AN

OBJECT-ORIENTED EXPERT SYSTEM SHELL

WITH IMAGE DIAGNOSIS

A Thesis

presented to the Department of Computer Science

of The Chinese University of Hong Kong

in partial fulfilment of the requirements

for the Degree of Master of Philosophy

by

CHAN WAI KWONG SAMUEL

May 1991

325556 thesis
QA
76.64
C42

"Better is the end of a thing than its beginning; and the patient in spirit is better than the proud in spirit.

Wisdom is good with an inheritance, an advantage to those who see the sun.

For the protection of wisdom is like the protection of money; and the advantage of knowledge is that wisdom preserves the life of him who has it."

Ecclesiastes

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr.K.S.Leung, for his guidance, advice and encouragement during the research for this thesis. I also wish to thank Dr. Felix Wong, a gynaecologist, for his medical knowledge support in developing ESCUM. I would like to acknowledge the support of the Department of Computer Science of The Chinese University of Hong Kong for this research project.

I dedicate this dissertation to my wife, Medereana, who has remained a devoted and loving wife through the long and sometimes stressful days of the past two years.

ABSTRACT

There are some object-oriented expert system building tools which are in common use, however, they do not have any image diagnosis power nor have any features that support spatial information. The Object-Oriented expert system shell (OOI) is designed to incorporate the Image processing abilities. Its power and flexibility are especially designed for image recognition and diagnosis. The shell consists of several generic objects used to describe a two dimensional image.

The shell employs a robust control strategy which minimizes the amount of domain-specific control knowledge. A set of knowledge sources are designed. Each of them contains a set of specialized method or meta-knowledge, and works independently of each other in an object-oriented approach. The highly modular architecture of the system enables designer to modify any one of them easily without worrying about any unexpected side-effects.

Although the image and control objects are stored individually as set of independent objects, there are a lot of spatial and conceptual relationships between them. The image objects in OOI are organized in hierarchy with several levels of abstraction. They are stratified into several levels with which image objects in the universe could be represented. It is designed in order to capture the inheritance structure of image data. This object based representation provides a clean method for concept abstraction.

TABLE OF CONTENTS

CHAPTER 1. OVERVIEWS

1.1 Introduction

Image diagnoses or understanding is a very promising field for its potential applications in various domains such as robotics, military application, medical examination etc. Even though attempts have been made to design a general vision system [Hanson78], nevertheless, it seems to be too ambitious and not realistic. In Binfords' paper [Binford82], a good survey and critique of the state of the art in general model-based image analysis systems were presented. Vogt [Vogt86] pointed out ten major problems in developing image analysis programs and discussed their cases extensively. They are summarized as follows :

(a) **Assessment of Image Quality:**

To assess the quality of an image is the first step in image understanding. Due to the inadequate understanding of the image subject and acquisition process, it is a difficult task to measure and describe the image quality. Besides, the image quality often changes depending on the locations in the image.

(b) **Use of Operators:**

There are many different operators, or algorithms, for a specific image processing task, such as region-orientation or edge detection operators. They

are designed based on different image models. There are no formal criteria for selecting operators. Furthermore, the optimal parameters used in the operators can only be calculated instructively by trial and error methods.

(c) **Failure to Use Known Design Knowledge, Rules or Facts:**

In general, only a few types of entities can be identified, typically lines and regions. Knowledge about them is domain-independent, or called "hard-knowledge". It is difficult to merge declarative knowledge ("soft-knowledge") in a conventional approach for image understanding.

(d) **Evaluation of Analysis Result:**

The process of evaluation is very important in realizing flexible image analysis. Thus how to evaluate the analysis result and adjust parameters is an important problem in designing image analysis process.

Recognition tasks can be roughly classified into the following three main classes [Nevatia78]:

i. Identification of objects whose presence and location are approximately known: Models are exactly defined by their shape and size.

ii. Identification of objects whose presence and location in the image are unknown: The set of the models is small and well-known.

iii. Interpretation of scenes where objects can occur in any physically permissible relationships: The world of the models is large and not explicitly known.

All these classes, especially the last two, need a large amount of extra image knowledge. This knowledge is about the models of the world and the methods that must be used to perform the recognition task.

1.2 Image Understanding and Artificial Intelligence

Special-purpose vision systems have shown considerable success within their limited task domains, as shown in section 1.4. To date, however, there have been no general purpose systems that work effectively across a variety of domains. It is believed that special-purpose systems have succeeded because they are able to define, structure and apply knowledge relevant to their task domains. In fact, knowledge in vision includes domain-independent knowledge such as lighting, location of the image grabbed as well as domain-dependent object knowledge about attributes, relations and object-specific control for recognizing objects in the scenes. Object knowledge can encompass three-dimensional structure, two-dimensional appearance, geometric and co-occurrence relationships amongst objects and object parts. Control knowledge addresses the efficient extraction, organization and matching of image information to store models and the ordering of constraints to insure efficiency and consistency in the evolving interpretation.

Systems working in restricted domains perform specific recognitions. General-purpose systems, on the other hand, require generalized knowledge and inference machinery that may not be applicable to a specific task with the same efficiency and reliability. Indeed, general-purpose techniques are often unable to solve nontrivial problems. Chapman [Chapman87] stated that "Such problems are only solved by use of large, domain-specific knowledge base. It has become almost an axiom of artificial intelligence that powerful problem solving in any realistic domain requires a large amount of knowledge". This has led to the notion of "expert" or *knowledge-based systems.* The power of an expert system derives from the knowledge it possesses not from the particular formalism but inference schemas it employs [Feigenbaum77]. This assertion can be extended to all complex cognitive problems, such as natural language and automatic image understanding, in which human reasoning capabilities are required.

Recently, many pieces of work in the field of image processing stress the utility of using artificial intelligence tools to obtain enhanced performances in the scene understanding task. A great effort has been devoted to image analysis and understanding. Many algorithms have been developed for image processing and understanding. These algorithms can be classified into three levels, low, intermediate and high as shown in figure 1.1

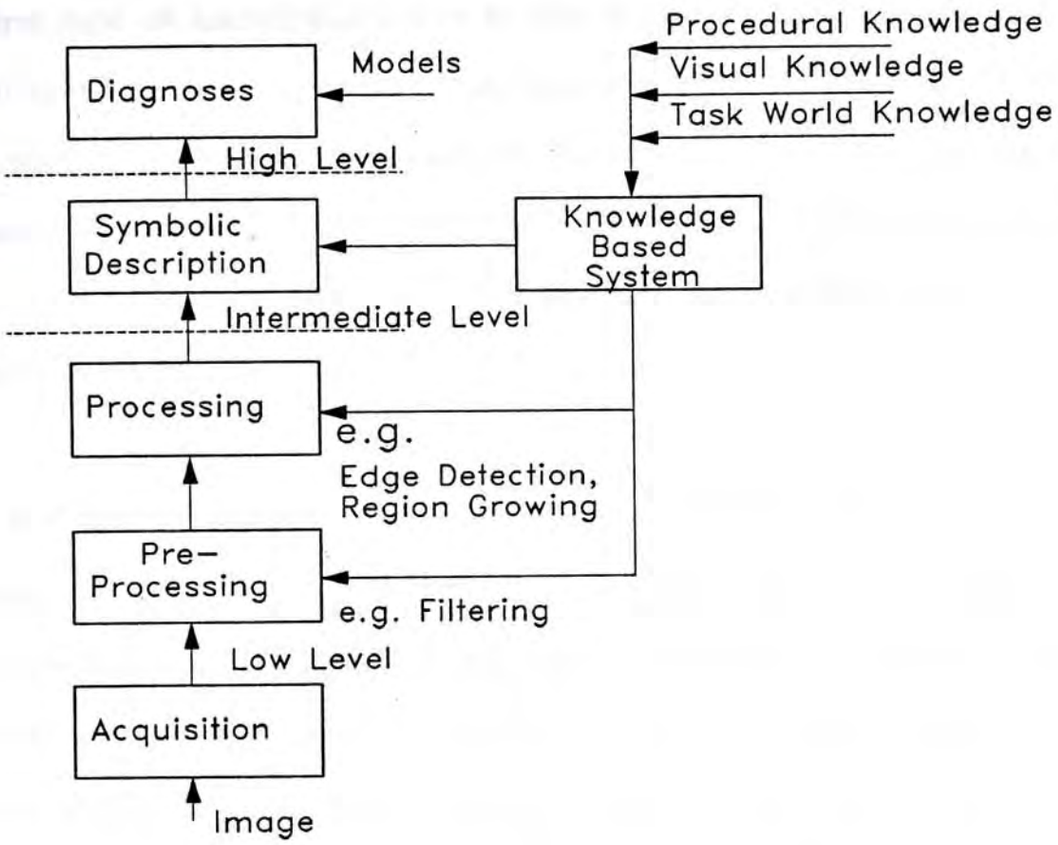

Figure 1.1 A Typical Knowledge Based Vision System

Each box in figure 1.1 corresponds to a specific processing stage, from image filtering up to the description of an image in terms of known primitives for understanding. In fact, knowledge used in each of these stages, can be classified into

(i) procedural knowledge,

(ii) visual knowledge and

(iii) task world knowledge.

The first type of knowledge refers to the determination of algorithms and operators, the selection and setting of related parameters, and the combination of different algorithms. The second type of knowledge concerns physical technique such as for generating an image, or handling different orientations in object surfaces and illumination conditions. And the third type is the high level knowledge about the image domain being considered.

1.3 Object-Oriented Programming and Artificial Intelligence

In recent years, the computer science community has shown increasing interest in the object-oriented programming (OOP) paradigm. Many researchers have pointed out OOP's software engineering benefits, such as ease of reuse, modularity, and extensibility [Cox86]. The artificial intelligence community has also shown interest [Stefik86], as evidenced by many object-oriented extensions to conventional programming languages such as C, Pascal and Lisp.

OOP techniques have been used in many practical AI systems, including system development environments and domain-specific knowledge-based applications. The experience of building these systems has revealed a number of inherent benefits of the object-oriented approach. For example, objects encapsulate both data and processing capabilities in a single entry, facilitating the representation of both knowledge and the entities of the domain being modelled. Since objects can be viewed as computer-based, executable instances of corresponding entities in the problem domain, the object-oriented approach focus on the constructions of computer equivalent of these entities with closely-mapped abstracted structures and capabilities. The constructed systems can

then be inhabited by the same entities as those in the real problem domain. This feature has been exploited in intelligent simulation research [Myler88]. Moreover, these ideas apply equally well to conceptual entities, such as goals and rules in an expert reasoning system, as to concrete objects.

OOP offers the advantages of modularity, which eases the task of an AI system developer. Modularity combines with the additional object-oriented features of data hiding and polymorphism to allow seamless integrations of heterogeneous knowledge structures and inference mechanisms. Although different objects might employ disparate representations internally or behave differently, they can all share an identical set of messages or protocol. On the other hand, although different types of assertions could have different ways of responding to messages, all are externally represented and accessed in a uniform way.

OOP also facilitates other strategies for the constructions of intelligent systems. For example, one approach asserts that a knowledge-based task can be accomplished by decomposing it into several agents, each containing and representing a chunk of complete knowledge needed to accomplish the overall task. By applying the object-oriented metaphor of intelligent communicating agents, the computational system can be viewed in terms of individual experts that accomplish goal via interaction. In fact, these techniques have been fully exploited in the project and are discussed in the following chapters.

1.4 Related Works

Starting from the late 70's, there is growing attention to meet image processing needs using the tools offered by artificial intelligence methodologies.

Nagao and Matsuyama [Nagao80] have presented a system for the segmentation and classification of high resolution colour aerial photographs. In this system, the extraction of complex geographical and cultural structures (such as roads, rivers, forests, residential and agricultural areas) is based on a variant of a production system in which the rules are actually a complex visual subsystem. They use multi-spectral, shape and adjacency properties to explicitly characterize each goal region in a knowledge based description. It was arguably the first to achieve a significant measure of success in a fairly complex natural scene domain.

The ACRONYM system [Brooks84], developed at the Stanford Artificial Intelligence Lab, is an example of a model-based system which incorporates a detailed 3-D geometric description of the object target of the recognition process. The identification strategy generates hypotheses based on edges extracted from gray-levels image, then verifies the consistency of the positions of the edges and vertices with the geometrical constraints obtained from the models. So far results have been reported on the recognition of airport photographic images.

Another interpretation system, VISIONS, has been proposed by Hanson and Riseman [Hanson78]. In this system, the knowledge is represented in a hierarchical structure with layers, such as objects, volumes, surfaces and regions. At each level of the hierarchy a hypothesis-and-test paradigm is used to construct a scene description

from the image data.

McKeown et al. have proposed the SPAM [Mckeown85], a knowledge-based system of over 500 rules, organized into five processing phases, for the interpretation of airport scenes from aerial photos and map data base. In this system, existing maps and domain specific knowledge are used for the coordination and the control of the segmentation process: objects are hypothesized and verified via consistency rules starting from fragments generated in the low level process.

In all the works reported above, a specific domain knowledge is used in the high level step in order to generate hypotheses and to drive the low-level processing, but essentially no explicit knowledge-based low level tools are employed. A totally different approach is offered by Nazif and Levine [Nazif84]. In their works for low level image segmentations, an expert system is proposed. It is based only on the general knowledge of the laws on perceptual grouping.

1.5 Discussions and Outlines

The general approach to vision problems discussed in section 1.2 exhibits several drawbacks, the most important of which are :

(i) lack of flexibility ; and

(ii) lack of clarity and modular representation of knowledge.

Both drawbacks are due to the fact that knowledge is embedded in code. Therefore application of these systems to tackle other environments different from that

of their original designs would require complete redesigns and implementations. A very important aspect to be pointed out is that several low level algorithms are based on domain-independent knowledge. For example, image filtering, smoothing or edge detection, can be designed as tools which can be called by an independent high level module. These algorithms can be regarded as "general" although their parameter setting depends on particular applications.

In order to overcome the drawbacks, one has to understand what kind of knowledge is useful for the vision problem resolution. Treating the image understanding problem as a perception problem, it is possible to identify two main knowledge sources. The first one is from the perceptual grouping laws of visible entities; the other derives directly from explicit descriptions of the objects to be recognized in the image. Normally these two types of knowledge are used in a hierarchical fashion with emphasis on one source at the expense of the other and there are well separated from each other.

For these reasons, an object-oriented expert system shell for solving image related problems (OOI) is proposed, in which the prior knowledge is separated from the basic algorithms. The basic algorithms are stored in a low level vision kernel while the high level recognition knowledge is represented by rules and methods in an object-oriented form. The main tasks of OOI are :

(i) planning and use of each algorithm for preprocessing.

(ii) selecting of parameters in accordance with the prior knowledge about image.

(iii) intaking the expert rules and knowledge from users and making diagnose of image.

(iv) synthesis the final results.

The next chapter describes object-oriented software systems. An object-oriented expert system shell (OOI) is proposed in chapter 3. The control and strategies used in OOI is described in chapter 4. In order to extract meaningful information from images, some image processing algorithms are presented. They are discussed in chapter 5. In Chapter 6, all the pictorial information used in the inference are discussed. The detail of the rule structure and the model based reasoning used are revealed in chapter 7. A knowledge acquisition module is used to communicate with the knowledge engineers for the acquisition of domain knowledge. This module is discussed in chapter 8. The analysis scheme in OOI were applied to obtain meaningful segmentation and diagnosis of the medical cell images. A medical expert system, ESCUM, is built under a real life environment. Implementation details are described in chapter 9, followed by a conclusion in the final chapter.

CHAPTER 2. OBJECT-ORIENTED SOFTWARE SYSTEMS

2.1 Introduction

Object-oriented approaches to software development have received increased emphasis since the early 1980s. These new software development methods are expected to be used widely due to features such as information hiding, modularity, abstraction and localization [Booch87]. Initially, object-oriented methods were applied primarily during the implementation phase using object-oriented languages. Recently, object-oriented paradigms have been applied to earlier phases of the software development process. Numerous efforts regarding object-oriented design approaches are found in the literature [Bailn89, Booch86, Booch87, Meyer88]. More recently, object-oriented analysis techniques have been used to initiate the object-oriented software development process [Coad90]. In this chapter, a comparison on the traditional view of the software systems and the object-oriented software system is laid out.

2.2 Traditional Software Systems

Data/Procedure-oriented software systems is regarded as the traditional software system. They are composed of a collection of *data* that represents some information and a set of *procedures* that manipulates the data. Things happen in the system by invoking a procedure and giving it some data to manipulate. As an example of a software system, consider a system for managing windows that occupy rectangular areas on a display screen. The windows contain text and have titles. They can be moved around

the screen. A window-management system implemented as a data/procedure system would include data representing the location, size, text contents, and title of each window on the screen. It would also include procedures that move a window, create a window, tell whether a window overlaps another window, replace the text or title of a window, and perform other manipulations of windows on a display. To move a window, a programmer would call procedure that moves windows and pass to it the data representing the window and its new location.

A problem with the data/procedure point of view is that data and procedures are treated as if they were independent when, in fact, they are not. In a properly functioning system, appropriate choices of procedures and data must be made. In fact, the problem has been addressed by adding features such as data typing and type checking to the traditional programming systems.

2.3 Object-Oriented Software Systems

Instead of using two types of entities representing information and its manipulations independently, an object-oriented system has a single type of entity, an *object* that represents both. An object contains data and the procedures that manipulate them. Information is manipulated by sending a *message* to the object containing the information.

When an object receives a message, it uses its own procedures to manipulate its own data. The object to be manipulated is called the receiver of the message. A message has a symbolic name, called the message selector, that describes the type of

manipulation desired. The crucial feature of messages is that the selector is only a name for the desired manipulation which describes *what* the programmer wants to happen, but not *how* it should happen. The message receiver contains the description of how the actual manipulation should be performed. The programmer of an object-oriented system sends a message to invoke a manipulation, instead of calling a procedure.

Of course, procedures have names as well, and their names are used in procedure calls. However, there is only one procedure corresponding to a procedure call, so a procedure name specifies the exact procedure to be called and exactly what should happen. A message, however, may be interpreted in different ways by different receivers. So, a message does not determine exactly what should happen; the receiver of the message does.

If the earlier example of the window-management system were implemented as an object-oriented system, it would contain a set of objects representing windows. Each object would describe a window on the screen and the manipulation of the window it represents -- for example, how to move it, how to determine whether it overlaps another window, or how to display it. Each of these manipulations would correspond to a selector of a message.

The description of a single type of manipulation of an object's information is a procedure-like entity called a *method.* A method, like a procedure, is the description of a sequence of actions to be performed by a processor. However, unlike a procedure, a method cannot directly call another method. Instead, it must send a message. *The important thing is that methods cannot be separated from objects.* When a message is

香港中文大學圖書館藏書

sent, the receiver determines the method to execute based on the message selector. The set of messages that an object can respond to is called its *protocol*. The external view of an object is nothing more than its protocol while the internal view of an object is somewhat like a data/procedure system. An object has a set of variables, sometimes called its private variables, and has a set of methods that describes what to do when a message is received. The values of the private variables play the role of data and the methods play the role of procedures. Only the procedures of an object can manipulate the data of the same object. The main characteristics of an object-oriented system is summarized in the following section.

2.4 Characteristics of an Object-Oriented Systems

Object-Oriented Programming is a software development method that puts emphasis on or starts with the data structures rather than on the control flow involved in solving a problem. Object-orientedness is not a precisely defined term, and has been given a variety of interpretations by different authors [Fikes85, Stefik86, Stroustrup88]. However, the common features are as follows [Bhanu89]:

(i) **Objects**

An object is a triple (ID,S,M), where ID is an object identifier that uniquely identifies the object, state S is a set of state variables (attributes), and method M is a set of methods. Each state variable may contain its value or point to another object.

Method M contains a set of triples (s,p,m) where s is the selector, p is the parameter list, and m is the method corresponding to the selector s. The selector of a method uniquely identifies that method within the object. The parameter list and method correspond to formal parameters and procedure body in conventional languages or a set of rules in expert systems.

(ii) **Classes and Instances**

Most object-oriented systems make a distinction between the description of an object and the object itself. Many similar objects can be described by the same general description. A *class* describes a whole set of related objects. Each object described by a class is an *instance* of that class. The class describes all the similarities of its instances. Each instance contains the information that distinguishes it from the other instances. This information is stored in instance variables which are a subset of the private variables. The *values* of the instance variables are different from instance to instance. The difference in response by two different instances is a result of their different instance variables. The methods in a class use a set of names to refer to the set of instance variables. When a message is sent, the names in the invoked method refer to the instance variables of the message receiver. Some of an object's private variables are shared by all other instances of its class. These variables are called *class variable* and are part of the class.

(iii) **Inheritance**

Another mechanism used for implicit sharing in object-oriented system is called *inheritance.* An object inherits the attributes of another object, changing any attributes that distinguish the two objects. Some object-oriented systems have inheritance between all objects, but most systems provide it only between classes. A class may be modified to create its subclass. A subclass inherits everything about its superclass. The following modifications can be made to a superclass to form its subclass:

(a) adding instance variables and
(b) providing new methods and message for the subclass.

(iv) **Message Passing**

A message is a triple (r,s,v) where r is the receiver of the message, s is the selector and v is a list of parameter values. When a message (r,s,v) is sent to receiver, r, the method that corresponds to the selector s of the receiver r, is executed with v as the actual parameters of the method.

In general, objects accept messages as inputs and, if appropriate, generate other message as outputs. When an object receives a message, it checks its collection of methods to see which one corresponds to the message. If the object does not have a method to handle a specific message, it then checks with its parent to see if the parent has a method to deal with such a message. If that fails, the parent's parent is checked and so on.

(v) **Encapsulation**

Each object is an integral and autonomous entity. It has got the required resources to manifest its state and behaviour. It can be viewed as a shielded and encapsulated entity. Its data are private to itself and hidden from others. Its interaction with other objects and the outside world is through sending and receiving messages. Encapsulation restricts the fall out effects due to changes in specifications, design, etc.

(vi) **Data Abstraction**

Data abstraction is the principle which governs a program not to make any assumptions on the implementations and internal representation. In OOP, any object can be requested for providing information or service by sending a suitable message. The requester is not bothered with how the object provides the information or service, but what it asks for. The how part, i.e. the implementation or information generation, is internal to the supplying object. Thus the emphasis is on what rather than how as far as the interaction among objects is concerned. A programmer can then concentrate on the problem solving rather than the implementation details. Abstraction allows a programmer or a knowledge engineer to work at appropriate abstraction levels at various stages of development.

(vii) **Dynamic Binding**

In conventional programming languages, the operators and functions are bound to their respective operations at the time of compilation. This is called static binding. In such cases each operator has got, in general, a unique name and a unique operation. Some languages such as 'Ada' have a 'operator overloading' feature, which enables a single operator to have more than one operation.

In an object-orient approach, the binding of an operator (message) to a particular method takes place at run time. The operator (message) 'print' sent to an object invokes the method in that object at run time. This is a dynamic binding between the message and the method. Naturally the same message 'print' can be sent to different objects to activate different methods. Thus a single message can have different responses during run time, overloading the operator names. This characteristic is termed polymorphism [Pascoe86].

Almost all the systems that claim to be object-oriented have objects and support class hierarchies and inheritance. Such systems are called level 1 object-oriented systems. More sophisticated object-oriented systems have methods or procedures attached to class-objects. They allow dynamic binding and creation of instance-objects at runtime. The most sophisticated object-oriented systems also support multiple inheritance as shown in figure 2.1 :

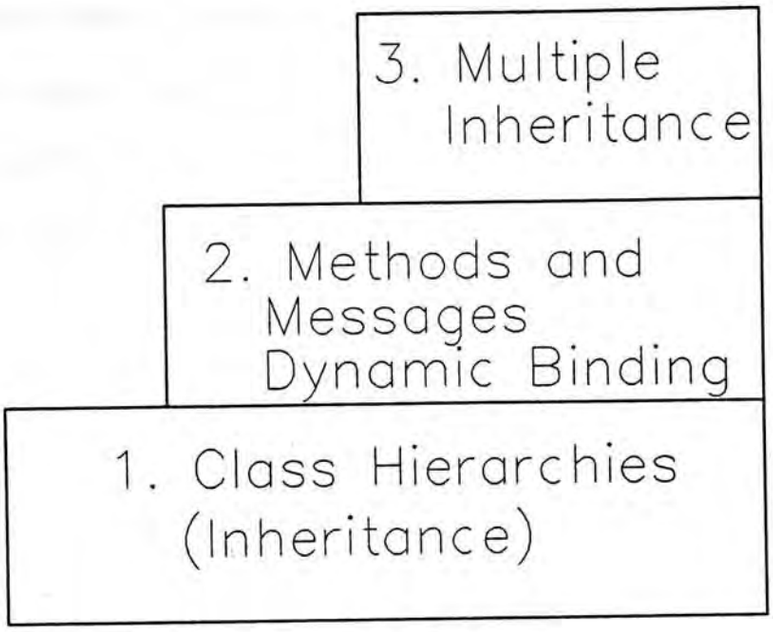

Figure 2.1 Levels of Object Orientation

2.5 Knowledge Representation in Image Recognition

In a scene, every computer program contains knowledge about the problem it is solving. One needs both a large amount of knowledge and some mechanisms for manipulating the knowledge to create solutions to new problems. This scenario agrees with the field of artificial intelligence (AI) where the concern is to "write down descriptions of the world in such a way that an intelligent machine can come to new conclusions about its environment by formally manipulating these description" [Brachman85]. Therefore, knowledge representation is a crucial aspect in all AI problems.

In the last twenty years, a broad variety of knowledge representation languages have been proposed ranging from moreorless direct derivatives of first-order logic to schemata based on cognitive psychology or applied computer science like association networks, production systems, or procedural languages. Some of them are discussed in the following sections :

2.5.1 Rule-Based System

A classic way to represent human knowledge is the use of IF/THEN rules. The satisfaction of the rule antecedents gives rise to the execution of the consequents -- some action is performed. Such production rule systems have been successfully used to model human problem-solving activity and adaptive behaviour. More recently, substantial knowledge-based systems have been constructed using this formalism, for example the R1/XCON computer configuration system, implemented in the OPS5 production rule language. Production systems exhibit useful modularity, in that rules are independent of each other, and of the rest of the system. Each rule encodes a 'chunk' of independent domain knowledge. Furthermore, simple chaining methods can be used to implement inference procedures which are like those used by humans. Production rule systems can be built which seem to model closely human problem solving in some domains.

Unfortunately, experience programmers will find out that while developing knowledge systems, it is common for the set of rules in a system to follow a cyclical pattern. It is an illusion that by simply adding more and more rules, a system can

be developed to handle more and more complex problems. Nevertheless, a system with several hundreds of rules may be developed. As the number of rules increases, so do two other factors:

(i) The difficulty of conceptualizing the system impeding development.

(ii) The mass of rules searched during the inference process, reducing processing efficiency as shown in figure 2.2

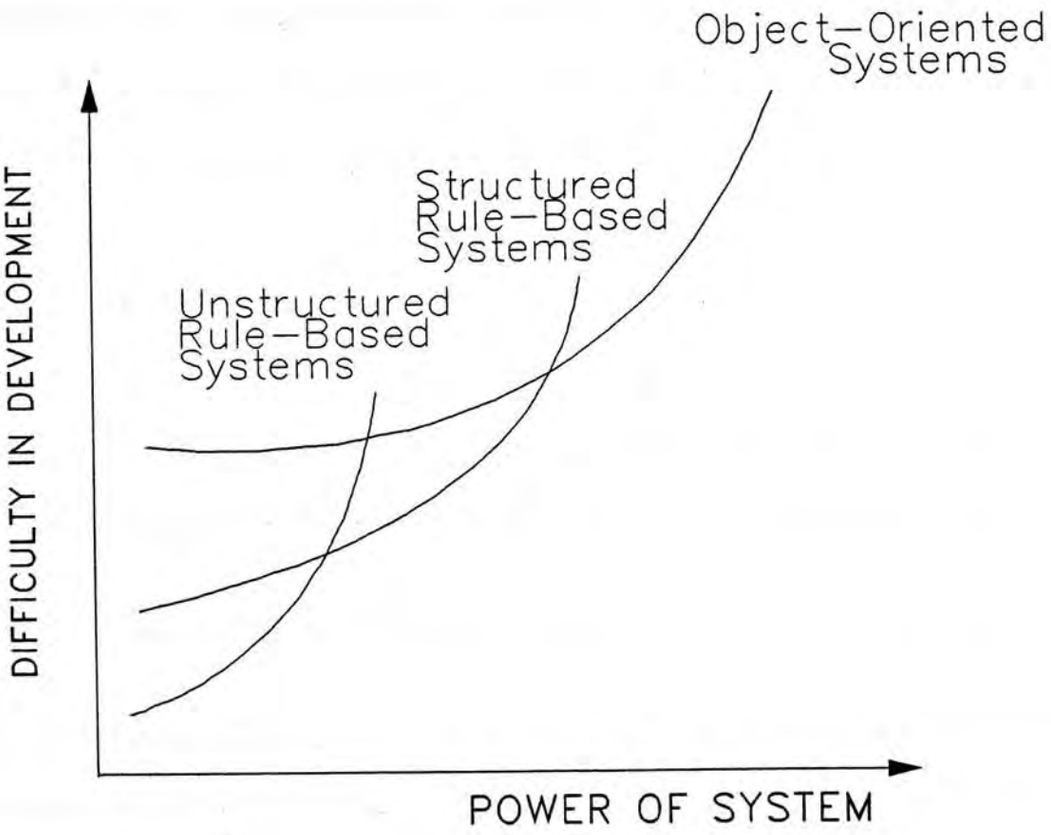

Figure 2.2 Ease of Use Versus Power in Unstructured, Structured, and Object-Oriented Knowledge Systems [Gallagher88].

Furthermore, because of the independence of the rules from each other and from the control strategy and the rule sets have no intrinsic structure, it is impossible to determine rigorously properties of the system's behaviour by static analysis. It is necessary to test the system with the data of interest to see what it will do. Since realistic production systems cannot be exhaustively tested, they cannot be used in safety-critical applications.

Like all other systems, different methods of representing their functions are required at different levels of complexity. Each "layer" or "level" of description represents another increment in the systems abstraction [Frisch82]. This leads naturally to structured objects knowledge representation.

2.5.2 Structured Objects

The use of structured objects as a means of representation stemmed from the coming together of ideas generated in a number of different lines of research. These objects have been given a wide variety of names, such as the followings:

(i) "Schemas", by the psychologist Bartlett [Bartlett32] in his work on memory.

(ii) "Frames", by Minsky [Minsky75] in the seminal paper in which he described the process of understanding images and natural languages; and subsequently in several languages such as FRL [Roberts77] and UNITS [Stefik79].

(iii) "Scripts" by Schank and Abelson [Schank77] which describe the relations between events in standard situations.

(iv) "Prototypes" or "Units" in the language KRL conceived by Bobrow and Winograd [Bobrow77], who use the work of Rosch [Rosch75] on categorisation of concepts.

(v) "Objects" in many languages, for example SMALLTALK [Goldberg83], LOOPS [Bobrow83, Bobrow85], FLAVORS [Moon80], ORBIT [Steels83], FORMES [Cointe83] and C++ [Stroustrup88].

2.5.3 Object-Oriented Knowledge Management

The "one dimensional" representation of knowledge using facts (data) and procedures gets very involved when one tries to manage more complex knowledge structures. In a number of procedural languages, such as Ada, Pascal and certain extensions to LISP, object-oriented techniques have been adopted to remedy this. This approach enables one to define objects such that facts and procedural knowledge are packaged into a single unit. Furthermore, specialized object classes can be constructed in a manner which allows the more specific objects to "inherit" knowledge from more generally defined object classes. This yields a great saving in cost with respect to the gathering and representation of information.

Object-Oriented knowledge representation have evolved from attempts to implement the "semantic network" class of models for human memory that emerged from cognitive psychology in the early 1970s [Quillian68, Shastri88]. Object-Oriented programming technologies [Stroustrup88] emphasize on the underlying concepts (objects) in an application area and the linkages between them. This is

somewhat distinct from a strictly rule-based approach in representing knowledge. For example, one might model an image or picture knowledge for classification and choosing pixels in rule-based format [Nazif84]. An alternative method, however, is to emphasize on the "structural relationships" among the relevant concepts in the image.

Object-oriented systems allow the developer to specify the hierarchical relationships among concepts (knowledge) as described in section 2.4.

Associated with each object may be rules or procedures that specify what to do to obtain the value for an attribute, or what to do when a value for an attribute is found [Fikes85]. These rules or procedure are sometimes referred to as methods. The purpose of such systems is usually to find the values for the attributes of a particular object.

2.5.4 Object-Oriented Expert System Building Tools

The development of new expert systems is changing rapidly -- in terms of both ease of construction and time required -- because of improved expert system building tools. The core of an expert system consists of a knowledge base and an accompanying inference engine that operates on the knowledge base to develop a desired solution or response. The following are surveys of some of the current object-oriented expert system building tools in common use [Gevarter87].

(i) **ART**

ART is a versatile tool that incorporates a sophisticated programming workbench. It runs on advanced computers and workstations such as those produced by Symbolics, LMI, TI, Apollo and VAX. ART's strong point is viewpoints, a technique that allows hypothetical *nonmontonic* reasoning. ART is primarily a forward-chaining system with sophisticated user-defined pattern matching; the pattern matching is based on an enhanced version of an indexing scheme derived from OPS5. Object-oriented programming is made available by attaching procedures (active values) to objects (the objects are called schemata). Applications particularly suited for ART are planning/scheduling, simulating and configuration generation.

(ii) **KEE**

KEE, which runs on advanced AI computers, is the most widely used programming environment for building sophisticated expert systems. Important aspects of KEE are its multifeature development environment and end-user interfaces, which incorporate windows, menu, and graphics. KEE also offers a variety of reasoning and analysis methods, included object-oriented programming, forward and backward chaining of rules, hypothetical reasoning, a predicate-logic language, and demons. KEE has been used for applications in diagnosis monitoring, real-time process control and planning, design and simulation.

(iii) **X-I**

System X-I, developed by Leung[Leung90], includes a flexible knowledge-acquisition module, an inference engine (backward chaining), a database interface, fuzzy retrieval facilities, and external methods interface. The expert system

shell is developed on VAX computers using a VMS operating system. The shell's object-oriented approach to knowledge representation supports data and knowledge abstraction. The approach also encourages modular designs, which improve the efficiency of knowledge acquisition and management.

(iv) **Picon**

Picon is designed as an object-oriented expert system shell for developing real-time, on-line expert systems for industrial automation and other processes that are monitored with sensors during real-time, such processes are found in some aerospace and financial application. Picon operates on LMI Lambda/Plus Lisp machine and TI Explorer. Picon supports both forward and backward chaining.

2.6 Concluding Remarks

Pictorial information, which is two-dimensional (or "spatial"), does not fit naturally into the linear first-order logic framework. The systematic use of networks for knowledge representation with inheritance property began with Quillian [Quillian68]. Many associative network formalisms were lacking in two respects : logical and heuristic adequacy [Jackson86]. Minsky's [Minsky75] frames tried to integrate a declarative and procedural systems. However, the object-oriented approach provides a more elegant paradigm for knowledge representation which allows representation of hierarchical information. Also, it provides the necessary framework for representing pictorial information storing data explicitly at various levels of abstractions. The conceptual

organization of the object-oriented expert system design is based on a hybrid of objects (as the primary scheme) and rules (as the secondary scheme). Although there are some object-oriented expert system building tools which are in common use, they do not have any image diagnosis power nor have any features that support spatial information. An object-oriented expert system shell (OOI) is designed. Its power and flexibility especially designed for image recognition and diagnosis will be discussed in following chapters.

CHAPTER 3. SYSTEM DESIGN AND ARCHITECTURE

3.1 Introduction

The problems that characterize the application of artificial intelligence techniques in image analysis may be stated as follows:

(a) The analysis is based on two different types of knowledge: the domain knowledge and the knowledge of the techniques for image processing. The interactive combination of these two types of knowledge can be used to control the process of image analysis. The control can focus at a specific diagnosis or goal to avoid deviations and errors.

(b) Another problem is the heuristics that characterize both types of knowledge. For example, the knowledge used in natural scenes for image recognition cannot substitute that of medical domains although the basic image processing techniques are common. *Experience* is more important here than in other fields.

It therefore seems justifiable to set an objective to build a knowledge-based system that try to create a link between, say, the medical knowledge base and the image processing system to acquire, store and use knowledge on image analysis methods.

In this chapter, a recognition philosophy is first discussed. A system architecture of an object-oriented expert system shell is introduced and investigations are made on

using an object-oriented approach as the basis for an image diagnosis system. The declarative part of the knowledge base is organized as objects. An object based representation provides a clear method for concept abstraction. In the following sections, an outline of all object models including image and control objects is given.

3.2 Inheritance and Recognition

Inheritance is a form of reasoning that leads an agent to infer properties of a concept based on the properties of its ancestors. For example, if a physician knows that "Cancer cells have thick vessel surrounding them", then given that "There are cancer cells in a sample of tissue", one may infer that "There are thick vessels in the sample of tissue". This kind of reasoning, often referred to as default reasoning, is commonplace. One can even argue that it is the quintessence of common sense reasoning. In general, inheritance may be defined as the process of determining properties of a concept, C, by looking up properties locally attached to C, and if such local information is not available, by looking up properties attached to concepts that lie above C in the conceptual hierarchy.

Recognition is the dual of the inheritance problem [Shastri88]. While inheritance seeks a property value of a given concept, recognition seeks a concept that has some specified property values. The recognition problem may be described as follows: "Given a description consisting of a set of properties, find a concept that best matches this description". It is noticed that the properties of concepts are not necessarily available

locally, and may have to be determined via inheritance. For this reason, recognition may be view as a very general form of pattern or concept matching. Target patterns are the set of patterns to which an input pattern is to be matched. The target patterns are organized in a hierarchy. The matching of input pattern A with a target pattern T_i involves matching properties in A with properties local to T_i and those inherited from its ancestors. In fact, inheritance and recognition are important forms of limited inference [Frisch82]. These two complementary forms of reasoning probably lie at the core of intelligent behaviour and act as precursors to more complex and specialized reasoning processing. Figure 3.1 illustrates the role of inheritance and recognition.

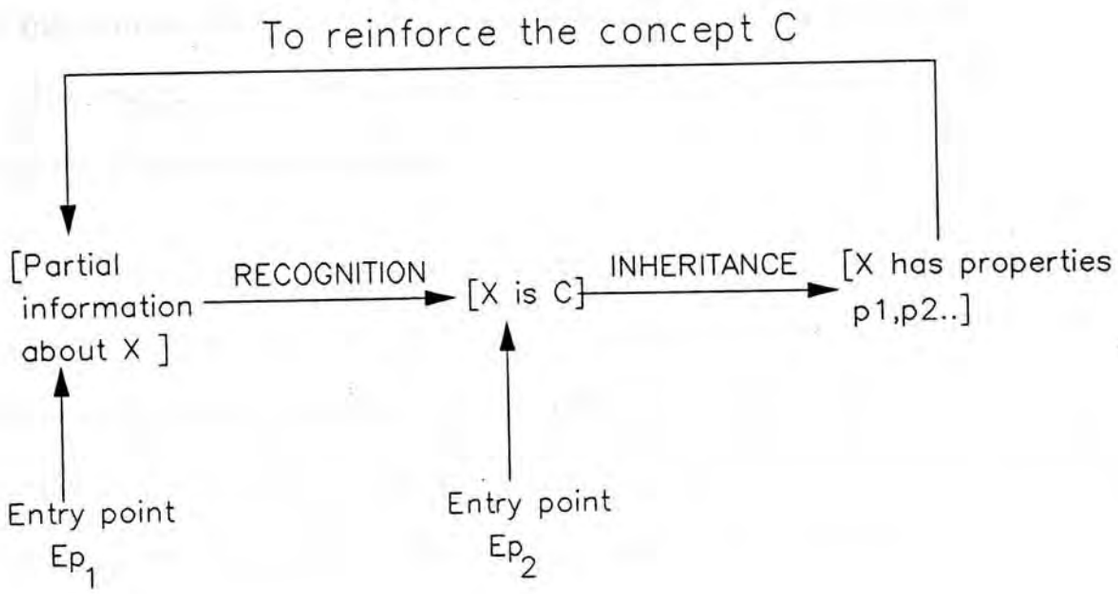

Figure 3.1. Relationship between Inheritance and Recognition.

In figure 3.1, entry point Ep_1 corresponds to recognition queries. A recognition query corresponds to a request for generalized pattern matching and is initiated whenever a mental process has a partial description of an unknown entity "X" or the class to which "X" belongs. Here "X" may refer to a physical object, the precondition part of a production rule, a frame, an object or an action etc. Entry point, Ep_2, corresponds to inheritance queries. Here an internal process for identifying the class of "X" tries to ascertain the value of some properties of "X". In addition to common sense reasoning tasks, one can identify many other cognitive subtasks that would require inheritance as an intermediate step. Besides their ubiquity, inheritance and recognition

are significant because in spite of operating with a large knowledge base, human beings perform these inferences effortlessly and extremely fast -- often in a few hundred milliseconds. This suggests that inheritance and recognition are perhaps basic and unitary components of symbolic reasoning.

In order to capture inheritance structure of image data in the above discussion, a spatial object representation is used. It is because objects can be created at various levels of spatial resolution and the information about the hierarchical links between objects can be stored along with the object themselves. Consider an image I, made up of three regions I_1, I_2 and I_3, and let each of these regions be made up of subregions as shown in figure 3.2

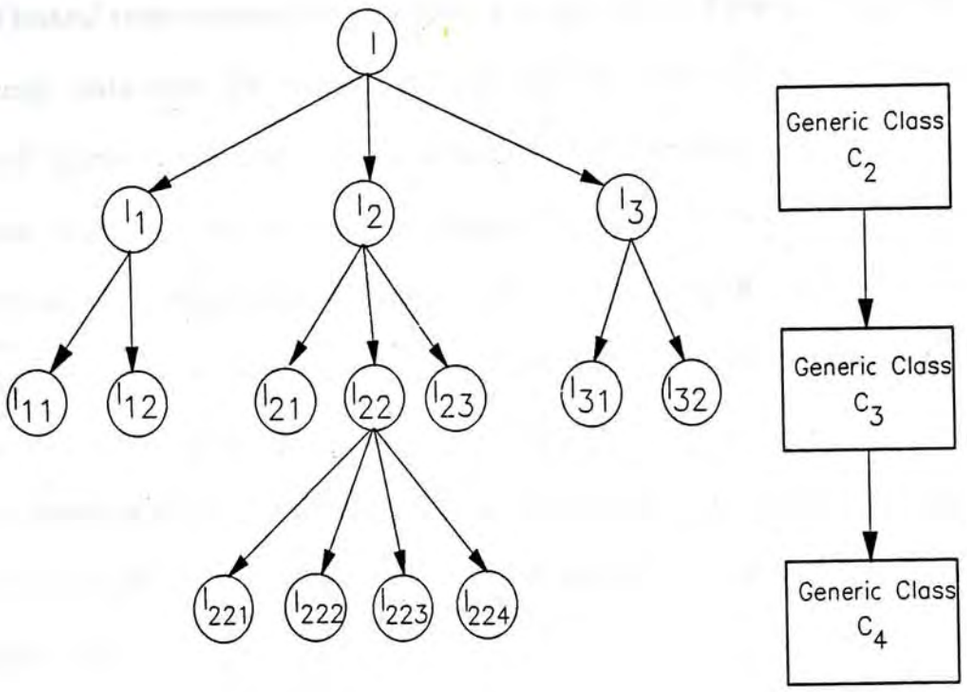

Figure 3.2 Object Hierarchy of Image, I

Each of the I's is represented as an individual object and belongs to one of a set of generic classes C_1, C_2,....., C_n, wherein there is one generic class for each level of spatial resolution. Thus I_1, I_2 and I_3 are instances of C_2. I_{11}, I_{12}, ...I_{32}, are instances of class C_3, and so on.

An object based representation provides a clean method for concept abstraction. Although the image data may be stored individually as a set of independent objects, there are a lot of spatial and conceptual relationships between various component entities. Suppose that a group of spatial objects have some features or conceptual properties in common. With an object model, it is easy to create a new subclass which has this group of objects as its instances. This new class object (subclass) would then represent abstraction of such properties. Supposing objects I_{11}, I_{21}, I_{22}, I_{23} and I_{31} share a set of common spatial properties P_1, a subclass C_{p1} having these five objects as its instances can be created. C_{p1} will then be a subclass of the generic class of C_3 as shown in figure 3.3.

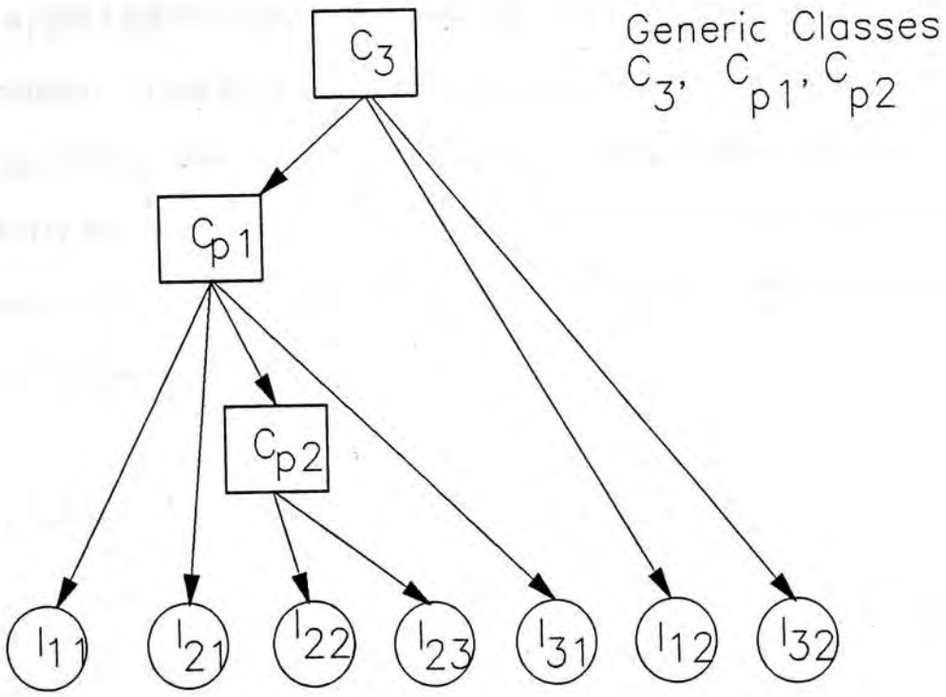

Figure 3.3 Conceptual Abstraction by Subclassing

In the design of our expert system shell for medical image understanding, called OOI, knowledge is organized in hierarchical objects and with several levels of abstraction. OOI relies on metaknowledge to guide the hierarchical reasoning processes. Hierarchical segmentation and reasoning techniques are used similar to that of a human viewer who can see very quickly which areas of an image are region interiors and which are likely to contain boundaries. Armed with this "coarse" information, the

brain can then apply higher resolution to specific areas of interest to separate regions with more precision. Hierarchical reasoning techniques [Tanimoto75] mimic this behaviour by sacrificing unimportant details and using higher levels of knowledge abstraction at early stages in order to form approximate ideas on what an image is. If the recognition process fails at some levels of abstraction, the system uses lower level where more details are available.

3.3 System Design

A primary objective in computer vision research is to construct image diagnosis systems which can analyze images by constructing interpretations based on object models. Interpretation refers to the mapping between objects (e.g. cells, cytoplasm) in the object model and image structures (e.g. regions, patches, points) in the image. During analysis, an image diagnosis system needs to perform the following two types of tasks:

- Segmentation : the task of grouping pixels together to construct image structures that can be associated with the objects in the given model.

- Interpretation : the task of constructing mappings between image structures and objects.

Segmentation is practical when sufficient knowledge is available on the image to be processed and the image structures to be computed. The knowledge base increases

as the interpretation process develops, leading to more reliable segmentation. Many systems were constructed in the past [Mckeown85, Levine78]. Most systems integrate segmentation and interpretation using one of the following types of analysis :

(1) Bottom-up analysis : The image structures are extracted from the image and are interpreted as instances of objects in the model. For example, when a large and black elliptical region is extracted, it is interpreted as a nucleus of a cell.

(2) Top-down analysis : The appearance of the object is first determined, and the associated image structures are then extracted. For example, suppose an image understanding system wants to find a cell, the image understanding system invokes the cell model and establishes the descriptions of the specific image structures to be extracted from the image.

It is generally accepted that image understanding systems should incorporate both bottom-up and top-down analyses. Some systems use only one type of analysis. For example, MSYS [Barrow76], used bottom-up analysis. The Image structures are first segmented from an image. A set of initial labels are assigned to these image structures. Then, geometric constraints between labels are used to filter out inconsistent labelling. Garvey's system [Garvey76], on the other hand, used top-down analysis. In the system, a goal is first constructed. The system then matches the goal represented as a template with the image. Other systems [Hanson78, Nagao80] incorporate both

types of analysis but use ad hoc rules to determine which type of analysis is to be used at what stage during the analysis. Such systems often require a large set of domain dependent control knowledge to direct the analysis of the image understanding system.

It is the goal of this research to develop an object-oriented expert system shell for image diagnosis, code named OOI. The shell employs a robust control strategy which minimizes the amount of domain-specific control knowledge required in specific applications.

The conceptual organization of the object-oriented expert system design is based on a hybrid of objects (as the primary scheme) and rules (as the secondary scheme). Object are a good way for representing knowledge, as already discussed in chapter 2. On the other hand, rules are a good way for representing dependencies between concepts and thus are appropriate for representing the strategies and controls within the instances. The rationale for this is argued in Johnson's [Johnson85] and Aikins' papers [Aikins83].

In the following system architecture, a general framework which enables image diagnosis systems to integrate both bottom-up and top-down analyses is discussed.

3.4 System Architecture

Figure 3.4 shows the system architecture of the object-oriented image diagnosis system (OOI) which is a expert system building tool for expert systems incorporated

with image diagnosis. If an user provides image object models to OOI, the results of the analysis will be made available to the user through the user-consultation kernel.

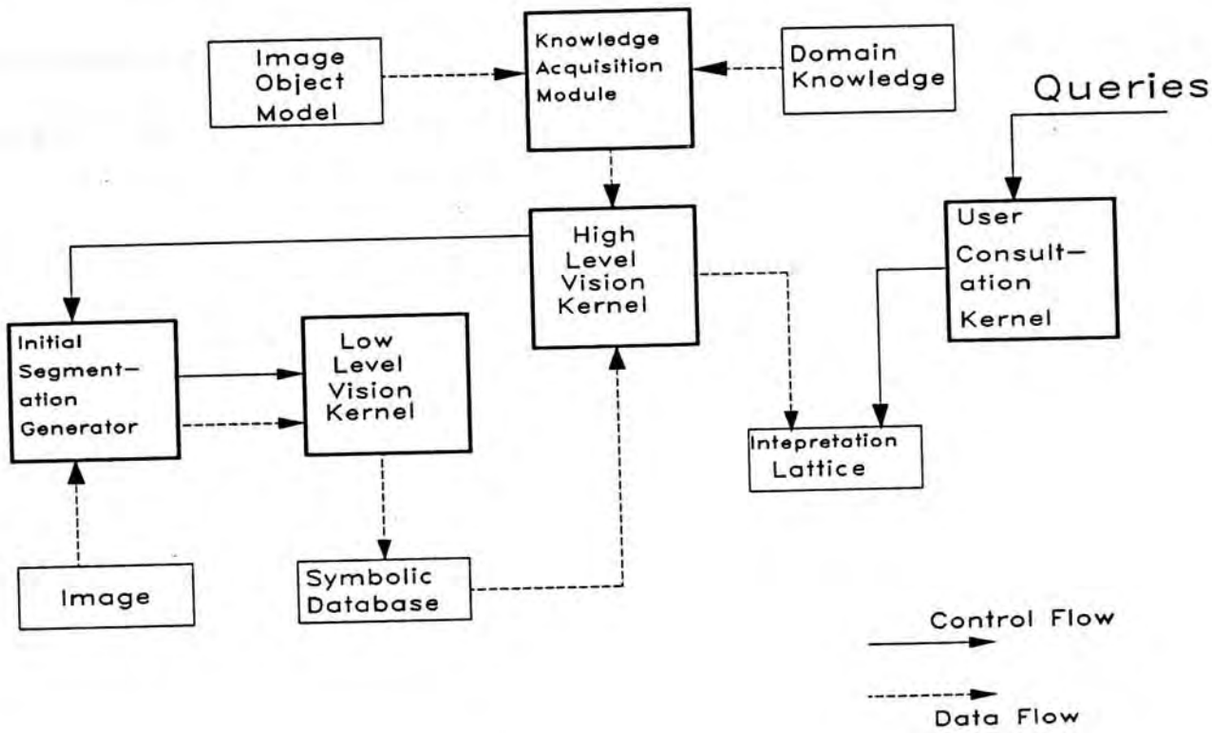

Figure 3.4 System Architecture for OOI

The image is first segmented by a general purpose low level vision kernel (LLVK). It is initiated by the initial segmentation generator (ISG). The segmentation results are recorded in the symbolic database. The high level vision kernel (HLVK) uses the image object model either to interpret image structures already extracted, in a bottom-up

sense, or to search for image structures that have not been discovered yet using a top-down approach. During the analysis, the HLVK coherently constructs an interpretation lattice from an image object model for the input physical image as shown in figure 3.5. This lattice is used by the user-consultation kernel (UCK) to answer all the queries from the user. Domain knowledge and the image object models are input from user through the knowledge acquisition module. The module is discussed in chapter 8.

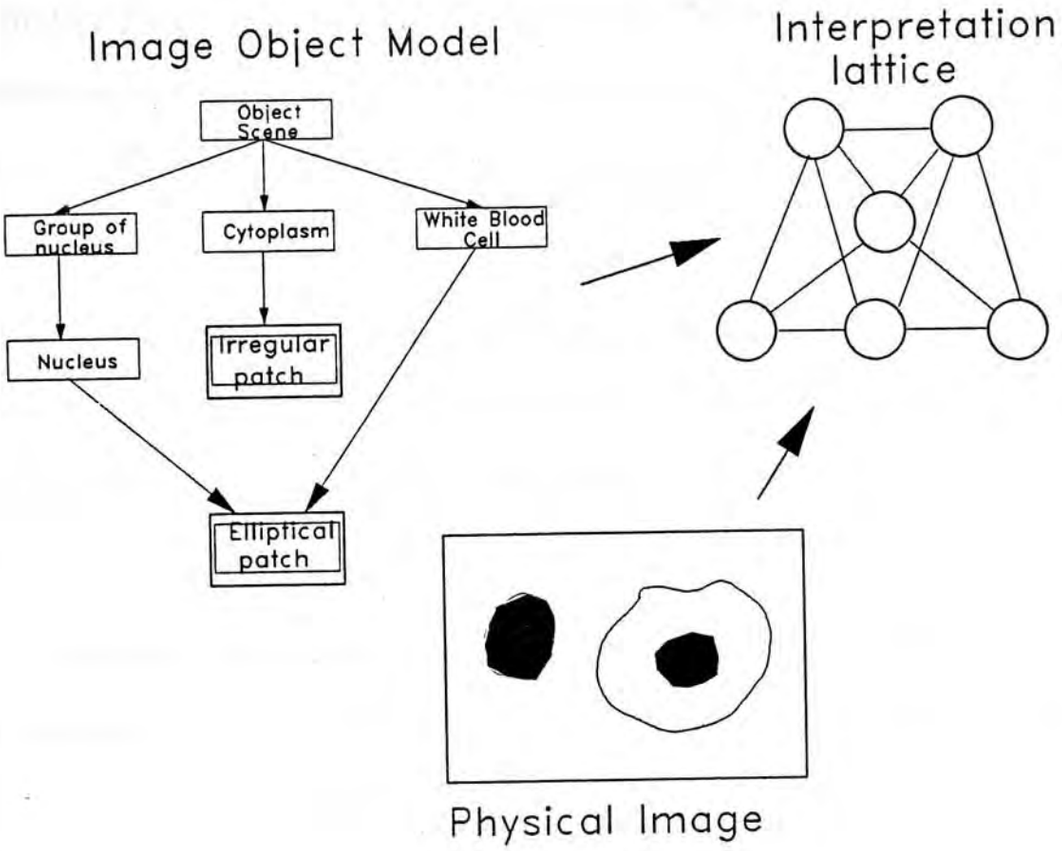

Figure 3.5

Mapping between Physical Image, Object Model and Interpretation Lattice

3.4.1 The Low Level Vision Kernel

The LLVK is formulated as a domain-independent segmentation kernel. It receives a list of instructions from the ISG. In the ISG, different instruction sets are stored to tackle different natures of the input images. For example, the segmentation techniques used in the natural scene is totally different from that used in the cell

recognition. The ISG is also influenced by the HLVK. Whenever the certainty of the major interpretations done in the HLVK is below a certain threshold, the HLVK will re-initialize the LLVK, through the ISG, to restart the segmentation process according to another instruction set.

The LLVK uses general segmentation techniques to extract the image structure. The goals of segmentation include enhancement or modification of the image to improve its appearance or to highlight information, measurement of image elements, classification or matching of image elements in the image.

The low level image processing algorithms can be classified in many ways. If an algorithm changes a pixel's value based only on its value, it is called a point process. Whilst if the algorithm changes a pixel's value based on its value and that of its neighbouring pixels, it is called an area process. Details of the algorithms in the LLVK are described in the chapter 5.

3.4.2 The High Level Vision Kernel

The HLVK uses object models to interpret the data recorded in the symbolic database and construct the interpretation lattice. The HLVK employs both bottom-up and top-down control schemes. All the control strategies are based on the object-oriented paradigm. There are two main types of objects in OOI, image and control objects. The behaviour of an object is embodied in methods which are

procedures for performing activities or a set of rules to be fired. All activities in an object-oriented system are caused by communication between objects by sending messages to each other requesting some information or actions.

Rules are used in the bottom-up process to make a plan for a rough interpretation of the scene. A plan is a set of hypotheses about the image with a relatively high certainty factors. The rules for the top-down process make a detailed semantic description of the scene. The top-down analysis works in the framework of region growing. The process includes a series of iterations of merging and labelling operations. Based on the information on the scene provided by the bottom-up analysis, the top-down control scheme performs an efficient analysis by focussing its attention.

Hierarchical reasoning is used in HLVK. All the system knowledge is organized in hierarchical levels with several levels of abstractions and relies on meta knowledge to guide the recognitions. Details of the hierarchical reasoning can be found in chapters 4, 6 and 7.

A scene description in the form of an interpretation lattice is built as the result of a top-down and bottom-up process. An interpretation lattice contains all the physical links between recognized regions. Inside the interpretation lattice, all attributes and semantic descriptions of each merged and labelled patch can be obtained. In addition, the spatial relations between each region are included. The lattice also acts as a spatial database to supply the necessary information to the user

queries through the user-consultation kernel. The analysis process is completed when all regions extracted from the LLVK are interpreted and assembled into the interpretation lattice.

3.4.3 User Consultation Kernel

Potentially, OOI can construct all possible interpretations for each region in an image and put them into the interpretation lattice. The goal of the analysis is provided for the user consultation kernel (UCK) by the user as a query. Whenever the UCK is activated, it matches the goal with the interpretations already constructed in the lattice. OOI needs not only to give an interpretation, but to select the most suitable one having the highest certainty factor, among many interpretations, as its conclusion. If the interpretation is found, UCK will answer the user's query and explain why this interpretation is drawn.

3.5 Structure of the Image Object Model

In OOI, an image model is represented as a graph structure of nodes and arcs as shown in figure 3.6. Each image object class represented by a node in the graph structure, is described by the object based system. Object classes are used to model abstract image objects in the problem domain such as "nucleus" or "cytoplasm". Each object class may have many associated descriptions that are defined by attributes.

Relations between objects represented by arcs in the graph structure, are described by rules and links. Rules are used to build up the relations between objects while links are used to describe the hierarchical relations between objects.

The object based system can be seen as an encapsulation of attributes and methods inside itself. It contains database handling facilities allowing the system to intake, verify and manipulate its attributes. All the main characteristics, such as spectral, position properties and relative positions among the different objects of an image scene are stored in the attributes of the objects.

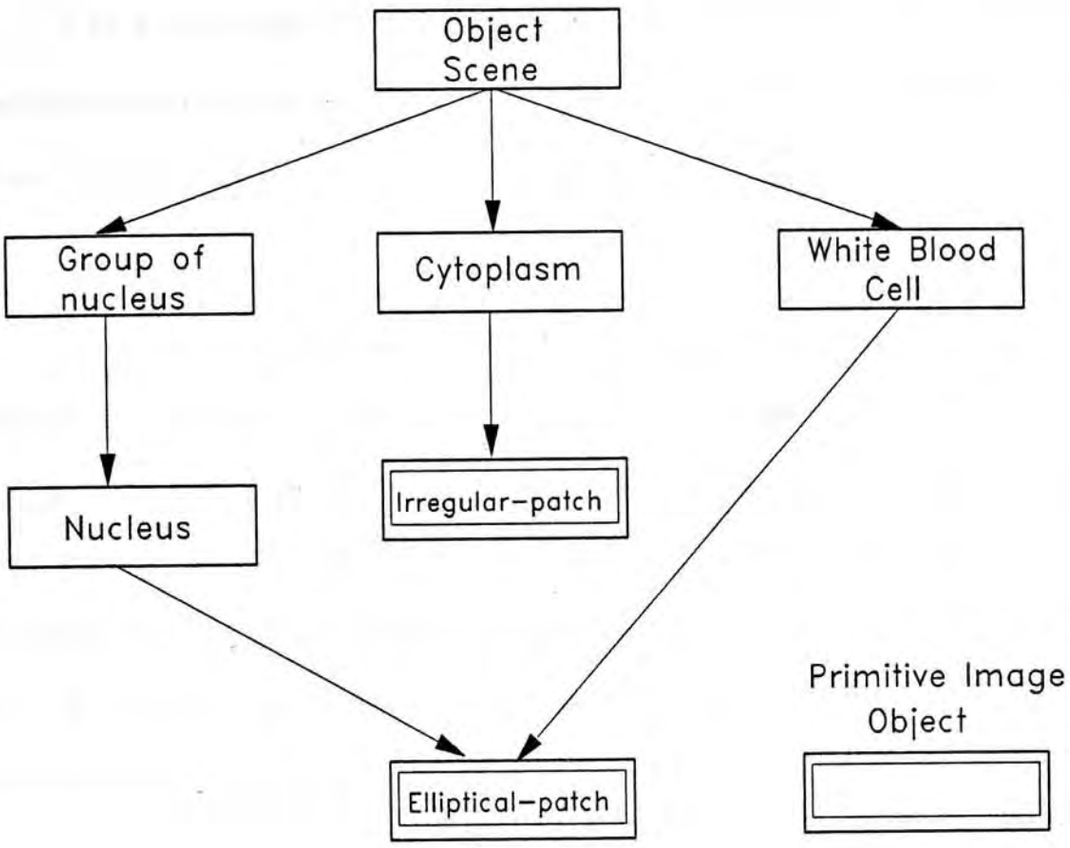

Figure 3.6 Object Model in Cancer Cells Investigation

3.5.1 Image Object Model in Object-Oriented Form

An object model hierarchy often involves generalization/specialization. The terms generalization and specialization can be defined as follows:

If C is a concept then $|C|$ means the set of instances of C. A concept G is a *generalization* of a concept C iff $|C| \subseteq |G|$, conversely, concept C is a *specialization* of a concept G.

In OOI, generalization and specialization relations are described by COMPRISED-OF and COMPONENT-OF links inside each image object. Link COMPONENT-OF describes an object which is the generalization of the current object while link COMPRISED-OF describes objects which are its specializations. All the knowledge recorded in a father object can be inherited by all the objects that are linked to it by the COMPRISED-OF link. The following image object classes are examples in C++ form.

Object-Class NUCLEUS-GROUP;

Attributes:

object-id

metric-attributes

shape-attributes

topological-attributes

Private rules & methods:

rules

methods

Friend rules & methods:

// Friend is a reserved word in C++

// Methods declare in friend can be shared with other objects.

rules

methods (which are shared with other classes)

Public rules & methods:

rules

methods

Links:

Comprised-of: nucleus

End-Object-Class

Object-Class NUCLEUS;

Attributes:

object-id

metric-attributes

shape-attributes

topological-attributes

Private Rules & methods:

rules

methods

Friend Rules & methods:

rules

methods

Public rules & methods:

rules

methods

Links:

component-of : nucleus-group

comprised-of : elliptical-patch

End-Object-Class

Object-Class ELLIPTICAL-PATCH;

Attributes:

object-id

metric-attributes

shape-attributes

topological-attributes

Private Rules & methods:

rules

methods

Friend Rules & methods:

rules

methods

Public rules & methods:

rules

methods

Links:

component-of : nucleus

End-Object-Class

It should be noted that some of these attributes and methods might represent defaults, and use the information inherited from its antecedent to provide values for particular instances of the class of objects described by the stereotype.

3.5.2 Image Objects Hierarchy

Figure 3.7 is an example illustrating the overview of objects in OOI. The box labelled PHYSICAL IMAGE represents a collection of image patches extracted from the LLVK. In addition to the PHYSICAL IMAGE, the system consists of several generic objects used to describe a two dimensional picture. They are patch object (Pat_object), region object (Reg_object) and entity object (Ent_object). It is stratified into several levels with which image objects in the universe could be represented. The levels are not all as distinct and well ordered as the figure might imply, but its

spirit can be accepted for discussion purpose.

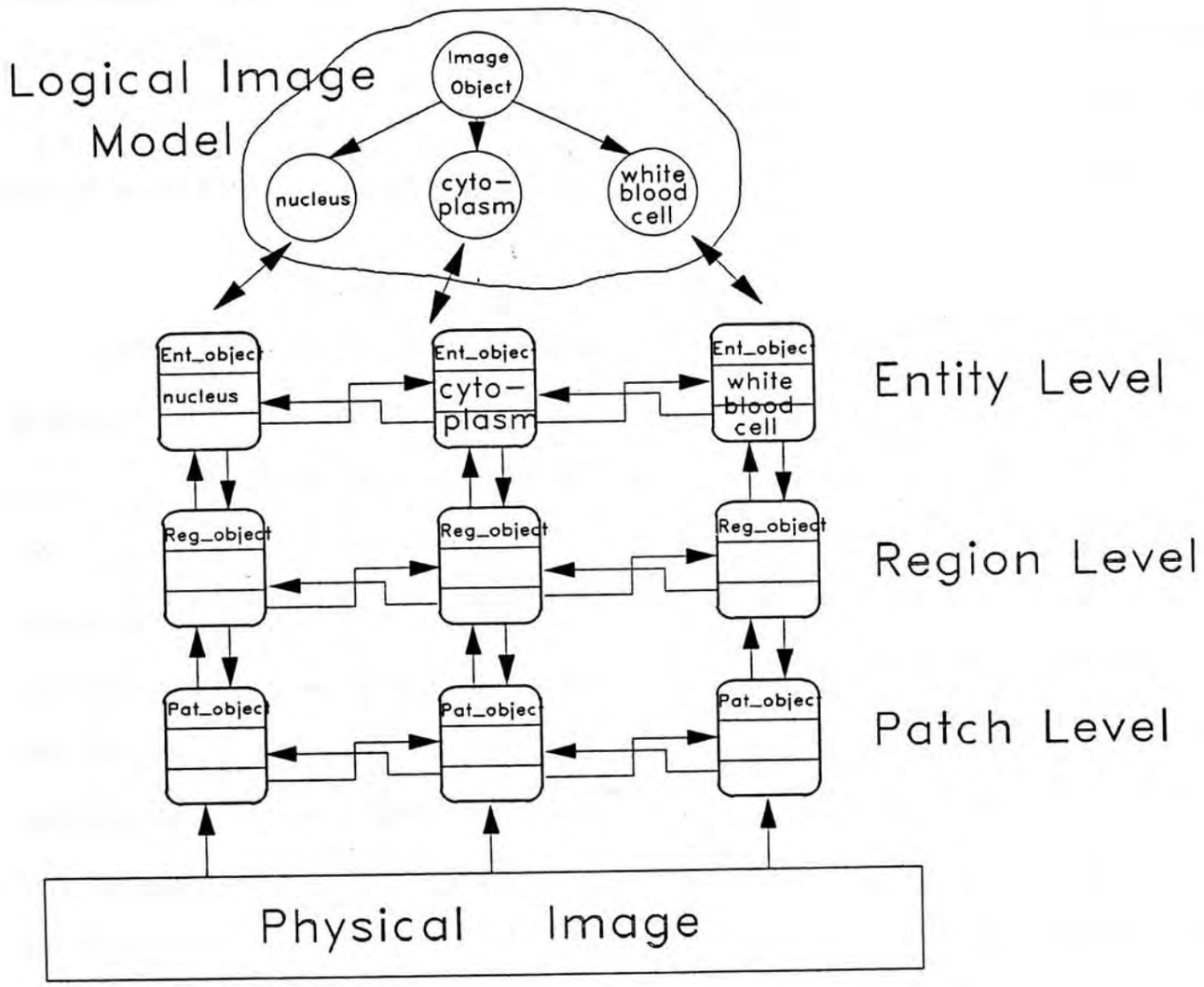

Figure 3.7 Image Objects in OOI

The arrows in figure 3.7 indicate allowed message paths between different objects. For example, a pat_object may send messages to an Reg_object, or to the pixel objects within the PHYSICAL IMAGE, and it receives messages only from the same set of objects. This design restriction on message paths is a means of enforcing a logical separation between different image representation levels. Messages are forced to pass through objects which incorporate boundaries between levels.

The physical structure of the *entities* objects can be described in terms of their geometrical properties. These might include the locations or areas that they occupy, along with their spatial relationships to other entities in the world. The Ent_object at the top level of figure 3.7 embodies the semantic meaning of the image. Since entity may composed of more than one region patch, it can be described locally in terms of their occupied *region* patches. The Ent_object may send messages to and receive messages from the Reg_object, which encapsulate the local properties and characteristics of regions. Typically, in order to let an Ent_object perform its function, Ent_object sends a message to a Reg_object requesting it to perform its function, and to supply it with any data or parameters that Ent_object might need. The Reg_object may sent a messages to an Ent_object informing it of the results or errors which may affect the future flow of control.

The remaining levels of the picture are all image description terms. Edges in the entities or regions can be described by *segments*. Segments may appear in

the pat_object. The segments can be produced using edge detectors, such as Sobel or zero crossing technique described in chapter 5. The lowest level representation for an image is the *pixel* level, which is shown in the PHYSICAL IMAGE in figure 3.7. A digitalized image is represented as a two-dimensional array of integers, with dimension 512 X 512, each representing an intensity level (gray level). The array is arranged so as to retain the geometry of the image plane of the camera. Similarly to the upper level, reg_objects send messages to or from the pat_objects in order to activate methods encapsulated inside the pat_objects or vice versa.

3.6 Reasoning in OOI

The reasoning in OOI is a generate-test paradigm and can be achieved in the following steps:

1. Feature extraction
2. Hypothesis generation
3. Predication
4. Verification
5. Deduction/Conclusion

The generate-test paradigm involves making hypotheses of the models after extracting the possible features in the input image scene. These hypotheses are then tested to see if the object is indeed present in the image. If the hypothesis test is positive, then the matched scene features are removed from the consideration and hypothesis

is generated. If the hypothesis test turns out to be negative, then another hypothesis is generated. Hypotheses are continuously generated and verified until a complete description of the scene is made using all available information.

To generate hypotheses, the vision system extracts distinct features from the scene and matches these features within the stored model features. The features from the scene have numerical properties that can be used to differentiate between features and to find its location in the scene. Based on these matches, the system hypothesizes a model and a location in the scene.

In chapter 4, a more detailed discussion on the control mechanism and messages handling within each object level is presented.

3.7 Concluding Remarks

In order to construct image diagnoses system with domain knowledge separated from control mechanism, an object oriented expert system shell is proposed in this chapter. The architecture of the shell has been discussed. The shell consists of several generic objects used to describe a two dimensional image. The shell employs a robust control strategy which minimizes the amount of domain-specific control knowledge. It incorporates both bottom-up and top-down analyses. Details in control and strategies can be found in chapter 4.

CHAPTER 4. CONTROL AND STRATEGIES

4.1 Introduction

A problem could be better solved by a number of experts, each using his/her own knowledge to tackle a particular aspect of the problem. The term 'knowledge source' (KS) is used as a set of knowledge used by an individual 'expert' to tackle a particular aspect of the problem [Erman80]. The knowledge sources have the following properties :-

(a) The domain knowledge needed to solve a problem is partitioned into KSs.

(b) KSs are kept separate and independent.

(c) Each KS is used to transform information in one level, possibly using information on the same or other levels of the hierarchy, into information of the same or other levels.

(d) Each KS may use algorithms or heuristic knowledge, forward or backward reasoning.

According to Feigenbaum, in any problem-solving activities, a hierarchy of knowledge for solving the problem is necessary [Feigenbaum78]. As shown in figure 4.1, at the lowest level on the control plane is a set of knowledge sources (KSs), the tasks of which are to make the primary inferences. They are organized according to their levels of abstraction and is called "Hypothesis formation" level. The next level is

the "meta" level, which has knowledge about the capabilities of the KSs in the hypothesis-formation level. This level is called the "KS-activation" level. KSs on this level represent a policy on the knowledge utilization. The highest level is the strategy level, which analyzes the quality of the current solution to determine what region of the data to analyse next. It also determines what kind of strategy to use.

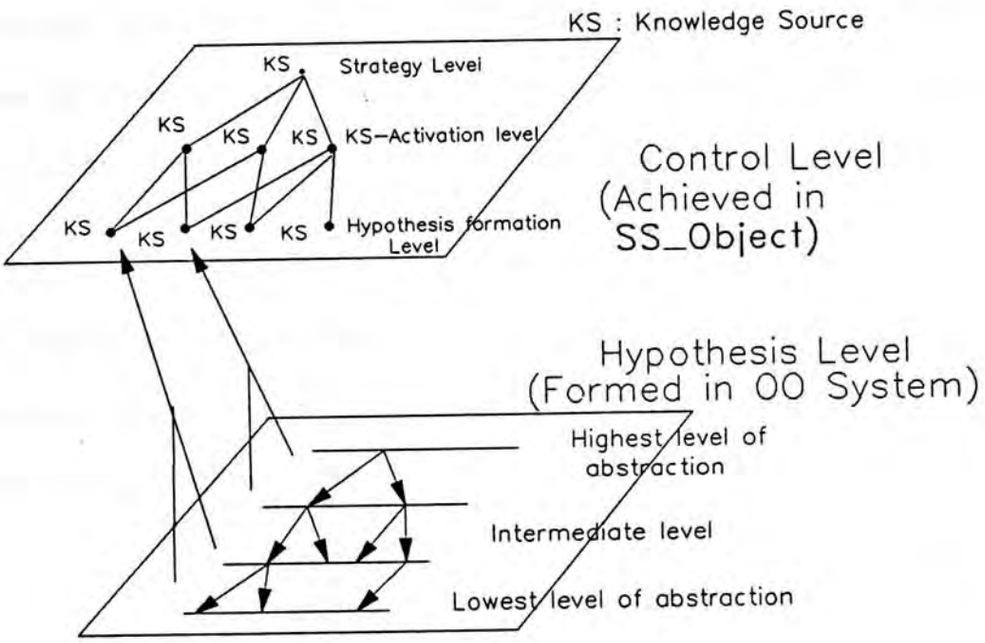

Figure 4.1 The Hypothesis and Control Plane

In order to achieve an effective control in OOI, a set of knowledge sources, called control objects, are designed. Figure 4.2 gives a detailed view of the structure of an image object level described in chapter 3, i.e. entity, region or patch levels and

its relationships with the set of control objects.

In the design, control tasks are achieved by six classes of objects, which are divided into two main categories, consultation and operation. The purpose of consultation objects is to hold information and give advice and strategies to the operation objects. Audience, Priority Table (PT_object), Intrinsic Hypothesis (IH_object) and the Domain Knowledge Base (DKB_object) are the consultation objects. Whenever they are invoked to make some decisions, they will send messages to pre-specified objects, in the directions shown in figure 4.2. Since they behave like information centres among the objects, they have relatively high message flowing rate between them.

In contrast to the consultation objects, Scheme Scheduler (SS_object) and Task Scheduler (TS_object) are the operation objects. A lot of methods are embedded inside the operation objects or its descendants. They receive messages from the consultation objects and execute them as advised.

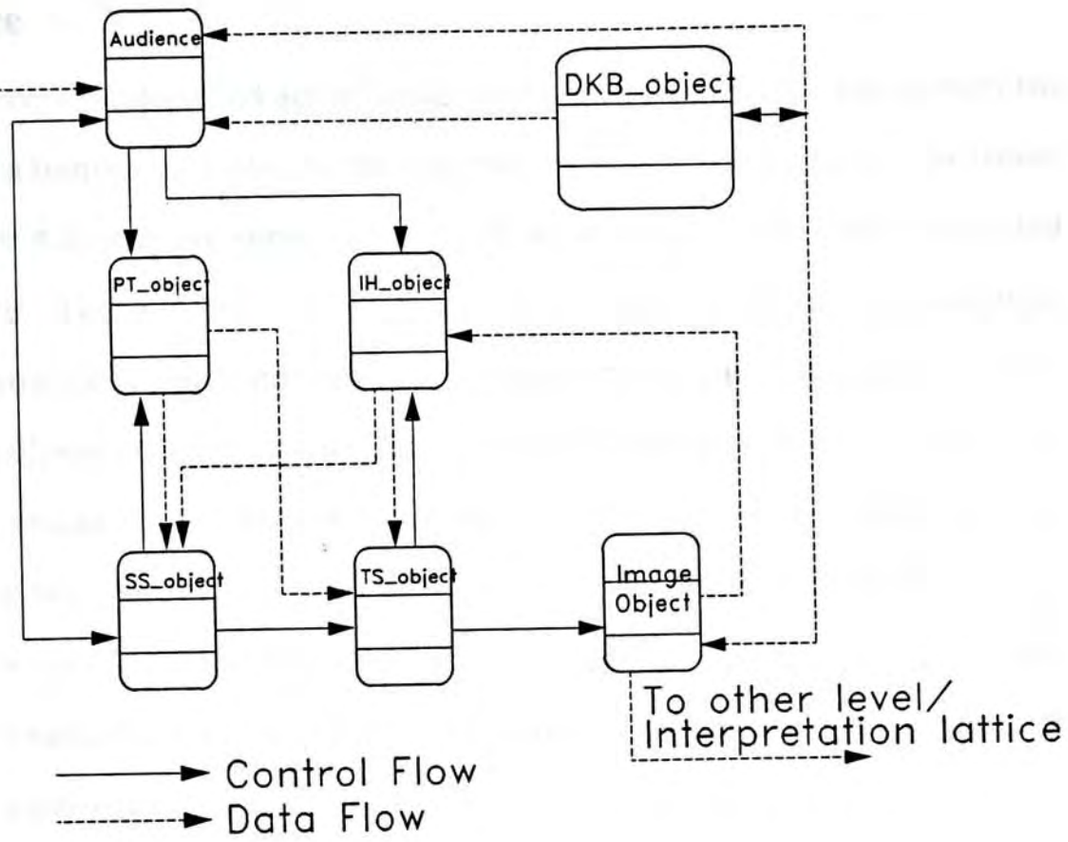

Figure 4.2 Control Objects in each Image Level

4.2 Consultation Class Objects

The consultation class objects are detailed in the following subsections:

4.2.1 Audience

The currently interested set of image patches are input into the system for analysis. If it is a bottom-up process, the interested set is transferred from the lower level as in figure 4.2, and vice verse. The selections of image patches are controlled by the audience. There is only one instance of an audience object. Its role is to interpret and analyse input from currently interested set of image patches and to initiate the job allocation to other consultation and operation objects. Messages are sent to the PT_object and IH_object to inform the presence of the image patches. According to the strategies stored in the scheme scheduler, image patches are filtered before analyses and suitable candidates of image patches can be selected by the audience. For example, in a bottom-up process, image patches with large areas are chosen by the audience.

4.2.2 Intrinsic Hypothesis (IH_object)

IH_object is used to store the hypothesis made for each image patch. It is independent of level of the image object. The certainty of each hypothesis is revised and updated after the image patch traversing each layer. If the certainty of a hypothesis is greater than a preset threshold, the object is said to be identified and it is put into the interpretation lattice. IH_object can also be used to determine the area of interest at the current level. That is, hypothesis from a simpler vision level may be used to reduce reasoning time at a more complex level. For instance, if a nucleus were confirmed at the pat_object with a high certainty, IH_object will reduce the reasoning time in the Reg_object by passing the image patch directly into the

香港中文大學圖書館藏書

Ent_object level. Task Scheduler (TS_object) at the region level may suggest to look for some other particular features at the same level, which may help confirm or refute the hypothesis proposed from its lower level. The confirmation of a hypothesis in the IH_object at one level most likely changes the deduction of another TS_objects.

4.2.3 Priority Table (PT_object)

Corresponding to each level, as shown in table 4.1, there is a priority table (PT_object) which contains all the tasks pertinent to the Scheme Scheduler (SS_object) from which TS_objects are activated. The TS_objects are sorted according to the priority of the tasks. The priorities of each attribute in the PT_object are determined during knowledge acquisition. This may be regarded as a form of meta-knowledge which contributes to the reduction of search space.

Attributes	Tissue 1	Tissue 2	Tissue 3	Tissue 4	Overall
Size	2	2	1	1	1
Gray-level	3	1	3	2	2
Centroid	4	4	2	3	4
Curvature	1	3	4	4	3

Table 4.1 An Example of Priorities of Each Attribute in a PT_object

4.3 Operation Objects

In contrast to the consultation objects, Scheme Scheduler (SS_object) and Task Scheduler (TS_object) are operation objects. A lot of methods are embedded inside the operation objects or its descendants. They receive messages from the consultation objects and execute them as advised.

4.3.1 Scheme Scheduler (SS_object)

SS_objects obtain the domain knowledge from the DKB_object to decide on the scheme to be explored. The SS_object comprises two major levels of control, the activation level with a set of metarules and the strategy level. These two controls are used in non-primitive tasks which do not directly manipulate the patch of interest. The metarules govern the policy on the knowledge utilization. They contain the meta-knowledge about the capabilities of the messages passed from objects of other levels. The strategies in the strategy level are used to analyze the quality of the current solution. Since the SS_objects can be organized, if applicable, in a hierarchy, a parent SS_object is supported by a group of child SS_objects. The SS_objects are global objects which are used in different levels of image objects and they are used to keep the main goals or objectives of the system consistent throughout the reasoning.

4.3.2 Task Scheduler (TS_object)

Tasks are embedded in a TS_object. Each task is composed of methods and rules to perform the task. They are all different and triggered by the SS_object. The TS_objects are local in each image level. The mechanism of TS_object is shown in

detail in section 4.5.

4.4 Taxonomy of Image Objects in OOI

An object is an abstraction for defining both static and dynamic properties of entities being modelled and the relationships between them. The possibility of defining inheritance hierarchy of objects and putting objects together to form a more complex object, plus mechanisms allowing object communication and interaction are all intrinsic to the object-oriented paradigm. Abstractions essential to the development of the system are all built in the taxonomy of the image objects in OOI.

Herein, since the number of the image objects is numerous and the messages passing from the TS_objects to the image objects are crucial, all the image objects in OOI adopt a general object template. The messages passing are governed by the TS_objects, in order to ensure the modularity and the consistency in the whole system.

4.4.1 Object Template

The template of an image object includes

(i) the definition of the attributes,

(ii) the tasks and life cycles and

(iii) the conditions that must be fulfilled so that a task can happen.

4.4.2 Attributes

Associated with each object, there are two main types of attributes namely, **constant** attributes whose values do not change and **state** attributes, also called **instance variables**, whose values can be changed. For example, in OOI,

```
const Image_ID;
int   Area;
```

The former is a constant while the area of the patch is a instance variable. The area may be changed after region merging. Constraints can be imposed upon the values of the attributes. A very trivial example is

```
Area >= 0
```

The values of all attributes of an object are called the **situation** of the object. Of course, an object can evolve from a situation to another situation provided that at least a value of an attribute is changed.

4.4.3 Tasks and Life Cycles

Tasks are defined in an object as the abstractions for incorporating the dynamics. Tasks are classified into **Birth**, **Update** or **Death** categories. It is possible to have more than one task in each category. In most situations, a task leads to the modification of the value of an attribute. Examples of tasks are

new,

growing,

merging,

abortion and

mature.

To "new" an image object is a birth task to initiate a new image object for processing. The growing of an image object during region growing is an update task. If the image object is merging to another object, the original image object is considered dead. Two other death tasks, abortion and mature, correspond to giving up the deduction in the image object and to achieving the successful deduction respectively.

A life cycle of an image object involves a sequence of birth, updated and death tasks. The sequencing of these tasks is implicitly restricted which means that a life cycle must start with a birth task, followed by a sequence of update tasks and terminated with a death task. After the death task, no more tasks can be applied in the image object.

4.4.4 Object Security

Restrictions can be set during the possible life cycles of an object. Security

is achieved by imposing **guards** in the following form.

{C} e

where C is a necessary condition for task e to happen. However, this condition does not oblige e to happen. It only indicates that e is permitted to happen. It provides the necessary condition, but not the sufficient condition, for the task to happen. For example,

{area > = 20} mature

4.5 Message Passing

Messages are passed from the SS_objects to activate the TS_objects. TS_objects, described in section 4.3.2, contain all the primitive and potential tasks acting on the image objects. Each TS_object consists of descriptions about the actions. The roles of the TS_objects are to **select, evaluate** and **execute** the tasks. With the help of the security guards in each image object, described in section 4.4.4, they hold some conditions to check the validity of the tasks occurrence. Trigger conditions, preconditions and reject conditions are checked during the selection, evaluation and execution phases as follows:

Once a message is sent from a SS_object, the set of TS_objects which are interested in the message will be selected and given a FRESH state. During the **selection** phase, the trigger conditions of these fresh TS_objects will be examined and a TRIGGER state will be given to the suitable candidates. If the preconditions of the triggered

TS_objects are met in the evaluation phase, they will be given a PRECONDITION state. The reject conditions of these TS_objects are checked in the execution phase. The survived one is executed and given an EXECUTION state. Moreover, if the current inference sequence cannot arrive at a conclusion, re-examination of the fail TS_objects will be conducted.

4.6 Strategies

A symbolic description of the input scene is made via an object-oriented region analyser. This analyser tries to segment an input image into meaningful regions and assigns object label to each of them. Reasoning mechanism is extremely important in the design. The strategies in the strategy level of the SS_objects are used to determine the reasoning mechanism in the system. The mechanism is designed as a highly object-oriented hierarchical approach, working from bottom-up and top-down.

Initially, each image patch is in a form of an instance of a physical_image class, pat_object described in chapter 3. They are stored in a linked list. First of all, data driven approach is used. Each production rule in Domain Knowledge Base (DKB) checks the status of all instances in the linked lists, and executes the associated action when its condition is satisfied. The successful candidates, which are called the key-patches, are stored in another linked list for another phase of investigation. In the second phase, a model driven approach is used for the remaining minor patches. The process will iterate until most of the patches have been labelled. The process is shown in figure 4.3.

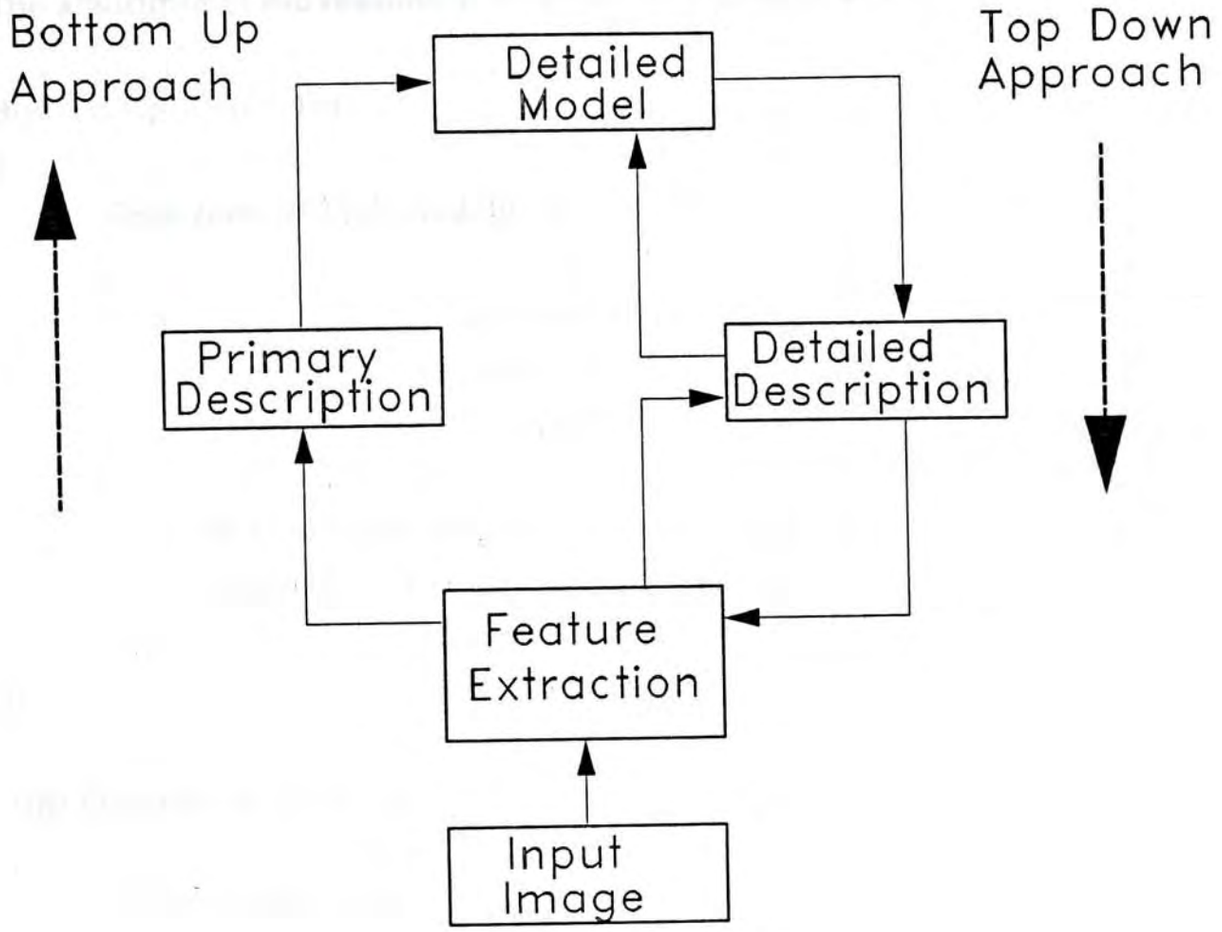

Figure 4.3 Combination of Top-Down and Bottom-Up Image Analysis

The algorithm of the reasoning strategies is described as follows:

```
Bottom-Up(linked_list)
{
      While (not(NULL(linked_list)))
      {
            For each image object in linked_list,
            using rules in Domain Knowledge Base (DKB),
            performs evidence combination;

            /* Append keypatches into key_linked_list */
            append(key_linked_list(keypatch));
      }
};

Top-Down(high_order_linked_list, linked_list)
{
      While (not(NULL(key_linked_list)) and (certainty_factor > THRESHOLD)
      {
            merging(key_linked_list, linked_list);
            matching(Image_models, linked_list);
            updating(high_order_linked_list, linked_list);
      }
};

Reasoning(physical image)
{
      int i = 1;

      /*    Initialize an array of linked list,
            array_linked_list[1] contains image objects, pat_objects
```

```
            array_linked_list[2] contains image objects, reg_objects
            array_linked_list[3] contains image objects, ent_objects
        */

        Initialization;

        While ( i < number_of_levels )
        {
            Bottom-Up(array_linked_list[i]);
            Top-Down(array_linked_list[i+1],array_linked_list[i]);
            i++;
        }
    }
```

Reasoning based on abstract knowledge has been recognized as a superior approach [Sacerdoti74]. It is less time consuming to first attempt to solve a problem by ignoring unimportant details and using higher level of knowledge abstraction. When a solution to the problem is found with a high enough confidence value, all that remains to be done is to account for details that support the hypothesis. The two approaches are discussed in follows:

4.6.1 The Bottom-Up Approach

A "plan", containing a group of hypotheses, is generated as a representation of the crude structure of the input scene. The plan is represented by a set of image object labels with their corresponding degrees of certainty. In this phase, hypothesis formations are based on the image primitives. They are used as the basis for the image analysis. The approach is data-driven. The primitive image objects are

allocated in the interpretation lattice with a set of certainty factors, as shown in figure 3.5. All allocations are based on the rules in the control objects. The format of the rules is composed of three parts as follows:

IF <fuzzy predicates>
THEN <hypothesis>
WITH <certainty>

For example,

Rule N01:

IF gray-level-of <patch, A> is low and
size-of <patch A> is medium
THEN <patch A> is a nucleus
WITH C.F. = 0.8

In the traditional bottom-up approach, difficulties lie more in the segmentation and feature extraction than in the recognition itself. If the segmentation is successful, the feature extraction becomes reliable, and the recognition can be simple. Good recognition is not achievable without an accurate segmentation of the whole pattern. In order to generate the plan efficiently, patches with large areas or salient features are first selected from the segmented image. It is reasonable to assume that most of them correspond to large parts of objects in the scene and that

they can be extracted from the image data rather reliably in the segmentation process. Since they have a larger area relative to the other patches, they have a relatively high signal to noise ratio (S/N) and is particular suitable to be a pioneer. These patches are called keypatches. It will be desirable that certain reliability measures be given to each of the keypatch, and the total reliability measures are calculated depending on the larger structures constructed from the combination of significant keypatches. This measure will be useful in the selection of candidate models. It should be possible to grasp the rough structure of the scene by assigning object labels to the larger structures. The labels may have multiple values containing different certainty factors corresponding to different hypotheses toward the larger structure. For example,

```
(Structure 001 :
        (nucleus        0.9)
        (blood_cell     0.7)
        (garbage        0.1))
```

Small patches in the segmented image may disturb the evaluation of the relations between the keypatches. Hence, merging of these patches with the keypatches has to be done. The merging technique is discussed in chapter 6. When a small patch touches more than two keypatches, similarity factors are computed. The merging depends on the similarity of gray levels and the compactness of the region which would be obtained if the small patch and the keypatch were merged. The

compactness criterion guides the merging operation to obtain regions with smooth boundaries. In order to evaluate the global relations between regions, smooth boundaries are rather crucial. The keypatch which obtains the highest similarity factor and potential compactness is selected. The bottom-up analysis is the repetitive application of parsing of adjacent pattern components into a higher level conceptual element.

4.6.2 The Top-Down Approach

The concept of top-down analysis is introduced initially in pattern analysis with the concept of syntactic analysis. Top-down starts with a model. The assumption for an object is first made without any reference to the input image. Then the model under assumption is decomposed into lower picture components according to the rules of hierarchical model description. And the check is performed to see whether these components exist in the image in the prescribed positions. In OOI, methods and rules, embedded in each level of control objects, are used to examine each patch in the segmented image which has not yet been interpreted. They are also used to verify the hypotheses developed as well. Hypotheses formation in this phase is based on expectation, where the expectation is inferred from knowledge about the domain. This approach is also called model-driven. The image primitives are used solely to verify the expectation.

In OOI, the model is organized as hierarchical objects as shown in chapter 3. Control objects include a set of rules and methods which describe properties and relations with other image objects. For example, at region level, the DKB_object for a cell consists of two major components: nucleus and cytoplasm. Each component object is composed of several attributes which collectively infer the component object to be a specific part of a cell. Each attribute has an associated weight which indicates the importance of the attribute compared to the others. In identifying a cell in the image, its component objects are first be investigated, based on the model developed. The methods and rules stored in the component objects are "watching" the symbolic database or the interpretation lattice. Whenever the predicates in the condition part is fulfilled, the hypotheses are upgraded. The image primitives are used to verify the component objects, in consequence, the cell is identified.

4.7 Concluding Remarks

The control in OOI is thoroughly described in this chapter. A set of knowledge sources are designed. Meta-Knowledge and methods are embedded inside these knowledge sources. Communication between them are achieved by message passing. All the image objects in OOI adopt a general template in order to ensure the modularity and the consistency in the whole system. The strategies in OOI are combination of both bottom-up and top-down approaches.

CHAPTER 5. IMAGE PROCESSING ALGORITHMS

5.1 Introduction

Image processing is the science of modifying and analyzing pictures. Most of the algorithms implemented in OOI are flexible and have relatively low processing and memory costs. The algorithms are expressed in computer programs as methods in the system.

The goals of image processing include enhancement or modification of the image to improve its appearance or highlight information, measurement of image elements, classification or matching of image elements, and recognition of items in the image. Image measurement makes few assumptions about what things are in the pictures, while classification and recognition require successively more knowledge about what can appear in the image.

The image processing algorithms can be classified in many ways. If an algorithm changes a pixel's value based only on that its value, it is called a point process. Whilst if the algorithm changes a pixel's value based on the value of that pixel and the values of its neighbouring pixels, it is called an area process.

Extraction of useful information from an input image in the project begins with low-level operations. Some of them are more general techniques whilst some of them are more specific, which are much more suitable in the medical domain. All the image processing algorithms implemented for enhancement, pixel classification, edge detec-

tion and segmentation in OOI are described in the following sections.

5.2 Image Enhancement

There are two possible approaches to enhance image features. One is to increase the contrast of suspicious areas and the other is to reduce their background variations. In OOI, image enhancement includes spatial filtering and feature enhancement. Convolution mask is commonly used in spatial filtering whilst feature enhancement is achieved using histogram manipulation.

5.2.1 Spatial Filtering

Spatial filtering is an area process, which uses neighbourhood information to modify the pixel value or assess the existence of some properties at an image point. Spatial filtering can "sharpen" the image's appearance by accentuating intensity changes and provide many other useful enhancements. Some of these include finding objects by matching images using matched filter, measuring image properties, making assertions about object edges in the image, removing noise, and blurring or smoothing the image.

Traditionally, a common approach to spatial filtering has been to analyse the two-dimensional Fourier power spectrum of the image. A busy texture, relative to a smooth one, has considerable power concentrated at spatial frequencies. Directionality in the image is represented by a directional bias in the two-dimensional spectrum.

Co-occurrence matrics have been used with some success by a number of research workers [Haralick79]. The co-occurrence statistic **P(i,j,x,y)** describes the probability that gray level values i and j occur at points separated by a vector (x,y). Much of the success of the co-occurrence matrix methods has stemmed from their application to satellite imagery featuring essentially planar surfaces.

Besides using co-occurrence matrics, convolution technique is used in OOI. Convolution masks have been used with considerable success for the task of spatial filtering [Pratt78]. They are noteworthy for their simplicity and because they are computational less expensive than conventional Fourier methods and co-occurrence matrices. A convolution mask is a discrete approximation to a two-dimensional convolution integral [Duff83]. Convolution is a classic image processing algorithm commonly used for spatial filtering and finding image features. The convolution operation replaces a pixel's value with the sum of that pixel's value and its neighbour, each weighted by a factor. The weighting factors are called the convolution kernel. To convolve an image area, a sliding kernel matrix operates over each row of pixels in the matrix image. At each point, the kernel values multiply the image values under it, sum the result, and replace the pixel at the centre of the kernel with the value. The equation is

[Eqn. 5.1] $$H(i,j) = \sum_{m=-k}^{m=k} \sum_{n=-k}^{n=k} M(m,n) * I(i+m, j+n)$$

where the operator, M is the convolution mask and I is the gray level of a pixel in the image. The mask weights, M(m,n) are generally signed integers and satisfy a zero sum constraint, which results in a zero response in areas of uniform intensity and removes any bias from image. The mask, also called a filter of kernel, is of size (2K + 1) X (2K + 1) elements. In OOI, a kernel having of size 3 X 3 elements is used.

A view of convolution is that it performs spatial frequency filtering. In sound, frequency is the number of times per second a waveform repeats. In two-dimensional images, spatial frequency is the number of times per unit distance that a pattern repeats. As with a one-dimensional signal, an image can be broken down into a series of sine and cosine waves in form of spatial frequency. Quickly changing image intensities are represented by high spatial frequencies, while slowly changing intensities are represented by lower spatial frequencies. Transformation can be done both horizontally and vertically. This can be accomplished by using a fast Fourier transform. Apart from using the Fourier transform, a suitable kernel can be chosen to detect a certain band of frequencies.

For instance, edges and other sharp transitions (such as noise) in the gray levels of an image contribute heavily to the high-frequency content of its Fourier transform. It then follows that blurring is required by attenuating a specified range of high-frequency components. This can be achieved in using the following kernel:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Since high-frequency components are "filtered out", and information in the low-frequency range is "passed" without attenuation, this method is commonly referred to as *low pass filtering*.

On the other hand, if one want to select high spatial frequencies, the following kernel is used. It is also called as a *high pass filter*.

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

This is often called a *Laplacian filter* because it approximates an unoriented second-derivative operation. Since edge have high spatial frequencies (sudden intensity changes), this kernel selects edges of any directions. It might be used as an "edge detector" for image analysis.

5.2.2 Feature Enhancement

In OOI, feature enhancement is achieved by using histogram manipulation. Histogram manipulation is a widely used idea for it is easy to derive a histogram from an image by recording the number of pixels at particular gray levels. The histogram often yields useful information about the nature of the image. To achieve the enhancement, it usually involves some transformation of the gray levels:- What transformation will be used depend on the expected effect, and can be deduced from the histogram. Histogram equalization is the histogram transformation. The idea of histogram equalization is to enhance the use of "underused " gray levels and to

damp down the use of "overused" ones. Very frequently, the algorithm are more clearly described by supposing that gray levels are continues in the interval $0 \leq r \leq 1$ and describing the transformation T(r) and $T^{-1}(s)$ as continuous function.

Assume the histogram behaves as a probability density function, P_r describing the distribution of the input gray level r. Now if T^{-1} satisfies the conditions of being monotonic increasing and single valued, according to elementary probability theory [Fisz63]

$$P_s(s) = \left[P_r(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)}$$

To visualize what this means in the case of equalisation, consider the transformation,

[Eqn. 5.2] $$s = T(r) = \int_0^r p_r(u)du \qquad 0 \leq r \leq 1$$

where s is the cumulative distribution function of r. Since P_r is a probability density function, T^{-1} will be single valued and monotonic on the interval [0,1]; further, it is immediately clear that

$$\frac{ds}{dr} = P_r(r)$$

and hence

$$P_s(s) = [P_r(r)\frac{1}{P_r(r)}]_{r=T^{-1}(s)}$$

$$= [1]_{r=T^{-1}(s)}$$

$$= 1 \qquad 0 \le s \le 1$$

So $P_S(s)$ should always be 1 regardless of the nature of the function $T^{-1}(s)$; such a density function corresponds to an equalised histogram, and the theory underlying the equalisation transform derived in the discrete case.

5.3 Pixel Classification

Segmentation is basically a process of pixel classification. The aim of image segmentation is to divide images into several meaningful areas. With such methods, "Objects of interest" can be extracted from quite complex scenes for further analysis or interpretation. Thus segmentation represents an important early stage in image analysis and image identification after the image enhancement. The image is segmented into subsets by assigning the individual pixels to classes. For example, when a image is segmented by thresholding its gray levels, the pixels are classifying into "dark" and "light" classes, in an attempt to distinguish dark objects from their light background. Similarly, in edge detection, pixels are classified into "edge" and "non-edge" by thresholding the response of some difference-operators that have high values when the rate of change of gray level is high.

Definition

Segmentation of a discrete image signal f(m,n), where $\{0 \le m \le M-1 \cap 0 \le n \le N-1\}$, is the division of f into disjoint nonempty subareas f_1, f_2, ...,f_p which satisfy the following criteria of uniformity E:

(a) $\cup_{i=1}^{P} f_i = f$

(b) f_i is connected $\forall i$ with $i = 1,..,P$.

(c) $\forall f_i$ the criterion of uniformity $E(f_i)$ is satisfied.

(d) $E(f_i \cup f_j)$ is not satisfied for any union of two neighbouring f_i, f_j.

The choice of suitable criteria of uniformity depends critically on the nature of the images to be segmented. In general, there are two approaches in segmentation, such as edge-oriented and region-oriented methods. Edge-oriented image segmentation is a process in which each image pixel is assigned a gradient value which can be a complex vector, i.e. it can have magnitude and direction. Various techniques for the computation of such gradient images are described in sections 5.4. Some of them are implemented in OOI. Edge-oriented methods generally lead to incomplete segmentation. To compensate the deficiency of the edge method, region-oriented methods are widely used in OOI. Section 5.5 gives the details.

5.4 Edge Detection Methods

Edge detection is a special case of pixel classification, based on local property values. Local features usually involve abrupt changes in gray level, which may take several geometrical forms:

(1) An edge : the gray level is relatively consistent in each of two extensive adjacent regions, and changes abruptly as the border between the regions is crossed. An ideal edge has a step like cross section.

(2) A line or curve : the gray level is relatively constant except along a thin strip. In cross section, this yields a sharp spike. It should be mentioned that lines often occur in association with edges, e.g. highlights on edges of blocks; membranes separating parts of a cell; roads running between fields bearing different crop types.

5.4.1 Local Gradient Operators

Difference operators are the common tools for detecting edges. They yield high values at places where the gray level is changing rapidly. Edge detections are based on an approximation of an operator, originally for continuous functions, adapted to the case of digital images. Within the realm of continuous two dimensional functions, the gradient f(x,y) is given by

[Eqn. 5.3] $$\nabla f(x,y)=\left(\frac{\partial f}{\partial x}+\frac{\partial f}{\partial y}\right).$$

Its magnitude is

[Eqn. 5.4] $$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

and the orientation of the gradient vector is

[Eqn.5.5] $$\alpha = \tan^{-1}\frac{\left(\frac{\partial f}{\partial x}\right)}{\left(\frac{\partial f}{\partial y}\right)}$$

But when used on the discrete problems, like digital images, x, y and f(x,y) are non-negative integer numbers so that the partial derivatives must be approximated with finite differences along the two orthogonal directions x and y, obtaining

$$\nabla_x f(x,y) = f(x,y) - f(x-1,y)$$

$$\nabla_y f(x,y) = f(x,y) - f(x,y-1)$$

and for any orientation,

$$\nabla f(x,y) = f(x,y)\cos\alpha + f(x,y)\sin\alpha$$

Finally the digital approximation to the gradient of f(x,y) is given by

[Eqn. 5.6] $$|\nabla f(x,y)| = \sqrt{\nabla_x f(x,y)^2 + \nabla_y f(x,y)^2}$$

Since this expression may be cumbersome, generally the digital gradient is considered to be either the sum of the absolute values of the two directional increments or the maximum between these two increments:

$$|\nabla_x f(x,y)| + |\nabla_y f(x,y)| \qquad \text{or}$$

$$\max(|\nabla_x f(x,y)|, |\nabla_y f(x,y)|)$$

These approximations are dependent on orientation. Another practical approximation which is often used, is due to Roberts [Roberts65], and may be written as

[Eqn. 5.7] $$f(x,y) = \max(|f(x,y) - f(x+1,y+1)|, |f(x+1,y) - f(x,y+1)|)$$

Since this approximation computes the finite differences of an element located at $\left(\frac{x+1}{2}, \frac{y+1}{2}\right)$, equation 5.7 may be considered an approximation to the continuous gradient at that position. On the other hand, if a three by three neighbourhood is considered, there is one well known approximation to the gradient, **Sobel** operator. The Sobel operator introduces weights in the summation of the values of the elements as shown in the following convolution masks.

$$H_1 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \qquad H_2 = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

The use of convolution mask in edge detection is easier to follow since all weights on the corresponding positions of the neighbourhood can be observed. In OOI, both the Roberts and Sobel operators are implemented for the edge detection.

5.4.2 Zero Crossing Method

The principle of the zero-crossing method is, instead of finding the maxima of a profile, to find the zero-crossings of the differentiated profile. The method was proposed by Marr and Hildreth [Marr80] to detect the edges of the features extracted in the initial image processing. They argued that the best operation was to apply the Gaussian operation to the original image for smoothing, and then apply the Laplacian operation.

The two-dimensional Gaussian distribution of the image is represented by:

[Eqn. 5.8] $$G(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\pi\sigma^2}\right)$$

where σ is the standard deviation. The Laplacian operation is then also applied to the points f(x,y) of the image as follows:

$$\nabla^2 G(x,y)*f(x,y)=\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}G(x-\xi,y-\eta)f(\xi,\eta)d\xi d\eta$$

$$=\left\{\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)G(x,y)\right\}*f(x,y)$$

The above combined operator $\nabla^2 G$ can be simplified as follows:

[Eqn.5.9] $$\nabla^2 G(r) = \left(\frac{r^2 - 2\sigma^2}{2\pi\sigma^6}\right) \exp\left(\frac{-r^2}{2\sigma^2}\right)$$

where $r = (x^2 + y^2)^{\frac{1}{2}}$.

Zero-crossings of a given image convoled with $\nabla^2 G(r)$ will provide its edge locations. On a two-dimensional grid, a zero-crossing is said to occur wherever there is a zero-crossing in at least one direction.

5.5 Regional Approaches in Segmentation

Regional approaches attempt segmentation of an image into regions according to regional image data similarity (or dissimilarity), as opposed to edge-enhancing and edge linking. After using a multi-level threshold method in OOI, a region growing technique, implemented using a graph-theoretic approach, is then applied.

5.5.1 Multi-level Threshold Method

Images, for example, containing light objects on a dark background or dark objects on a light background can be segmented by means of a simple threshold operation. The following relationship exists between the input signal f(m,n) and the output signal g(m,n):

[Eqn. 5.10] $$g(m,n) = \begin{cases} I_1 \text{ for } 0 \le f(m,n) < S \\ I_2 \text{ for } S \le f(m,n) \le f_{max} \end{cases}$$

where I_1 and I_2 are two arbitrary values with $I_1 \neq I_2$ ($I_1 = 0$ and $I_2 = 1$ are usually selected) and S is the intensity threshold to be used. Equation 5.10 is a typical case of the point operation. By suitable choice of S, pixels with value I_1 in the output image represent the objects and those with the value I_2 represent the background, or vice versa.

If an image contains more than two types of regions, it may still be possible to segment it by applying several thresholds. For images containing P objects with different characterizing intensity regions, equation 5.10 becomes

[Eqn. 5.11] $$g(m,n) = I_i \quad for \quad S_{i-1} \le f(m,n) < S_i$$

where $i = 1,2,...,P$, $S_o = 0$, $S_p = f_{max} + 1$, and $f_{max} = MAX\{f(m,n)\}$

For example, before automatic diagnosis of an optical microscope picture of cells is possible, separation of image into areas representing "cell plasma", "cell nucleus" and background is necessary. In the scene of blood cells, the nucleus is generally darker than the cytoplasm, which is in turn darker than the background. Thus the histogram has three peaks, and the picture can be segmented using two thresholds that separate these peaks. These two thresholds divide the gray scale into three ranges, which are displayed as black, grey and white.

Even when there are only two types of regions, it may be advantageous to use two thresholds in order to reduce the noisiness of the threshold picture. In OOI, multi-level thresholds are allowed to remove the noise as well as to separate the main features in the image.

5.5.2 Region Growing

A region is a connected group of pixels in the video buffer that is delineated by some sort of boundary. The aim of the region growing is to divide an area into a number of intraconnected subregion. In OOI, a line-adjacency algorithm for growing a region is used. Its general strategy is to locate each group of horizontally connected pixels in the interior of the region. Using a simple recursive call, this algorithm starts at a seed pixel known to be in the region's interior. It scans left and right to find the ends of the seed pixel's row, then grows the entire row. The algorithm proceeds by locating all groups of horizontally connected pixels that are vertically adjacent to the group it just scanned. Each time it finds an adjacent group of not-yet-filled pixels, and then the growing mechanism is called recursively to fill them. The algorithm terminates when all interior pixels have been filled. The algorithm can be described in the following pseudo code:

```
Region_Grow(seed_i, direction)
{
    For the seed_i
        Scan the left endpoint of seed line segment;
        Scan the right endpoint of seed line segment;
```

```
        Grow the line with the both endpoints;

        if (adjacent rows of pixels is detected in same direction) {
                get another seed, seed_j;
                Region_Grow(seed_j, direction);
        }

        if (adjacent rows of pixels is detected in opposite direction) {
                get another seed, seed_k;
                Region_Grow(seed_k, another_direction);
        }

}
```

A line-adjacency graph [Shani80] is an essential idea in the algorithm. It is a diagram of the connections between the adjacent segments in the interior of a region. The problem of growing a region is equivalent to traversing its line-adjacency graph in such a way that all nodes in the graph are visited. In practice, traversing the line-adjacency graph is relatively easy. In OOI, the algorithm is implemented. Experiments results showed that this growing technique is effective, not only for artificially patterns, but also for complex natural scene as shown in figure 9.1(g), 9.2(g) and 9.3(g). The excellent characteristics of region growing facilitate the extraction of homogeneous regions in the segmentation process.

5.6 Image Processing Techniques in Medical Domain

Conventional low-pass filtering techniques may be inappropriate for enhancing some medical images, for example the mammograms, the X-ray for the detection of breast tumors, because they tend to blur the image and cause further loss of edges. It is suggested [Ioannidis84] that a non-linear filter, the median filter, is particularly suitable for enhancing medical images due to its ability to provide both noise reduction and edge preservation. The two-dimensional medium filter is defined as follows :

For a window W(i,j) centres at image coordinates (i,j), the medium filtering output is

[Eqn. 5.12] $$\hat{G}_{ij} = median\{G_{\alpha\beta} \ : \ (\alpha,\beta) \ \in \ W(i,j)\}$$

where G_{ij} is the gray level of the pixel at image coordinate (i,j).

Lai [Lai89] proposed a modified median filter, Selective Median Filter (SMF). The two-dimensional SMF is defined as follows:

For a window W(i,j) centres at image coordinates (i,j), the selective median filtering output is

[Eqn. 5.13] $$\hat{G}_{ij} = median\{G_{\alpha\beta} \ : \ (\alpha,\beta) \ \in \ W(i,j) \ \wedge \ |G_{\alpha\beta} - G_{ij}| < T\}$$

where T is a threshold and G_{ij} is the gray level of pixel at image coordinate (i,j). Thus, in computing the median, the set of pixels is restricted to those with a difference in gray level no greater than T. By adjusting T, the amount of edge smearing can be

controlled. This modification of the median filter is related the selective averaging schemes developed for linear filters that have shown good results in improving the edge preserving power of medical images.

In general, to achieve sufficient noise suppression, one needs either a filtering technique allowing a large window size, or the filter has to be applied repeatedly to the image. SMF acts as a low-pass filter in homogeneous areas. As their window size increases, they respond with increasingly narrow pass-bands. Huang [Huang79] has shown that as the window size increases, noise is reduced but distortion is introduced into the actual signal. Therefore, in using a SMF, OOI achieves sufficient noise reduction successfully.

5.7 Concluding Remarks

In this chapter, all the image processing algorithms implemented for enhancement, pixel classification, edge detection and segmentation in OOI are described. The low level vision kernel (LLVK) uses these segmentation techniques to extract the image structure in OOI. Basic properties of each elementary extracted region are calculated and stored as primary attributes. Details are discussed in chapter 6.

CHAPTER 6. PICTORIAL DATA MANAGEMENT IN OOI

6.1 Introduction

After segmentation, basic properties of each elementary region, such as the average gray level, area size, location and some fundamental shape features are calculated and stored as primary attributes together with their region numbers. All the primary attributes extracted are discussed in section 6.2. On the other hand, since well-organized description of the pictorial data is essential for realizing an effective top-down and bottom-up control scheme, a relational database and relational graph are designed to store these spatial data and relations. They are summarized in section 6.3. Since the image primitives, extracted in the segmentation are very simple. In order to recognize objects, various specialized features of regions have to be calculated. A lot of image functions are built inside the control objects, as described in chapter 4. They are used to calculate the complicate attributes, based on a set of primitives. All these functions are described in section 6.5. In OOI, there are two main types of access functions in the image objects. One provides the base level access inside the image objects while the second type is accessible by other image objects. These two functions are discussed in this chapter.

6.2 Description of Basic Properties

As described in chapter 5, elementary regions are analyzed. The properties of regions, boundary segments, vertices, holes and line segments are described in each

descriptive element. Table 6.1 shows the features used for the description as the properties of descriptive elements in OOI. Several functions are designed in OOI to extract attributes from images, such as local contrast function, VH-ratio, Touching_ratio and Surrounding_ratio. Details are described in following sections.

Descriptive Element	Primary Features
Region	Area; Mean gray level; Contour length; Number of holes; Contrast; Minimum bounding rectangle (MBR)
Boundary Segment	Length; Contrast
Vertex	Position
Hole	Contour length
Line Segment	Distance from origin; Length; Position of end points

Table 6.1 Primary Features for Each Descriptive Element

For a region, the following features are described;

(i) **Area**

The area of a set A is simply the number of points in A.

(ii) **Mean Gray Level (MGL)**

The mean gray level of a region A is defined as:

$$MGL = \frac{\sum_{i}^{A} \text{Gray level at point } i}{\text{Area } A}$$

(iii) **Contour Length**

The contour length (perimeter) is the total length of the boundaries which surround the region. In fact, the contour length of a set A can be defined in a number of different ways. Some possible definitions [Rosenfeld82] are:

1. The sum of the lengths of the chain codes of all the borders of A.

2. The sum of arc lengths of all these borders, regarded as 8-curves.

3. The sum of the areas of the borders of A.

In the design, perimeters are calculated using the third method. It is straightforward to compute the contribution of each border point and sum these contributions during a scan of A. Nevertheless, it should be pointed out that the perimeter of a digitized region often grows exponentially as the digitization becomes finer while the area, on the other hand, tends to be a finite limit.

(iv) **Contrast**

The local contrast function C(i,j) for a pixel p(i,j) in an image is defined as :

$$C(i,j) = \frac{|P_c(i,j) - P_s(i,j)|}{\max\{P_c(i,j), P_s(i,j)\}}$$

where $P_c(i,j)$ and $P_s(i,j)$ are the average gray-level values of the pixels corresponding to the "centre" and the "surrounding" regions, respectively, with the centre at point p(i,j). The contrast function thus defines serves as the basis of the intralevel link property. For example, if the region in the image is uniform, the contrast values in that region will be quite low. The contrast value will be much higher in the presence of an edge or at the boundaries of a sharp feature. Textured regions will have higher contrast values accordingly.

(v) **Minimum Bounding Rectangle (MBR)**

The location of an elementary region in an image is represented by two coordinate pairs, (LX,LY), (RX,RY), called minimal bounding rectangle (MBR), as shown in fig.6.1 [Nagao80]. When one wants to get the two-dimensional image of an elementary region, one needs only to scan within the rectangular area specified by (LX,LY), (RX,RY), which save a great deal of processing time.

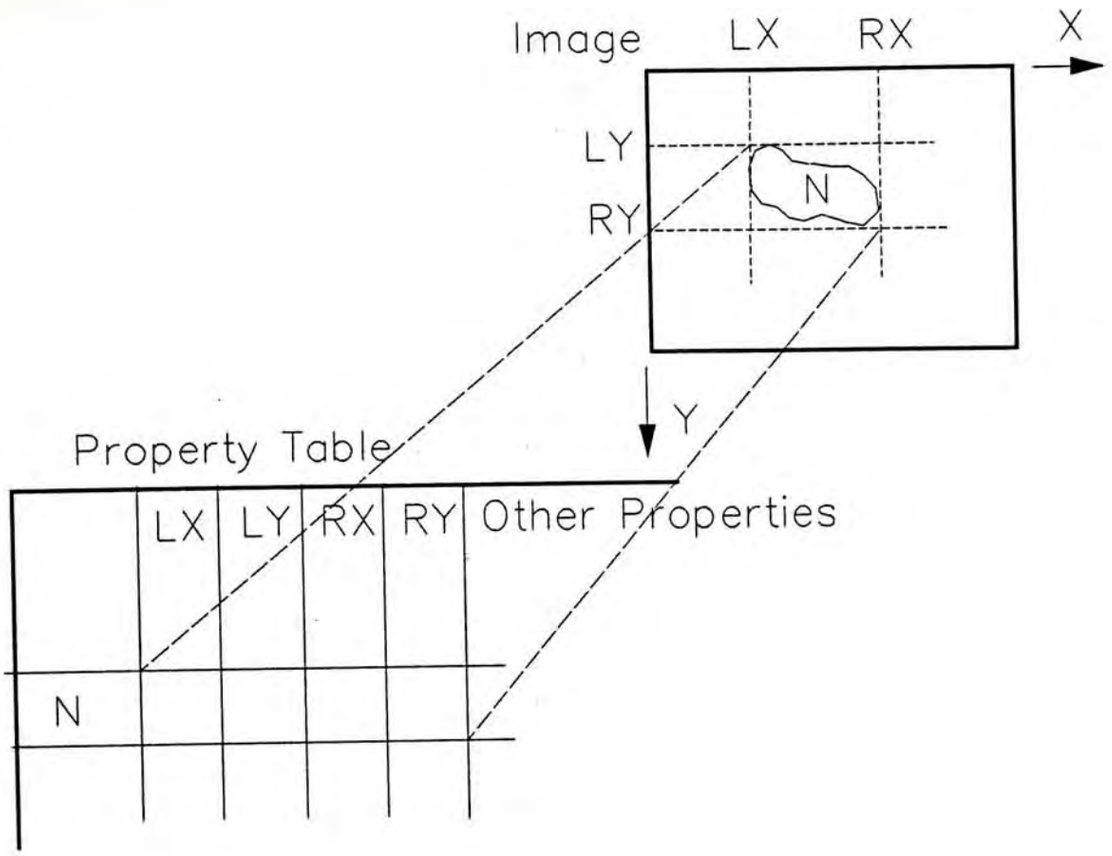

Fig. 6.1 Minimum Bounding Rectangle and the Property Table of an Image

On the other hand, in calculating the shape features of an elementary region, minimal enclosing rectangle (MER) is used. FIT, ELONG, and DIREC can be deducted from it. These features are calculated by the following process :

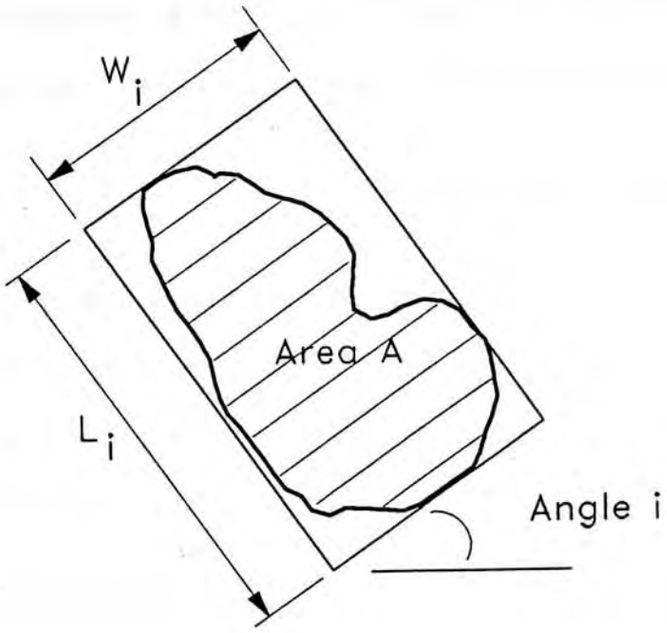

Fig. 6.2 Minimal Enclosing Rectangle

(a) The minimal enclosing rectangle (MER) of the region is shown in figure 6.2. MER is defined as the one whose F = (area of the region)/(area of the encasing rectangle whose sides are parallel with the coordinate axes) takes the maximum when the enclosing rectangle is rotated through 90° at intervals. FIT is defined as the maximum of F_i where

$$F_i = \frac{L_i \times W_i}{A}$$

$$i = 0°, 10°, \ldots\ldots, 80°$$

where A is the area of region.

(b) ELONG, the elongation of the region, is defined by ELONG = L/W, where L and W denote the length of the long and short sides of the MER.

(c) DIREC denotes the direction of the long side of the MER.

All subsequent analysis problems take each elementary region as an primitive image patch or a part of patch to be recognized. They consult the property table, as shown in figure 6.1, in order to examine its properties or deduce the results of the analysis. Beside of using the basic properties of the primitive image patch in deduction, spatial relations are also considered and they are discussed in the following section.

6.3 Description of Relations

In order to grasp the topological relations of an input image, or answer the queries concerning the content of the image, its features and structure have to be input by the user. This is essential in using the model based approach. A relational database and relational graph are designed in OOI to store these spatial relations and they are discussed in the following sections.

6.3.1 Relational Database of Pictorial Data

Table 6.2 illustrates an example of relation called PHYSICAL_PATCH of degree 8 defining each patch with 8 fields:

(i) ID(patch identity number),

(ii) S_X (starting X-coordinate of the patch),

(iii) S_Y (starting X-coordinate of the patch),

(iv) AREA,

(v) MGL(Mean Gray Level),

(vi) RHT-OF (RIGHT_OF),

(vii) LFT-OF (LEFT-OF) and

(viii) PART-OF.

The domain PYSICAL_PATCH, for example, is the set of all valid physical patch extracted from the Low Level Vision Kernel (LLVK). Each row of the table represents an n-tuple of the relation. In this tabular representation of a relation, the following properties, which derive from the definition of a relation, should be observed :

1. no two rows are identical,
2. the ordering of the rows in not significant, and
3. the ordering of the columns is significant.

Relation: **PHYSICAL_PATCH**

ID	(S_X,S_Y)	AREA	MGL	RGH-OF	LFT-OF	PART-OF
1	(121,34)	23	11	2		NUCLEUS
2	(11,32)	11	13		1,3	BLOOD_CELL
3	(99,21)	103	165	2		CYTOPLASM
4	(112,78)	145	34	11		CYTOPLASM
5	(63,56)	1003	74		31,35	NUCLEUS
6	(2,67)	234	234	32		BLOOD_CELL
7	(90, 234)	435	246	18	15, 34	NUCLEUS
8	(345,65)	235	12		22	CYTOPLASM

Table 6.2 Physical Patch Relation

Many spatial data structures are developed and used to describe pictorial patterns in pattern recognition. A specific problem can be solved more efficiently by using specially designed data structures than by using general purpose database models. However, the database approach is more convenient for data shared by different users and applications. The approaches to database representation can be categorized as hierarchical, network and relational. The relational database is chosen because it employs simple tabular structures that are conceptually simple to understand and thus easy to access.

6.3.2 Relational Graphs and Relational Databases

The concept of relational graphs is used in syntactic pattern recognition to represent the structural information of a given patterns [Ku74]. A relational graph is a labelled directed graph consisting of a set of labelled nodes and several sets of labelled edges, which are ordered pairs of nodes. The nodes in a relational graph represent subpatterns and image pattern primitives, and the edges represent the relations between subpatterns and primitives. The following definitions and example give the detailed representation of the image model in OOI:

Definition :

A relational graph over the graph alphabet $V = V_N \cup V_E$ is a 4-tuple $\omega = (N, E, \mu, \epsilon)$ where

(1) N is a finite, nonempty set of nodes;

(2) $E \subseteq N \times N$ is a set of ordered pairs of nodes in N, called edges:

(3) V_N is a finite nonempty set of node labels (primitive or subpattern descriptions);

(4) V_E is a set of edge labels (relation descriptions);

(5) $\mu : N \rightarrow 2^{V_N}$ is a function called node interpreter, where 2^{V_N} is the power set of V_N, the set of all the subsets of V_N; and

(6) $\epsilon : E \rightarrow 2^{V_E}$ is a function called edge interpreter.

For example, given an image *I*, as shown in figure 6.3, a relational graph $\omega = (N, E, \mu, \epsilon)$ over graph alphabet $V = V_N \cup V_E$ is represented as in figure 6.4 where

$N = \{ I, U, B, C, x, y, z \}$

$E = \{$ (U, I), (B, I), (C, I), (x, U), (y, B), (z, C),
(B, C), (B, U), (C, B), (U, B), (U, C)}

V_N = { *Image, Nucleus, Blood-cell, Cytoplasm, Line-segment* }

V_E = { *part-of, left-of, right-of, inside-of* }

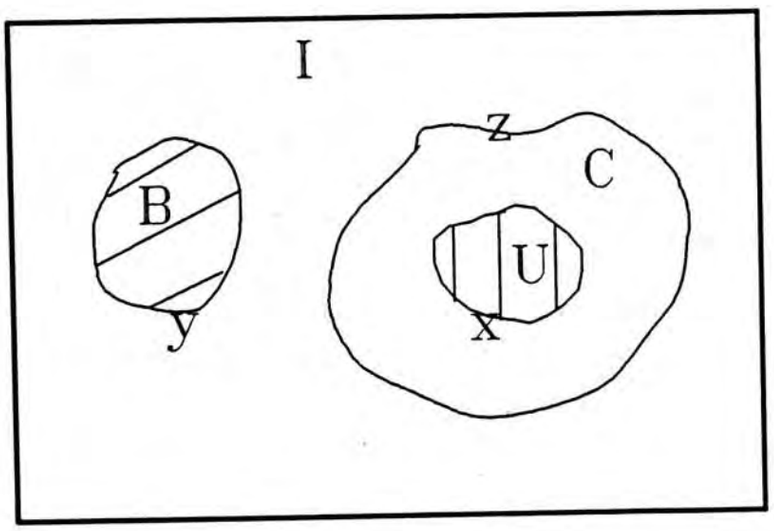

Figure 6.3 An Label Image

$\mu(I)$ = { Image }

$\mu(U)$ = { Nucleus }

$\mu(B)$ = { Blood-cell }

$\mu(C)$ = { Cytoplasm }

$\mu(x)$ = { Line-segment }

$\mu(y)$ = { Line-segment }

$\mu(z)$ = { Line-segment }

$\in(U, I)$ = { part-of }

$\in(B, I)$ = { part-of }

$\in(C, I)$ = { part-of }

$\in(x, U)$ = { part-of }

$\in(y, B)$ = { part-of }

$\in(z, C)$ = { part-of }

$\in(B, C)$ = { left-of }

$\in(B, U)$ = { left-of }

$\in(C, B)$ = { right-of }

$\in(U, B)$ = { right-of }

$\in(U, C)$ = { inside-of }

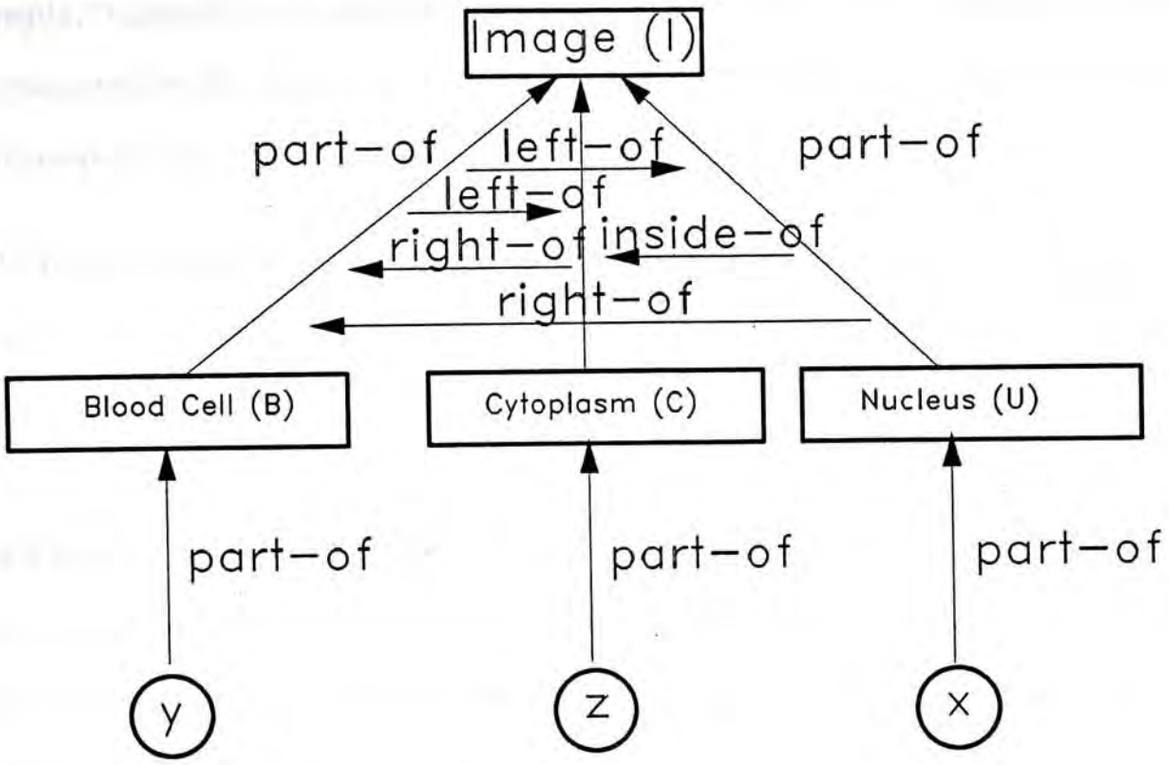

Figure 6.4 A Relational Graph of Image I

In this example, pictorial entities such as line segments, nucleus, blood-cell, cytoplasm and the whole picture are represented as nodes, and the relations between nodes are described in terms of structural descriptions such as "part-of", "left-of", "right-of" and "inside-of" as shown in figure 6.4.

The relations of the relational databases can be considered as data structures for representing relational graphs. In fact, for a relational graph such as the one in figure 6.4, each set of edges corresponds to a binary relation of the relational databases. An edge label corresponds to the name of a relation, and each tuple of this relation corresponds to a pair of nodes connected by the edge with this label.

For example, "right-of" is an element of the set of edge labels, V_E. A pair of nodes (U, B) connected by the edge "right-of" in figure 6.4 correspond to is a tuple (U, C) in the relation of "rgt-of" in table 6.2

In fact, all the topological relations in OOI are expressed by the concept of relational graphs. Users are requested to input these relations so as to support an effective deduction in the expert system.

6.4 Access Functions in Image Objects

There are two main types of access functions in the image objects. The first type provides the base level access within image objects. This type functions are all stored in the private part of an object that probably would not be used by other objects. The second type are accessible by other image objects. It includes the user accessible functions, described in section 6.4.2 and image functions described in section 6.5.

6.4.1 Basic Access Functions

There are three basic access functions for an image object as follows:

i. *Lookup*,

ii. *Store* and

iii. *Remove*.

The function "*Lookup*" takes two arguments, a slot name and an object name. It is used to find out whether the slot is defined in the object, or its antecedents. The function "*Store*" supports the basic storing facility and takes three arguments, an

object name, a slot name and the value to be stored. The function "*Remove*" takes one to three arguments, the first being an object, the second a slot name and the third a value. It is implemented as a derived function in C++, which allows functions with a variable number of parameters. If only the object name is given, the object is removed from the system. If an object name and a slot name are given, then the slot is removed from the object, along with all its values. If a value is also given, then just that value is removed.

They are all stored in the private part of an object that probably would not normally be used by an end user of the system.

6.4.2 User Accessible Functions in Objects

User accessible functions are functions that are all stored in the public parts of the image objects and can be inherited to their descendants. Associated with each slot attribute, either explicitly or implicitly, there are three corresponding access-slots functions:

i. *To_Get_Value*,

ii. *To_Put_Value* and

iii. *To_Delete_Value*.

They are similar to the basic access functions, as described in section 6.4.1, except that they can be accessed by an user.

6.5 Image Functions

The extractions of attributes of an image must be supported by high-speed processing when they become necessary in the analysis process. One possible scheme is to calculate all the required features beforehand. But this is not practical for the following reasons:

(1) It is wasteful to calculate features which may not be used in the analysis process.

(2) A large amount of storage would be necessary to store all the calculated features.

To solve the problem, features used in the image analysis can be divided into two types: primary features and secondary features. The calculation of a primary feature is dealt with directly in the Low Level Vision Kernel (LLVK), such as area, and generally is time consuming. On the other hand, the secondary features, such as compactness of a region, can be calculated quickly from a set of primary features. Based on such considerations, only primary features are entered into the High Level Vision Kernel (HLVK). The secondary features will be calculated from the primary features of the descriptive elements and their relationships when needed in the analysis process. Image functions are used to achieve this objective. They can be classified into three main categories as follows:-

6.5.1 Unary Image Operations

These operations concentrate on the local properties of image objects. They take one image object each time and return a value after calculation. For example,

the metric operation, *Find_Centre*, uses the primary features, dimensions of minimum bounding rectangle (MBR), to find the centroid of the area of the object. Some other attributes are listed in tables 6.3.

Features of a region	Compactness; VH-ratio; Diameter; Intensity; Saturation.
Features between two regions	Contrast of border; Spatial relationships; Touching-ratio; Surrounding-ratio

Table 6.3 Secondary features functions derived from primary features

The details of the features of a region are given as follows:

(i) **Compactness**

It is a topological attribute in the sense that it does not depend on the size or shape. This can be calculated from the area and the contour length of a region. Compactness is defined by

$$Compactness\ \ Ratio\ \ (C.R.) = \frac{(Area\ \ of\ \ region)}{(Perimeter^2)}$$

(ii) **VH-Ratio**

Crude shapes of a region such as vertically-long or horizontally-long can be defined based on the VH-ratio, which is computed using

$$VH-ratio = \frac{LY-RY}{RX-LX}$$

where LX, RX, LY and RY are the coordinates of the MBR, defined in section 6.2.

(iii) **Diameter**

Diameter is defined as the greatest extent of an area A in any direction. Readily, this is equal to the greatest distance between any two points of A. The extents of A in various directions, or the distances between points of A, are sometimes useful as descriptive properties. The diameter of area A is defined as L_i, as shown in figure 6.2.

(iv) **Intensity and Saturation**

If a color image is grabbed, intensity and saturation of the image can be defined. These can be calculated from the mean intensities of (Red, Green, Blue) of the region.

$$r = \frac{R}{(R+G+B)}, \qquad g = \frac{G}{(R+G+B)}, \qquad b = \frac{B}{(R+G+B)}$$

$$\text{Intensity} = \frac{(R+G+B)}{3},$$

Saturation = 1 - 3 X min(r, g, b)

6.5.2 Binary Relation Operations

Binary relation operations take two image objects each time and return a value after calculation. The names of the binary relation functions can be found in table 6.3.

(i) **Contrast of Border**

This can be computed by averaging the contrast of the boundary segments included in the intersection of the contours of the two regions.

(ii) **Spatial Relationship Operations**

They are employed to retrieve picture objects satisfying certain spatial relationships or to test spatial relationships among picture objects. For example, *AKO(a kind of), left-of, right-of, is_west, is_east, is_north, is_south, is_inside, is_touch, is_through, is_between.* etc. These commands are used to find image objects to the west (east, north, south...) of an image object and store the result in a slot.

(iii) **Touching_Ratio**

Touching_ratio of two regions, region1 to region2, is calculated from the length of the border between the two regions and the contour length of region1.

$$Touching\_ratio_{region1} = \frac{Border\ \ length}{Contour\ \ length\ \ of\ \ region1}$$

(iv) **Surrounding_Ratio**

The degree of a region being surrounded by another region is reflected by the surrounding_ratio. The surrounding_ratio of region1 and region2 is computed as the ratio of the overlapped area of the MBRs of the two regions to the whole area of the MBR of region1.

$$Surrounding\_ratio_{region1} = \frac{Overlapped\ \ area\ \ of\ \ MBR}{Whole\ \ area\ \ of\ \ region1's\ \ MBR}$$

6.5.3 Update Operations

Commands, such as *merging, union* and *intersection,* are useful when two or more different image objects of the same geographic region are to be integrated into a new image object. For example, in region image analysis, it is often necessary to merge several regions into one. In such a case, a new region should be calculated from the features of the old regions without referring back to the image arrays. In OOI, this can be performed as follows.

i. The feature f of the new region can be calculated by the weighted average

$$f = \frac{\sum_{i=1}^{N} f_i \otimes A_i}{\sum_{i=1}^{N} A_i}$$

where A_i and f_i denote, respectively, the area and the feature of the i-th old region and $\otimes$ is a generic binary operator.

ii. Minimum bounding rectangle (MBR) -- The MBR(Xmin, Xmax, Ymin, Ymax) of the new region can be obtained as the MBR of the MBRs of the old regions.

$Xmin = minimum \{Xmin_i\};$

$Xmax = maximum \{Xmax_i\};$

$Ymin = minimum \{Ymin_i\};$

$Ymax = maximum \{Ymax_i\};$

where $\{Xmin_i, Xmax_i, Ymin_i, Ymax_i\}$ denotes the MBR of the i-th old region.

6.6 Concluding Remarks

The image primitives, extracted in the segmentation are very simple. In order to recognize objects, various specialized designed features of regions have to be calculated. As described in chapter 4, image functions are built inside the control objects. They are used to calculate the complicate attributes, based on a set of primitives. In this chapter, an overview of all image functions in OOI are presented. Several functions are designed in OOI to extract attributes from images, such as local contrast function, VH-ratio, Touching_ratio and Surrounding_ratio. Moreover, a concept of relational graph is discussed in order to represent the spatial relations between image objects.

CHAPTER 7. KNOWLEDGE MANAGEMENT

7.1 Introduction

Initially, each image patch is in a form of an instance of a physical_image class. They are stored in a linked list. Conceptually, each production rule in a domain knowledge base (DKB) checks the status of all instances in the linked lists, and executes the associated action when its condition is satisfied. The details of the rule structure are discussed in section 7.2. The successful candidates, which are called the keypatches, are stored in another linked list for another phase of investigation. In the second phase, a model driven approach is used for the remaining minor patches. Merging is done so as to group the minor patches into larger patches. Moreover, matching is performed in an object-oriented approach at this level, as summarized in section 7.3. The larger patches are then analysed by the rules stored in the DKB. Methods which are stored in the control objects, described in chapter 4, are also used for merging, creating and labelling the patches. Fuzzy reasoning, described in section 7.4, is employed in OOI. All the control schemes are carried out until most of the image patches are labelled.

7.2 Knowledge in A Domain Knowledge Base

The knowledge block for an object holds the description of the properties which must be satisfied by a region corresponding to that object. It also holds the description of the relations which must be satisfied between the regions corresponding to objects. In OOI, knowledge stored in the Domain Knowledge Base (DKB) is in form of rules.

Most of the rules in the DKB are used in the bottom-up reasoning in analysing the image patches. Hypothesis is generated and a certainty factor is used to determine the reliability of the deduction. They are discussed in the following sections.

7.2.1 Structure of Rules

In OOI, rules are used to represent expert knowledge to describe relations among objects and are handled in a flexible way. They are all stored in ASCII format and users can edit any rules by using a text editor. An external full-screen editor is incorporated in OOI for the user to edit all the rules.

The format of a rule is shown as follows:

[RULE <rule code>
IF (Ante_part)
THEN Conseq_part
] Certainty is <Certainty factor>

<Ante_part> ::= <Proposition> | <Prop_list>
<Conseq_part> ::= <Attribute> <Operator> <Value>
<Prop_list> ::= <Proposition> <Connector> <Prop_list>
<Proposition> ::= <Attribute> <Operator> <Value> <Donation_factor>
<Operator> ::= IS | IS_NOT | <Num_operator>
<Num_operator> ::= > | > = | < | < = | = | < >
<Attribute> ::= *<Letter> <Letters>

<Value> ::= <Attribute> | <Numbers>

<Certainty_factor> ::= <Numbers>

<Donation_factor> ::= <Numbers> | <Fuzzy number> | nil

<Numbers> ::= <Number> <Numbers> | nil

<Letters> ::= <Letter> <Letters> | <Number> <Letters> | nil

<Connector> ::= AND | OR

<Number> ::= 0 | 1 | 2 | 3 | ...

<Letter> ::= A | B | C | D | ...

The antecedent part of a rule may consist of multiple propositions connected by AND/OR and the antecedence of each rule is stored in the memory as a AND-OR graph. An AND-OR graph is useful for representing the solution of problems that can be solving by decomposing them into a set of smaller problems, all of which must then be solved. The decomposition, or reduction, generates arcs which are called AND arcs. An AND arc may point to any number of successor nodes, all of which must be solved in order for the arc to point to a solution. An example of the structure of a AND-OR graph used in OOI is shown in figure 7.1

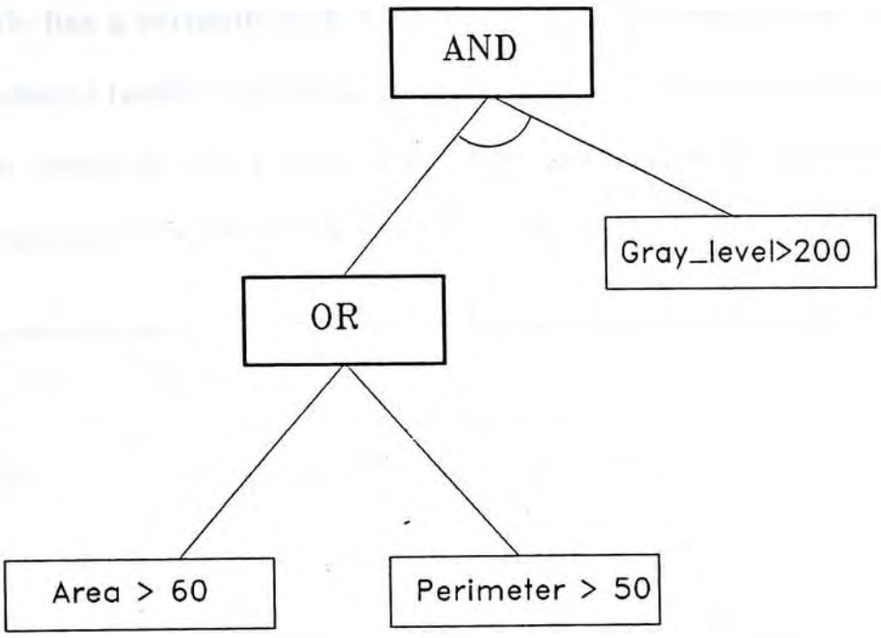

Figure 7.1 AND-OR Graph Used in OOI

The AND-OR graph in figure 7.1 is equivalent to the antecedence of the following rule:

[RULE a1

IF (((area > 60) OR (perimeter > 50))

AND (gray_level > 200))

THEN patch is nucleus

] Certainty is 0.9

Each rule has a certainty factor attached. The certainty factor is a numeric value. If the certainty factor is omitted, absolute certainty is assumed (i.e. Certainty is 1). Rules are stored in rule nodes, which are declared as a structure of C. The information contained in a rule node is as follows:

Rule_code	Rule code of a rule
Ante_part	A pointer pointing to the AND-OR tree of the antecedent proposition
Conseq_part	The structure of the consequent proposition
CF	The certainty of that rule
Active	A flag which decides whether the rule is active in the current context

Table 7.1 Information Contained in Rule Node

Each rule is uniquely identified by the rule code. Ante_part is a pointer to the AND-OR tree. The structure of a node of the AND-OR tree is as follows:

Attr_name	Name of the attribute in the node
Conn	Represents the type of the node (AND, OR, PROPOSITION)
Attr_value	The value of the attribute in the proposition
Left	Points to the left son of the node
Right	Points to the right son of the node
DF	Donation factor of the proposition

Table 7.2 Structure of the Node of the AND-OR Tree

The field *Conn* is used to indicate the current type of the node. If its value is PROPOSITION, indicating that a node contains a proposition, Attr_name and Attr_value will be filled with values. Donation factor, discussed in section 7.4, is used optionally and the default values is 1.0. However, if the value stored in *Conn* is either AND or OR, it means that the current node is a logical connective node which does not contain a proposition. The fields *Left* and *Right* will be used to point to the nodes containing the left and right operands respectively.

7.2.2 Hypothesis Generation

Each rule for the bottom-up process has a fuzzy predicate which describes properties or relations between objects. It also has a certainty factor which indicates the uncertainty of the knowledge it relies on. When a rule is applied to a region to

check whether the region can be labelled as an object, it examines the pictorial features of the region, described in chapter 6, and produces a fuzzy certainty factor which indicates the degree of satisfaction of the property. The predicates can also represent other knowledge, such as knowledge for inferences based on the background or history of patients, which are also input in rule form. In order to generate a reliable deduction, the Scheme Scheduler object, SS_object, activates the rule base to examine every region. For every combination of rules and regions, the SS_object receive a fuzzy certainty factor. Based on the set of fuzzy certainty factors, the SS_object assigns an object label to each of the region and computes the degree of correctness. The mathematical model in the inference is described in section 7.4.

In a production system architecture, it is an important and difficult problem to resolve the conflicts between the rules which work concurrently and independently to label or modify the regions. In OOI, each production rule in the DKB tries to assign a label to a patch whenever the patch satisfies the condition attached to the rule. It is usually the case that a patch simultaneously satisfies the conditions of several production rules whose associated actions try to assign different labels to the patch. The problem of conflict resolution can be solved: Whenever a patch is interpreted by executing an action in a rule, every rule which are giving a different interpretations to the patch are declared inconsistent if the certainty factors are smaller. i.e. only the hypothesis with the largest certainty factor is used.

7.2.3 Inference Engine

The inference engine of OOI is a recursive one. In OOI, backward chaining is used, not only because of search economy but also for the focused attention it gives to the problem, especially for the diagnosis problems. In the process of diagnosis under backward chaining, a reasoning tree is built from the goal to the leaves. Figure 7.2 shows a sample reasoning tree. The reasoning tree contains the context of the current consultation.

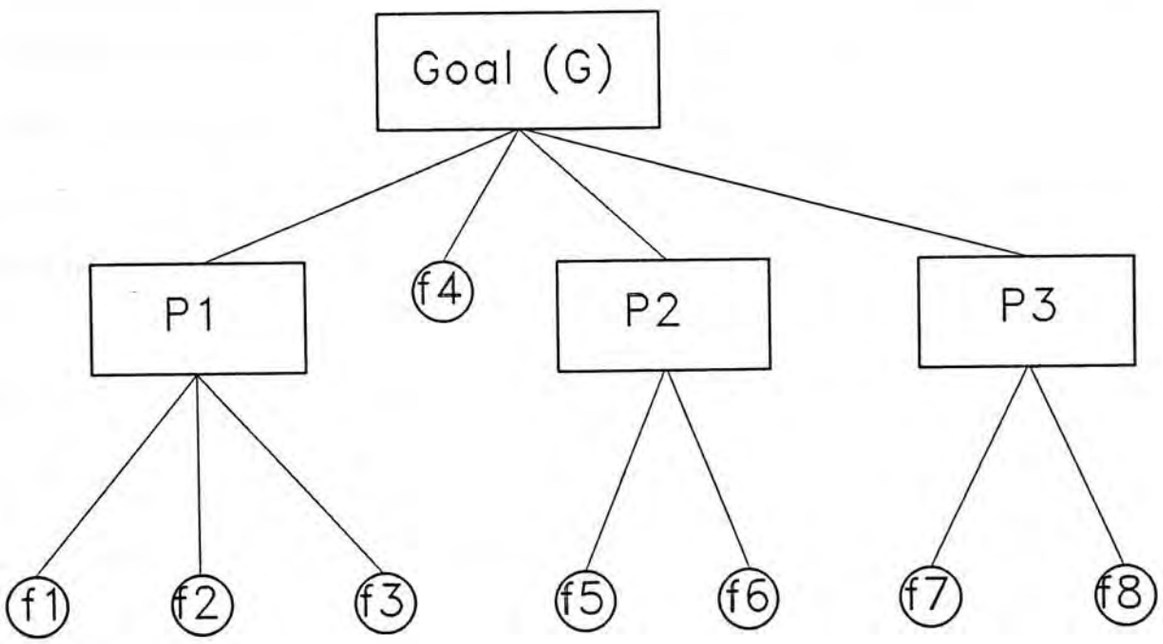

Figure 7.2 A Sample Reasoning Tree

For locating the possible rules efficiently, rules are indexed in the *Rule_used* and *Rule_updated* fields. Moreover, the chaining process defines a depth first, left-to

right traversal of the tree. For instance, in the above reasoning tree, the system will first evaluate the value of P1. This causes in the system to search for the values, f1, f2 and f3, from the image objects sequentially. Then the fact f4 is tested and evidence combination will be performed to get the certainty of G. The process will continue until all the possible paths to evaluate the goal (G) have been considered and the whole reasoning tree has been built traversed.

7.3 Model Based Reasoning in OOI

Reasoning that incorporates a model is called model-based reasoning. In order to build model-based systems, one needs tools for representing objects, the "things" in the world, and processes, the behavior of things in the world. In OOI, classes and instances are used to represent things while object-oriented programs and rules are used to represent processes.

7.3.1 Merging and Labelling

In OOI, rules are built inside the objects implicitly. They are used mainly in merging and labelling the image patches into larger patches at each level. The action part of the objects includes a list of actions which manipulate the scene description. The manipulation is a combination of several methods.

(i) Physical Patch Level Methods

A patch is assigned with a label.

(ii) Region Level Methods

The patch is merged to form regions. If the description of the region has not been created yet, a new region is created.

(iii) Entity Level Methods

When a new region is created at the region-level operation, the region is associated with a meaningful object entity. Pointers are used to link up regions belonging to that object as shown in figure 7.3. If the description of the entity has not been created yet, a new entity is created and the region is associated with it.

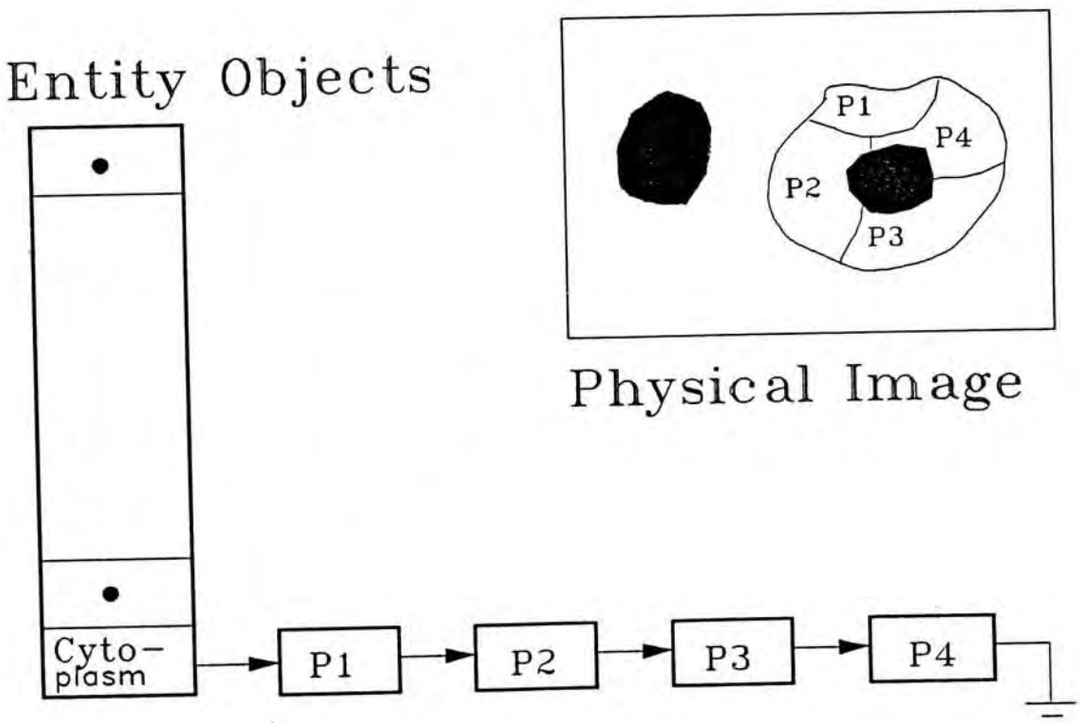

Figure 7.3 Pointers Used in Linking Up Regions

7.3.2 Vision Model

Besides labelling and merging the image patches into larger patches, identification is also made between image patches and objects at the three levels of the image objects hierarchy, described in section 3.5.2. A mapping from image patches to the object model is made, making use of the information gained through the acquisition module. This can be viewed as predication of image features. The model of the objects are input by users. The model representation scheme used in OOI is able to represent the class of objects which the system is required to recognize. Table 7.3 shows the generalized model of nucleus.

Name of generalized model		: nucleus
Derived class name		: nucleus
Area	(small, large)	: small
Mean gray level	(low, high)	: very low
Perimeter	(short, long)	: short
Background	(low, high)	: medium
Compactness	(low, high)	: high
Topological Properties		: (inside cytoplasm)

Table 7.3 Generalized Model of Nucleus

Fuzzy values are accepted and interpreted during reasoning. Relational operators, such as inside and left-of, are used to describe the spatial relations. Details

of the relational operators can be found in chapter 6. Based on this generalized model, an abnormal nucleus can be defined in an object-oriented approach, as shown in table 7.4.

Name of generalized model		: nucleus
Derived class name		: abnormal nucleus
Area	(small, large)	: medium
Mean gray level	(low, high)	:
Perimeter	(short, long)	: medium
Background	(low, high)	:
Compactness	(low, high)	: very high
Topological Properties		:

Table 7.4 Derived Model of Nucleus

All the empty slots in the derived model will inherit its attribute values from its super model. A matching process producing certainties is performed in top-down fashion. Correct matches imply the consistency of the detailed model at this stage.

7.4 Fuzzy Reasoning

As the system performs reasoning along the top-down and bottom-up hierarchies, some intermediate results are generated from the analysis. In order to find the promising solution and achieve the most effective reasoning, each antecedent part of a rule is assigned a donation factor (DF) according to the importance and the faith

in the feature used for the recognition of the specific object.

For example:

IF (area is large [DF_1 = 0.3]) AND

(gray_level is low [DF_2 = 0.7])

THEN patch is nucleus

Certainty is 0.7

where $\sum_{i=1}^{N} DF_i = 1$

The rule above shows that in order to decide whether a patch is a nucleus, gray_level is more important than area. Since a donation factor is relative quantity, the sum of the donation factor is normalized to be one.

The donation factor may be a crisp number or a fuzzy number. The definition of a fuzzy number is given in the Zadeh's papers [Zadeh65, Zadeh83]. If a donation factor is a fuzzy number, it can be conveniently represented as a 4-tuple (W_1, W_2, W_3, W_4), and $\forall W_i \in [0, 1]$. W_2 and W_3 denote the interval in which the membership degrees are equal to 1. W_1 and W_4 denote the interval in which the membership degrees are non-zero. For example, "around 0.7" would means (0.5, 0.7, 0.7, 0.9).

When a rule is checked and executed, a similarity confidence value (SCV) is computed. The SCV is based on the fuzzy certainty value of the feature, used in the antecedent part, and the donation factor (DF) of the antecedent part. The formulae used to calculate the SCV of a conclusion is as follows:

Rule : IF A_1 [DF_1] and
A_2 [DF_2]
THEN C is V [FN_R]

FACT : A_1^* (FN_1)
A_2^* (FN_2)
CONCLUSION C is V^*

$$SCV = \left(\sum_{i=1}^{N} DF_i \times FN_i \right) \times FN_R$$

where N = 1, 2
DF_1, DF_2 are donation factors of propositions A_1,A_2 respectively
FN_1, FN_2 are fuzzy certainty factors of facts A_1^*,A_2^* respectively
FN_R is the fuzzy certainty factor of the rule

A SCV is computed for each of the candidate rules for each feature used in the high level matching. Evidence combination is performed in OOI.

7.5 Concluding Remarks

In this chapter, the details of the rule structure in Domain Knowledge Base (DKB) are presented. In the first phase of diagnosis, most of the rules in DKB are used in bottom-up reasoning of analysing the image patches. In the second phase, matching is performed during model driven approach. The generalized model in object-oriented approach has been shown. A similarity confidence value (SCV) is introduced in the fuzzy reasoning in OOI.

CHAPTER 8. KNOWLEDGE ACQUISITION AND USER INTERFACES

8.1 Introduction

Knowledge acquisition is the transfer and transformation of problem-solving expertise from some knowledge sources to a knowledge base. Knowledge acquisition is a bottleneck in the construction of expert systems. In the narrowest sense, it involves obtaining knowledge that goes into an expert system. In a broader sense, it entails the entire knowledge engineering process. The knowledge engineer's job is to act as a go-between to help an expert to build an expert system. It includes interviewing an expert, developing a knowledge map, a diagram which shows the inter-relationships of knowledge, and encoding the knowledge into a knowledge base, and then testing and refining the knowledge with the expert and the users until the system matures. This knowledge engineering process can be summarized in figure 8.1.

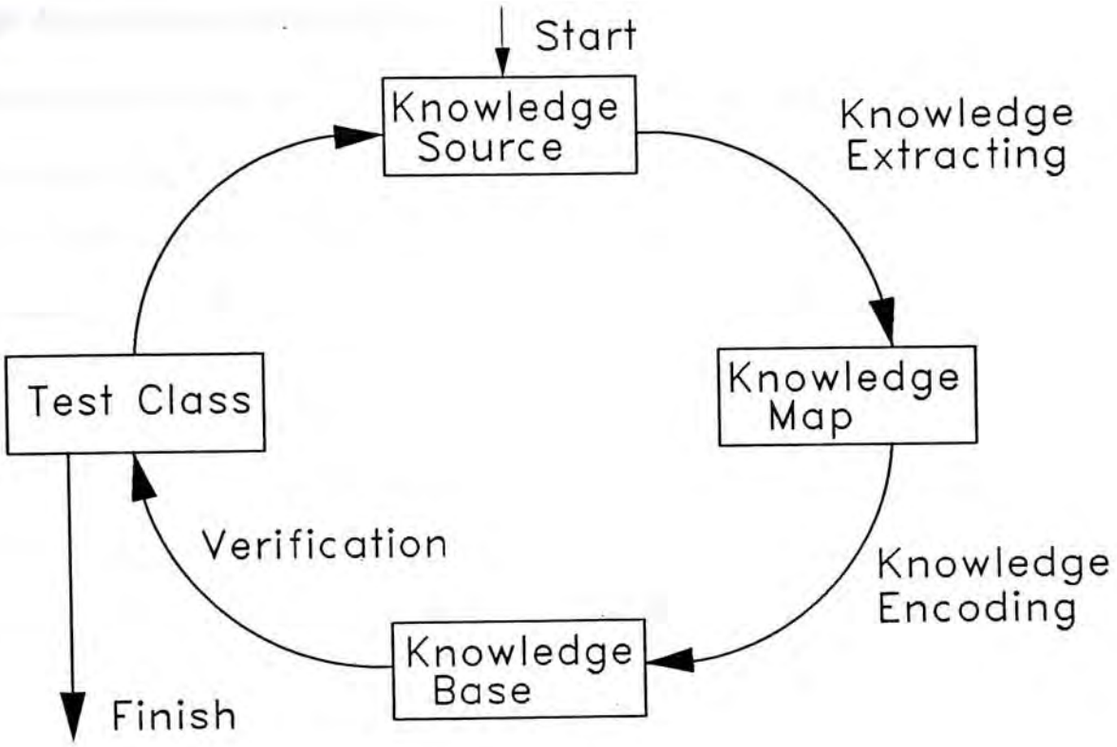

Figure 8.1 The Knowledge Engineering Process

One of the most difficult aspects of the knowledge engineer's task is helping the expert to structure the domain knowledge, to identify and formalize the domain concepts. However, in this chapter, a narrower discussion on the knowledge acquisition is presented. It focuses on how the knowledge is encoded and input into the shell. A knowledge acquisition module is used to communicate with the knowledge engineers for the acquisition of domain knowledge. This module is discussed in section 8.2. A pop-up window is used as the user interface in the shell and they are discussed in section 8.3.

8.2 Knowledge Acquisition Subsystem

The acquisition module in OOI mainly consists of three sub-modules:

(i) Rule Management Module

(ii) Attribute Management Module

(iii) Model Management Module

They are discussed as follows:

8.2.1 Rule Management Module

Knowledge is elicited from the domain experts by the knowledge engineer and encoded by the rule management module in a rule form. A text editor is used for inputting the rule. The format of the rule is shown in chapter 7. These rules are stored in rule node, which are declared as a structure in C. The rule node is a most complicated data structure. Each rule is uniquely identified by the rule code. They are organized in the knowledge base by a hash table, as shown in Figure 8.2.

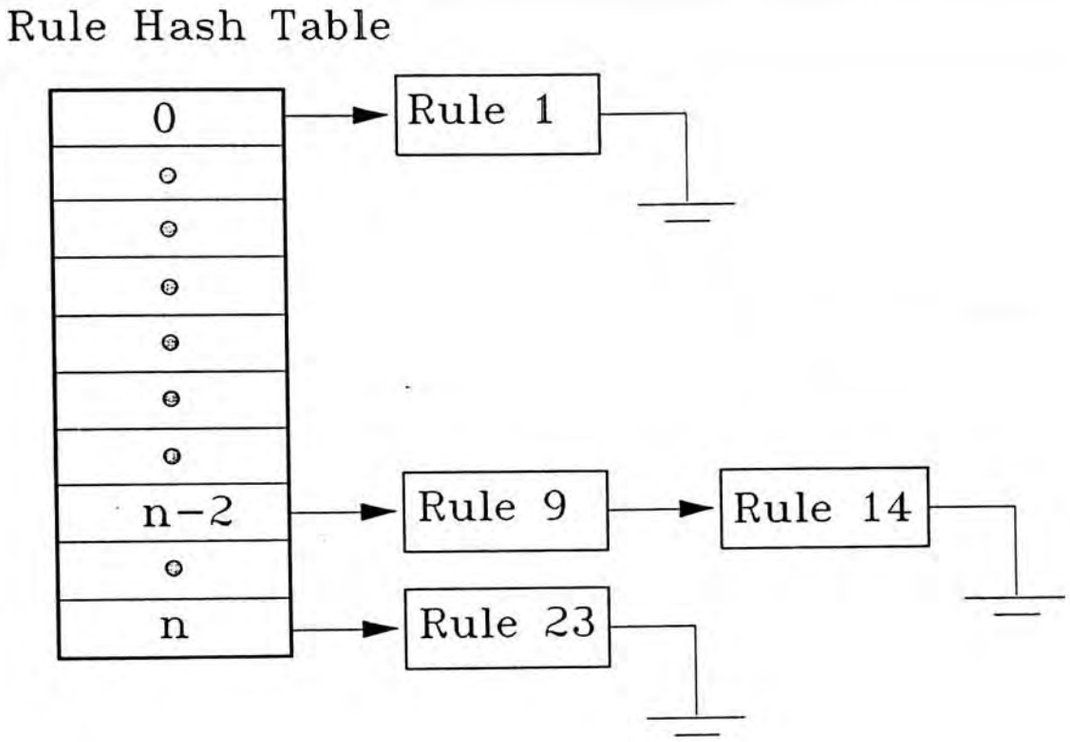

Figure 8.2 Rules Hash Table

The rule management module is responsible for the manipulation of all the rules in the knowledge base. It provides some operations, which enable users to create, edit, view and delete rules in its domain knowledge base.

8.2.2 Attribute Management Module

In order to build up the rule structure, all the attributes that are used inside the rules should be defined beforehand and it is done by the attribute management

module. The module enables users to define, modify, list and display the attributes. All the rule attributes that can be defined in OOI can be classified into following five categories:

(i) Numeric Attribute

Most of the image attributes belong to numeric attributes, for example, area, perimeter and gray level. To declare a numeric attribute, user are requested to input the lower and upper limits of the attribute. It is done to counter-check the error that would be introduced in passing the attribute from the LLVK to HLVK. In the following example, X_coordinate_upper_corner_MBR must first be declared as a numeric attribute before the rule can be formed.

e.g. IF (X_coordnate_upper_left_corner_MBR < 10)
THEN nucleus is False

(ii) Logical Attribute

Logical attributes are of Boolean type which can have the values YES or NO and True or False only. In the example above, nucleus is declared as a logical attribute since it is designed to have a Boolean value only.

(iii) Multiple-Valued Attribute

Multiple-valued attributes can be enumerated as a finite list of pre-defined elements. It can have several values simultaneously. An obvious

example can be found in the labelling the keypatches. The label may have multiple values containing different certainty factors corresponding to different hypotheses toward the keypatch.

e.g.

Rule 001

IF (area > 100) and (nucleus_adjacent is True)

THEN keypatch is cytoplasm

Rule 002

IF (gray_level < 50) or (garbage is False)

THEN keypatch is nucleus

In this example, keypatch is defined as a multiple-value attribute since it has values "cytoplasm" and "nucleus" simultaneously.

(iv) Fuzzy Attribute

Fuzzy attributes have fuzzy sets as their values. To declare a fuzzy attribute, user are requested to define the fuzzy set corresponding to the fuzzy attributes. The most common fuzzy set is (high, medium, low). Some of the fuzzy attributes and their corresponding fuzzy set are shown in table 8.1.

Fuzzy Type	Fuzzy Sets
Length	long medium short
Compactness	high average low
Region	large normal small

Table 8.1 Fuzzy Type and Fuzzy Sets

(v) Function Attribute

In order to enhance the ability of the shell, linkage with other external image subroutines are necessary. Function attribute are designed to accomplish this requirement. They will activate a external function and return a value to OOI.

For example :

Rule 101

IF (find_hole is true)

THEN patch_with_hole is true

The attribute, find_hole, is declared as a external function and a Boolean value is expected in return.

8.2.3 Model Management Module

As described in chapters 4 and 7, model-based reasoning is done in the top-down approach in recognition. Image models are input by users. The model management module is designed to handle the knowledge from the users. The functions of this module are summarized in table 8.2.

Function	Description
Define	Enable user to define a new image model
List	List out all the image models defined
Modify	Modify an existing model
Delete	Delete an existing model
Exit	Exit to main menu

Table 8.2 Functions Available in Model Management Module

Since it is sometimes difficult for a user to use a crisp value for an attribute of a model, a fuzzy value of the attribute can be handled by OOI. The model management module will interpret the input fuzzy value and then convert the fuzzy value into a value interval. The interval is stored in a model object and used in the

model based reasoning. Furthermore, topological properties, described in chapter 6, can be input through this module. All the topological properties are separated by parentheses. Examples can be found in chapter 9.

8.2.4 Methods of Knowledge Encoding and Acquisition

The methods of knowledge acquisition are closely related to the form of knowledge representation, which are different for different expert systems. The conventional methods of knowledge acquisition can be classified into the following types:

(i) Knowledge Editor

A knowledge editor is a language editor in which the input of the knowledge will be guided according to the specific syntax available. Some of rule-based languages, such as EMYCIN and KAS, are examples of the knowledge editor. Used interactively, the editors prompt the users for the portions of a rule or declaration, supply defaults, correct spelling, and perform type checking as necessary. Such an editor can remove the drudgery of discovering and correcting syntax errors in a new rule as well as expedite the process of adding new rules and types to the program. Languages with relatively simple and fixed syntax, such as EMYCIN, have editors that find all syntax errors.

(ii) Traditional Editor

Languages like OPS5 and PROLOG, for which the definitions allow a

variety of syntactic forms, generally do not have editors that warn about the unusual constructs, even though the construct might be syntactically incorrect. The domain knowledge can be input through a traditional editor or word processor. The advantages in using these editors are that domain knowledge can be input quickly and modified easily. However, knowledge engineers have to remember all the syntax and the overall conceptual structure of the knowledge representation.

(iii) Interactive Editing

The third mode is to create a knowledge bases interactively through interrogations. Using this mode of input, a knowledge engineer will be asked questions by the system until a whole knowledge base has been created. The advantage of the interactive editing is that a knowledge engineer does not have to remember any syntax nor even to know the overall concept of the knowledge representation. Knowledge engineers only have to know several keywords to answer the questions. The main disadvantage are that the whole process of creating a knowledge base is very time consuming, and once the process has started, it can only be stopped at the end of certain stages. Moreover, modification is difficult and inefficient, since a series of questions have to be resolved in order to locate the piece of knowledge to be modified.

In the expert system shell, OOI, two modes of inputs are supported to cater for the need of different levels of users. One of the modes supports knowledge engineers to input knowledge through a traditional editors. This mode of input provides a quick means of knowledge input for those

knowledge engineers who are familiar with the system and the knowledge representation. The other mode supports knowledge to be input through interrogations. This mode of input caters for those knowledge engineers who do not have a clear concept of the knowledge representation.

8.3 User Interface in OOI

The user interface requirements for expert systems have evolved considerably since the days when a consultation system first conducted a dialogue with the user. These changes are due in part to the transition from expert systems to expert advisory systems, the increasing use of models of knowledge, and the increasing size and complexity of the system. Because the user is actively involved in the decision-making process, there is increased necessity for the user interface to support the user's cognitive task. This requirement encourages the use of the interface as a model world metaphor, which is directly supported by model of knowledge. Under the current research in user interface [Hendler88], the following requirements must be met by the user interface.

(i) Immediate Feedback

As well as displaying the domain in the screen, the displays must allow the user to act directly upon them, and then provide immediate feedback on the effects of the user's actions. Rapid feedback in terms of changes in the behavior of objects not only allows for the modification of actions even as they are being executed, but also supports the feeling of acting directly on the objects themselves. It removes the perception of the computer as an intermediary by pro-

viding continual representation of system state.

(ii) Recoverability

The end users of expert systems tend to be skilled knowledge workers, who may or may not be experts in the precise area addressed by the system. Aside from the fact that such users tend to be less patient with "unfriendly" systems, the success of the system is directly related to the users' ability to try out various decisions on the system and then change their minds. They must be able to recover easily from the situations they do not want to be in and to experiment with different responses without much penalty.

Recoverability implies more than standard undo facilities and may impose computational requirements on the underlying system for maintenance of history list, checkpoint capabilities, or even truth maintenance. The ability to guess at an answer to a question with little or no penalty, and to see the implications of the guess quickly, makes the system easier to learn to use.

(iii) Appropriate Granularity

As applications become larger and more complex, an overall requirement of the interface is to assist knowledge engineer in dealing with complexity. This implies that the interface design issues become more important as the application becomes larger. To assist in dealing with complexity, the interface must, first assist the knowledge engineer in decomposing the problem, and, secondly, help him to understand the relationships between the component parts of the problem. It is achieved in creating hierarchies indicating both menu and

submenu relationships.

In OOI, immediate feedback is achieved by checking all the input from a user and error messages are used to prompt the user immediately when failure occurs. Accelerator key may be used for the experience user, by typing the capital letters. For the recoverability, the user can escape from any environment by pressing the <ESC> key in the keyboard. OOI is built using a sequence of menu and submenu, as shown in section 8.3.2.

8.3.1 Screen Layout

The screen layout of OOI and the main menu is shown in Figure 8.3.

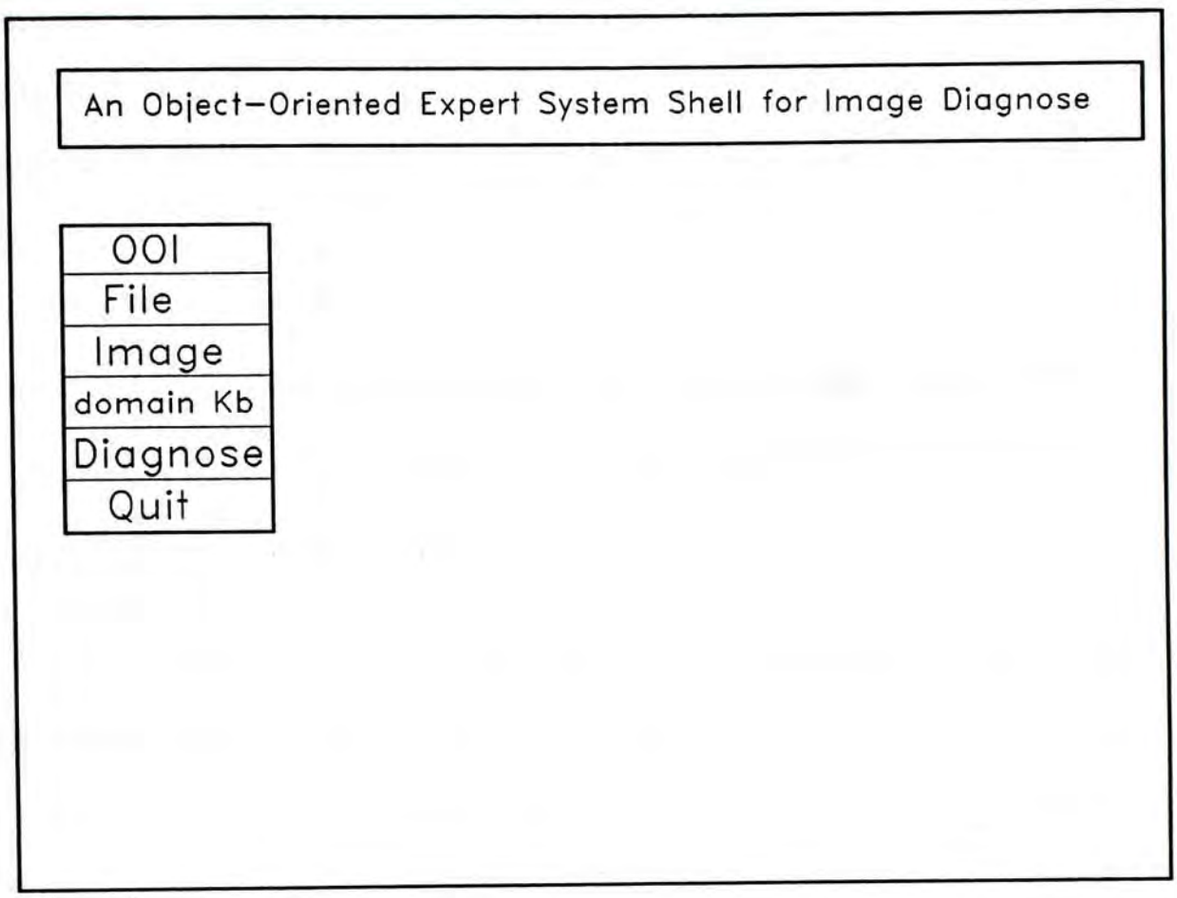

Figure 8.3 Screen Layout and Main Menu of OOI

A pop-up window is used in the design. The pop-up window is a portion of the screen that is used for a specific purpose. When the window appears, what is currently on the screen is saved and the window is displayed. When the application using that window is finished, the window is removed and the original contents of

the screen restored. In OOI, it is possible to have several windows on the screen at the same time. Functions with similar properties are group in the same window. This is done to increase the granularity of the system, as described above.

8.3.2 Menus and Options

As shown in figure 8.3, a user may select one of the five options from the main menu. The functions of these options are briefly described as follows:

(i) File Option

This option is used to load and save the knowledge into the system. All the attributes created by the system can be saved into a file and loaded upon requested. When this option is selected, another window contained sub-menu will be displayed. The description of all the options in this sub-menu are shown in table 8.3

Option	Description
Load	Attributes, which are previously defined and stored in file, are retrieved into the system by using this command.
Save	This will save up all the attributes created, which are currently defined by the knowledge engineer, into a file.
Shell	Exit to the DOS temporarily.
New	It will reset the system and all the parameters defined will be reinitialized.
Exit	Exit to main menu.

Table 8.3 Sub-menu in File Option

(ii) Image Option

This option allows knowledge engineer to define all the image attributes and models which are used in the system. Users can also activate the low level vision kernel to preprocess an image so as to capture the primary image patches. The sub-menu in Image option is detailed in table 8.4.

Option	Description
Model	Knowledge engineer can use this sub-menu to input an image model which are used in top-down approach, as described in chapter 7. The functions available in this sub-menu can be found in table 8.2
Attribute	This option is responsible to acquire the attributes, described in section 8.2.2. Details are summarized in table 8.5.
LLVK	The low level vision processing kernel is activated in this option. All the primary attributes are extracted and assembled in a file for diagnosis.
Exit	Exit to main menu.

Table 8.4 Sub-menu in Image Option

As described in the section 8.2.2, image attributes must be defined before any diagnosis can be done. These attributes are input by the knowledge engineer. The attribute management module is designed to exploit the knowledge from the users. The options in image-attribute sub-menu can be summarized in the table 8.5.

Option	Description
Define	Enable knowledge engineer to define a new attribute.
List	List out all the attributes defined.
Modify	Modify any existing attribute.
Delete	Delete a existing attribute.
Exit	Exit to main menu.

Table 8.5 Options in Image-Attribute Sub-menu

(iii) Domain Knowledge Base Option

Domain knowledge are stored in term of rules in the knowledge base. These rules can be loaded from a file if they already exist. Knowledge engineer can also activate an editor to edit a new set of rules, if appropriate. The format of the rules are shown in chapter 7. The options available in this sub-menu are :

Option	Description
Load	Enable knowledge engineer to load a predefined rule set from file. If the file doesn't exist, an editor is loaded for engineer to edit a new rule set.
Edit	This enables the engineer to edit a specific rule.
View	It will display all the rules which already exist inside the knowledge base.
Delete	Delete an existing rule.
Exit	Exit to main menu.

Figure 8.6 Sub-menu in Domain Knowledge Base Option

(iv) Diagnosis Option

In this option, parameters are set for the inference. For example, image objects and goal of the diagnosis can be built. User can activate the inference engine to accomplish the diagnose. The result of the diagnosis will be manifested in a file afterward. The options available in this sub-menu are shown in figure 8.7.

Option	Description
Build	Primary image patches are retrieved from file and image objects are built. These image objects serve as the fundamental objects in diagnosis.
Goal	This option is used to establish a goal for the backward chaining reasoning in diagnosis.
Start inference	It will commence the diagnose process.
Restart	This option is used to reset the system and values of instance variables to the initial state.
Exit	Exit to main menu.

Figure 8.7 Sub-menu in Diagnosis Option

8.4 Concluding Remarks

A major bottleneck in the development of expert systems occurs during the knowledge acquisition phase. Domain expertise, in fields such as medicine and law, cannot currently be easily encoded into the knowledge base. However, in OOI, a text

editor is used for inputting rules in the rule management module while model management module is designed to handle image models. These two modules enhance the capability of the knowledge acquisition in OOI.

CHAPTER 9. IMPLEMENTATION AND RESULTS

9.1 Introduction

The analysis schemes described in the preceding chapters and sections have been applied to obtain meaningful segmentation and diagnosis of medical cell images. Each image is grabbed through an electronic microscope. The microscope used is a polarizing petrographic microscope with a trinocular head which will allow the attachment of a camera. Images are shot on a number of slides of normal and abnormal cells. A image grabber, PCVISION *Plus* frame grabber, is used to digitize the images. The PCVISION *Plus* frame grabber is a video digitizer and frame memory capable of digitizing standard RS-170/330 (or CCIR) video input and storing the digitized image in a special on-board frame memory. The image can be simultaneously displayed on a video monitor [Imaging87]. The digitization is performed with a 512×512 size with 8-bit density resolution. Each image has 256 gray levels.

The implementation of OOI is carried out in PC environment. The systems for the preliminary segmentation and the high level reasoning analysis are implemented in C++ and it is run under MS-DOS. Since it is inefficient to have 256 Kbytes (512×512 bytes) of image pixels resident in the user memory of DOS environment, expanded memory is used. The discussion of using expanded memory is in section 9.2. The total number of C++ statements in the programs is about 25,000. An expert system, ESCUM, are designed to diagnose the cell images of the cervix uteri. A set of rules and models are designed. They are analysed and shown in section 9.3. Some real images are studied

and the results are presented in section 9.4.

9.2 Using Expanded Memory

Expanded memory provides MS-DOS programs with access to maximum of 32 megabytes of RAM memory beyond the 640 K bytes managed by MS-DOS. Unfortunately, expanded memory cannot be used by MS-DOS applications automatically. Each program must be specifically written to recognize and use it. In OOI, all programs using expanded memory is following a protocol. Each program is modified according to the following steps:

(i) Detect the presence of the expanded memory manager (EMM). EMM is a software component that provides the software interface defined by Lotus-Intel-Microsoft (LIM) Expanded Memory System (EMS) between the application program and the underlying expanded memory system.

(ii) Determine whether a sufficient number of expanded memory pages is available to the application.

(iii) Obtain the address of the start of the page frame.

(iv) Allocate expanded memory page.

(v) Map expanded memory pages into the page frame.

(vi) Read, write or execute data in expanded memory.

(vii) Return expanded memory pages to the EMM before the application terminates.

9.3 ESCUM

9.3.1 General Description

ESCUM(Expert System for Cervix Uteri Morphometry) is a medical expert system in the domain of the diagnosis of cancer cells in cervix uteri. Cervix uteri is the lower portion of the uterus. It is a tubular structure measuring approximately 4 cm in length and about 3 cm in diameter [Koss79]. The epithelium (a closely-packed sheet of cells arranged in one or more layers) of the cervix is of two types. The ectocervix, protruding into the vagina, is linked by non-keratinized stratified squamous epithelium similar to that of the vagina. The stratified epithelium extends approximately to the external os where it changes to tall columnar epithelium covering the endocervical canal [Novak74].

During sexual maturity, the stratified squamous mucosa of the normal adult woman undergoes cyclical changes in response to ovarian hormone stimulus. The stratified squamous epithelium can be distinguished into the basal, middle and superficial layers. The basal layer is the layer of epithelial regeneration, and is composed of one row of small elliptical cells [Koss79].

Cells in the middle layers gain more cytoplasm as they mature towards the surface while the nuclear size remains more or less the same. This layer can be subdivided into the parabasal and intermediate layer. The parabasal layer consists of smaller cells adjacent to the basal layer, and the intermediate layer consists of larger cell next to the superficial layer [Koss79].

The superficial zone is composed of flattened, larger cells with smaller and pyknotic nuclei. The cytoplasm of most superficial cells contain precursors of keratin although keratin only develops under abnormal conditions [Ham74].

9.3.2 Cervical Intraepithelial Neoplasia (CIN)

The transformation zone around the squamocolumnar junction of the cervix is where most of the CIN and eventually cervical cancers develop [Anderson87]. The term Cervical Intraepithelial Neoplasia (CIN) was first used by Richarts to include both dysplasia, of all grades, and carcinoma (a malignant epithelial tumour) in situ into a single entity [Richarts67]. The columnar epithelium which covers the transformation zone undergoes reserve cell hyperplasia (any condition in which there is an increase in the number of cells in a part) leading to squamous metaplasia. During the early stage of the metaplastic process, a carcinogen (a substance which cause living tissues to become carcinomatous) introduced may stimulate a malignant transformation of the cells when there is host susceptibility [Anderson87]. It is not clear at what stage in the metaplastic process does the changing epithelium become non-susceptible to the transformation. It seems that other than the actively transforming epithelium, the mature metaplastic, squamous and columnar epithelium are resistant to mutagenic change [Anderson87].

CIN was further classified into three grades by others [Buckley82, Ferenczy82]. CIN I and CIN II are equivalent to mild and moderate dysplasia while CIN III includes both severe dysplasia and carcinoma in situ. In CIN I and II, the nuclei are enlarged, pleomorphic (occurrence in more than one form), slightly irregular

in outline, hyperchromatic with aberrant chromatin structure and prominent nucleoli. The nucleocytoplasmic (NC) ratio is high. In CIN III, undifferentiated basaloid cells with high NC ratio occupy more than a half of the epithelium. Cells shows marked nuclear pleomorphism, little cytoplasm and loss of polarity.

9.3.3 Development of ESCUM

In diagnostic histopathology (recognition of changes in cells) are based on subjective interpretation of morphological features. In the continuous lesions, there are gradual transitions from completely benign to extremely poorly differentiated. Arbitrary criteria have to be devised for their classification. Subjective interpretation of such criteria are likely to result in diagnostic shifts and consequently disagreement between different interpreters [Baak84].

Morphometry, which is the measurement of the different forms of an organism, has the advantages of reproducibility, accuracy, objectivity and the potential to detect differences which may escape the human observer or are neglected by subjective judgement [Baak84]. Morphometry is therefore desirable in the area of intraepithelial neoplasia (the growth of a neoplasm within the cells).

Morphometric study of the intraepithelial neoplasia and associated lesions of cervix uteri is conducted to have a view into the maturation kinetics of the lesions, feasibility of objective discrimination of the different grades and legitimacy of various grading systems. Biopsies are used in the study of cervix uteri. Epithelial cells of the biopsies are sampled from surfaces layer of the epithelia. In the traditional

approach, measurement is done by manual tracing of the monitor images using the Videoplan Image Analyzer (VIA) without an expert system such as ESCUM. The following attributes are computed manually:

(a) Mean nucleus area

(b) Mean nucleus perimeter

(c) Mean maximum nuclear diameter

(d) Mean minimum nucleus diameter

(e) Mean nucleus polarity

(f) Mean cell area

(g) Mean cell perimeter

(h) Mean nucleocytoplasmic (NC) ratio

(i) Mean nuclear contour index

The development of the ESCUM is carried out in two phases. The first phase is based on the morphometric study of the cervix uteri. It computes all the attributes listed above. Object models are designed and used to diagnose epithelial cells of the biopsies which are sampled from surfaces layer of the epithelia. They are used in top-down approach as shown in tables 9.1-9.5. The models "nucleus", "cytoplasm", "blood cell" and "garbage" are defined. The "abnormal nucleus" of nucleus is also defined as a derived class of the "nucleus".

Name of generalized model		: nucleus
Derived class name		: nucleus
Area	(small, large)	: small
Mean gray level	(low, high)	: very low
Perimeter	(short, long)	: short
Background	(low, high)	: medium
Compactness	(low, high)	: high
Topological Properties		: (inside cytoplasm)

Table 9.1 Generalized Model of Nucleus

Name of generalized model		: cytoplasm
Derived class name		: cytoplasm
Area	(small, large)	: medium
Mean gray level	(low, high)	: medium
Perimeter	(short, long)	: long
Background	(low, high)	: very high
Compactness	(low, high)	: low
Topological Properties		: (inside background)

Table 9.2 Generalized Model of Cytoplasm

Name of generalized model		: white blood cell
Derived class name		: white blood cell
Area	(small, large)	: small
Mean gray level	(low, high)	: very low
Perimeter	(short, long)	: short
Background	(low, high)	: very high
Compactness	(low, high)	: high
Topological Properties		: (inside background)

Table 9.3 Generalized Model of White Blood Cell

Name of generalized model		: garbage
Derived class name		: garbage
Area	(small, large)	: very small
Mean gray level	(low, high)	: very high
Perimeter	(short, long)	: short
Background	(low, high)	: very high
Compactness	(low, high)	: low
Topological Properties		: (inside background)

Table 9.4 Generalized Model of Garbage

Name of generalized model		: nucleus
Derived class name		: abnormal nucleus
Area	(small, large)	: medium
Mean gray level	(low, high)	: very low
Perimeter	(short, long)	: medium
Background	(low, high)	: medium
Compactness	(low, high)	: very high
Topological Properties		: (inside cytoplasm)

Table 9.5 Derived Model of Nucleus

The second phase of ESCUM is based on the historical data of patient. This phase of development is used to confirm the clinical suspicion arising from the image diagnosis. The medical knowledge is represented as production rules and is elicited from a gynaecologist. The knowledge includes the following areas:

(a) age,

(b) diet,

(c) family history,

(d) sexual life and diseases,

(e) menstrual history, and

(f) social life and culture.

The construction of the second phase of ESCUM is an incremental process and the knowledge base of ESCUM has been refined for several times. As ESCUM is a diagnosis system, it can be built effectively by adjusting the rules in domain knowledge base. The rules in the ESCUM are shown in appendix A.

In diagnosis, ESCUM has two goals as follows:

(a) **Mean Nucleocytoplasmic (NC) Ratio**

It is a ratio between the area of nucleus and cytoplasm. As mention in section 9.3.2., the nucleocytoplasmic (NC) ratio is high in CIN I and II while undifferentiated basaloid cells with high NC ratio occupy more than a half of the epithelium in CIN III [Wai90]. The mean NC ratio for each category are shown in table 9.6.

Cells in Cervical Biopsy	Mean NC Ratio
Normal	0.078 ± 0.026
CIN I	0.195 ± 0.051
CIN II	0.309 ± 0.057
CIN III	0.464 ± 0.079

Table 9.6 Mean Nucleocytoplasmic Ratio of Cells in Cervical Biopsy

(b) **Cell_Diagnosis**

It is a multiple-valued type object goal, with a certainty factor and has the following expected values :

(i) normal,

(ii) CIN_I,

(iii) CIN_II,

(iv) CIN_III and

(v) Cancer.

They indicate the stages to which the cells belong.

The two goals in ESCUM are related to each other. The inference of Cell_Diagnosis depends on the result of the mean NC ratio. Figure 9.1 shows the structure of ESCUM.

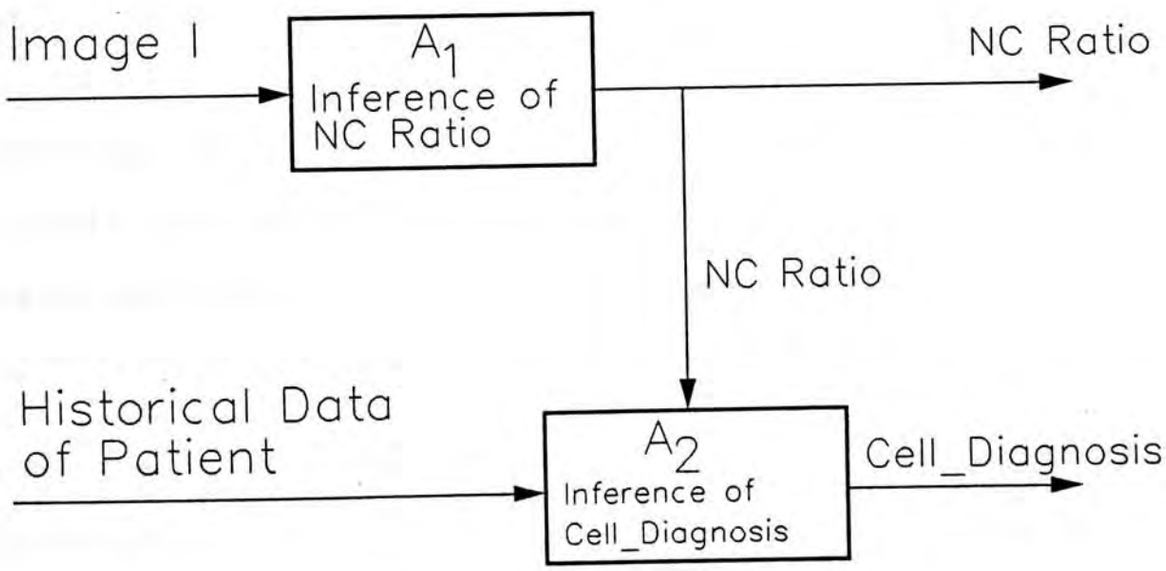

Figure 9.1 Structure of ESCUM

In Figure 9.1, an image of the cells of cervix uteri, I, is grabbed and input into A_1, inference of NC ratio, that produces the mean NC ratio. After giving the ratio to user, the NC ratio and the historical data of patient are input into the second level of inference. The output of A_2 is the Cell_Diagnosis.

9.4 Results

Figure 9.1(a) is a digitized input image of a normal cell of cervix uteri. A clear cell can be seen at the right hand side of the image. It can be seen that a nucleus is embedded inside the cytoplasm. Besides, some black spots appear near the cell. They are white blood cells. Since the grabbed image is a noisy image and it is relatively dark, a low-pass filter is used to filter out the high-frequency components of the image. Figure 9.1(b) shows the result of the low-pass filtering. Figure 9.1(c) shows that the brightness of the image is increased. The histogram of the image is shown in figure 9.1(d). Multi-level thresholds are used to segment the image as well as to separate the main features in the image. Figure 9.1(e) illustrates the result. Sobel operator, a gradient operator, is used to extract the edge from the image. Result is shown in figure 9.1(f). 80 regions are grown in using the line-adjacency graph technique described in section 5.5.2. The grown regions are highlighted by dotted lines shown in figure 9.1(g).

The preliminary segmented regions are then analyzed by the rules and models embedded in ESCUM. Since the appearances of the white blood cells and the nucleus are quite similar, rules are used to distinguish them.

After analysing the image in figures 9.1(a)-(g), ESCUM identifies the nucleus and the cytoplasm properly. The NC ratio is 0.05. It is identified as normal cervix uteri cell and the patient is diagnosed to have no cancer with certainty factor 0.84. Figures 9.2(a)-(g) show another example, the NC ratio is 0.32 which is classified as a cell in grade CIN II. However, if the historical data of the patient is also included in the diagnosis as shown in figure 9.1, the patient is identified to be in the CIN_III stage with certainty factor 0.85. Figures 9.3(a)-(g) show the result of another example in which the nucleus to cytoplasm ratio is 0.41. Again, it is identified as an abnormal cell image and belongs to the cell of CIN III. Cancer is diagnosed after merging with the historical data of the patient with certainty factor 0.9. These deductions coincide with that of the expert.

9.5 Concluding Remarks

In this chapter, implementation details are discussed. A sample expert system, ESCUM, is designed to diagnose the cell images of the cervix uteri. Three real-life images are grabbed and tested. Preliminary results obtained are satisfactory.

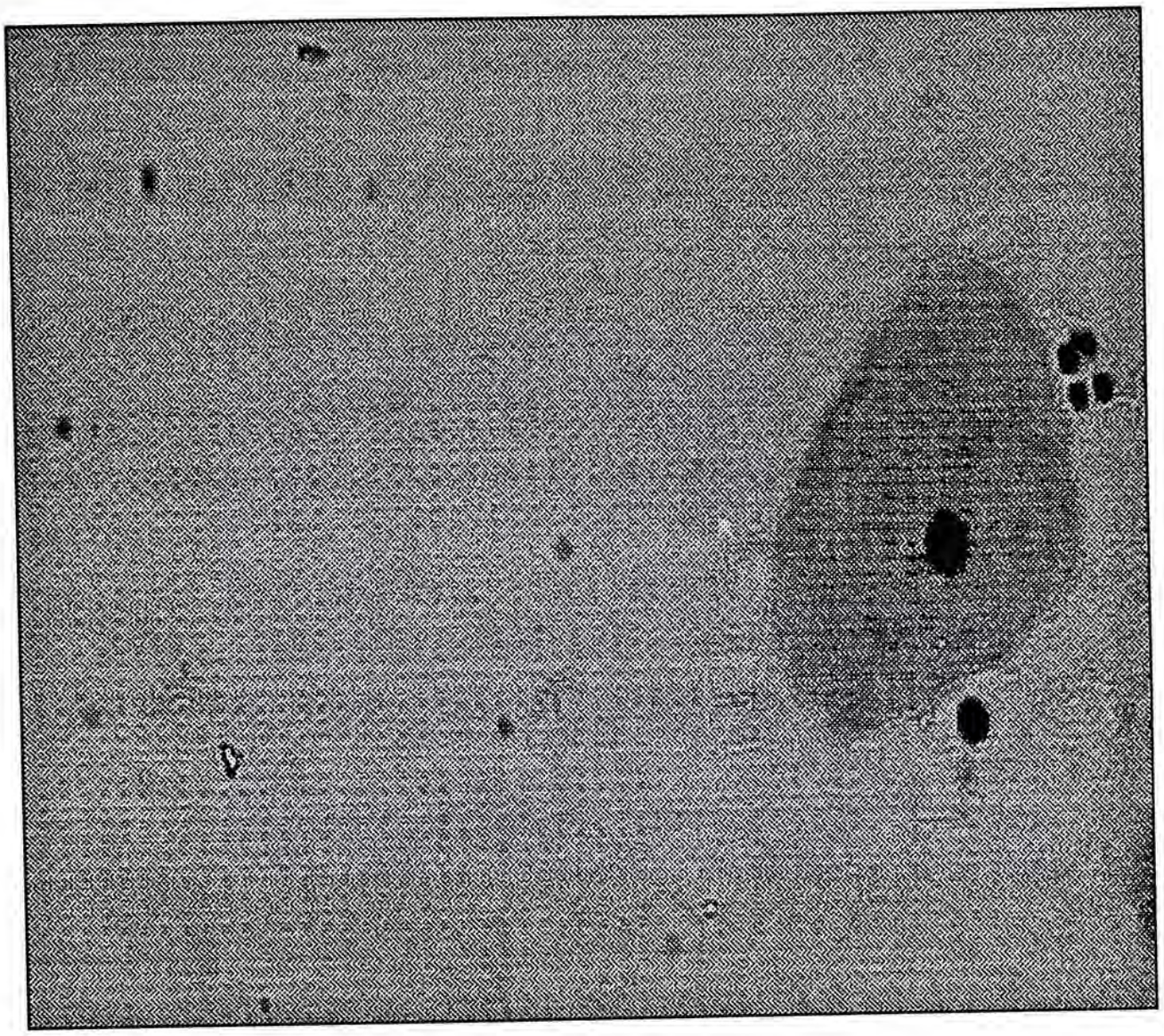

Figure 9.1(a) The Input Image

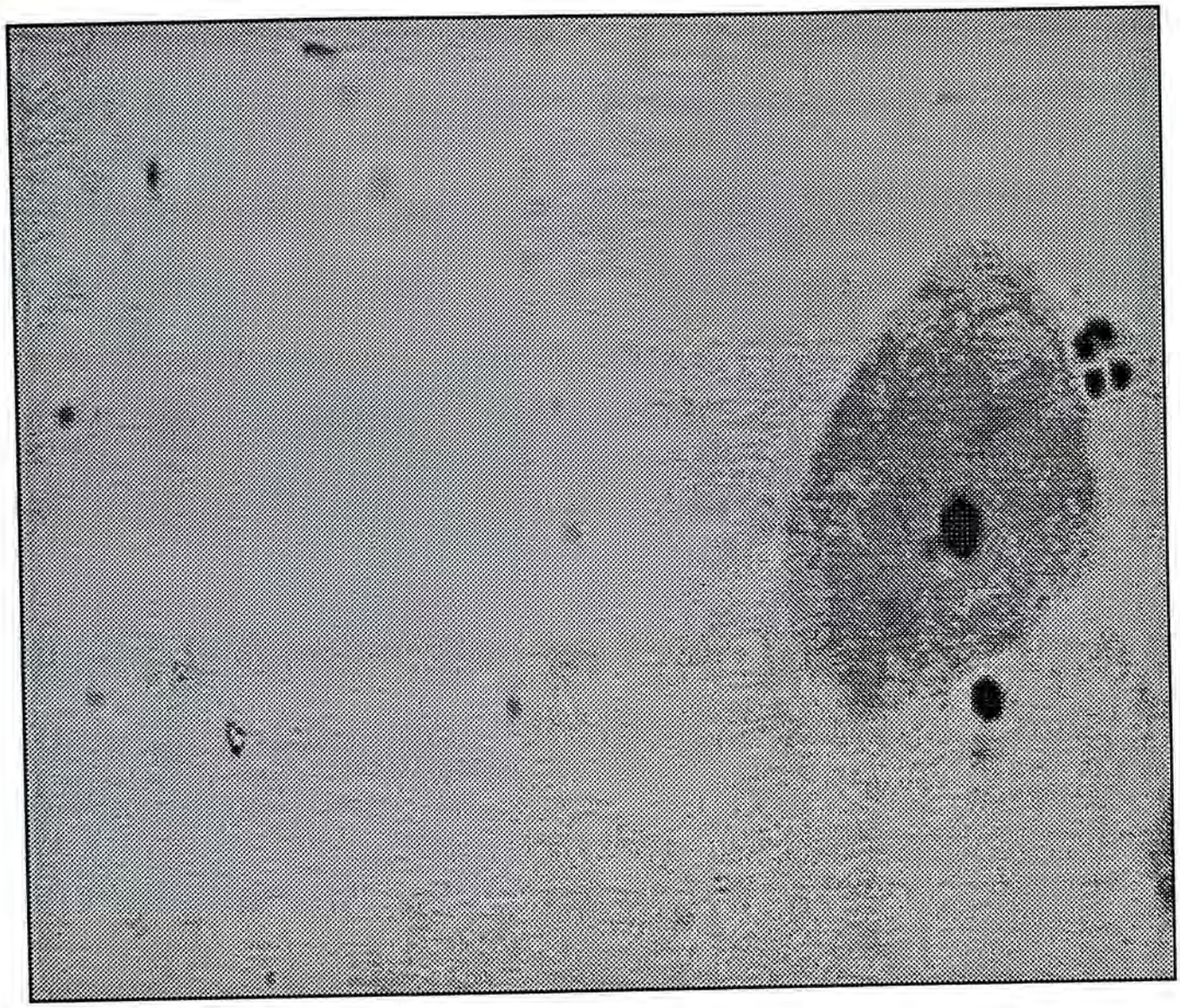

Figure 9.1(b) Input Image after Low Pass Filtering

Figure 9.1(c) Input Image with increased Brightness

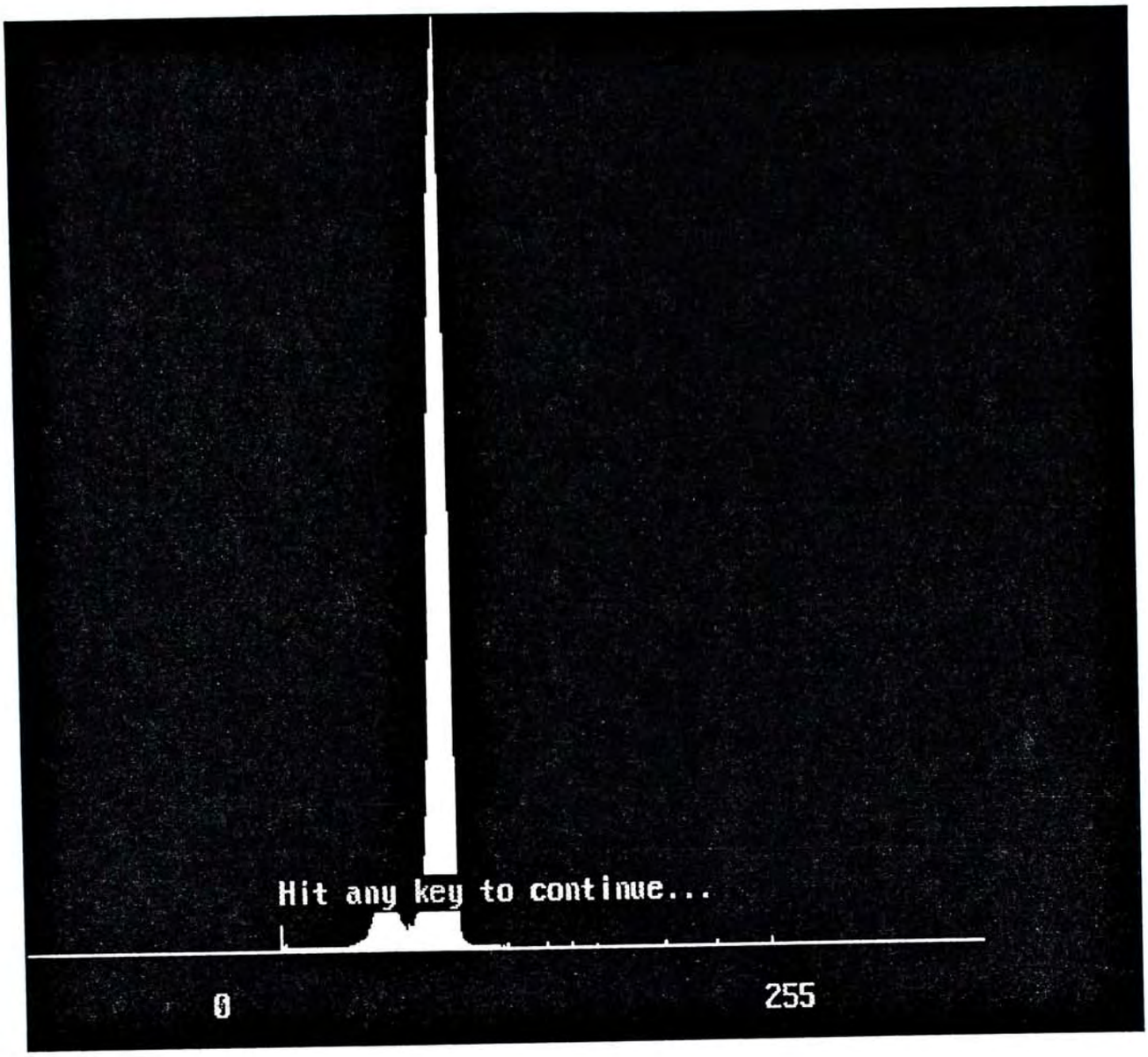

Figure 9.1(d) The Histogram of the Input Image

Figure 9.1(e) Image after Applying Multi-level Threshold

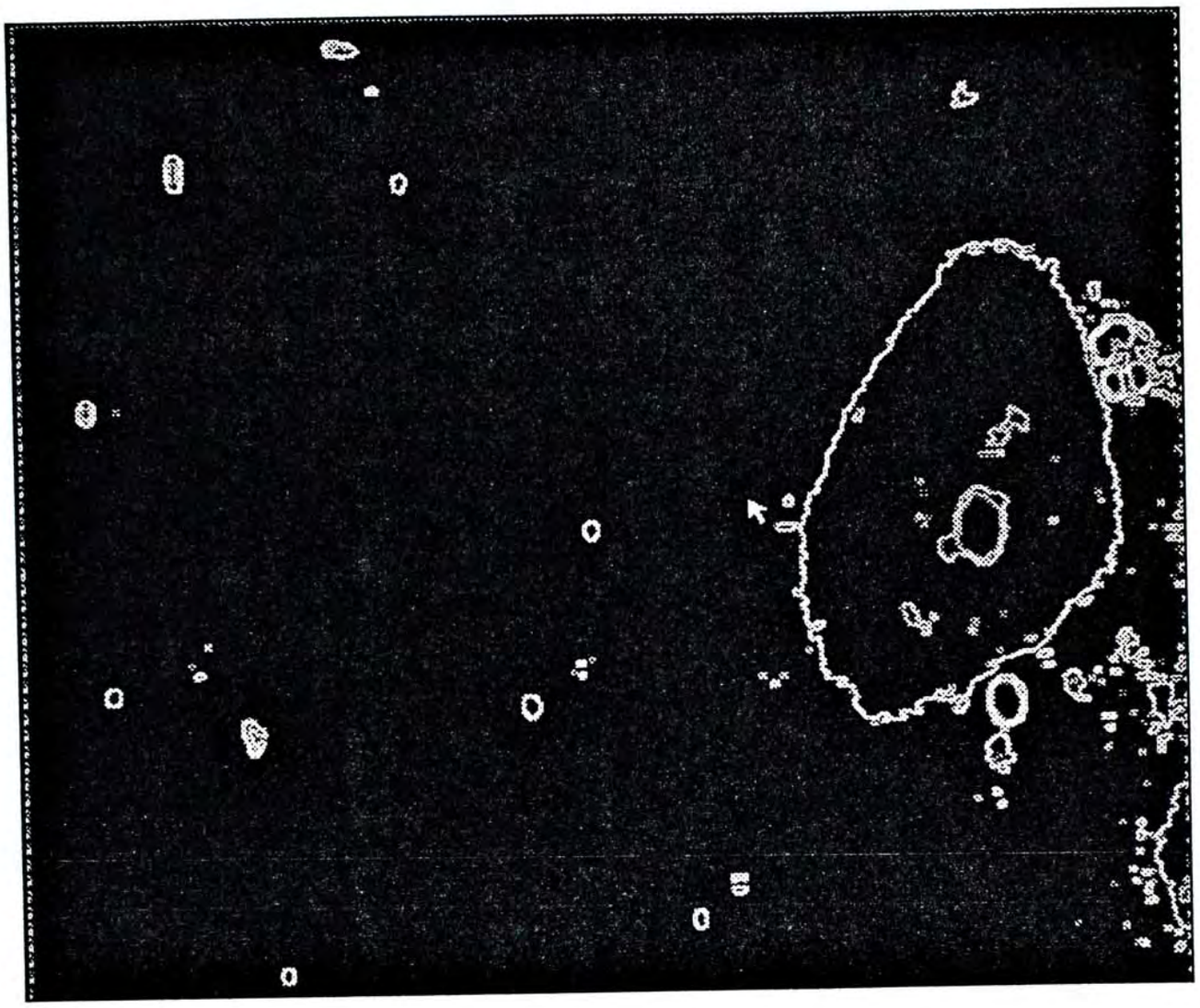

Figure 9.1(f) Image after Sobel Operator

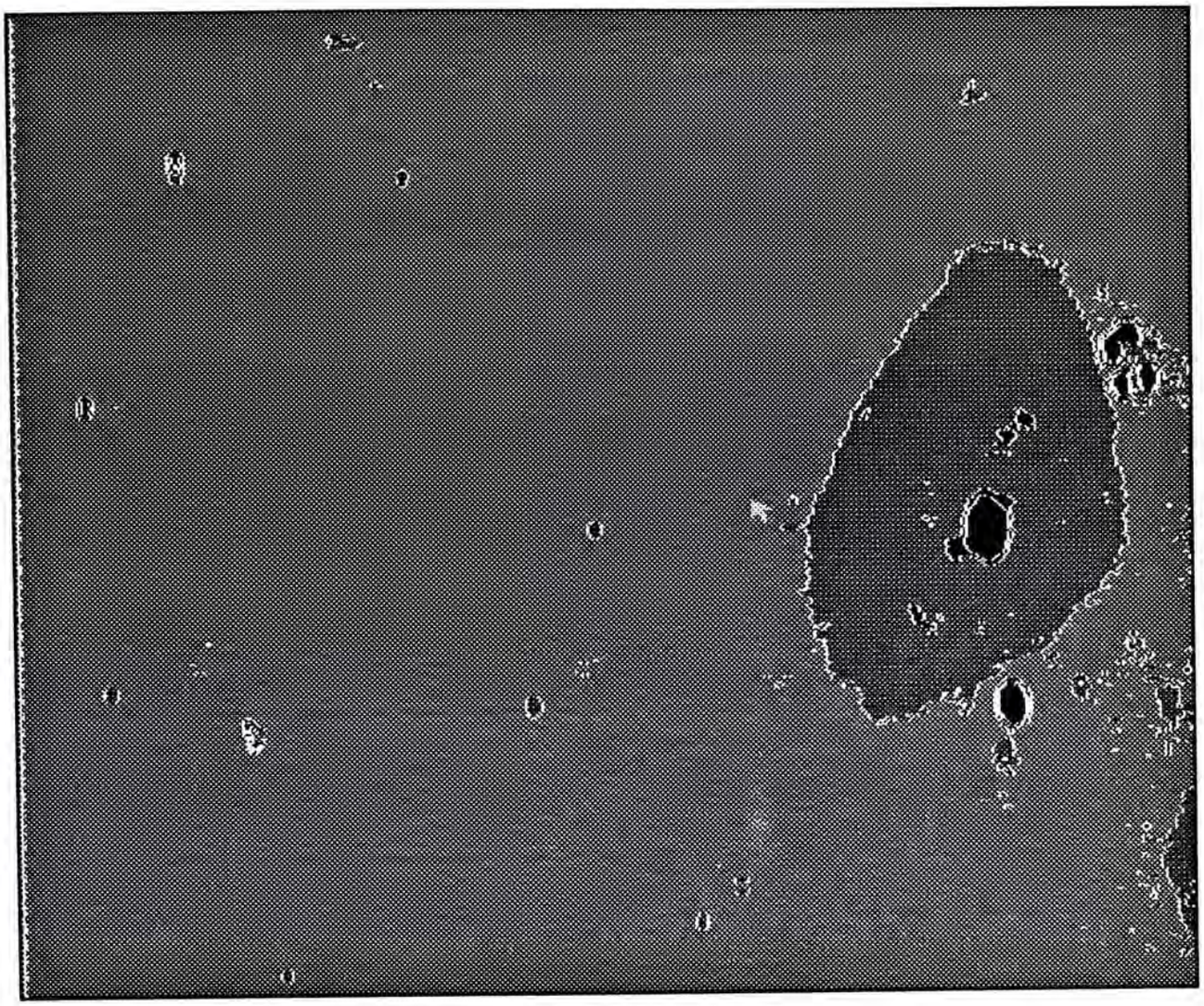

Figure 9.1(g) Image after Region Growing

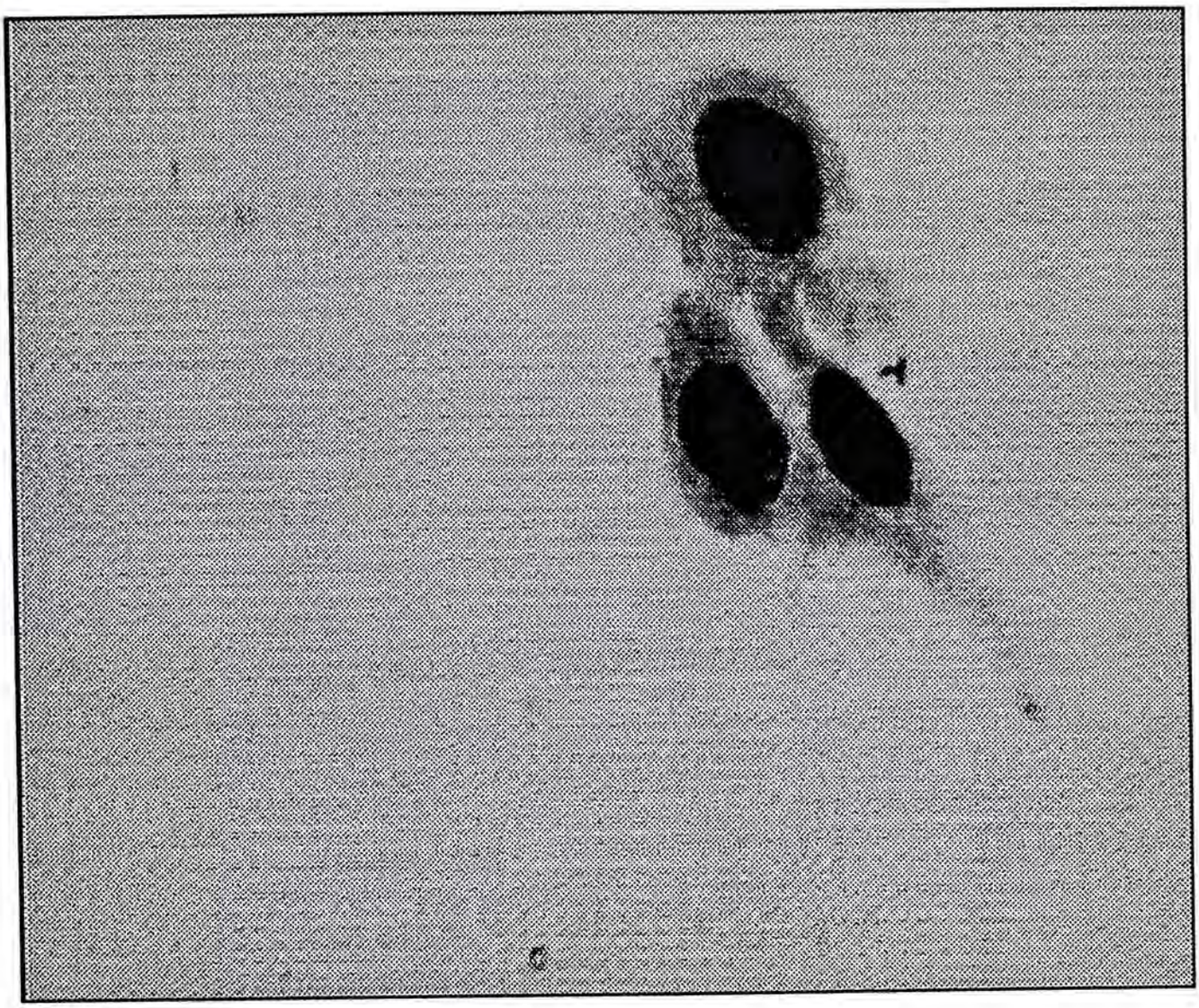

Figure 9.2(a) The Input Image

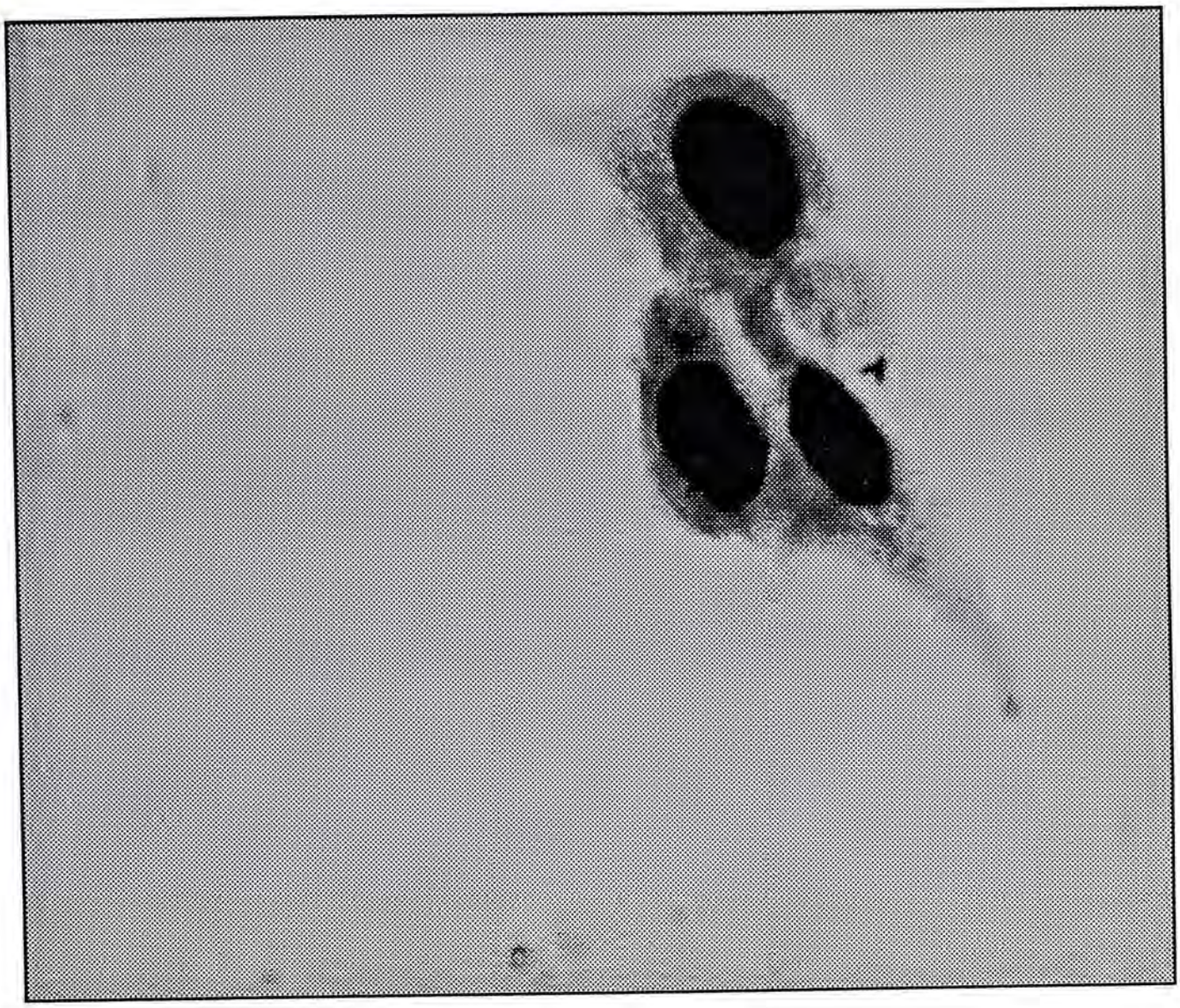

Figure 9.2(b) Input Image after Low Pass Filtering

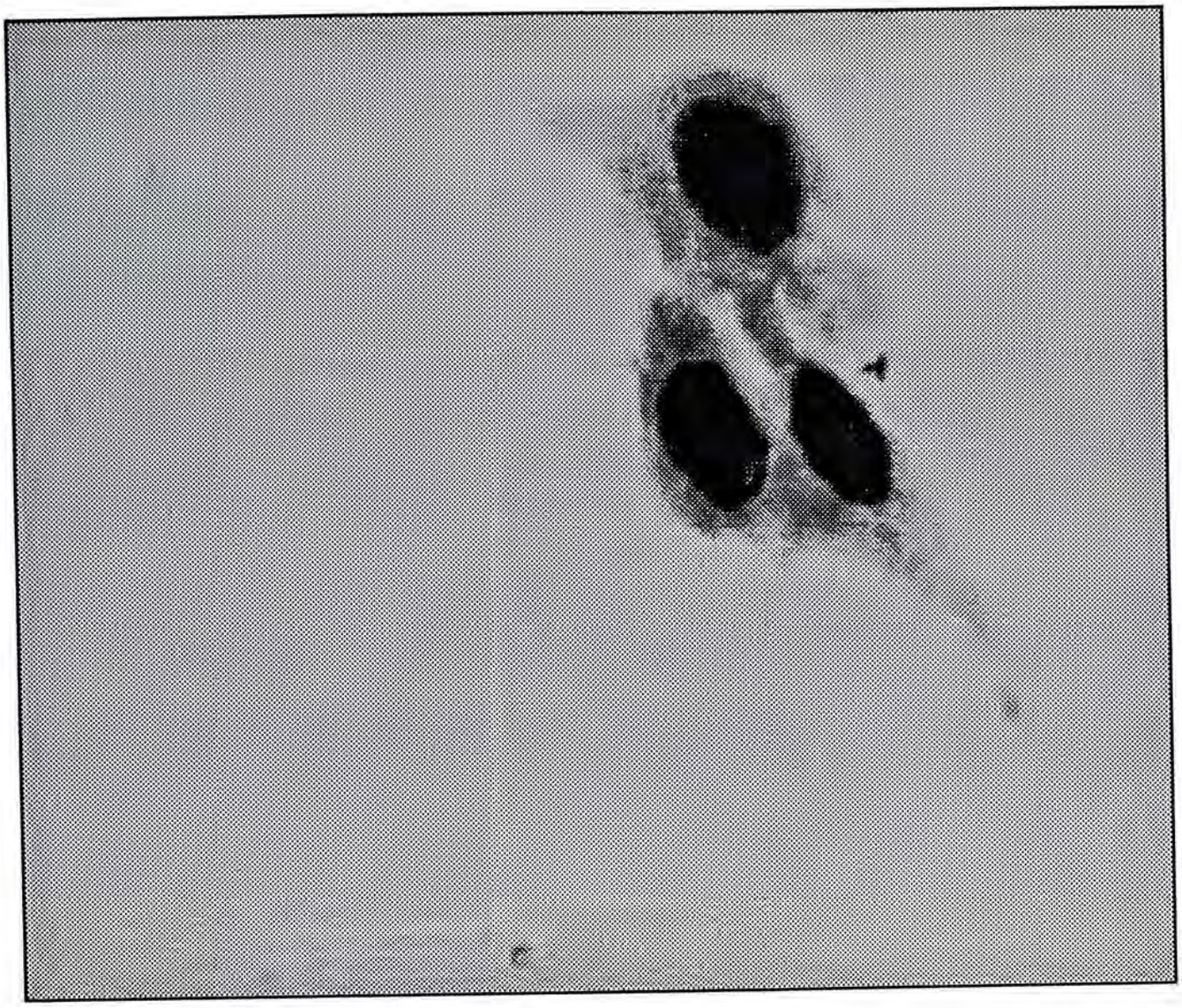

Figure 9.2(c) Input Image with Increased Brightness

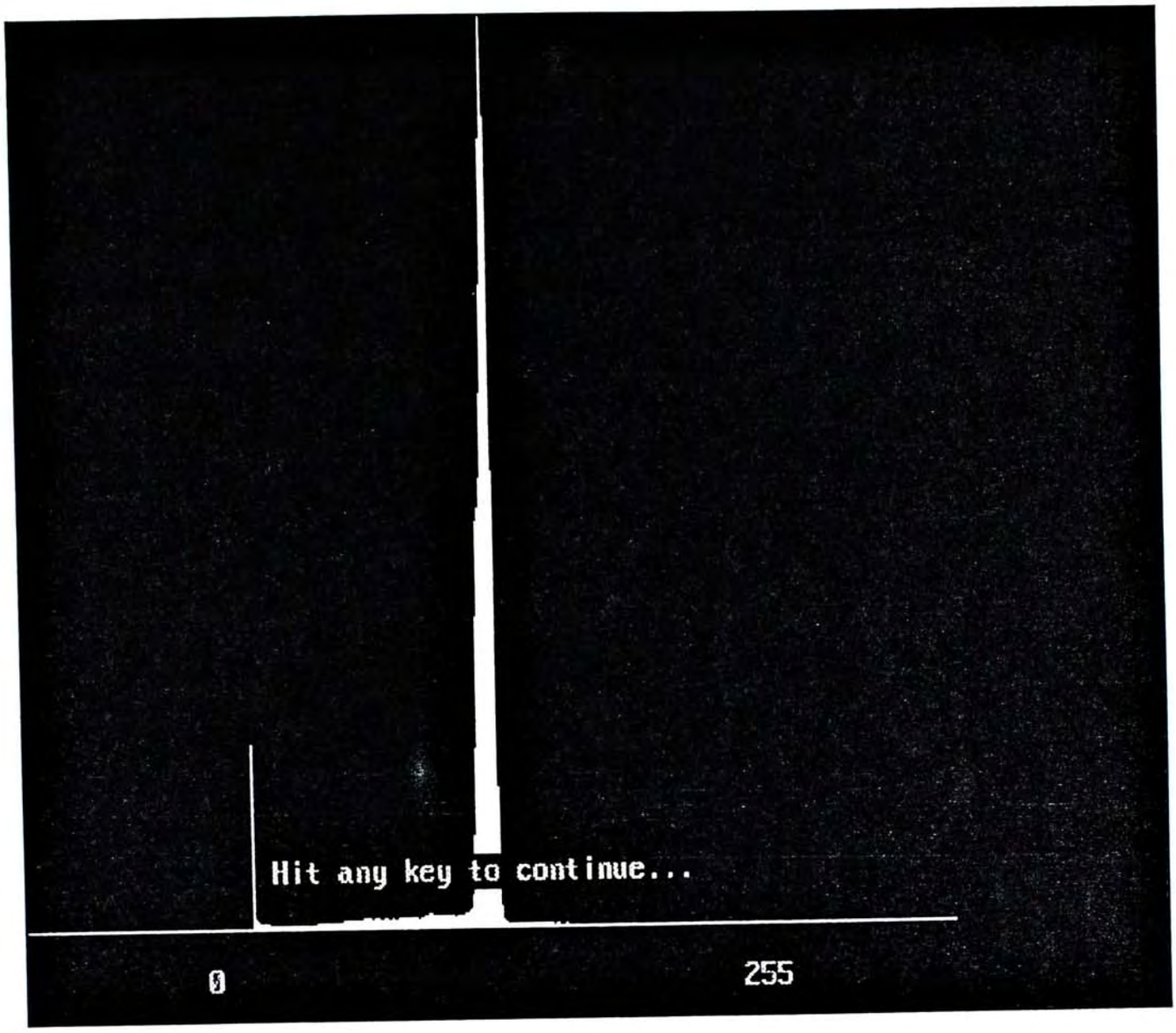

Figure 9.2(d) The Histogram of the Input Image

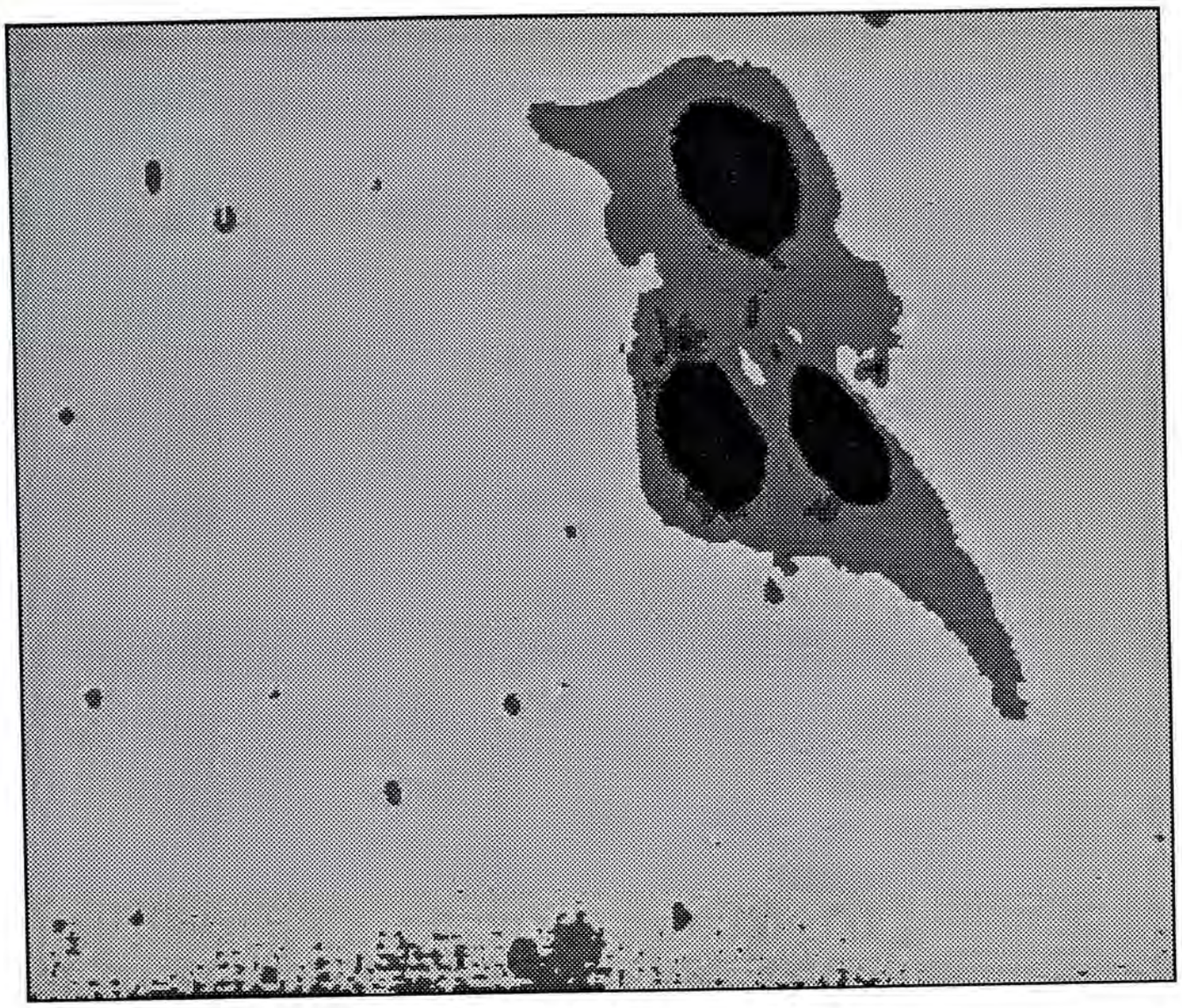

Figure 9.2(e) Image after Applying Multi-level Threshold

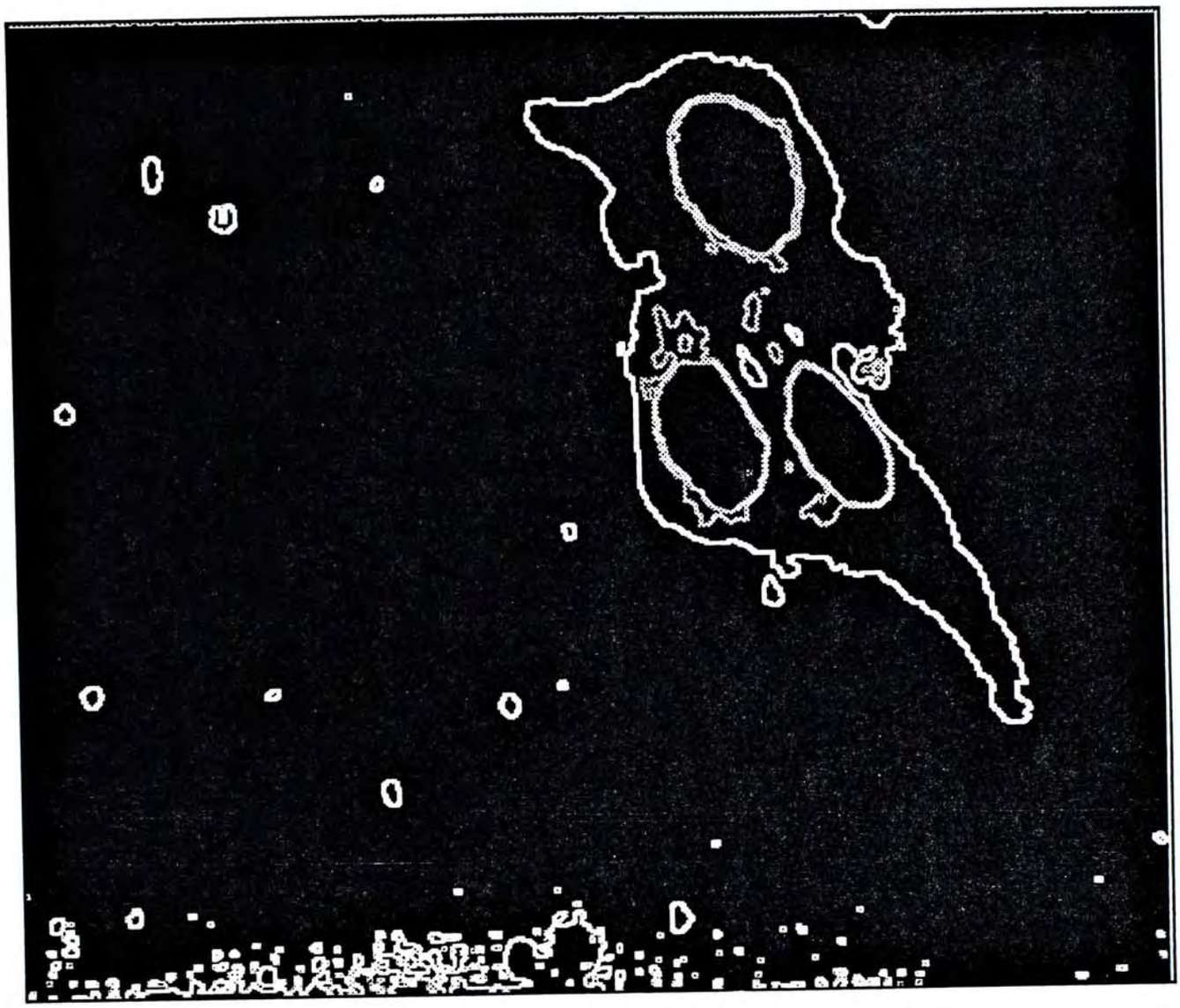

Figure 9.2(f) Image after Sobel Operator

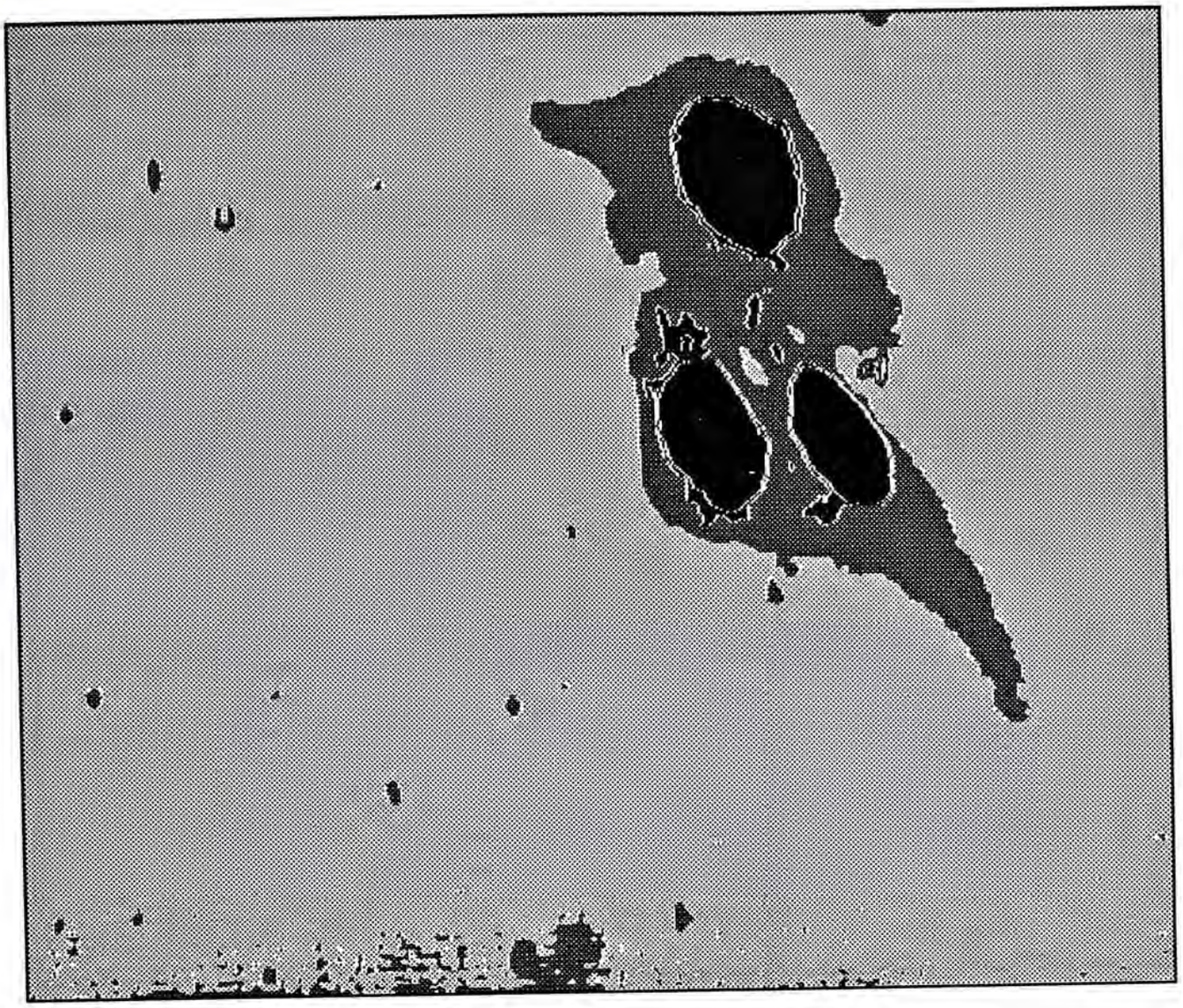

Figure 9.2(g) Image after Region Growing

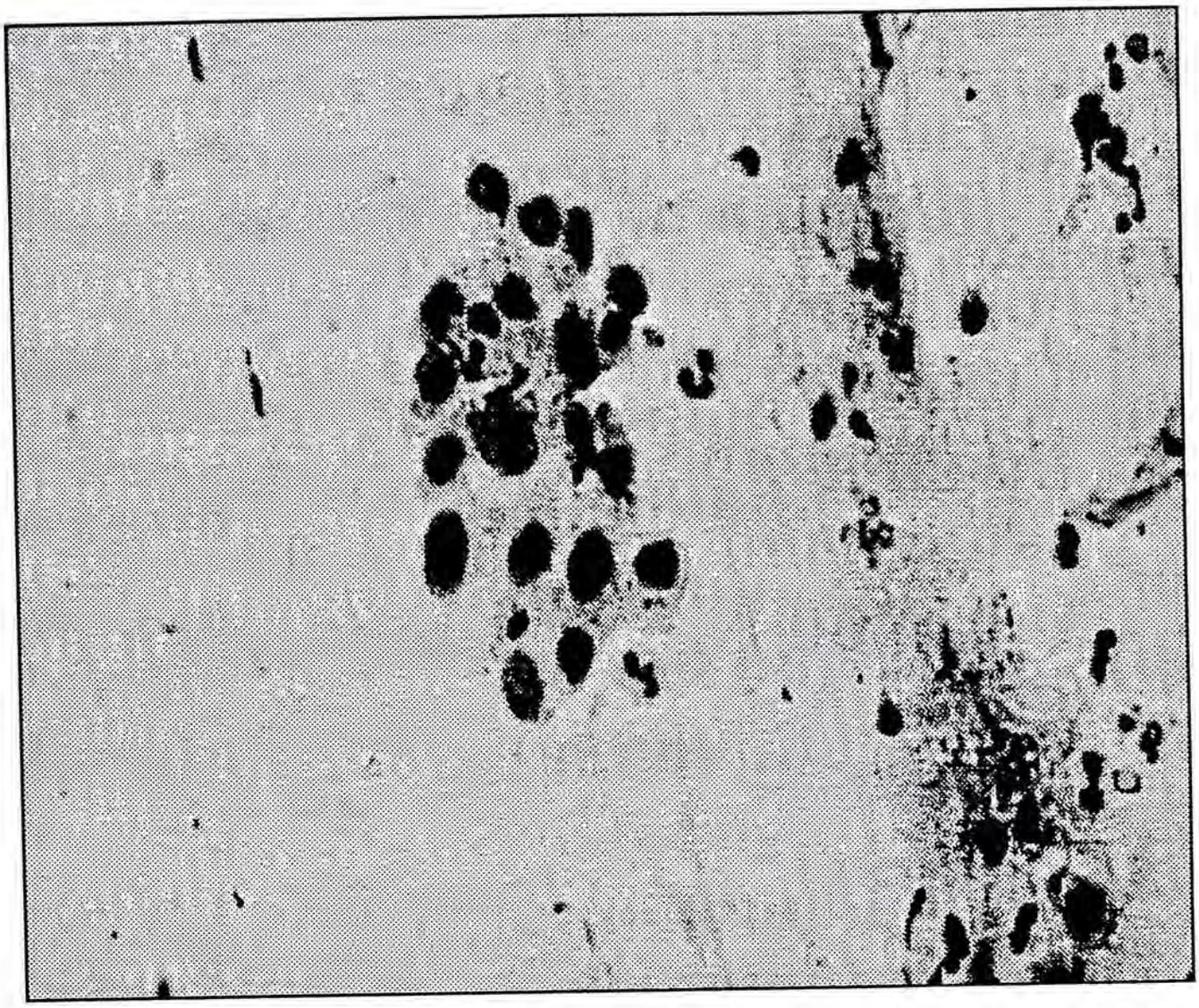

Figure 9.3(a) The Input Image

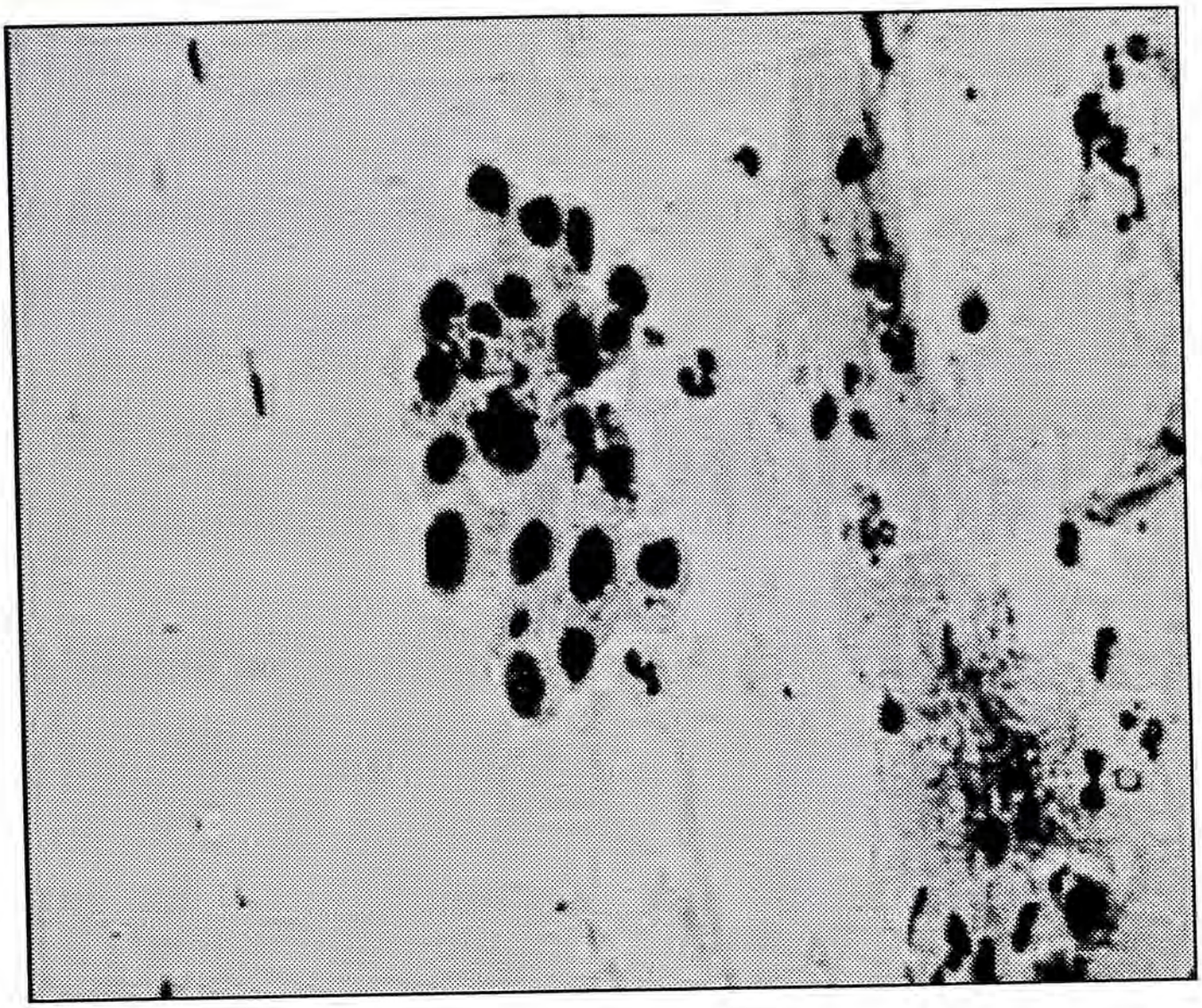

Figure 9.3(b) Input Image after Low Pass Filtering

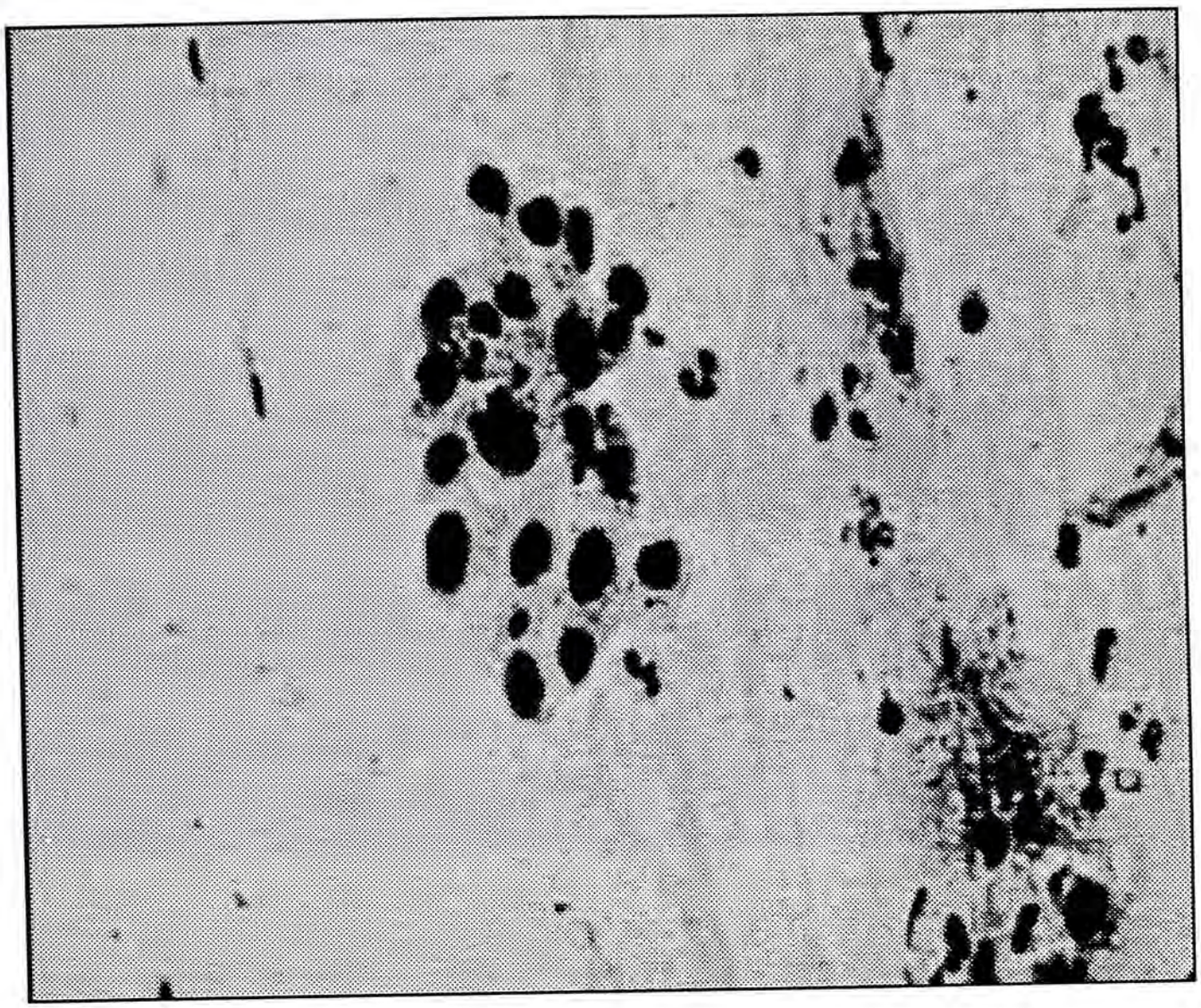

Figure 9.3(c) Input Image with Increasing Brightness

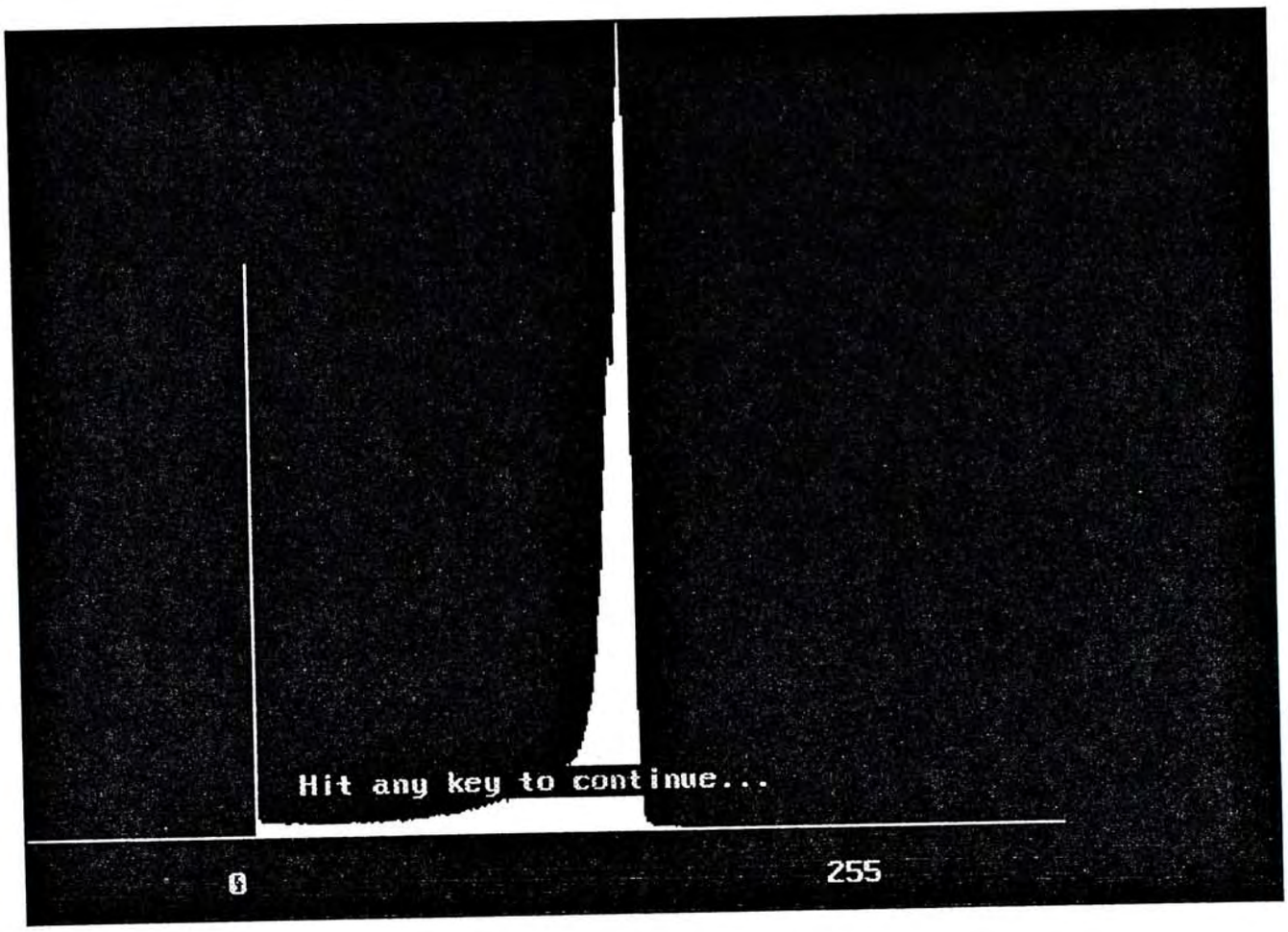

Figure 9.3(d) The Histogram of the Input Image

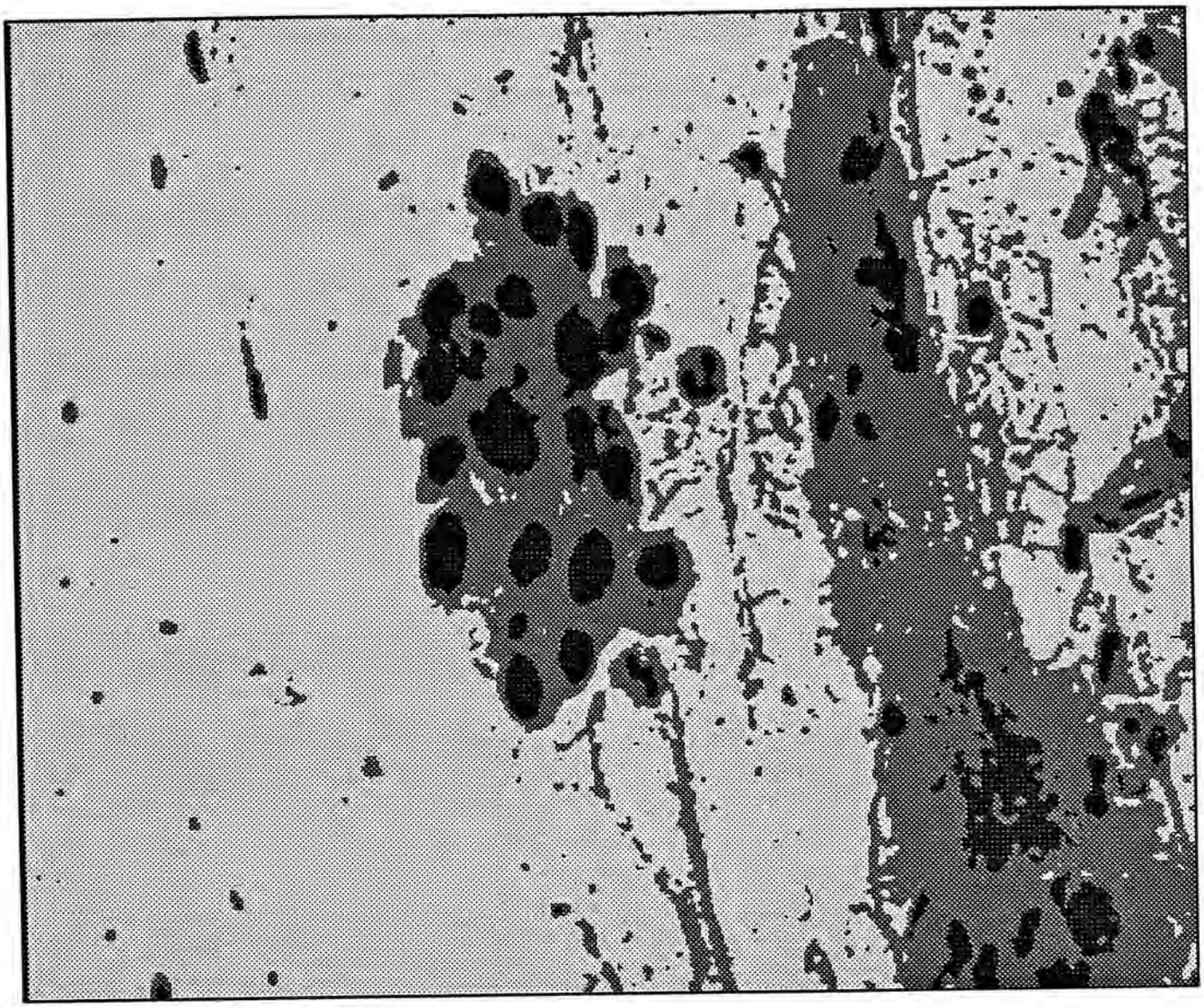

Figure 9.3(e) Image after Applying Multi-Level Threshold

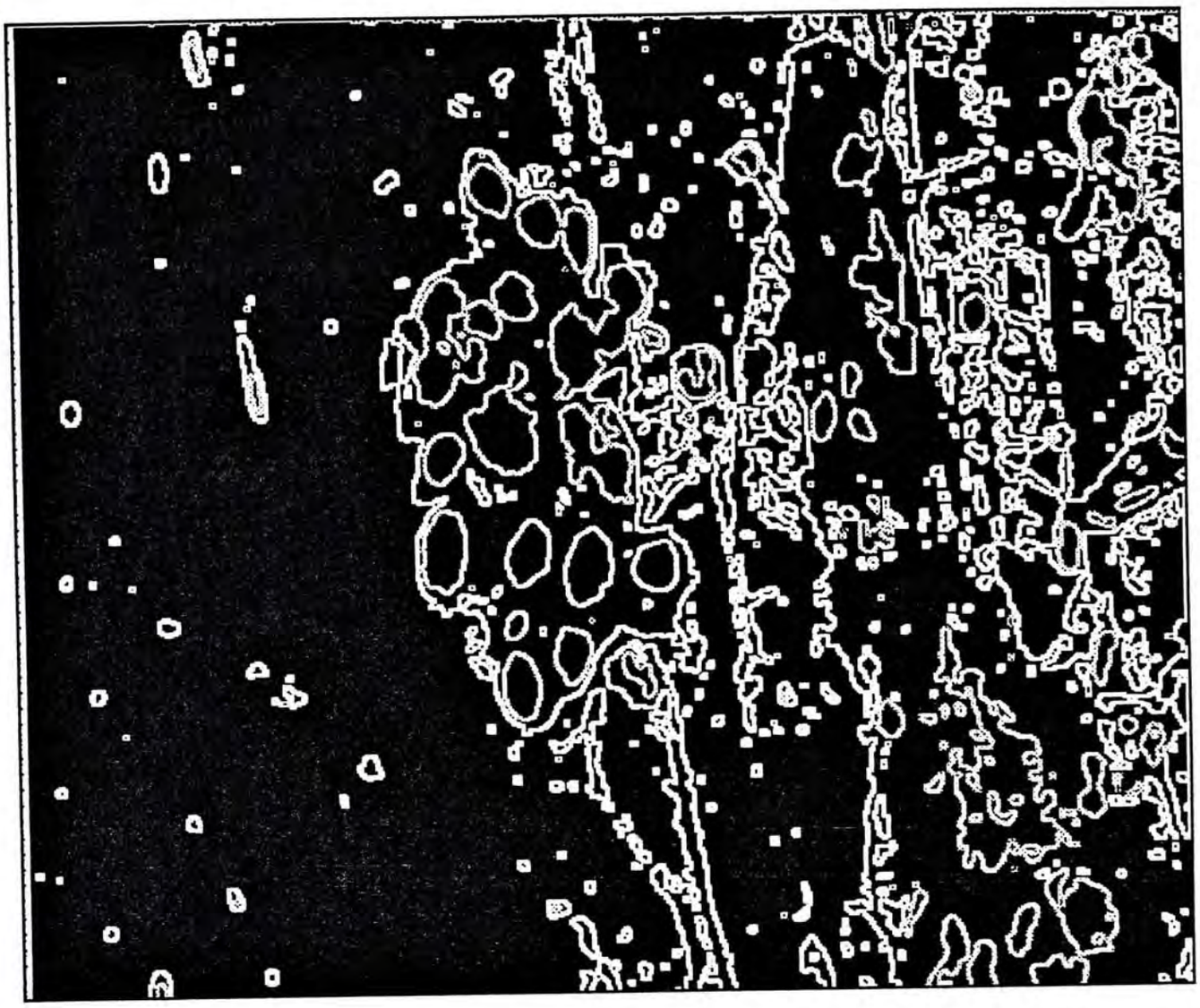

Figure 9.3(f) Image after Sobel Operator

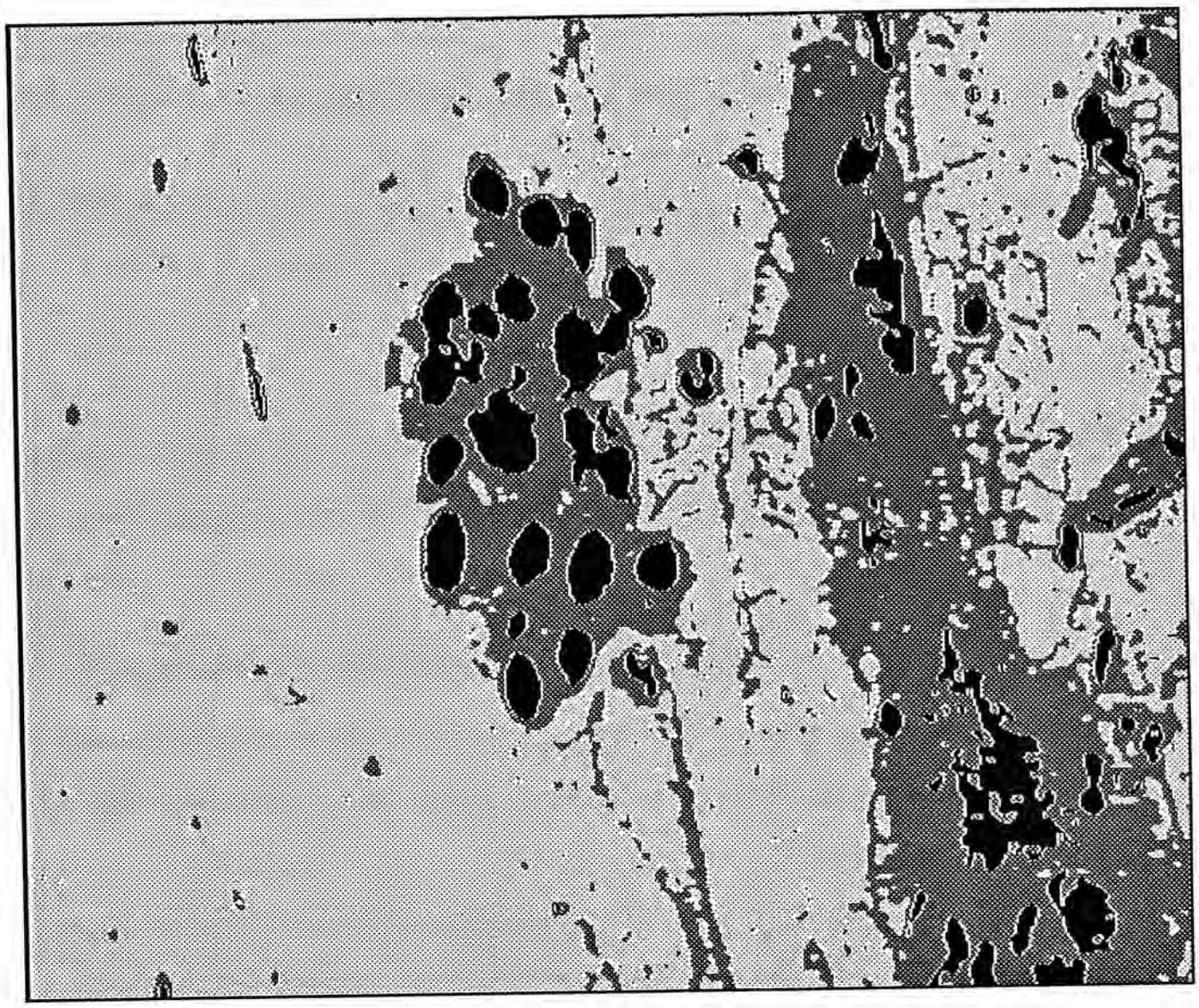

Figure 9.3(g) Image after Region Growing

CHAPTER 10. CONCLUSION

10.1 Summary

Building an image diagnosis expert system is a challenging and sophisticated process. By using an expert system shell, both the knowledge engineer and domain expert need not worry about data manipulation and knowledge representation. Only the relevant domain knowledge is required to be specified. Moreover, the algorithms, the procedures and the control structure will also be handled by the shell. This is the reason why OOI is constructed.

In this research, an object oriented expert system shell for image diagnosis is proposed and implemented. The system separates both the domain knowledge from the control mechanism. The architecture of the shell has been discussed. The shell consists of several generic objects used to describe a two dimensional image. The main achievements of OOI are summarized as follows:

(i) Although there are some object-oriented expert system building tools which are in common use, they do not have any image diagnosis power nor have any features that support spatial information. The object-oriented expert system shell (OOI) is designed to incorporate the image processing abilities. Its power and flexibility especially designed for image recognition and diagnosis has been discussed.

(ii) A variety of low level operators are available in doing pre-processing. A new region growing technique, based on line-adjacency graph, has been developed

for the segmentation preprocessing. Experiments results showed that this growing technique is effective, not only for artificially patterns, but also for complex natural scene as shown in figure 9.1(g), 9.2(g) and 9.3(g). The excellent characteristics of region growing facilitate the extraction of homogeneous regions in the segmentation process.

(iii) Although the image objects may be stored individually as set of independent objects, there are a lot of spatial and conceptual relationships between them. The image objects in OOI are organized in hierarchy with several levels of abstraction. They are stratified into several levels with which image objects in the universe could be represented. It is designed in order to capture the inheritance structure of image data. This object based representation provides a clean method for concept abstraction as well.

(iv) A lot of image functions are built inside the control objects. They are used to calculate the complicate attributes, based on a set of primitives. These functions will surely enhance the capability of the system.

(v) The shell employs a robust control strategy which minimizes the amount of domain-specific control knowledge. A set of knowledge sources are designed. Each of them contains a set of specialized method or meta-knowledge, and works independently of each other in an object-oriented approach. The highly modular architecture of the system enables designer to modify any one of them easily without worrying about any unexpected side-effects.

(vi) Communication between objects are achieved by message passing. Since the number of the image objects is numerous and the messages passing are crucial, all the image objects in OOI adopt a general template in order to ensure the modularity and the consistency in the whole system.

(vii) The strategies in the strategy level of the Scheme Scheduler (SS_object) are used to determine the reasoning mechanism in the system. The mechanism is designed as a highly object-oriented hierarchical approach, working from both bottom-up and top-down.

10.2 Areas of Future Work

All the functions mentioned in the previous chapters have enabled OOI to perform as an efficient and reliable expert system shell in image diagnosis. However, it is admitted that there are many parts to be improved in OOI. Main topics which will contribute to the great advancement of OOI are:

(i) **Parallel Processing**

Image understanding systems include many image processing processes, such as preprocessing, segmentation, and calculation of properties of regions and edges. These processes generally take much time especially for large images. Even though OOI attempts to reduce processing time by processing some primitive attributes in advance, it still requires long processing time for analyzing a large amount of images. The case will be worse if a continue sequence of

images are grabbed and diagnosed. Therefore, it will be necessary to design special hardware architecture and software environment so that OOI can process images in parallel.

(ii) **Introduction of Three Dimensional Information**

OOI can process only two dimensional image. It is incapable in examining any three dimensional objects. Although one can estimate locations of three-dimensional objects by encoding rules about shadow, this does not work well for complex scenes. Therefore, three-dimensional information should be introduced to enhance the power of its recognition power.

APPENDIX A. RULE BASE OF ESCUM

```
[Rule P1
        IF      ( small_area is TRUE ) and
                ( dark is TRUE ) and
                ( background < 40 )
        THEN
                pat_object is nucleus
] Certainty is 0.9

[Rule P2
        IF      ( medium_area is TRUE ) and
                ( gray_level < = 10 )
        THEN
                pat_object is abnormal_nucleus
] Certainty is 0.85

[Rule P3
        IF      ( small_area is TRUE ) and
                ( dark is TRUE ) and
                ( perimeter > 50 )
        THEN
                pat_object is abnormal_nucleus
] Certainty is 0.7

[Rule P4
        IF      ( medium_area is TRUE ) and
                ( medium_gray_level is TRUE )
        THEN
                pat_object is cytoplasm
] Certainty is 0.78

[Rule P5
        IF      ( tiny_area is TRUE ) and
                ( medium_gray_level is TRUE )
        THEN
                pat_object is cytoplasm
] Certainty is 0.7

[Rule P6
        IF      ( medium_gray_level is TRUE ) and
                ( medium_area is TRUE ) and
                ( background > 40 )
        THEN
                pat_object is cytoplasm
] Certainty is 0.95
```

```
[Rule P7
        IF      ( tiny_area is TRUE ) and
                ( perimeter < 50 ) and
                ( low_compact is TRUE)
        THEN
                pat_object is garbage1
] Certainty is 0.85

[Rule P8
        IF      (( large_area is TRUE )  or
                 ( medium_area is TRUE)) and
                ( bright is TRUE )
        THEN
                pat_object is garbage2
] Certainty is 0.8

[Rule P9
        IF      ( tiny_area is TRUE ) and
                ( dark is TRUE ) and
                ( background > 40 )
        THEN
                pat_object is blood_cell
] Certainty is 0.8

[Rule P10
        IF      ( tiny_area is TRUE ) and
                ( dim is TRUE )  and
                ( background > 50 )
        THEN
                pat_object is blood_cell
] Certainty is 0.7

[Rule R1
        IF      ( pat_object is cytoplasm ) and
                ( perimeter > 200 ) and
                ( background > 35 )
        THEN
                reg_object is cytoplasm
] Certainty is 0.85

[Rule R2
        IF      ( pat_object is cytoplasm ) and
                ( perimeter > 200 ) and
                ( medium_gray_level is TRUE )
        THEN
                reg_object is cytoplasm
] Certainty is 0.65
```

```
[Rule R3
        IF      ( pat_object is nucleus  ) and
                ( background > 20 ) and
                ( background < = 60 )
        THEN
                reg_object is nucleus
] Certainty is 0.9

[Rule R4
        IF      ( pat_object is blood_cell ) and
                ( dark is TRUE ) and
                ( background > 20 ) and
                ( background < = 60 )
        THEN
                reg_object is nucleus
] Certainty is 0.7

[Rule R5
        IF      ( pat_object is  abnormal_nucleus ) and
                ( background > 20 ) and
                ( background < = 60 )
        THEN
                reg_object is abnormal_nucleus
] Certainty is 0.9

[Rule R6
        IF      ( pat_object is abnormal_nucleus )
        THEN
                reg_object is abnormal_nucleus
] Certainty is 0.6

[Rule R7
        IF      ( pat_object is nucleus ) and
                ( background > 60 )
        THEN
                reg_object is blood_cell
] Certainty is 0.8

[Rule R8
        IF      ( pat_object is abnormal_nucleus ) and
                ( background > 60 )
        THEN
                reg_object is blood_cell
] Certainty is 0.7

[Rule R9
        IF      (( pat_object is garbage1 ) or
                ( pat_object is garbage2 )) and
                ( background > 100 )
```

```
        THEN
                reg_object is garbage
] Certainty is 0.8

[Rule R10
        IF      (( pat_object is garbage1 ) or
                ( pat_object is garbage2 ))
        THEN
                reg_object is garbage
] Certainty is 0.65

[Rule E1
        IF      ( reg_object is nucleus ) and
                ( high_compact is TRUE )
        THEN
                ent_object is nucleus
] Certainty is 0.9

[Rule E2
        IF      ( reg_object is nucleus )
        THEN
                ent_object is nucleus
] Certainty is 0.7

[Rule E3
        IF      ( reg_object is blood_cell ) and
                ( high_compact > 0.7 )
        THEN
                ent_object is blood_cell
] Certainty is 0.9

[Rule E4
        IF      ( reg_object is blood_cell )
        THEN
                ent_object is blood_cell
] Certainty is 0.7

[Rule E5
        IF      ( reg_object is abnormal_nucleus ) and
                ( compactness > 0.6 )
        THEN
                ent_object is abnormal_nucleus
] Certainty is 0.85

[Rule E6
        IF      ( reg_object is abnormal_nucleus )
        THEN
                ent_object is abnormal_nucleus
] Certainty is 0.7
```

```
[Rule E7
        IF      ( reg_object is cytoplasm ) and
                ( medium_compact is TRUE )
        THEN
                ent_object is cytoplasm
] Certainty is 0.8

[Rule E8
        IF      ( reg_object is cytoplasm )
        THEN
                ent_object is cytoplasm
] Certainty is 0.7

[Rule E9
        IF      ( reg_object is garbage ) and
                ( medium_compact is TRUE )
        THEN
                ent_object is garbage
] Certainty is 0.8

[Rule C1
        IF      ( compactness < = 1 ) and
                ( compactness > 0.6 )
        THEN
                high_compact is TRUE
] Certainty is 0.8

[Rule C2
        IF      ( compactness < = 0.6 ) and
                ( compactness > 0.4 )
        THEN
                medium_compact is TRUE
] Certainty is 0.8

[Rule C3
        IF      ( compactness < = 0.4 ) and
                ( compactness > 0 )
        THEN
                low_compact is TRUE
] Certainty is 0.8

[Rule A1
        IF      ( area > 9000 )
        THEN
                large_area is TRUE
] Certainty is 0.8
```

```
[Rule A2
        IF      ( area > 800 ) and
                ( area < = 9000 )
        THEN
                medium_area is TRUE
] Certainty is 0.8

[Rule A3
        IF      ( area < = 800 ) and
                ( area > 200 )
        THEN
                small_area is TRUE
] Certainty is 0.8

[Rule A4
        IF      ( area < = 200 ) and
                ( area > 0 )
        THEN
                tiny_area is TRUE
] Certainty is 0.8

[Rule G1
        IF      ( gray_level > 80 )
        THEN
                bright is TRUE
] Certainty is 0.8

[Rule G2
        IF      ( gray_level < = 80 ) and
                ( gray_level > 35 )
        THEN
                medium_gray_level is TRUE
] Certainty is 0.8

[Rule G3
        IF      ( gray_level > 12 ) and
                ( gray_level < = 35 )
        THEN
                dim is TRUE
] Certainty is 0.8

[Rule G4
        IF      ( gray_level < =12 )
        THEN
                dark is TRUE
] Certainty is 0.8
```

```
[Rule H1
        IF      ( sexual_disease is TRUE) and
                ( age_of_first_sex < 20 ) and
                ( NC_Ratio > 0.36 )
        THEN
                Cell_Diagnosis is cancer
]Certainty is 0.9

[Rule H2
        IF      ( abnormal_bleeding is TRUE) and
                ( sexual_partner > 2 ) and
                ( NC_Ratio > 0.25 ) and
                ( NC_Ratio < = 0.36 )
        THEN
                Cell_Diagnosis is CIN_III
]Certainty is 0.85

[Rule H3
        IF      ( alcoholic is TRUE )  and
                ( smoker is TRUE ) and
                ( NC_Ratio > 0.25 ) and
                ( NC_Ratio < = 0.36 )
        THEN
                Cell_Diagnosis is CIN_III
]Certainty is 0.7

[Rule H4
        IF      ( abnormal_bleeding is TRUE ) and
                ( number_of_pregnancy > 4 ) and
                ( NC_Ratio > 0.15 ) and
                ( NC_Ratio < = 0.25 )
        THEN
                Cell_Diagnosis is CIN_II
]Certainty is 0.75

[Rule H5
        IF      ( number_of_pregnancy > 4 ) and
                ( sexual_partner > 2 ) and
                ( smoker is TRUE ) and
                ( NC_Ratio > 0.25 ) and
                ( NC_Ratio < = 0.36 )
        THEN
                Cell_Diagnosis is CIN_III
]Certainty is 0.9

[Rule H6
        IF      ( number_of_pregnancy > 4 ) and
                ( sexual_disease is TRUE ) and
                ( NC_Ratio > 0.15 ) and
                ( NC_Ratio < = 0.25 )
```

```
        THEN
                Cell_Diagnosis is CIN_II
]Certainty is 0.8

[Rule H7
        IF      ( age_of_first_sex < 20 ) and
                ( NC_Ratio > 0.36 )
        THEN
                Cell_Diagnosis is cancer
]Certainty is 0.8

[Rule H8
        IF      ( alcoholic is TRUE ) and
                ( number_of_pregnancy  > 4 ) and
                ( NC_Ratio > 0.15 ) and
                ( NC_Ratio < = 0.25 )
        THEN
                Cell_Diagnosis is CIN_II
]Certainty is 0.6

[Rule H9
        IF      ( sexual_disease is TRUE) and
                ( abnormal_bleeding is TRUE ) and
                ( NC_Ratio > 0.25 ) and
                ( NC_Ratio < = 0.36 )
        THEN
                Cell_Diagnosis is CIN_III
]Certainty is 0.95

[Rule H10
        IF      ( sexual_disease is TRUE ) and
                ( abnormal_bleeding is TRUE ) and
                ( NC_Ratio > 0.36 )
        THEN
                Cell_Diagnosis is cancer
]Certainty is 0.95

[Rule H11
        IF ( NC_Ratio < 0.1 )
        THEN Cell_Diagnosis is normal
]Certainty is 0.8

[Rule H12
        IF      ( sexual_disease is TRUE) and
                ( abnormal_bleeding is TRUE ) and
                ( NC_Ratio > 0.1 ) and
                ( NC_Ratio < = 0.15 )
        THEN
                Cell_Diagnosis is CIN_I
]Certainty is 0.9
```

APPENDIX B. GLOSSARY FOR OBJECT-ORIENTED PROGRAMMING

Class

A class is a description of one or more similar objects. For example, the class Apple, is a description of the structure and behavior of instances, such as appple-1 and apple-2. Loops and Smalltalk classes describe the instance variables, class variables, and methods of their instances as well as the position of the class in the inheritance lattice.

Class Inheritance

When a class is placed in the class lattice, it inherits variables and methods from its superclasses. This means that any variable that is defined higher in the class lattice will also appear in instances of this class. If a variable is defined in more than one place, the overriding value is determined by the inheritance order. The inheritance order is depth-first up to joins, and left-to-right in the list of superclasses.

Class Variable

A class variable is a variable stored in the class whose value is shared by all instances of the class.

Composite Object

A group of interconnected objects that are instantiated together, a recursive extension of the notion of object. A composite is defined by a template that describes the subobjects and their connections.

Data Abstraction

The principle that programs should not make assumptions about implementations and internal representations. A composite is defined by a template that describes the sub-objects and their connections.

Default Value

A value for an instance variable that has not been set explicitly in the instance. The default value is found in the class, and tracks that value until it is changed in the instance. This contrasts with initial values.

Delegation

A technique for forwarding a message off to be handled by another object.

Initial Value

A value for an instance variable that is computed and installed in the instance at object creation. Different systems provide initial values and/or default values.

Instance

The term 'instance' is used in two ways. The phrase 'instance of' describes the relation between an object and its class. The methods and structure of an instance are determined by its class. All objects in Loops (including classes) are instances of some class. The noun 'instance' refers to objects that are not classes.

Instantiate

To make a new instance of a class.

Instance Variable

Instance variables (sometimes called slots) are variables for which local storage is available in instances. This contrasts with class variables, which have storage only in the class. In some languages instance variables can have optional properties.

Lattice

In this document we are using 'lattice' as a directed graph without cycles. In Loops, the inheritance network is arranged in a lattice. A lattice is more general than a tree because it admits more than one parent. Like a tree, a lattice rules out the possibility that a class can (even indirectly) have itself as a superclass.

Metaclass

This term is used in two ways: as a relationship applied to an instance, it refers to the class of the instance's class; as a noun it refers to a class all of whose instances are classes.

Message

The specification of an operation to be performed on an object. Similar to a procedure call, except that the operation to be performed is named indirectly through a selector whose interpretation is determined by the class of the object, rather than a procedure name with a single interpretation.

Method

The function that implements the response when a message is sent to an object. In Loops, a class associated selectors with methods.

Mixin

A class designed to augment the description of its subclasses in a multiple inheritance lattice. For example, the mixin NamedObject allocates an instance variable for holding an object's name, and connects the value of that variable to the object symbol table.

Object

The primitive element of object-oriented programming. Objects combine the attributes of procedures and data. Objects store data in variables, and respond to messages by carrying out procedures (methods).

Perspective

A form of composite object interpreted as different views on the same conceptual entity. For example, one might represent the concept for 'Joe' in terms of JoeAsAMan, JoeAsAGolfer. JoeAsAWelder, JoeAsAFather. One can access any of these by view name from each of the others.

Polymorphism

The capability for different classes of objects to respond to exactly the same protocols. Protocols enable a program to treat uniformly objects that arise from different classes. A critical feature is that even when the same message is sent from the same place in code, it can invoke different methods.

Protocol

A standardized set of messages for implementing something. Two classes which implement the same set of messages are said to follow the same protocol.

Slot

See instance variable

Specialization

The process of modifying a generic thing for a specific use.

Subclass

A class that is lower in the inheritance lattice than a given class.

Superclass

A class that is higher in the inheritance lattice than a given class.

REFERENCES

[Anderson87] M.C.Anderson, "Premalignant and malignant disease of the cervix" in *Haines & Tayler Obstetrical & Gynaecological Pathology,* 3rd. ed. Churchill, Livingstone, 1987.

[Baak84] J.P.A.Baak, "Basic points in and practical aspects of the application of diagnostic morphometry" in *Pathology Res. Practice,* No.179, pp.193-199, 1984.

[Bailn89] S.C.Bailn, "An object-oriented requirement specification method" in *Communications of the ACM,* Vol. 32, No.5, pp.608-623, 1989.

[Barrow76] H.G.Barrow and J.M.Tenenbaum, " MSYS: A System for reasoning about scenes" in *Technical Note 121,* SRI International, Menlo Park, Calif., April 1976.

[Bartlett32] F.Bartlett, *Remembering, A Study in Experimental and Social Psychology,* London, Cambridge University Press, 1932.

[Bhanu89] Bhanu Prasad Pokkunuri, "Object oriented programming" in *ACM SIGPLAN Notices,* Vol.24, No.11, pp. 96-101, Nov., 1989.

[Bobrow77] D.Bobrow & T.Winograd, "KRL, Another perspective", in *Cognitive Science 3,* pp. 29-42, 1977. (KRL = Knowledge Representation Language).

[Bobrow83] D.Bobrow & M.Stefik, *The LOOPS Manual,* Xerox Palo Alto Research Centre, Palo Alto, 1983.

[Bobrow85] D.Bobrow, K.Kahn, G.Kiczales, L.Masinter, M.Stefik & F.Zdybel, "CommonLoops: merging common LISP and object-oriented programming" in *Report ISL-85-8,* Xerox Palo Alto Research Centre, Palo Alto, 1985.

[Booch86] G.Booch, "Object-oriented development" in *IEEE Transactions on Software Engineering,* Vol. SE-12, No.2, pp.211-221, Feb., 1986.

[Booch87] G.Booch, *Software Engineering with ADA: 2nd Ed.,* The Benjamin-Cummings Publishing Inc., Reading, MA, 1987.

[Brachman85] R.J.Brachman & H.J.Levesque, *Readings in Knowledge Representation,* Morgan Kaufmann: Los Altos, CA., 1985.

[Brooks84] R.A.Brooks, *Model-Based Computer Vision,* (Computer Science : Artificial Intelligence No.14), UMI Research Press, 1984.

[Buckley82] C.H.Buckley, E.B.Butler & H.Fox, "Cervical intraepithelial neoplasia" in *Journal of Clinical Pathology,* Vol.35, pp.1-13, 1982.

[Chapman87] D.Chapman, "Planning for conjunctive goals." in *Artificial Intelligence,* Vol.29, pp.333-377, July, 1987.

[Coad90] P.Coad & E.Yourdon, *Object-Oriented Analysis,* Yourdon Press, Prentice Hall, Englewood Cliffs, NJ, 1990.

[Cointe83] P.Cointe & X.Rodet, "FORMES: a new object-language for managing hierarchy of events", in *IFIP-83*, Paris, 1983.

[Cox86] B.J.Cox, *Object-Oriented Programming: An Evolutionary Approach,* Addison-Wesley, Reading Mass., 1986.

[Duff83] M.J.B.Duff, "Neighbourhood Operators ", in *Physical and Biological Processing of Images*, edited by O.J.Braddick and A.C.Sleigh, Springer-Verlag, Berlin, 1983.

[Ehrich88] H.D.Ehrich, A.Sernadas and C.Sernadas, "Abstract object types for databases" in *Advances in Object-Oriented Database System*, edited by K.Dittrich, Springer Verlag, New York, pp.144-149, 1988.

[Faloutsos87] C.Faloutsos, T.Sellis and N.Roussopoulos, "Analysis of object oriented spatial access methods" in *ACM SIGMOD (Management of Data)*, Vol.16, No.3, pp.426-439, Dec.,1987.

[Feigenbaum77] E.A.Feigenbaum, "The art of artificial Intelligence : themes and case studies of knowledge engineering", in IJCAI 5, pp.1014-1029, 1977.

[Feigenbaum78] E.A.Feigenbaum & H.Penny Nii, "Rule-based understanding of signals" in *Pattern-Directed Inference Systems*, edited by D.A.Waterman and F.Hayes-Roth, pp.483-501, Academic Press, 1978.

[Ferenczy82] A.Ferenczy, "Cervical intraepithelial neoplasia" in *Pathology of the Female Genital Tract*, edited by A.Blaustein, 2nd ed., Springer-Verlag, New York, Berlin, pp.156-177, 1982.

[Fikes85] R.Fikes and T.Kehler, "The role of frame-based representation in reasoning" in *Communication of ACM*, Vol.28, No.9, pp. 904-920, Sept., 1985.

[Fisz63] M.Fisz, *Probability Theoy and Mathematical Statistics*, John Wiley and Sons, New York, 1963.

[Frisch82] A.M.Frisch and J.F.Allen, "Knowledge retrieval as limited inference" in *Proceeding, sixth Conference on Automated Deduction*, edited by D.W.Loveland, Springer, New York, 1982.

[Fu74] K.S.Fu, *Syntactic Methods in Pattern Recognition,* Academic Press, New York, 1974.

[Gallagher88] J.P.Gallagher, *Knowledge Systems for Business, Integrating Expert Systems and MIS*, Prentice Hall, 1988.

[Garvey76] T.D. Garvey, "Perceptual strategies for purposive systems" in *Technical Note 117,* SRI International, Menlo Park, Calif., September 1976.

[Gevarter87] W.B.Gevarter, "The nature and evaluation of commercial expert system building tools", in *IEEE COMPUTER,* pp.24-41, May, 1987.

[Goldberg83] A.Goldberg and D.Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, 1983.

[Greene89] D.Greene, "An implementation and performance analysis of spatial data access methods" in *IEEE Data Engineering Conference*, pp.606-615, 1989.

[Guttman84] A.Guttman "R-trees: A dynamic index structure for spatial searching" in *ACM SIGMOD (Management of Data)*, Vol.14, No.2, pp.47-57, 1984.

[Hanson78] A.R.Hanson & E.M.Riseman, "VISIONS: A computer system for interpreting scenes" in *Computer Vision Systems*, edited by A.R.Hanson and E.M.Riseman, pp.303-334, Academic Press, New York, 1978.

[Haralick79] R.M.Haralick, "Statistical and Structural Approaches to Textures" in *Proceedings, IEEE,* Vol.67, pp.786-804, 1979.

[Ham74] A.W.Ham, *Histology,* 7th ed. Lippincott, Philadelphia, 1974.

[Hendler88] J.Hendler, "Designing Interfaces for Expert System" in *Expert Systems: The User Interface,* edited by J.A.Hendler, pp.1-13, Ablex Publishing Corporation, 1988.

[Huang79] T.S.Huang, G.J.Yang and G.Y.Tang, "A fast two-dimensional median filtering algorithm", in *IEEE Trans. On Acoustic, Speech and Signal Processing, ASSP,* Vol.27, No.1, Feb., 1979.

[Imaging87] Imaging Technology Inc, *PCVISIONPlus Frame Grabber User's Manual,* Massachusetts, 1987.

[Ioannidis84] A.Ioannidis, D.Kazakos and D. Watson, " Aication of median filtering on nuclear medicine scintigram images," in *IEEE Trans. on Pattern Recognition and Image Processing,* No.1, pp.33-36, 1984.

[Jackson86] P.Jackson, *Introduction To Expert Systems*, Addison-Wesley, 1986.

[Koss79] L.G.Koss, *Diagnostic Cytology and its Histopathologic Bases.* 3th ed. Lippincott, Philadelphia, 1979.

[Lai89] S.M.Lai, X.Li and W.F.Bischof "Automated detection of breast tumors" in *Computer Vision and Shape Recognition,* edited by A.Krzyzak, T.Kasrand and C.Y.Suen, World Scientific Press, pp.115-132, 1989.

[Leung90] K.S.Leung & M.H.Wong, "An expert-system shell using structured knowledge" in *IEEE Computer*, Vol.23, No.3, pp.38-47, March, 1990.

[Levine78] M.D.Levine, "A Knowledge-based computer vision system," in *Computer Vision Systems*, edited by A.Hanson and E.Riseman, New York, Academic Press, 1978.

[Levine80] M.D. Levine, "Region analysis using a pyramid data structure" in *Structured Computer Vision*, edited by S.Tanimoto and A.Klinger, Academic Press, pp. 57-100, 1980.

[Mckeown85] D.A.Mckeown, W.A.Harvey & J.Mcdermott, "Rule-based interpretation of aerial imagery" in *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.7, No.5, pp.570-585, 1985.

[Meyer88] B.Meyer, *Object-Oriented Software Construction,* Prentice Hall, Englewood Cliffs, NJ, 1988.

[Minsky75] M.Minsky, "A framework for representing knowledge" in *The Psychology of Computer Vision*, edited by P.H.Winston, McGraw Hill, pp. 211-277, 1975.

[Mohan88] L.Mohan and R.L.Kashyap, "An object-oriented knowledge representation for spatial information" in *IEEE Transaction on Software Engineering*, Vol.14, No.5, pp.675-681, May, 1988.

[Moon80] D.A.Moon & D.Weinreb, "FLAVORS: message-passing in the LISP machine", MIT, *AI Memo 602*, 1980.

[Myler88] H.R.Myler, "Object-Oriented Training Simulation," in *AI Paper 1988: Proc. Conf. AI and Simulation,* R.J.Uttamsingh, ed., Society for Computer Simulation, San Diego, Calif., 1988, pp.156-160.

[Nagao80] M.Nagao and T.Matsuyama, *A Structural Analysis of Complex Aerial Photographs*, Plenum Press, 1980.

[Nazif84] A.Nazif, M.D.Levine, "Low level image segmentation: An expert system" in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.6, No.5, pp.555-777, 1984.

[Novak74] E.R.Novak & J.D.Woodruff, *Novak's Gynecologic and Obstetric Pathology*, 7th ed. W.B.Saunders, Philadeliphia, 1974.

[Pascoe86] G.A.Pascoe, "Elements of object-oriented programming" in *Byte*, Vol.11, No.8, pp.139-144, 1986.

[Piper88] J.Piper and D.Rutovitz, "An investigation of object-oriented programming as the basis for an image processing and analysis system" in *Proceedings, the ninth International Conference on Pattern Recognition*, IEEE Computer Society Press, pp.1015-1019, 1988.

[Pratt78] W.K.Pratt, *Digital Image Processing*, John Wiley and Sons, New York, 1978.

[Quillian68] R.M.Quillian, "Semantic memory" in *Semantic Information Processing*, edited by M.Minsky, Cambridge, MIT Press, 1968.

[Richarts67] R.M.Richarts, "Natural history of cervical intraepithelial neoplasia" in *Clin. Obstetric Gynec,* Vol.10, pp.748-784, 1967.

[Roberts77] B.Roberts,I.Goldstein, "The FRL manual", in *MIT AI Laboratory Memo 409*, September, 1977.(FRL = Frame Representation Language)

[Rosch75] E.Rosch, "Cognitive representations of semantic categories", in *Journal of Experimental Psychology*, No.104, pp.192-233, 1975.

[Sacerdoti74] E.D.Sacerdoti, "Planning in a hierarchy of abstraction spaces" in *Artificial Intelligence,* vol.5, 1974.

[Schank77] R.Schank & R.Abelson, *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum Ass., Hillsdale, NJ.

[Shani80] U.Shani, "Filling regions in binary raster images: A graph-theoretic approach", in *SIGGRAPH Proceedings*, pp.321-327, 1980.

[Shastri88] L.Shastri, *Semantic Networks: An Evidential Formalization and Its Connectionist Realization*, CA/Pitman, London, 1988.

[Steels83] L.Steels, "ORBIT: An applicative view of object-oriented programming", in *Proc. European Conference on Integrated Interactive Computing Systems*, Stresa, Italy, 1-3 September 1982. North-Holland, Amsterdam.

[Stefik79] M.Stefik, "An examination of a frame-structures representation system" in *IJCAI-79*, Tokyo, 1979.

[Stefik86] M.Stefik and D.G.Bobrow, "Object-oriented programming: Themes and variations" in *AI Magazine*, Vol.6, No.4, pp. 40-62, 1986.

[Stroustrup88] B.Stroustrup, "What is Object-oriented programming ?" in *IEEE Software*, Vol.5, No.3, May, 1988.

[Szolovits78] P.Szolovits & S.G.Pauker, "Categorical and probabilistic reasoning in medical diagnosis", in *Artificial Intelligence*, Vol.11, pp.115-144, 1978.

[Tanimoto75] S.L.Tanimoto and T.Pavlidis, "A hierarchical data structure for picture processing"in *Computer Graphics and Image Processing*, Vol.4, No.2, pp. 104-119, June, 1975.

[Wai90] C.W.G.Wai, *Morphometric Studies of Intraepithelial Neoplasia and Associated Lesions in the Cervix Uteri and the Nasopharynx*, M.Phil. Thesis, Division of Clinical and Pathological Science, Chinese University of Hong Kong, 1990.

[Zadeh65] L.A.Zadeh, *Fuzzy Sets, Information and Control,* Vol.8, pp.338-353, 1965.

[Zadeh83] L.A.Zadeh, "The role of fuzzy logic in the management of uncertainty in expert systems" in *Fuzzy Sets and Systems,* Vol.11, pp.199-221, 1983.

CUHK Libraries
000325556