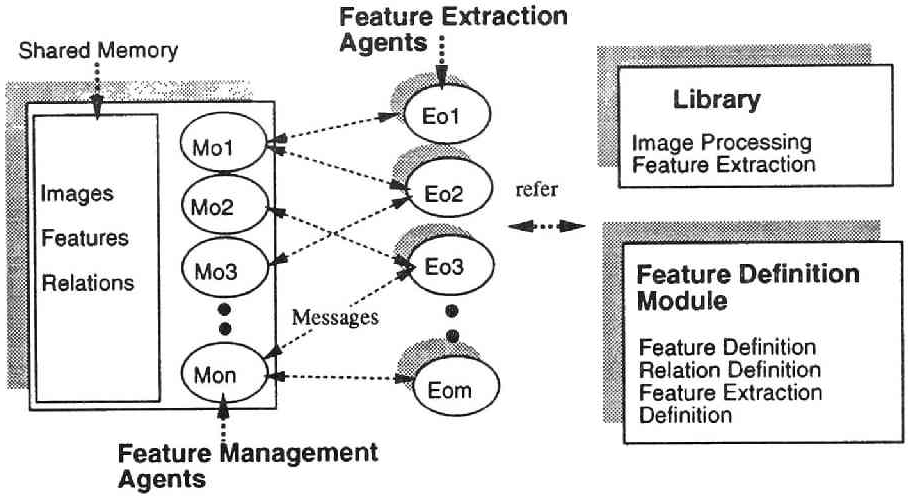


Figure 4.1: System Configuration

- (Off-line) features and their extraction methods are defined.
- Feature Management Agents are created.
- A feature is requested by the users or other systems.
- Feature Extraction Agents are generated from the feature definition.
- Feature Extraction Agents request features to Feature Management Agents. They apply feature extraction operations to the features (requested to Feature Management Agents) if they are given. Extracted features are then registered by the Feature Management Agents.
- Feature Management Agents store features, and provide features, if requested.

In the following sections, the agents and their execution are presented.

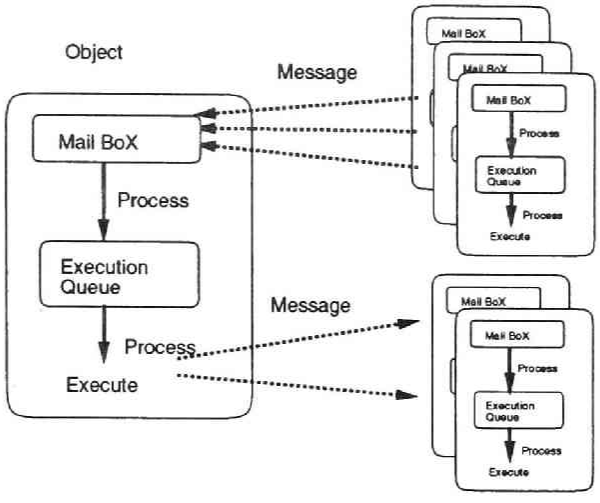


Figure 4.2: Concurrent object model

4.2.3 Concurrent Object Model

The multi agent model introduced into the PAFE system is realized with the software of the concurrent object model [YSHT87]. In this model, a process is exclusively assigned to each object for executing its action independently of the actions of other objects. An object sends messages to other objects and receives messages that affect the object's internal state and its action. This is shown in Fig. 4.2. This framework provides a good base for the implementation of the agent model.

For the implementation of this system, an object is realized as an object in an object oriented language to which two processes are attached for reading messages and executing messages, respectively. It has a mailbox to which any objects can send (write in reality) a message.

In this model, each object has its local memory and its own process for executing its job concurrently. Objects exchange information by message passing.

This system provides two types of message passing, one of which is 'send'

(only message sending) and the other is 'send-and-wait' (message sending and waiting for reply message) which is used when a reply value or synchronization is required. In this system, similar to ABCL/I, neither global time nor a scheduler exists and only local time (schedule) is valid. The only method for synchronization available is message passing.

This model is implemented by software simulation similar to ABCL/I. The simulation program was developed on LISP machine (Symbolics) using the ordinary process scheduler of Symbolics OS (Genera). Therefore the parallel processing of concurrent object model is simulated by the scheduler on a single processor.

4.3 Feature Extraction Agent

As was shown in Section 4.2, there are two types of agents in this system, which are called Feature Extraction Agent (EA) and Feature Management Agent (MA). An agent of either type works as a concurrent object. In this section, details of Feature Extraction Agents are presented.

The structure of this agent is shown in Table 4.1. EA holds the data of the feature to be extracted, the data for the extraction operations and the data for the features requested to the Feature Management Agents. The EA is automatically generated according to the feature definitions.

4.3.1 The Flow of Execution

The flow of EA's execution is shown in Fig. 4.3. The operations of an EA are composed of two phases. Each phase is triggered by a message from other agents, and finishes by sending messages to other agents.

In the first phase (say request phase), the EA is activated by receiving a hypothesized feature f_i^h as a request for feature extraction. The EA selects the extraction method from the given methods in the feature extraction definition. Then, it requests features by sending hypothesized features (f_j^h) to feature management agents (MAs), and becomes inactive until it gets other messages.

Table 4.1: Structure of the Feature Extraction Agent

identifier	:	unique id assigned to each object
feature-name	:	name of a feature to be extracted
component-features	:	components of a feature
component-relations	:	relations included in a feature
constraints	:	constraints for a feature
structure-variation	:	labeling variation for a feature
attribute	:	attribute definition for a feature
registration	:	registration definition for a feature
extraction-methods	:	extraction methods defined for a feature
current-operation	:	a method which is being executed
operation-list	:	methods which are to be executed
requested-feature-list	:	a list of features which are requested to feature management agents
detected-feature-list	:	a list of features which have already been sent from feature management agents
feature-management-agents	:	a list of feature management agents from which an agent requests features, and to which this agent sends extracted features.

In the second phase (say extraction phase), a message from an MA (a reply which include the detected features f^r) activates an EA for the execution of the feature extraction. The extracted features ($f^{r'}$) are sent to the MAs where they get registered. Finally, the EA makes a judgement about the result, and if the the judgement is positive, the EA sends a request to the MAs to stop further execution to extract requesting feature. Or it sends requests for features to the MAs again, if the result is not satisfactory.

Each feature extraction cycle initiated in response to a request for feature extraction consists of the following two phases.

1. **Request Phase:**

   activation: A hypothesized feature is sent from an MA to an EA as a request of the extraction. The EA selects an extraction method.

   messages: The features required for the extraction method are requested to MAs by messages from the EA.

   termination: The EA becomes inactive (The EA waits for messages).

2. **Extraction Phase:**

   activation: One of the MAs activates the EA by sending a message (Detected features are given).

   messages: Detected features are sent to MAs.

   termination1: A request for terminating the extraction of requested feature is sent to MA from the EA.

   termination2: The request for features required for the extraction is sent to MA again.

Any messages can be received even if the agent is executing in the above two phase. In other words, more than one request can be handled at the same time. If another request for the extraction of a different feature is sent to the

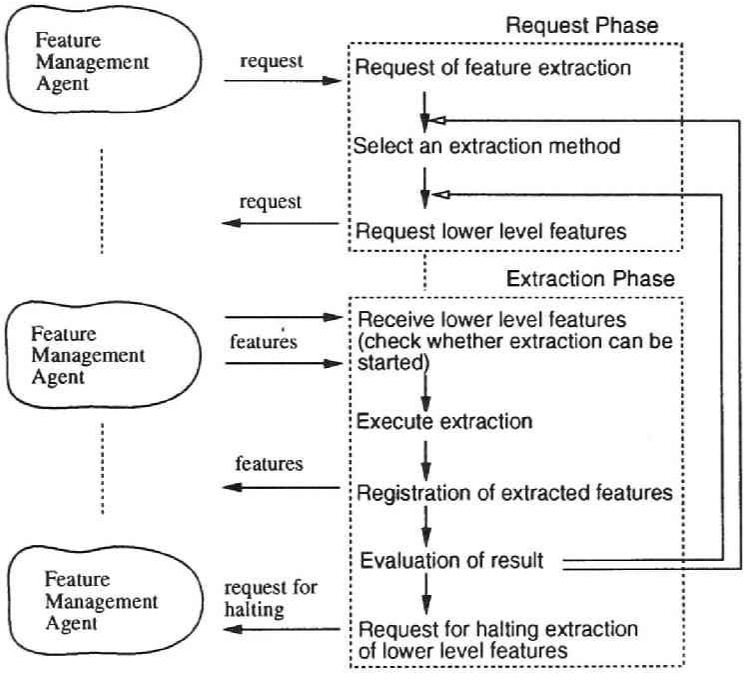


Figure 4.3: Flow of execution of the feature extraction agent

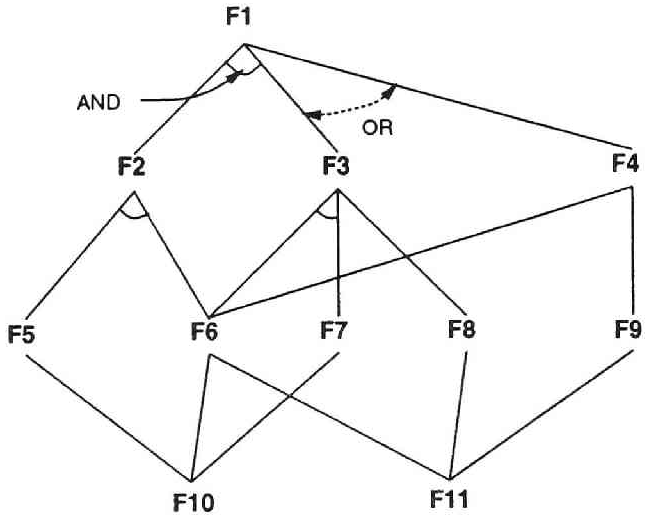


Figure 4.4: Graph for Feature Extraction

EA during the execution of the previous request, the EA can start a new cycle for the request independently on the previous request.

In the following sections, each operation is described in detail.

4.3.2 Selection of The Extraction Method

When a request (hypothesized feature) is sent by MA, EA is activated and then the extraction method is chosen to be executed. As more than one extraction method can be defined for a given feature, it is necessary for the agent to select one (or some) to execute. This corresponds to choosing some arc(s) in Fig. 4.4. The arcs related by AND lead to a sets of component feature nodes which must be chosen at the same time, while arcs that are related by OR (no AND is given) lead to feature nodes which can be chosen independently.

In selecting methods, global minimization of expected costs for the extraction of a feature in a distributed environment is a difficult problem. In

this system, therefore, the agent chooses the one method that is judged to be maximumally effective (this judgement is attached to the definition) for this particular extraction situation among all methods that have not been tried yet.

The number of the methods chosen at one time is important for the parallelism and efficiency of the execution. In this situation, two major strategies can be considered. One is to choose only one method, and the other one is to choose two or more methods at the same time. As we brought into our system the mechanism of the concurrent object model, simultaneous actions by different objects can occur. This means that some methods can be chosen in parallel. It is obvious, however, that a control strategy is required to avoid execution explosion and to make these objects work consistently. Therefore basically only one extraction method is chosen at one time for the above reason (The details are given in the next chapter, which is devoted to describe control strategy employed in the PAFE system).

4.3.3 Execution of Extraction

On receiving features from MAs, EA checks whether it is able to start execution. Assume that feature f_0 is defined as a structure consisting of features f_1, f_2 and a relation $r(f_1, f_2)$ between f_1 and f_2. Even if f_1 and f_2 were sent to the agent, it would not make sense to start execution without having detected $r(f_1, f_2)$. This dependency is detected previously when this agent is generated.

If the required input is only a single feature type, the EA will be ready to begin execution at any time after the input feature has become available. In other words, it is not necessary for the extraction agent to wait until the extraction of input features is completed. (The unmarked arcs in Fig. 4.5 correspond to this type.) If multiple features are necessary as input, the feature extraction agent basically has to wait for all input features to become available. The arcs marked with AND in Fig. 4.5 correspond to this type of extraction. (However, since waiting for all the features to become available is inefficient, we propose a new method which perform extractions incrementally. This method is described in the next chapter)

EA then starts the execution of feature extraction by searching or applying procedure defined in feature extraction definitions. Some methods may require only the application of image processing to input features (images), whereas others may require a search for component features to be used in feature synthesis in which relations and constraints between them have to be considered (This search was described in the previous chapter). The detected features are sent to MA, which initially requested the hypothesized feature from/ this agent.

4.4 Feature Management Agent

The Feature Management Agent (MA) manages the storage which holds images, features and relations that were extracted by EAs. The MS keeps the features which need to be extracted some time in the future, the data needed as input to the extraction operations, and the data obtained through feature extraction process by EAs. It provides stored features on request and identifies newly received features. As mentioned above, more than one of this agent of this kind can exist such as multiple MAs for only one kind of feature or MAs that are assigned to a set of tightly related features. For handling spatial spatial parallelism, more than one MA can be generated, each of which may be assigned to a spatially separated area (this is discussed in the Section 4.5).

The structure of the MA is shown in Table ??. There are two kinds of agents for feature management. One of them is a *mesh object* which really holds the feature data, and the other is a *plane object* which only delivers messages. This separation is introduced in order to handle the spatial division mentioned above. Multiple planes can exist as shown in Fig. 4.6. The reason for this division is that if all the features are assigned to only one mesh object of one plane object, the accesses to a mesh object may be concentrated and the communication to the mesh object may become bottle-necked.

The agent for the plane receives mainly requests for features from EAs, then, if required, duplicates the requests and and delivers them to each mesh object

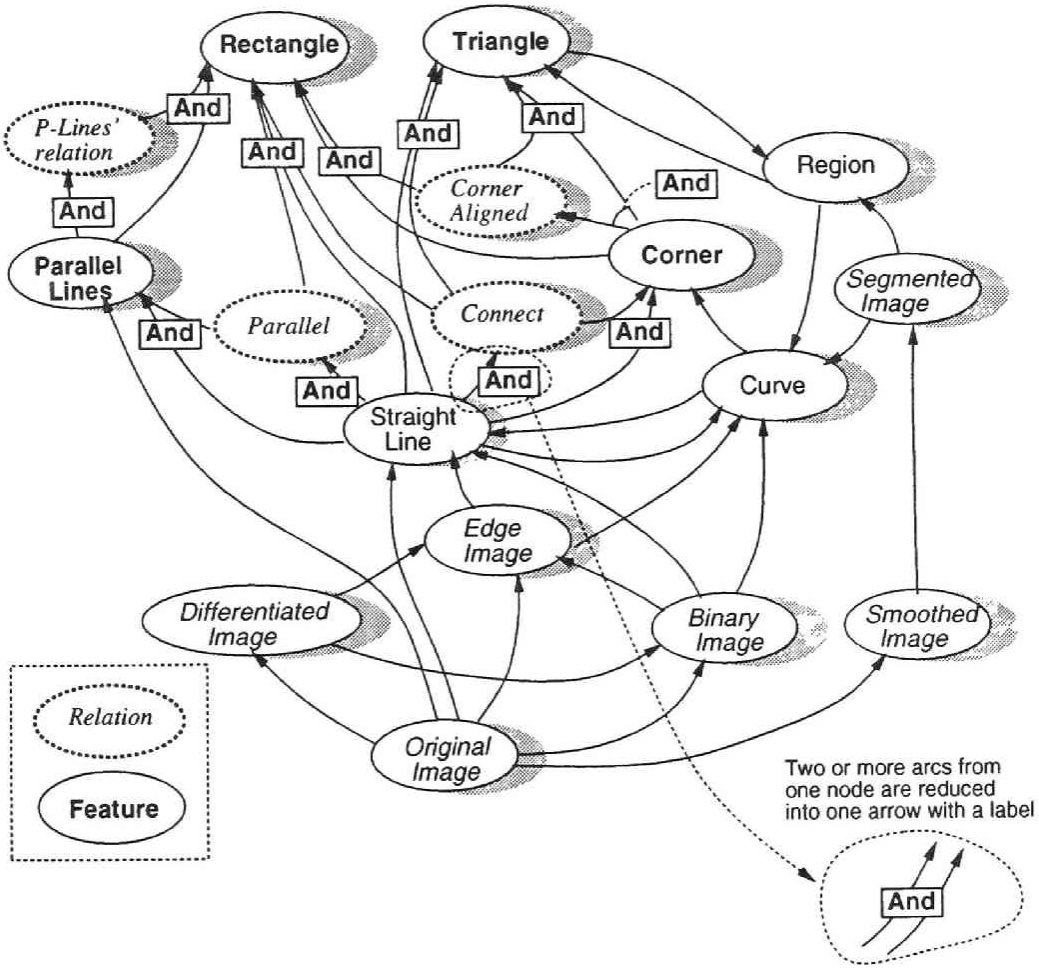


Figure 4.5: Execution of Feature Extraction

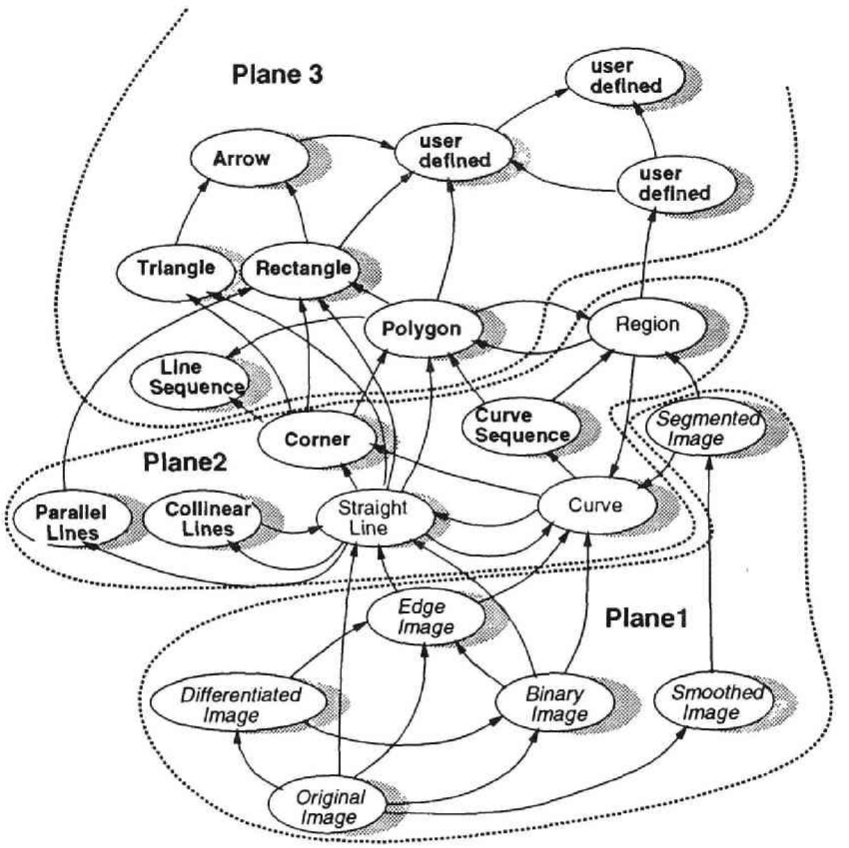


Figure 4.6: Example of Multiple Planes

Table 4.2: The structure of Feature Management Agents
(Agent for the management of the plane)

identifier	:	unique id assigned to each object
layer	:	level of a plane
window-size	:	the size of an image plane to be considered
mesh-number	:	the number of the spatial division
mesh-width	:	the width of each division (mesh)
mesh-array	:	an array whose elements are the pointers to mesh objects

whose area is overlapping or included in the area of the request. On the other hand, the extracted features are also sent directly from EA to plane objects because EA does not know to which mesh object the area of an extracted feature should be assigned. Other messages are sent directly from the EAs to mesh objects. However, the plane objects and the mesh objects can be treated as one agent in most cases. In the following, therefore, we do not mention this separation for simplicity unless it is essential for the discussion.

4.4.1 Flow of Execution

The flow of execution for the MAs is shown in Fig. 4.7. It is composed of two groups of operations. One is the phase of receiving requests for features. The other one is the phase of receiving the extracted features.

In the first phase (say request phase), MA is activated by receiving a hypothesized feature f_i^h, which is a request for the feature extraction from EAs, the users or other systems. It checks the identity of the requested feature with the previously requested features stored in the "detection-map" (shown in Table 4.3). If an identical hypothesized feature is found, the MA merges the requested one and identical one requested previously. Then, if candidate features are already extracted, the MA replies the features to the requesting

Table 4.3: The structure of the Feature Management Agent
(Agent for the management of the each mesh)

identifier	:	unique id assigned to each object
position	:	position in a plane
width	:	width of a mesh
center	:	center of a mesh
plane-object	:	pointer to a plane object
EA-feature-list	:	a list of pointers to feature extraction agents handling the features
EA-relation-list	:	a list of the pointers to feature extraction agents handling the relations
extraction-status	:	a map in which the status of feature extraction execution is stored
feature-list	:	a list of the extracted features
identification-method	:	a list of identification methods for features (mentioned in Section 3.2.2)

EA. If not, it sends the hypothesized feature (f_i^h) to the EA. If the EA which extracts that kind of feature does not exist, the MA creates new EA to extract it before sending the message. It then becomes inactive until it gets other messages.

In the second phase (say extraction phase), a message from EA (reply with the detected features) activates the MA. On receiving the message (that are reply with f^rs) from MAs, it becomes active again and identifies extracted features. The unique (that is, not identified features) extracted features ($f^{r'}$s) are sent to EAs that requested the feature previously.

The activation and termination for each phase are as follows.

1. **Request Phase:**

   activation A hypothesized feature is sent.

   termination1 If the candidate features for it is already extracted, it simply replies them.

   termination2 If the EA to extract the feature exists, send the request (hypothesized feature) to the EA. If the EA does not exist, it creates new EA for the feature.

2. **Extraction Phase:**

   activation Replied detected features from the EAs

   termination1 Extracted features are sent to the EAs which requested them previously.

Each execution for a request of a feature extraction has above two phases similar to the EA's action. More than one request can be handled at the same time. The order in which feature extraction requests are executed is not fixed, i.e. that request for which the needed feature become available first is executed first by the MA.

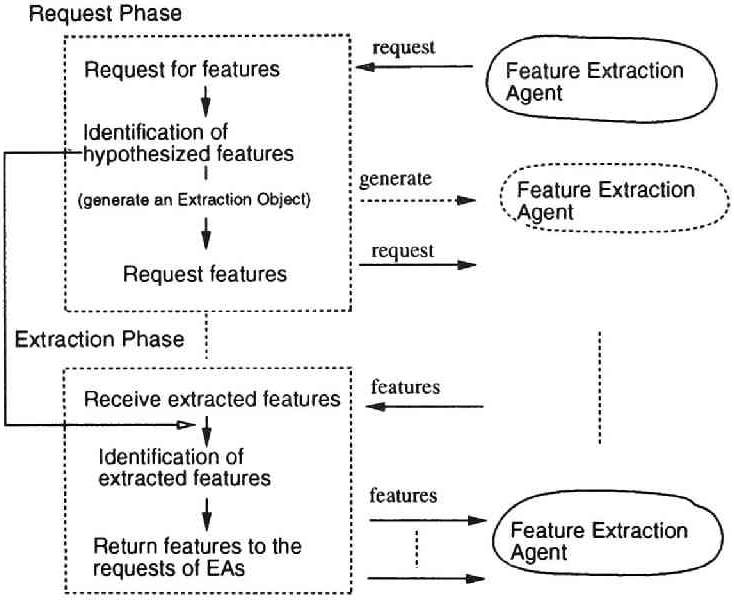


Figure 4.7: Flow of operations executed by MAs

4.4.2 Identification of Features

The identification operation is necessary, because essentially identical features may be requested or extracted more than one time. For example, this situation occurs in a case where more than two upper level features contain the same kind of feature and request, or in the case in which essentially identical features are extracted by applying more than one extraction method defined for a single kind of feature.

For hypothesized features, this identification avoids the repetition of essentially identical operations. For real features, it avoids the situation where more than one essentially identical features are stored at the same position for the same feature. In such situation more than one essentially identical upper level features to be extracted, and it would eventually lead to combinatorial explosion.

Identification of Hypothesized Features

The area of a hypothesized feature is expressed with a rectangle in which candidate features for it are expected to exist (mentioned in the previous chapter). The identification of hypothesized features is calculated by checking mutual inclusion of these areas of features. The hypothesized features are stored in the MA with a hierarchical organization as shown in Fig. 4.8.

In this way, the system can merge hypothesized features, if identical ones already exist in the current map. By merging them, the MA can avoid sending redundant extraction requests to EAs. Merging makes it possible to let features, whose areas are included in the areas of other features, share the candidate features of these superordinate features, and vice versa (some check for inclusion is still required).

Identification of Real Features

The area of a real feature is expressed with a minimum bounding rectangle that encloses the feature. The identification of a real feature is calculated by

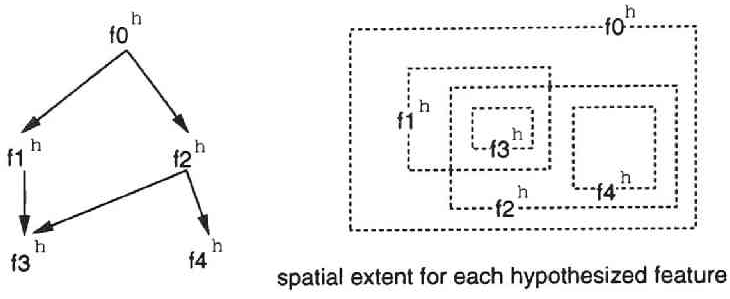


Figure 4.8: Management of Hypothesized Features

the following two operations.

Identification by Labeling Variations

First, for the structural features which have plural components, the variants of the component labeling are checked. The identification is necessary, because identical features may have been extracted. Consider that a rectangle can be extracted as a combination of the four lines (l_1, l_2, l_3, l_4), or as some other combination such as (l_2, l_3, l_4, l_1). If there already are identical features in the storage (managed by the MA), newly detected feature is rejected and abandoned.

Identification by area

Second, for all features, identification is checked with the ratio of the overlapping section to the non-overlapping section as shown in Fig. 4.9. When this ratio exceeds a priori threshold, two features of the same type are merged into a single feature. In this way, the system can reduce redundancy caused by the multiplicity of essentially identical features.

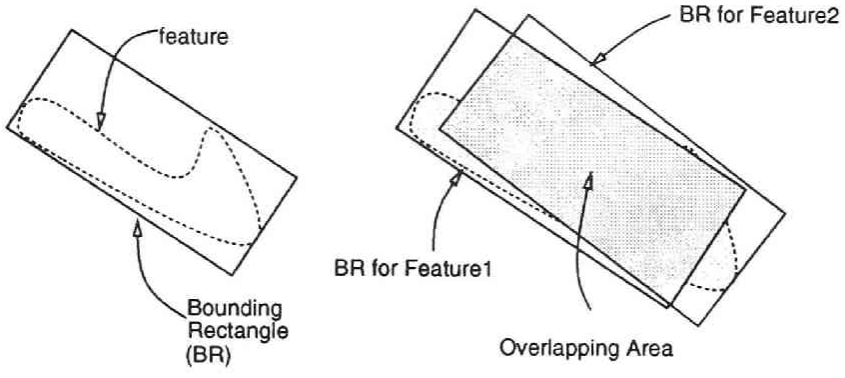


Figure 4.9: Identification of Real Features

4.5 Spatial Division of Feature Extraction

The objective of the PAFE system is not only to provide the system with a general feature description and extraction capability, but also to construct a system suitable for parallel processing. For this purpose, the spatial division of feature extraction is incorporated into the system. The Feature Management Agent and a Feature Extraction Agent are assigned to each spatially divided section. For the MA, a mesh object is assigned to each division as mentioned above. An EA is assigned to each division for each feature.

The system realizes spatial parallelism by their concurrent execution capability. This is effective especially for the extraction of spatially small features and image processing operations which are applied everywhere in the image. However, the extraction of a large feature, which belongs to more than one division, is not so easy because a large feature may lay across sections. However, the system avoids serious problems as follows:

image processing: Image processing composed of local operations such as differentiation can easily be divided. Image processing operations, which use global information such as histogram, are affected by the spatial

division. Therefore such operations are not performed separately in each section[2].

feature extraction from an image: There are two ways to handle this extraction case. One way is to extract parts of a large feature within each division separately and combine afterwards as shown in Fig. 4.10(a). The second way is to extract the entire feature in each extraction process and identify it as a single feature afterwards as shown in Fig. 4.10(b). An example of the former case would be the extraction of a curve, and an example of the latter case would be the extraction of a straight line.

feature extraction from a feature: The registration position of an input feature determines the agents to operate on it. In other words, a Feature Extraction Agent assigned to a particular division gets only the features that are registered in the division.

feature synthesis: If component features are scattered over several different divisions, the EA assigned to the respective divisions executes the extraction of features in their own division. Fig. 4.10 shows an example, where one of the component features of $\{X_i\}$ determines which of the EAs will execute the operation. In other words, the EA assigned to each division starts a search (for the extraction) with X_1 in the division as the first node. In this situation, several variants of of the same feature may be extracted in different meshes. Four identical rectangles, for example, would be extracted by four independent EAs if each side of the rectangle lies in a different division. The number of extracted rectangles, however, is the same as in the situation where the spatial division is not considered. Therefore, spatial division does not increase the total cost, if the number of component features are fixed.

On the other hand, the extraction process for variable-length features can be insufficient. The number of extracted identical features increases with

[2]In reality, some of them can be handled if they are implemented carefully

Table 4.4: Comparison between the blackboard and the multi agent models

	Blackboard Model	Multi Agent Model
Data Structure	flexible	flexible
Date Storage	the blackboard	an agent (or multiple agents)
Primitive Execution	a knowledge source (divided into small operational units)	an agent (a set of grouped operations)
Parallelism	implicit	explicit

spatial division, because the extraction process may include a procedure which only extracts the longest feature.

4.6 Comparison between the Blackboard Model and the Multi Agent Model

In this section, we compare the blackboard model and the multi agent model as a summary of this chapter. The following points are discussed below:

- The data structures used for image analysis
- Data storage
- Primitive procedures for image analysis
- The control mechanisms

Table 4.4 shows the comparison in compressed form.

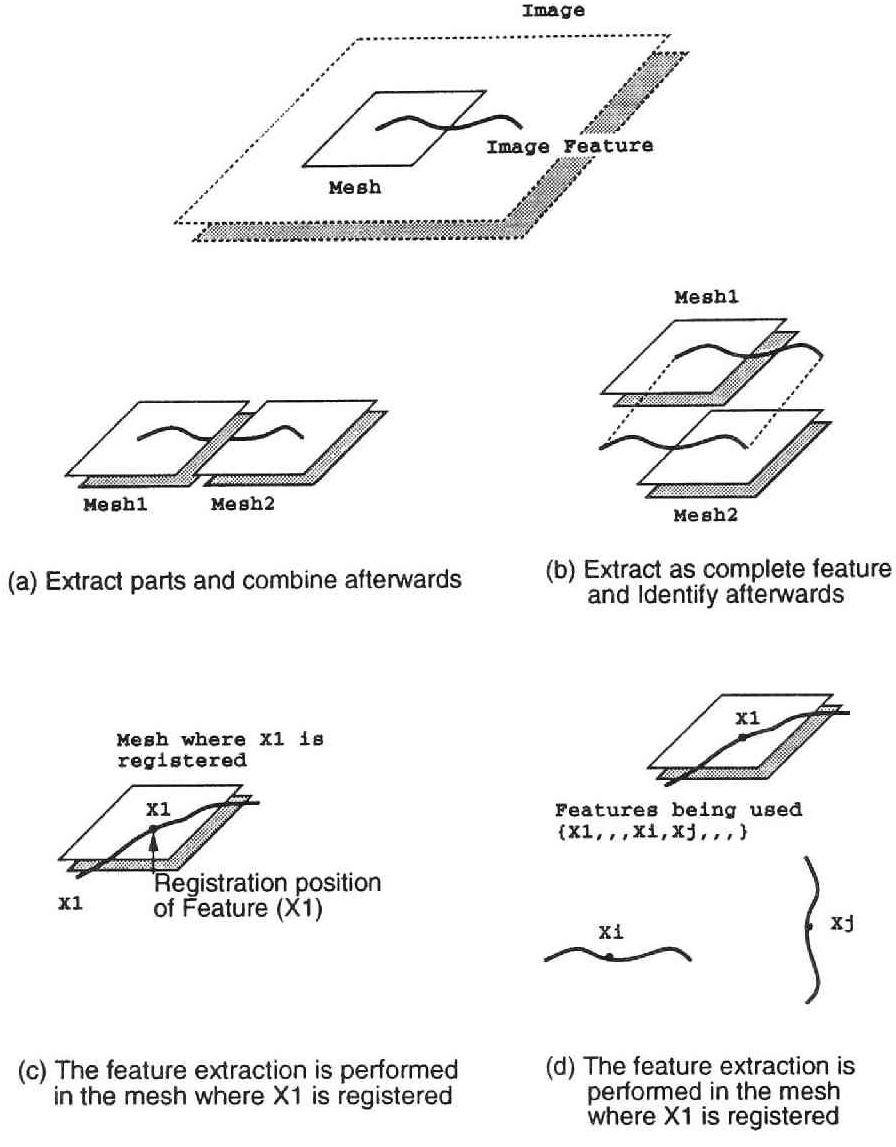


Figure 4.10: Spatial Division of Feature Extraction Process

The Data Structure

Although the blackboard model provides a flexible platform for managing various kinds of data, it does not provide the users the way to define the objects and features. Nor does the multi agent model. Our system provides a flexible way to define them independently of the blackboard model and the multi agent model.

Data Storage

The blackboard object in the blackboard model is a storage of data. In image analysis systems using the blackboard model, features and relations extracted during the recognition process are stored in the blackboard. The data can only be referred to by the knowledge sources. The data storage does not have any process for handling the data.

The data management function is also required in the multi agent model. It may be realized as an agent for data management, or may be realized by making each agent to have its own data independently. In PAFE, agents for data management are provided, which store and handle features and answer the queries by the other agents. The computation can be distributed because the agents for feature management have their own processes for managing data.

Primitive procedures

In the blackboard model, procedures for image processing and analysis are realized as knowledge sources. Each knowledge source has a definition in its condition part for activating it, and a definition in its action part for processing the data. Usually, a knowledge source consists of just one condition and one action.

By contrast, an agent in the multi agent model can respond to many activation patterns. It can be activated by receiving messages from other agents, or by one of its interior processes (when it includes multiple processes). In this sense the agent can be considered as an alternative of a set of rules organized

into a meaningful group. Or an agent can be considered as a small blackboard system embedded in a global blackboard system. In this system, they are automatically generated according to the feature definitions, which contain the methods (routines) for feature extraction. The computation can be distributed because the agents for feature extraction have their own processes for executing feature extraction.

Control mechanism

The control of the blackboard model was presented in Section 4.1. In each execution cycle of execution in the blackboard model, every possibility for each knowledge source to activate itself is checked. This phase consumes much execution time, and is a major obstacle to fast parallel implementation of the blackboard model. Though the possibility of parallel execution (of knowledge sources) does implicitly exist in this model, it causes serious problems such as the difficulties in connection with conflict resolution.

In contrast, the multi agent model incorporates explicit parallelism which allows each agent to independently execute its own jobs. Since mechanism for conflict resolution is not considered in this model, additional operations are required for keeping the consistency of the execution.

The goal of this system is the parallel and incremental feature extraction for the purpose of finding the possible existence of high level features, objects and so on. For this purpose, the problem of conflict resolution is not as serious as seen in image interpretation tasks. The method of extracting features incrementally together with operations for the identification check[3] can be a quite effective method.

Details of the control of feature extraction are presented in the following chapters.

[3]The operation to identify features and merges them if identical.

Chapter 5

Control of Agents in PAFE

In the preceding chapters, the framework for feature definition in PAFE and its system configuration were presented. In this chapter, the control mechanism for performing feature extraction through the cooperation of agents is described.

5.1 Introduction

There is a great need of developing a system that provides parallel and distributed processing. Many algorithms and systems, including special hardware, are proposed for low level image processing. On the contrary, the problems of intermediate and high level processing such as the extraction and recognition of structural features are not well investigated, although there are some systems reported [Dea89, Oht91].

To realize parallel processing in feature extraction, it is efficient to define features structurally as in our system. This makes it possible to divide the extraction of higher level features into lower level feature extraction. In our system, features and their extraction methods are organized in a feature extraction network as described in the preceding sections. They can be defined and extracted in multiple ways. Within the feature extraction network, different paths (that means extraction method) can be executed in parallel. Any combination of extraction methods such as combinations of top-down and bottom-up

can be dealt with in our framework. However, allowing each agent to choose its action arbitrarily (independent of each other and the status of the whole system) causes computational explosion. Mechanisms for controlling the agents are required to achieve efficient feature extraction.

First, in our system, several types of feature extraction are considered. Then, the strategy for controlling the agents' actions is proposed. According to this strategy, the agents realize flexible feature extraction with coarse grained parallel processing.

5.2 Feature Definition and Extraction Definition

5.2.1 Definitions

Features, relations and feature extraction are defined as follows:

$$\begin{aligned} f &::= (P, R, C, A, E) \\ r &::= (P, C, A) \\ e &::= (F_e, M, A_e) \end{aligned}$$

where (f, F) and (p, P) are the features (p is used in the description of a compound feature when a feature p is a component feature of the compound feature), (r, R) are the relations among features, C is the constraints among f and r, (e, E) are the methods for extracting the features, F_e is a set of features used in the extraction, M is the real procedure executed during extraction and A_e is the criterion value for choosing the method e. Each lower-case character expresses an element and each uppercase character expresses a set of elements.

These definitions can be organized hierarchically, in the case of which multiple extraction methods can be assigned to any feature. A feature without components can be defined by E, which means a procedural definition of the feature.

With this notation, feature extraction can be expressed as follows:

$$F_0 \Leftarrow e^1_{\{f_0\}} \odot (F_1, F_2, \cdots, F_n)$$

where the input features for extraction method e^1 are $f_1, \cdots, f_n$, and F_i is a set of candidate features for f_i. f_0 in the above notation can be thought of as a top-down extraction goal. $\odot$ means that e^1 is applied to a set of input features. This makes it possible to specify extraction which processes can not be executed without a goal or a model. Extraction based on multiple extraction methods can be expressed as follows:

$$\begin{aligned} F_0 \Leftarrow\ & e^1_{\{f_0\}} \odot (F_1, \cdots, F_n) \cup e^2_{\{f_0\}} \odot (F_1, \cdots, F_n) \\ & (= (e^1 \oplus e^2)_{\{f_0\}} \odot (F_1, \cdots, F_n)) \end{aligned}$$

where $\oplus$ expresses the operation for making union of the results which are generated by applying extraction methods (e^i) to the same input features.

The hierarchical combination of several extraction methods allows various kinds of flexible extraction. Intermediate features are also extracted during extraction execution of the target feature as shown in the feature extraction network 4.4. For instance, the situation where several extraction methods are applied for the extraction of f_0 can be expressed as follows:

$$F_0 \Leftarrow (e^{11} \oplus \cdots \oplus e^{1i})_{\{f_0\}} \odot ((e^{21} \oplus \cdots \oplus e^{2j})_{\{f_1\}} \odot (\cdots), \cdots, (e^{3l} \cdots)_{\{f_n\}} \odot (\cdots) \cdots)$$

Although the intermediate features, which are the input features for the extraction of higher level features, are shared during the execution of extraction, this is not explicitly expressed in this formula.

With this kind of extraction, any feature, which potentially can be extracted in some way, can be extracted by the following procedure:

For each target f_i, apply every existent extraction method e_j

In practice,it is impossible to apply every existent method, however, because of the following problems:

- The computation would explode, if all the possible combination of operations was carried out.
- Possible candidates of targets (goals) can not be specified before extraction starts (we can not know which new goals will be generated during execution without actually carrying out execution), and among these new goal features may be some which cannot be extracted with the present set of extraction methods.

Therefore, the several strategies are employed in the PAFE system. In this chapter, several types of operations needed for hierarchical extraction of features with feature network will be presented first. Then several strategies are considered.

5.2.2 Fundamental Extraction Operations

For the *hierarchical* extraction of features, the following two operations are essentially required:

(a) Receive the data, which are the result of previous executions, and apply selected operations to the data.

(b) Send the execution result for registration as newly extracted features.

These are very simple operations, which are always required for any kind of feature extraction. Next, we have to consider the following two operations:

(c) Select one or more extraction methods for extracting candidates for the target feature (f_0).

(d) Request features ($f_1 \sim f_n$) required by the selected extraction methods.

These two operations enable the top-down extraction of features by breaking down the higher level target feature (requested at the top level) into lower

level features, including the original image or some images which can be directly manipulated by the system.

Moreover, the evaluation of intermediate results is essential for the flexible and reliable extraction of features. The following two operations are performed according to the evaluation of intermediate extraction results:

(e) If enough results for the target feature f_j cannot be obtained, request *re-extraction* with a different set of extraction methods.

(f) Generate new goals (f_{new}) from the intermediate results.

Every operation executed in this system is expressed as a combination of the above operations (a) through (f).

5.2.3 Primitive Extraction Strategies

The combinations of six fundamental extraction operations mentioned above form several primitive strategies. Four of them are chosen in this system, as shown in Fig. 5.1.

The four feature extraction strategies are described below by utilizing the following notation:

f^h : a request for feature extraction (hypothesized feature)
f_r : an extracted feature
F_r : a set of extracted features f_r
$\Rightarrow$: the operation proceeds in the direction of the arrow
$\otimes$: an operation making combinations of features

In addition to these, the feature f_i can be expressed as $f_i(f_j, f_k)$ if it has f_j and f_k as component features. The operation of making combinations of the features are expressed as follows.

$$F_0^r \Leftarrow F_1^r \otimes \cdots \otimes F_n^r$$

Top-down extraction has three phases.

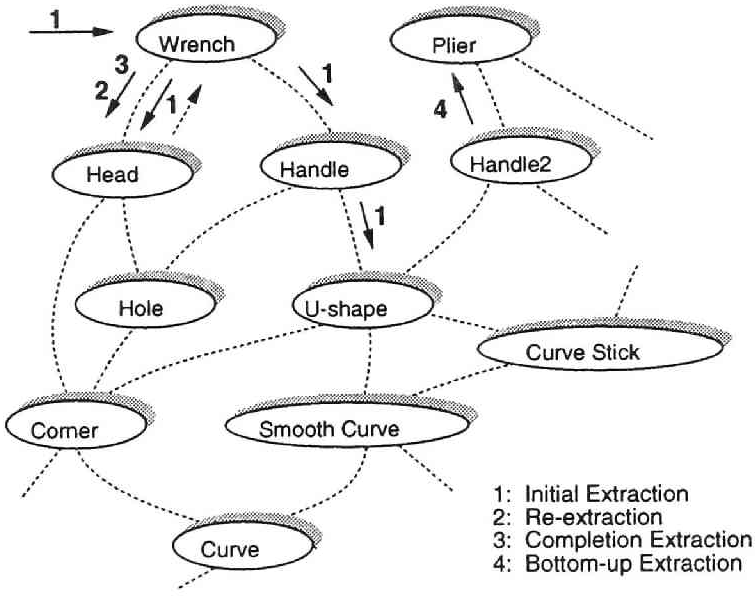


Figure 5.1: Several Primitive Strategies of Feature Extraction

Initial extraction: Given a request for a feature from an upper level agent (or user), the request is recursively broken into lower level feature requests until the request can be satisfied by an extracted feature (for instance, an original image). For example, in the case of the execution example shown in Fig. 5.1, the extraction of a wrench can be denoted by the following formula (this corresponds to operations (c) and (d) in the previous section. The execution corresponds also to (a) and (b)):

$$\begin{aligned} wrench^h_{init} = wrench^h_{init}(head^h_{init}, handle^h_{init}) &\Rightarrow head^h_{init}, handle^h_{init} \Rightarrow \cdots \\ Wrench^r = \{wrench^r(head^r_i, handle^r_j), ..\} &\Leftarrow Head^r \otimes Handle^r \Leftarrow \cdots \hookleftarrow \end{aligned}$$

Re-extraction: If enough results were not obtainable during the execution, the agents request component features again (this causes the *re-extraction* of low level features). This corresponds to operation (e) in the previous section.

$$\begin{aligned} (Wrench^r &\Leftarrow Head^r \otimes Handle^r \Leftarrow \cdots) \\ \hookrightarrow wrench^h_i &\Rightarrow head^h_i \Rightarrow \cdots \end{aligned}$$

Completion extraction: If some features score only moderately (that is, their score is below the acceptance level but higher than the rejection level), *completion extraction* for the confirmation of the feature is required as follows (this corresponds to operation (f) in the previous section):

$$wrench^r(head^r_i, handle^r_j) \Rightarrow wrench^h_{new}(head^h_{new}, handle^r_j) \Rightarrow head^h_{new}$$

In this case, a new extraction for a wrench-head ($head^h_{new}$) is requested for verifying the existence of an extracted wrench ($wrench^r(head^r_i, handle^r_j)$) with moderate score. A new goal for extraction ($wrench^h_{new}(head^h_{new}, handle^r_j)$) is generated during this operation.

The above three strategies are not enough for usual extraction of the features. Additional strategy is required for bottom-up extraction of features, in which higher level features are inferred from the extracted features according to part-of relationships or other relationships in feature definitions.

Inferring and/or extraction of upper level features: In bottom-up direction, a hypothesized feature which is suggested by some extracted features, which may be the components of the feature, is sent to MAs. For this purpose, there exists a table in which the type of the component features is related to be types of structural features. For example, an instance $handle2_i$ may suggest the existence of *plier* according to the feature definition shown in Fig 5.1, which causes the extraction of *head2* to be performed. This corresponds to operation (f) in the previous section.

$$handle2_i^r \Rightarrow plier_{new}^h(head2_{new}^h, handle2_i^r) \Rightarrow head2_{new}^h$$

5.2.4 Basic Control Strategy

In PAFE,strategies used in particular feature extraction situations are composed of the four primitive strategies described above. The basic idea of the combination of the strategies, which is taken in usual extraction task, is presented in this section.

First, let f_0 be a target feature which can be extracted through the extraction of component features f_1 and f_2. The computation without the changes of the target feature is as follows.

$$F_0^r \Leftarrow \bigcup_{f0}(e^{01} \oplus \cdots \oplus e^{0m})_{\{f0\}} \odot (e^{11} \oplus \cdots \oplus e^{1n})_{\{f1\}} \odot F_2^r$$

The *union* ($\bigcup$) expresses an operation for identification of extracted features mentioned in Section 4.4.2.

However, it is efficient to set a target which is only loosely defined ($f0_{loose}$), and then choose one or a small number of extraction methods for the extraction of f_1 and f_2. In this situation, $f0_{loose}$ may be a hypothesized feature whose position or size is not specified, so that extracted features of any size and of any position will match to it.

$$F_0^{loose} \Leftarrow e^{01}_{\{f0_{loose}\}} \odot e^{11}_{\{f1_{loose}\}} \odot F_2^r$$

If the results of the initial extraction with respect to $f0_{loose}$, are not satisfactory, *re-extraction* is required. For this purpose, a *re-extraction* request is sent to the respective EA through the MA. *Re-extraction* (that is $e^{02}_{\{f_0\}}$ in this case) takes place as follows:

$$F_0^{re\text{-}extraction} \Leftarrow e^{01}_{\{f0_{loose}\}} \odot (e^{11} \oplus e^{12})_{\{f1_{loose}\}} \odot F_2^r$$

If there is a candidate feature (f_0^r) for f_0, which is above the rejection threshold,but below the acceptance threshold, the system tries to perform *completion extraction* for that feature. For example, the following extraction operation may take place:

$$F_0^{completion} \Leftarrow e^{01}_{\{f0_{new}\}} \odot e^{12}_{\{f1_{new}\}} \odot F_2^r$$

With this basic strategy, a simple extraction of a requested feature can be performed.

5.3 Overall Control Strategy

The execution of several separate agents can carried out in parallel as was noted before. This implies that the potential for possible parallelism in this framework is quite high.

However, if each agent takes its own operation independently of other agents, it can easily cause computational explosion. For this purpose, more precise control strategy is required because this basic control strategy presented

in the previous section is not enough to constrain execution of all agents (EAs and MAs). The overall control strategy is presented in the following section.

In this strategy, we focus mainly on the two advantages of our system, which are a flexible combination of top-down search and bottom-up execution, and asynchronous incremental execution.

5.3.1 Combinations of operations

The features and their extraction methods are organized in a network as shown in Fig. 3.6. In this network, features can be extracted along the paths that connect nodes. An EA[1] is assigned to each node of the network, since one or more EAs are required for each feature. MAs are assigned to sets of features. This means that they construct a network of agents which is analogous to the feature extraction network used during execution (on request). The agents interact through messages whose contents are mainly hypothesized and real features. Thus, agents can interact in quite a simple way by sending hypothesized features, returning extracted features and so on.

Moreover, this system can focus on multiple targets and can employ multiple methods for their extraction as well as multiple types of operations (such as simple extraction, *re-extraction*, etc.) in parallel. An EA assigned to a node in the feature network is potentially able to carry out the above operations independent. Each EA requests features from other agents, executes feature extraction, and requests features again after having judged the extraction result. This can be considered as a kind of parallel graph search in which any node can be expanded or re-expanded (which corresponds to the trial of another OR-arc) independently.

Under this organization, the agents select their strategies as mentioned above. In this selection, the agents can potentially choose their strategies independently of the other agents. For example, an EA in charge of the feature f_1^h executes initial extraction while another EA executes *completion extraction*

[1]When spatial division is considered, plural EAs are created for one node

for f_2^h. Another example is the case of two parts (f_{part1}^h, f_{part2}^h) of a common parent (f_{whole}^h),whose extraction takes place independently of each other. Their operations would be related to each other only if there existed the ancestor-descendant relation in the feature extraction network between them.

The straightforward implementation, however, in which every agent can carry out any operation causes computational explosion in many cases. Obviously, a better global control strategy is required to make these agents work consistently and avoid computational explosion. For this problem, there are no general solutions reported, especially for concurrent objects or multi agent environments. We propose several constraints for choosing strategies (which will be described in Section 5.3.3).

5.3.2 Incremental execution

One advantage of using the concurrent object model (multi agent system) is not only the parallel execution of unrelated agents, but also data flow execution.

First, consider the operation of extracting structural features by combining component features. As the system can utilize hypothesized features instead of component features in its search when component features are not available, extraction of features across agents can be imagined as follows. Let the symbol $\otimes$ express "combination" similar to the direct product.

1. Component feature candidate f_1^h can be denoted as follows:

$$\begin{aligned}(candidates\ for\ f_1^h) &= (extracted\ features) + (hypothesized\ features)\\ F_1 &= F_1^r + F_1^h\\ &= \{f_{11}^r, f_{12}^r, \cdots, f_{1n}^r, f_{11}^{h'}, f_{12}^{h'}, \cdots\}\end{aligned}$$

2. The combinations of component feature candidates can be written as follows on the condition that $R(F_i, F_j)$ is already extracted:

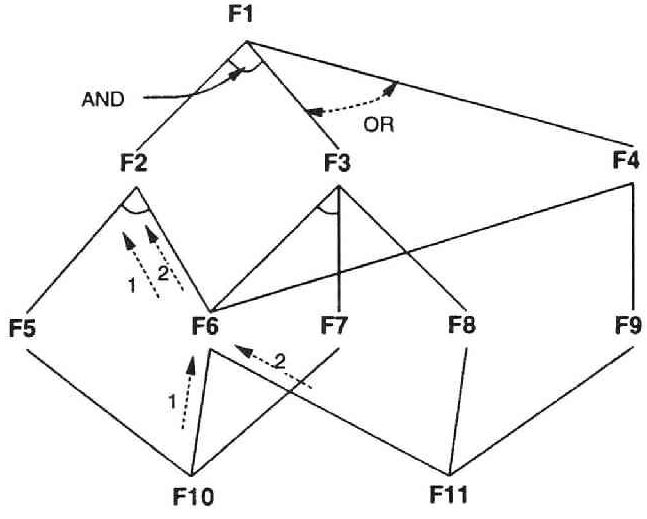


Figure 5.2: Example of Incremental Extraction

$$\begin{aligned} F_0 \Leftarrow\ & F_1 \otimes \cdots \otimes F_n \\ & = (F_1^r + F_1^h) \otimes \cdots \otimes (F_n^r + F_n^h) \\ & = \{f_{11}^r, ..., f_{11}^{h'}, ..\} \otimes \cdots \otimes \{f_{n1}^r, ..., f_{n1}^{h'}, ..\} \end{aligned}$$

where F means a set of candidate features for f^h (F^r is the set of extracted features, F^h is the set of hypothesized features). Relations also can be denoted by R and r in the same way. f_0 is composed of f_1 to f_n and satisfies the constraints $r(f_i, f_j)$'s.

The hypothesized features $f^{h'}$ are generated during the search. Therefore, execution can be carried out even if nothing is extracted. The consequence of this is that the search can occur whenever new candidates for any component features become available. Suppose that f_i (that is $\triangle F_i^r$) was extracted, then the new increment of f_0 (denoted $\triangle F_0^r$) can be calculated as follows:

$$F_0^{r'} \Leftarrow F_1 \otimes \cdots \otimes (F_i \cup \triangle F_i^r) \otimes \cdots \otimes F_n$$

$$\triangle F_0 \Leftarrow F_1 \otimes \cdots \otimes \triangle F_i^r \otimes \cdots \otimes F_n$$

This can be calculated economically by reordering the search and starting the search from f_i:

$$\triangle F_0 \Leftarrow \triangle F_i^r \otimes F_1 \otimes \cdots \otimes F_n$$

These mechanisms realize asynchronous incremental extraction of features. Consider the example shown in Fig. 5.2. The component features of F6 are sent from F10 and F11 asynchronously. Allowing for incremental execution, feature F6 and all the features extracted from F6 are extracted incrementally.

5.3.3 Overall control strategy

There are several factors one should consider when making a decision about the style of the overall control strategy of the system.

- The amount of parallel execution will increase,when the agents assigned to lower level features are allowed to do *re-extraction* (try another path) unrestrictedly.
- However, allowing unrestricted re-extraction also causes a large amount of unnecessary extraction operations to occur,lower level re-extraction might occur which would never take place, if related higher level features were extracted earlier. (Often the area for the extraction operation or even the set of target features to be extracted are constrained by the information provided by higher level features.)

Considering these factors, we propose an overall control strategy as shown in Fig. 5.3 and having the following properties:

- The feature that is initially requested (for the first time) as a component of an upper level feature is to be extracted immediately on request. It can be decomposed into requests of lower level features (component features, etc.).

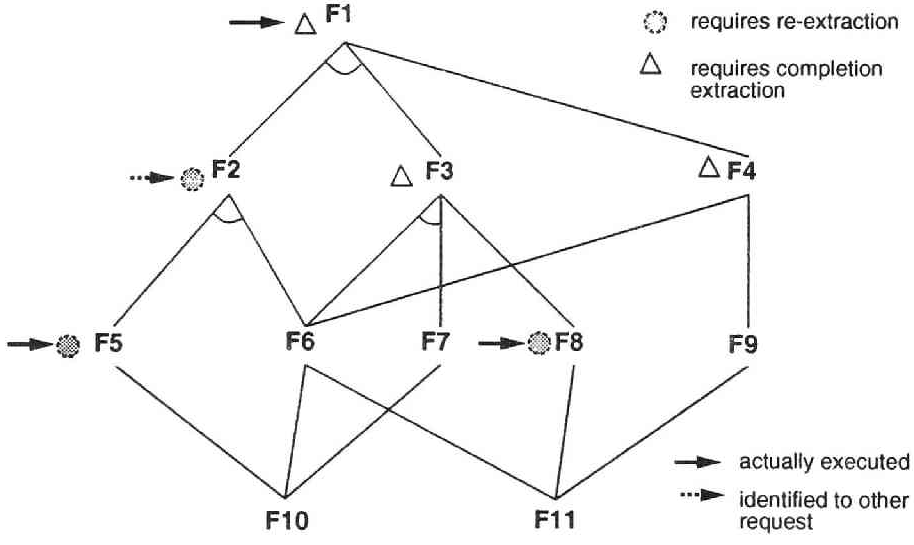


Figure 5.3: Overall Control Strategy

- The EA that did not receive enough lower features can request lower features again (*re-extraction*) only, if it is at the lowest level among all EAs that require *re-extraction* of lower level features.

- *Completion extraction* of a feature can be requested only, if the feature is at the highest level among all features that require *completion extraction*.

The point is that this strategy increases the rate of parallel execution by allowing the *re-extraction* of features that have not been sufficiently extracted yet, while avoiding unnecessary extraction operations by restricting completion extractions to higher level features.

In this strategy, we do not require an additional mechanism for the control of *re-extraction* operations, because *re-extraction* requests (hypothesized features) from EAs are not distinguished from lower level *re-extraction* requests. They are merged into one request (hypothesized feature) by MAs. For the example of Fig. 5.3, whether the request for F10 is a regular request from F5 or a re-extraction request from F2 via F5.

For the control of *completion extraction*, however, an additional mechanism is required, because the EA has to know whether it is at the highest level or not. This is realized by requesting a completion extraction request from the highest level EA. If that EA has no feature that requires completion, it sends a dummy request for completion extraction. The EAs, which receive a dummy request, can carry out completion extraction if necessary, or send their own dummy requests to lower level features. In this way, if one EA catches a dummy request it can start *completion extraction*, and none of the other EAs at a lower level than this EA will ever start completion extraction.

However, even this control strategy sometimes causes computational explosion, and further investigations are required for finding a better strategy.

Chapter 6

Experimental Result

In the preceding chapters, the PAFE system was described in detail. In this chapter, several experiments are presented to verify the efficiency of feature extraction performed by PAFE.

6.1 Experiments

This system was implemented on a Symbolics machine in Common Lisp. Most of the low level image processing is performed on SUN workstations when requested through RPC (remote procedure call) by the Symbolics machine.

The concurrent object models have been written in FLAVOR (an object oriented language based on Common Lisp). Each object has two processes for receiving messages and executing its own operations. Scheduling is left to the scheduler of the Symbolics machine.

The images to be recognized are images of a set of blocks that are mostly composed of straight lines, the images of a set of tools that are mostly composed of curves, and the images of a set of aerial photographs that are mostly composed of regional features. The definition of features were defined as described in Section 3.5.

6.1.1 Experiments for testing the basic operations

First, simple experiments not involving complicated control mechanisms are shown below. These demonstrate the abilities of feature definition in PAFE and the flexibility of the extraction process due to utilizing the feature extraction network.

Indoor Objects

The results of feature extraction defined in Section 3.5, are shown in Fig. 6.1 and Fig. 6.2 respectively. The structures for the extracted features are shown as well.

In example (1), a structured feature arrow was extracted as straight-line1 $\rightarrow$ straight-line2 $\rightarrow$ triangle (rectangle) $\rightarrow$ arrow. In example (2), the structured features shown in Fig. 3.3 were extracted. The numbers of extracted features including intermediate features are shown in Table. 6.1.

Table. 6.2 demonstrate the effectiveness of spatial division; it compares the execution time of feature extraction utilizing spatial division with extraction not utilizing spatial division (a whole plane is a single mesh). In this table, the execution time without spatial division is denoted by T_0, the execution time for a mesh, when spatially divided, is denoted by T_i, their maxima are denoted by $\max(T_i/T_0)$, the average are by $average(T_i/T_0)$, and the sum are by $\sum(T_i/T_0)$. Considering that features are not lying uniformly, the spatial division is effective except for variable-length features. (The reason was given in the previous chapter).

Aerial Photographs

The experiments in which PAFE is applied to aerial photographs are shown below. In these experiments, precise recognition is not pursued, because mechanisms of conflict resolution required for the precise recognition are not introduced into PAFE. The characteristic areas defined in Section. ?? are extracted as shown in Fig. 6.3 and Fig. 6.5.

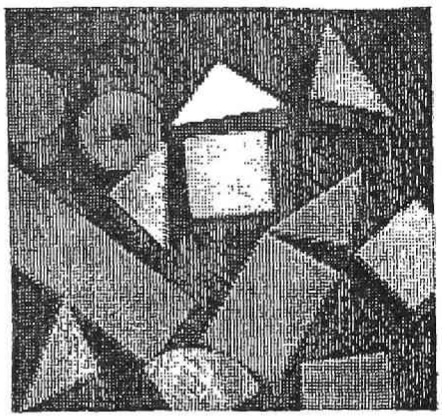
Table 6.1: Numbers of extracted features

Feature	Number	Feature	Number
Straight line1	52	curve-prim	15
Straight line2	5	curve	15
Corner	56	corner-curve	21
Line Sequence	68	flat-curve	11
Parallel lines	33	corner (beak)	27
Rectangle	3	hole	8
Arrow	4	u-shape	7
		wrench-handle	2
		wrench-head	1
		wrench	1
		pliers	1

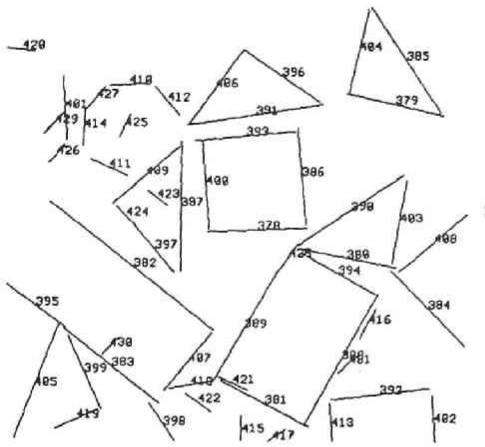
Table 6.2: Comparison of Execution Time

Feature (Relation)	Max ($\max(T_i/T_0)$)	Average ($\text{average}(T_i/T_0)$)	Total ($\sum(T_i/T_0)$)
Differentiated Image	0.28	0.26	1.06
Binary Image	0.39	0.38	1.54
Curve	0.39	0.25	1.0
Parallel (Relation)	0.14	0.12	1.93
Parallel lines	0.15	0.07	1.12
Rectangle	0.17	0.07	1.11
Triangle	0.14	0.05	0.81
Line sequence	0.75	0.22	3.52

The plane for Upper group is divided into 2×2 Meshes.
One for lower is divided into 4×4 Meshes.



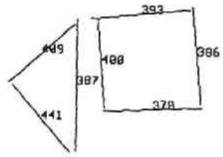
(a) Original Image



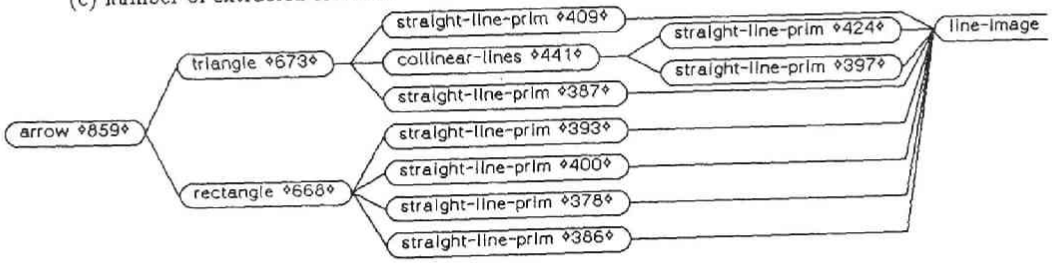
(b) Extracted straight-line-prim

Feature	Number
Straight line1	52
Straight line2	5
Corner	56
Line Sequence	68
Parallel lines	33
Rectangle	3
Arrow	4

(c) number of extracted features

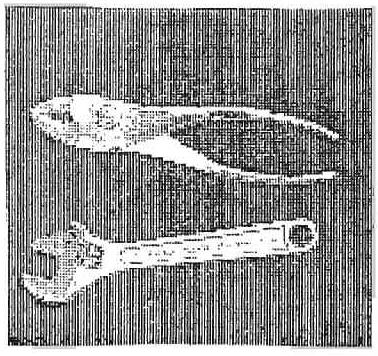


(d) extracted arrow

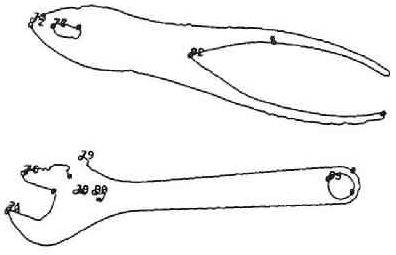


(e) feature network for extracted arrow

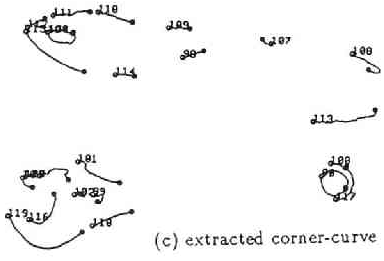
Figure 6.1: Experimental result for blocks



(a) Original Image



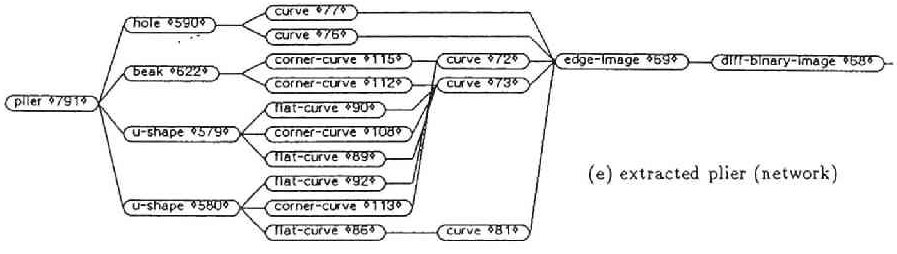
(b) Extracted curve-prim



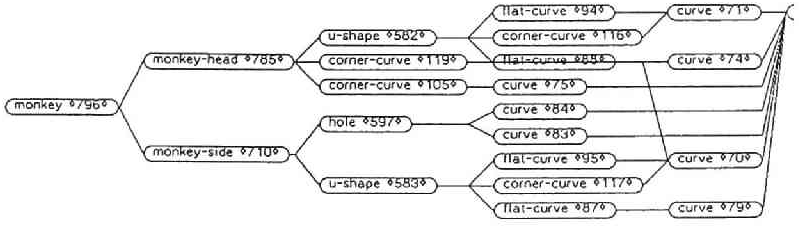
(c) extracted corner-curve

Feature	Number
curve-prim	15
curve	15
corner-curve	21
flat-curve	11
corner (beak)	27
hole	8
u-shape	7
wrench-handle	2
wrench-head	1
wrench	1
plier	1

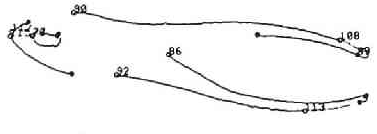
(d) number of extracted features



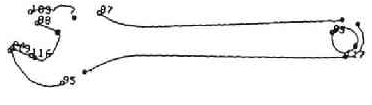
(e) extracted plier (network)



(f) extracted monkey-wrench (network)

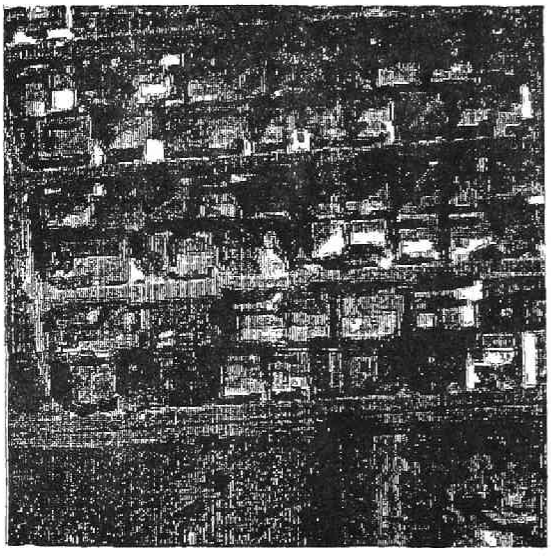


(e) extracted plier (curves)

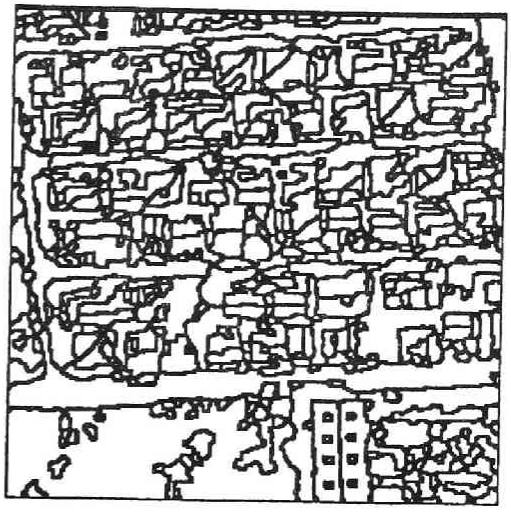


(f) extracted monkey-wrench (curves)

Figure 6.2: Experimental result for tools



(a) Original Image

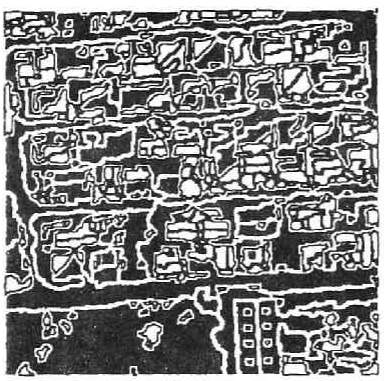


(b) Segmentation Result (Label Image)

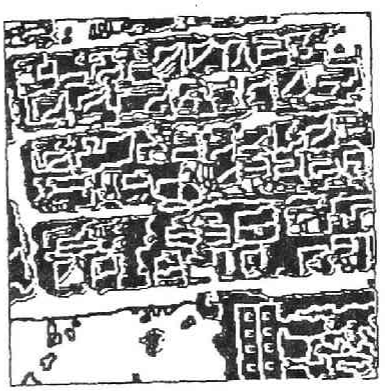
```
#<FI REGION ◇40◇>
  TYPE-OF:  REGION
  PART-OF:  NIL
  IDENTIFIER:  40
  COMPONENT-FEATURES:  NIL
  COMPONENT-RELATIONS:  NIL
  ATTRIBUTE:  (CENTROID (219.2115 25.482374)
               FIT    0.5212766
               DIRECTION 1.570796326794896 6d0
               ECCENTRICITY 1.0681819
               CONTOUR (#<ART-Q-247 665323025> #<ART-Q-247 665323415>)
               AREA 1078
               COLOR (R-MEAN 153.71057 G-MEAN 170.82468 B-MEAN 165.67255)
               REGION-IMAGE #<ART-Q-256-256 664436225>
               REGION-ID 12
               STARTING-POINT (208 0)
               MBR (#<ART-Q-4 665323056> #<ART-Q-4 665323057>)
  RELATIONS-TO-OTHER-FEATURES:  NIL
  POSITION:  (:POSITION (219.2115 25.482374) :AREA (219.2115 25.482374))
```

(c) An example for a structure of an extracted region

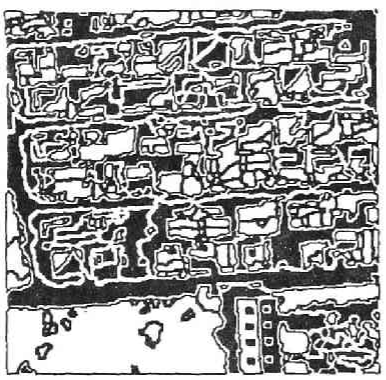
Figure 6.3: Experiment for Aerial Photographs (1)



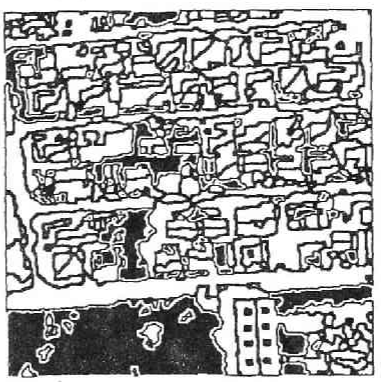
(a) Large Region



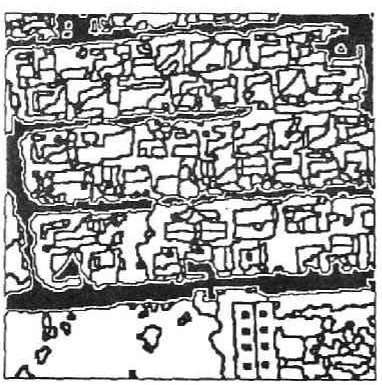
(b) Small Region



(c) Elongated Region



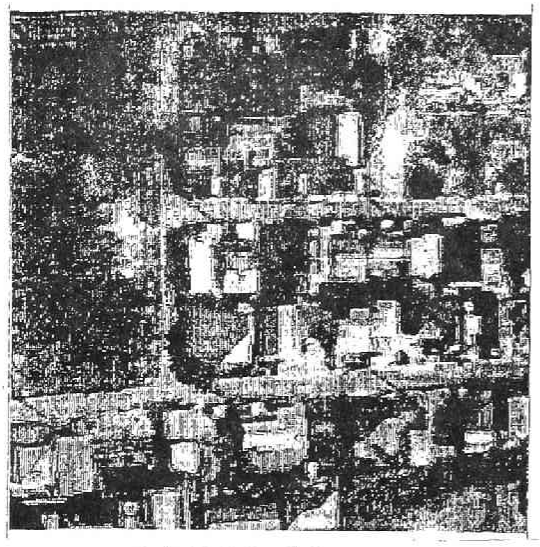
(d) Plant Region



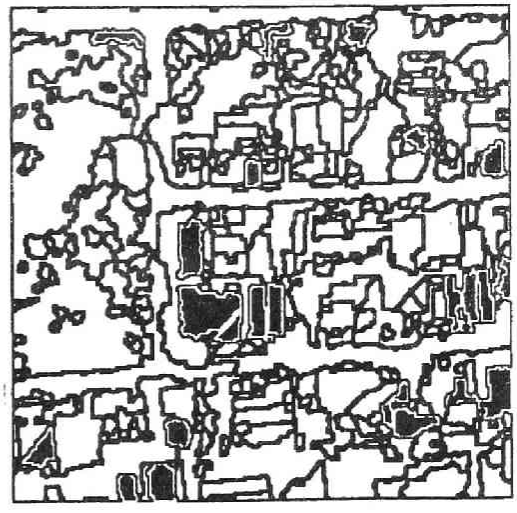
(e) Road Region

(f) House Region

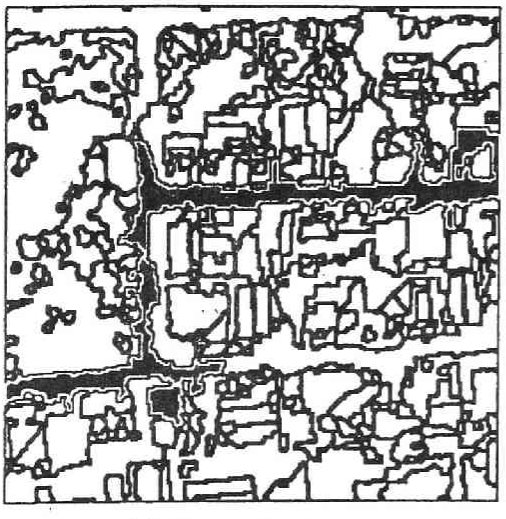
Figure 6.4: Experiment for Aerial Photographs (2)



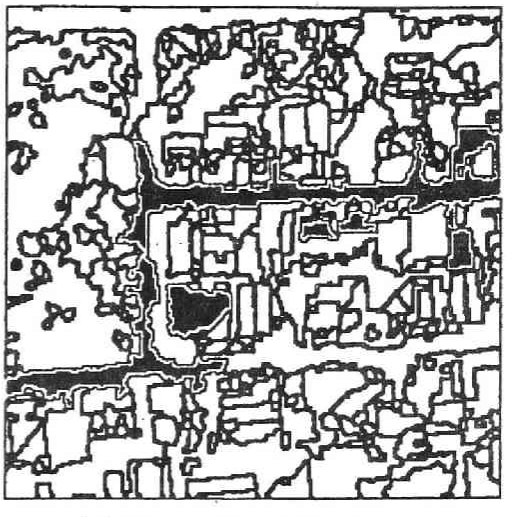
(a) Original Image



(b) House Region



(c) Road Region



(d) House-Road Region

Figure 6.5: Experiment for Aerial Photographs (3)

6.1.2 Experiments for testing the control mechanism

The experiment for the verification of the control mechanism is shown below. In this experiment, PAFE was applied to the tools defined above. An example of execution steps in the concurrent environment is shown in Fig. 6.6, Fig. 6.8.

The extraction process proceeded as follows (letters in parentheses refer to sub-images of Fig. 6.6 to Fig. 6.8).

1. The 'U-shape' feature was specified as a target by the user.
2. Several u-shapes were extracted by the initial extraction operation (c). (Features are presented by rectangle to distinguish them from each other)
3. The existence of upper level features (Handle, Head, Handle2) were suggested by them, and they were requested for extraction (d).
4. In this extraction Map, some handles were extracted (e).
5. 'Wrench' was suggested from 'Handle', and extraction was performed. Four candidates were extracted (f).
6. Completion extractions for the above wrenches were requested, and the extraction of 'Head' was performed (g).
7. As the results are not satisfactory, completion extraction operations for 'Head' are performed, and new heads are extracted as shown in (h).
8. This resulted in the extraction of a new feature of 'Wrench' whose extraction is completed with the results (i) and (j).
9. The extracted incomplete pliers are shown in (k).

An intermediate state of extraction execution is shown in Fig. 6.9, in which several processes are working simultaneously on the extraction of several features. A chart of time traces depicting the extraction of several features from

beginning to end execution is shown in Fig. 6.10, where the number of processes involved in the extraction of each feature is represented by the thickness of the trace lines. Another example of the extraction of tools form an image is shown in Fig. 6.11.

6.1.3 Discussion and Summary

Flexibility of Definition

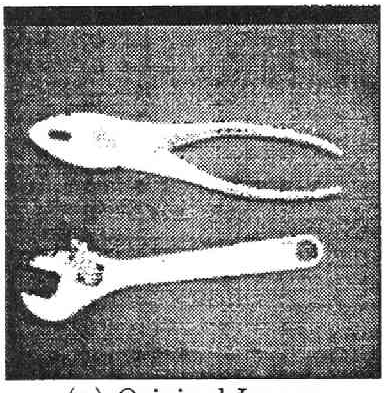
The flexibility of feature definition inherent in PAFE can be clearly observed through these the simple experiments. In the case of block-like objects which are composed of straight lines, the intermediate features are quite well expressed in our framework. Although writing feature descriptions for tools is more difficult than for blocks, they are well defined in this system, too. The same can be said of the regional features of aerial photographs.

Flexibility of Control

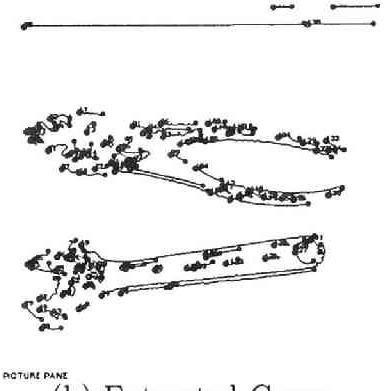
The experiments in Section 6.1.2 show the flexibility of our system. By allowing each agent to choose appropriate strategy independently, the extraction process as a whole proceeds in complicated but flexible ways. The system tries to extract all the features which are potentially extractable until there is no such feature left or until interruption by the user or other systems occurs.

Parallelism

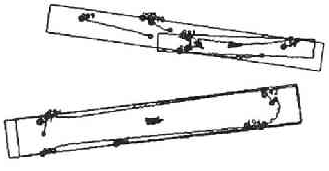
During initial extraction when the detection of a loosely specified feature is the target, the number of parallel processes executing simultaneously amounts a few, say up to ten. The reason for this is that only one target feature was specified in the experiments and no spatial parallelism was incorporated for the verification of the control mechanism. If plural targets were given to the system and spatial division was incorporated, the execution parallelism would increase.



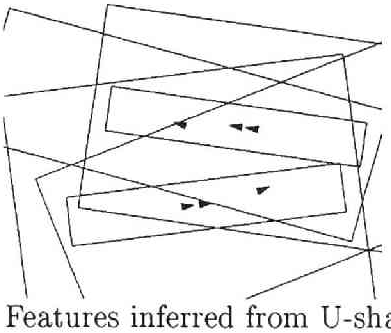
(a) Original Image



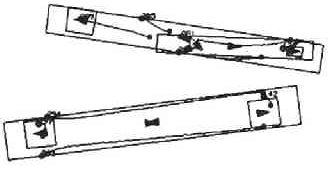
(b) Extracted Curve



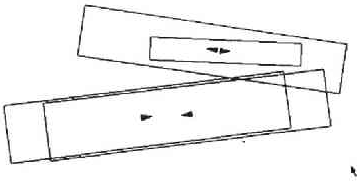
(c) Extracted U-shape



(d) Features inferred from U-shape

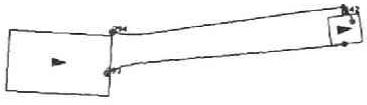


(e) Extracted Handle



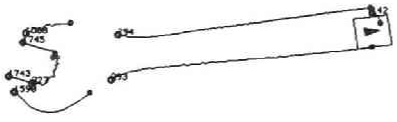
(f) Inferred Wrench

Figure 6.6: Feature Extraction Progress (1)



(g) Extracted Wrench

(h) Re-extracted Head



(i) Result for Wrench

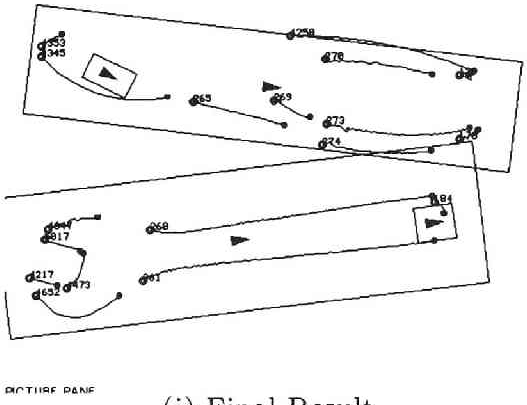
Figure 6.7: Feature Extraction Progress (2)

During the re-extraction or completion extraction phases, a number of processes were running simultaneously. Often their number was on the order of tens and sometimes more than one hundred. This occurred mainly because new targets has been created for completion extraction. Especially in the case of recognition of complicated images, the number of processes often increases beyond system capacity. (In this situation, we limit the number of processes which are running at the same time.)

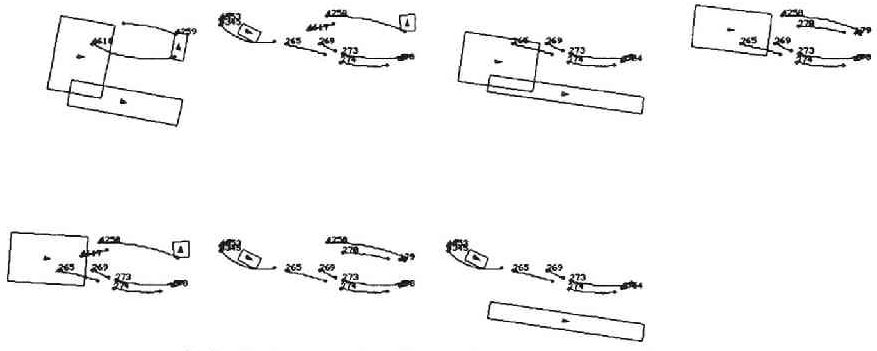
6.2 Summary

This chapter described some experimental results and the behaviors of a feature extraction system with multi agents. These experiments demonstrated the characteristics of the PAFE system as follows:

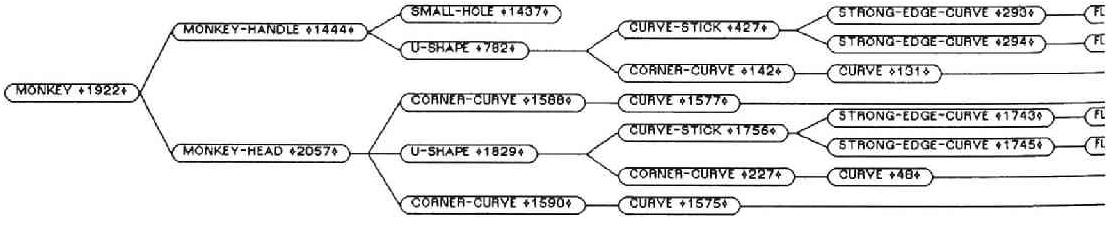
- parallelism (concurrency) of extraction execution



(j) Final Result



(k) Extracted pliers during execution



(l) Structure of Extracted Wrench

Figure 6.8: Feature Extraction Progress (3)

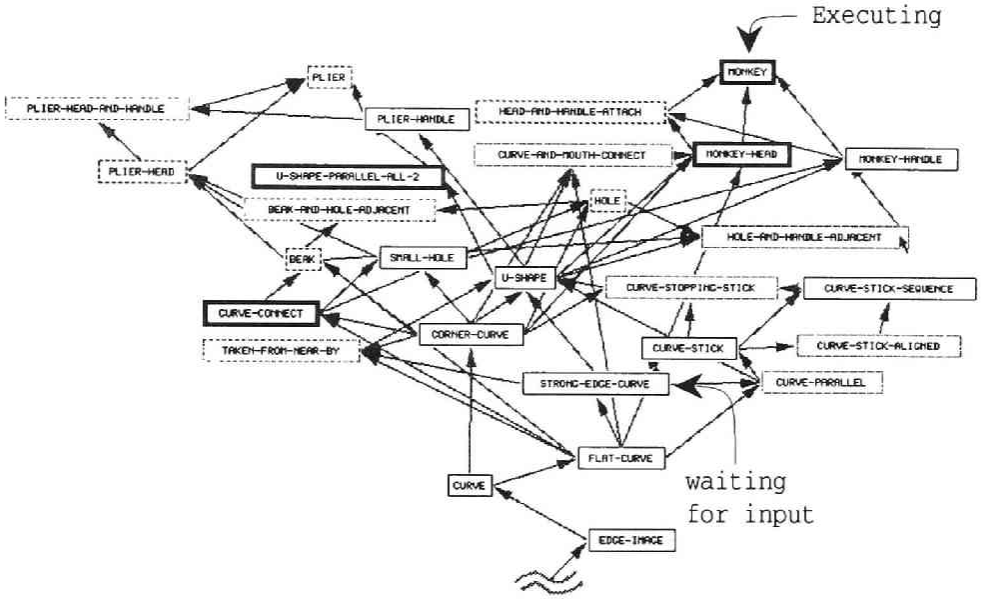


Figure 6.9: Some intermediate state of extraction execution

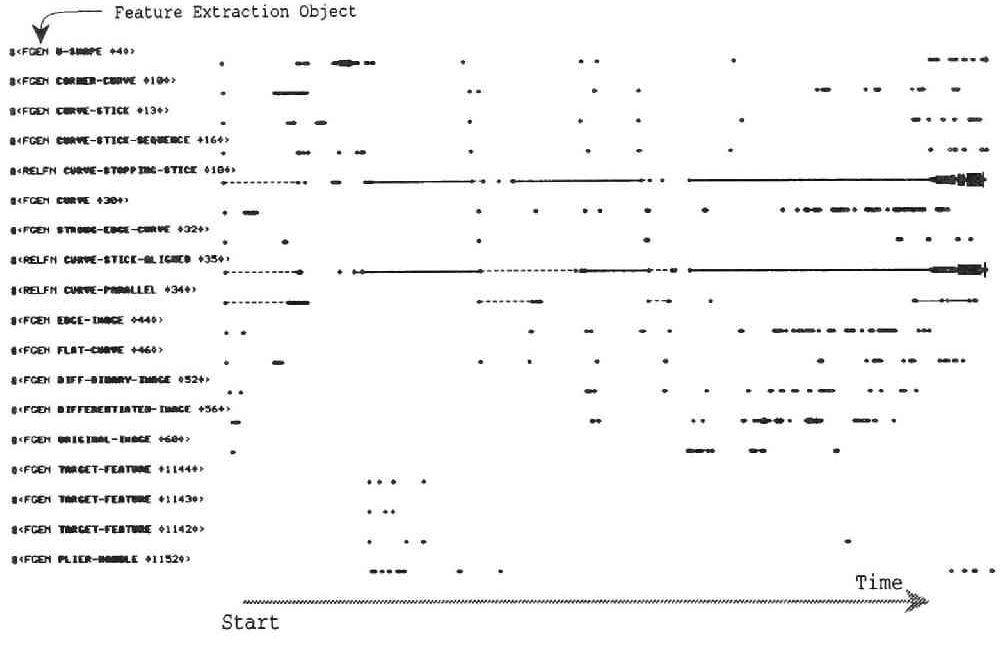
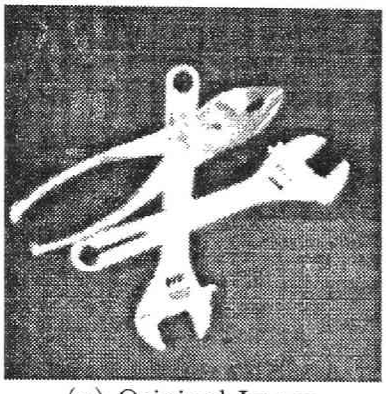
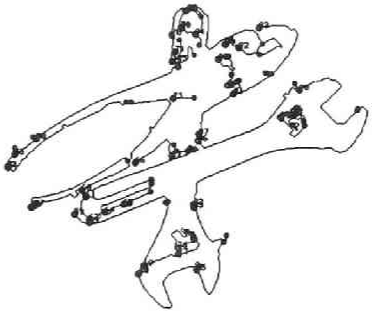


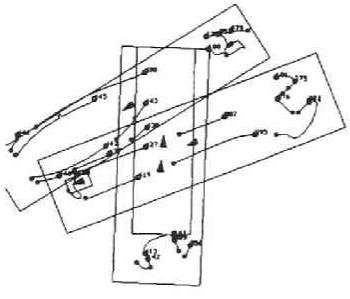
Figure 6.10: Time traces of the extraction of several features from beginning to end of execution (Thickness of Lines shows the number of executing process)



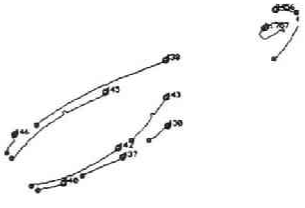
(a) Original Image



(b) Detected Curves

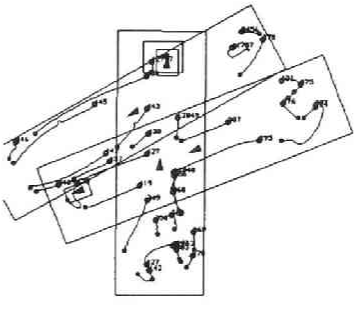


(c) Result (1)



(d) completion extraction (1)

(e) completion extraction (2)



(f) Final Result

Figure 6.11: Execution Result for Another Trial

- flexibility of execution
- efficiency of feature extraction by relying on a feature network as shown in Fig. 3.6.

Further investigations are required to establish a more efficient control strategy. At the same time, extending the utility of the feature network to encompass most of the recognition objects is an important problem to be tackled with.

Areas for the future work

There are two main areas for the extension of this research. One is the introduction competition between the agents. The other one is the integration of features which are extracted from the various kinds of sources.

1. Competition between the agents.
   In this system, only the cooperative actions of the agents were introduced. However, the competition between agents is sometimes useful for flexible and efficient feature extraction. For example, branch pruning of the parallel search could be realized by allowing only the agents with the highest score in the competition to continue the extraction.

2. The integration of features.
   The identification of extracted features is introduced into this system, needs to be supplemented by feature integration to achieve robust feature extraction.

Chapter 7

Conclusion

7.1 Summary

The objective of the research in this thesis is to construct a constructing a flexible image analysis system which utilizes local features efficiently. For this purpose, a framework for flexible feature description, a framework for performing image processing and further recognition tasks according to description of features, and a framework for flexible control of these processes were investigated in this research. Parallelism of execution is also investigated, since image analysis has a considerable degree of potential parallelism in many forms (spatial parallelism, etc.).

In this thesis, two systems for Image Processing and Recognition, which were developed in this research, were presented. First, a system for recognizing overlapping two dimensional objects (RTS) was described, which proposes a methodology for recognizing scenes where already known objects are partially occluded by each other. After that, a system (PAFE) for feature extraction was described which provides a flexible platform for defining structural features as well as primitive features, and which realize parallel execution of feature extraction processes. A method for the integration of locally residing features was discussed, too.

The purpose of this research is not to explore general features which are

applicable to every kind of applications but to construct a general platform which allows all kinds of features to be used without causing serious conflicts. Especially in both part of the research, integration of local features, which are fragments in any case, into some meaningful structure such as a part of an object, an object itself or a scene description was at the center of the discussion.

For this purpose, the following points were discussed and efficient methods were proposed in this thesis:

- An efficient method for recognizing overlapping objects.
- A framework for feature definition as the basis for realizing modularity.
  - a hierarchical definition of features
  - feature extraction with a feature network
- A framework of multiple agents allowing parallel execution of extraction process.
  - flexibility of execution control
  - parallel (concurrent) extraction execution
- A control strategy for feature extraction by the multiple agents.

The modularity of the feature definition is fitted to translate the feature extraction process into modular components, and to make parallel execution easier. The system organization with multiple agents, in which each of them perform relatively simple execution, realized a flexible control mechanism as a whole system.

Several experiments involving 2-D objects demonstrated the abilities of our system with regard to the above points:

- Scenes of blocks, tools and aerial photographs can easily be analyzed by our system.

- The flexibility of the extraction processes was demonstrated.
- A certain degree of parallelism was achieved.

7.2 Areas for future works

Further investigations are required in regard of the following areas which are promising extensions of this research. One area is the competition among agents. Another area is the integration of features which are extracted through various kinds of extraction processes from the different sources.

1. Preparation of a feature network which can satisfactorily be the base for the definitions of most objects.
   It is quite important to prepare a set of features which are useful for the extraction of many kinds of objects. This set is not necessarily a unique one. They can be several sets depending on the targets.

2. Competition among agents.
   In this system, only cooperative actions of the agents were introduced. However, competition among agents is sometimes useful for flexible and efficient feature extraction. For example, branch pruning of the parallel search could be realized by allowing only the agents with the best scores to continue the extraction.

3. Integration of features.
   In this system, the identification of extracted features is introduced. Further investigation for the integration of them is required to achieve robust feature extraction. For instance, reliability factors could be efficient base for integration of features which are extracted through various kinds of extraction processes with different reliability.

4. A more efficient control strategy.
   The control strategy for the multiple agent system needs to be extended beyond the scope reported in this thesis.

Bibliography

[AF86] N. Ayache and O. Faugeras. A new approach for the recognition and positioning of two-dimensional object. *IEEE Trans.*, Vol. PAMI–8, No. 1, 1986.

[AK87] K. Andress and A. Kak. A production system environment for integrating knowlodge with vision data. *Proc. Workshop, Spacial Reasoning and Multi-sensor Fusion*, pp. 1–12, October 1987.

[BGB79] R. Brooks, R. Greiner, and T. Binford. The acronym model-based vision system. *Proc. 6th IJCAI, Tokyo*, pp. 105–113, 1979.

[Bro91] R. Brooks. Intelligence without reason. *Artificial Intelligence*, Vol. 47, pp. 139–160, 1991.

[Dea89] B. Draper and et all. The schema system. *International Journal of Computer Vision*, Vol. 2, pp. 209–250, 1989.

[DMM85] W. Harley D. McKeown and J. McDermott. Rule-based interpretatin of aerial imagery. *IEEE Trans.*, Vol. PAMI–7, No. 5, pp. 570–585, September 1985.

[EF86] M. Eshera and K. Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE, Trans.*, Vol. PAMI–8, No. 5, pp. 604–618, September 1986.

[FP81] O. Faugeras and K. Price. Semantic description of aerial images using stochastic relaxation. *IEEE, Trans*, Vol. PAMI–3, pp. 633–642, 1981.

[Hae82] W. Haettich. Recognition of overlapping workpieces by model directed construction of object contours. *DSIA*, Vol. 1, No. 2–3, 1982.

[HR78] A. Hanson and E. Riseman. Visions: A computer system for intepreting scenes. *Computer Vision Systems, Academic Press*, pp. 303–333, 1978.

[Low87] D. Lowe. The viewpoint consistency constraint. *Computer Vision*, Vol. 1, No. 1, pp. 57–72, 1987.

[Mat88] T. Matsuyama. Expert systems for image processing – knowledge-based composition of image analysis process –. *Proc. 9th ICPR, Rome*, pp. 125–133, 1988.

[MH90] T. Matsuyama and V. Hwang. *SIGMA*. Plenum, 1990.

[Nag84] M. Nagao. Shape recognition by human-like trial and error random processes. *Proc., 2nd Int. Sympo., Robotics Reserch*, 1984.

[Nag86] M. Nagao. Toward a flexible pattern analysys method. *Proc. ICPR, Paris*, 1986.

[Nii86a] H. Nii. Blackboard systems : The blackboard model of problem solving, and the evolution of blackboard architectures. *The AI Magazine*, pp. 38–53, Summer 1986.

[Nii86b] H. Nii. Blackboard systems: Blackboard application systems, blackboard systems from a knowledge engineering perspective. *The AI Magazine*, pp. 82–106, August 1986.

[NM80] M. Nagao and T. Matsuyama. *A Structural Anarysys of Complex Aerial Photographs*. Plenum, New York, 1980.

[NN86] M. Nagao and S. Nakajima. Shape recognition system by variable size slit method – it's hardware and software –. *Proc. ICPR, Paris*, 1986.

[Oht87] Y. Ohta. *Knowledge-Based Interpretation of Outdoor Scenes.* Research Notes in Artificial Intelligence. Pitman Advanced Publishing Program, 1987.

[Oht91] Y. Ohta. Approaches to parallel computer vision. *IEICE Trans. Vol.E 74, No.2*, pp. 417–426, 1991.

[RB84] P. Rummel and W. Beutel. Workpiece recognition and inspection by a model-based scene analysis system. *Pattern Recognition*, Vol. 17, No. 1, pp. 141–148, 1984.

[Rum86] P. Rummel. Gss – a fast, model-based gray-scale sensor system for workpiece recognition. *IEEE Proc. CVPR*, 1986.

[SFO88] A. Sanfeliu, J. Font, and I. Orteu. An architecture based on hybrid system for analyzing 3d industrial scenes. *Proc. 9th ICPR, Rome*, pp. 368–370, 1988.

[SH86] T. Skordas and R. Horaud. Planning a strategy for recognizing partially occluded parts. *IEEE Proc. CVPR*, 1986.

[Tro81] H. Tropf. Analysis-by-synthesis search to interpret degraded image data. *Proc. Robot Vision and Sensory Controls.*, 1981.

[Wea87] C. Weems and et. all. Image understanding architechture. *Proc. Image Understanding Workshop, DARPA*, pp. 483–495, February 1987.

[Wey87] T. Weymouth. Incremental interference: Spatial reasoning within a blackboard architecture. *Proc. Workshop, Spacial Reasoning and Multi-sensor Fusion*, pp. 34–41, October 1987.

[YSHT87] A. Yonezawa, E. Shibayama, Y. Honda, and T. Takada. Modelling and programming in a concurrent object-oriented language abcl/1. *Object Oriented Concurrent Programming*, 1987.

List of Publications

List of Major Publications

[1] Yuichi Nakamura and Makoto Nagao, Recognition of Overlapping 2-D Objects, (in Japanese), Journal of JSAI, Vol. 3, No. 4, pp. 65–77, 1988,

[2] Yuichi Nakamura and Makoto Nagao, Recognition of Overlapping 2-D Objects by Local Feature Construction Method, Proc. 9th ICPR, 1988,

[3] Yuichi Nakamura and Makoto Nagao, A Blackboard System for Feature Extraction, (In Japanese), Journal of JSAI, Vol. 5, No. 3, pp. 354–366,

[4] Yuichi Nakamura and Makoto Nagao, Parallel Feature Extraction System with Multi Agents, Proc. 11th ICPR, 1992, (to be published),

[5] Yuichi Nakamura and Makoto Nagao, Parallel Search for Feature Extraction with Concurrent Objects, (In Japanese), Journal of JSAI, (to be published),

List of Other Publications and Oral Presentations

[1] Yuichi Nakamura and Makoto Nagao, Recognition of Overlapping Objects, (In Japanese), Proc. 34th Convention of IPSJ, pp. 1681–1682, 1987,

[2] Satoshi Ichikawa and Yuichi Nakamura, Numerical Analysis of Transmission Lines with Branches and Junctions, (In Japanese), Journal of IEICE, Vol. J67–Am, pp. 381–390, 1987,

[3] Yuichi Nakamura and Makoto Nagao, Recognition of Overlapping 2-D Objects, (In Japanese), Proc. CV49–4, IPSJ, 1987,

[4] Yuichi Nakamura and Makoto Nagao, A Blackboard System for Feature Definition and Extraction, (In Japanese), Proc. Symp. "Towards Advanced and High Speed Image Understanding", IEICE, pp. 25–30, 1989,

[5] Yuichi Nakamura and Makoto Nagao, A Blackboard System for Feature Definition and Extraction, (oral presentation), Symp. Expert Vision, IEICE, 1989,

[6] Yuichi Nakamura, The Modularity and Generality of Feature Extraction System, (oral presentation), AVIRG, February, 1991,

[7] Yuichi Nakamura and Makoto Nagao, A Feature Extraction system with Concurrent Objects, (In Japanese), Proc. PRU–91–36, pp. 17–26, 1991,

Abbreviations

JSAI Japanese Society for Artificial Intelligence

ICPR International Conference on Pattern Recognition, IEEE.

IPSJ Infomation Processing Sciety of Japan

CV Computer Vision

IEICE The Institute of Electronics, Information and Communication Engineers

PRU Pattern Recognition and Understanding

AVIRG Auditory and Visual Infomation Research Group