Design and Implementation

of an Image Understanding System:

DADS
Design and Implementation
of an Image Understanding System:
DADS
Proefschrift

ter verkrijging van de graad van doctor aan de
Technische Universiteit Delft, op gezag van
de Rector Magnificus, prof. drs. P.A. Schenck,
in het openbaar te verdedigen ten overstaan van
een commissie aangewezen door het College van
Dekanen op donderdag 29 maart 1990 te 16.00 uur.

door

Xiang Sheng Cheng

geboren te Shanghai, China

wiskundig ingenieur
Dit proefschrift is goedgekeurd door de promotor
prof. dr. ir. E. Backer

dr. ir. J.J. Gerbrants heeft als toegevoegd promotor
in hoge mate bijgedragen aan het totstandkomen van het proefschrift.
to Wei-Ling
Contents

Contents ..................................................................................................................... vii
Summary .................................................................................................................... xi

I DADS: A Distributed and Anomaly-Driven System for General Image Understanding 1

1 Introduction ............................................................................................................ 3
   1.1 The Issue of Image Understanding ................................................................. 3
   1.2 Past Achievements ......................................................................................... 7
   1.3 The Scope of PART I .................................................................................... 15

2 Image-Understanding Modelling ........................................................................ 17
   2.1 Preliminary Remarks on Certain Aspects of Visual Perception .................. 17
   2.2 DADS Modelling ......................................................................................... 21
   2.3 Context Hierarchy and Processing Blackboxes ............................................ 28
   2.4 Anomaly and Anomaly Handlers ................................................................. 35
   2.5 Further Discussion ....................................................................................... 37

3 System Design for DADS ............................................................................... 43
   3.1 Representational Specification of Data Classes ........................................... 44
   3.2 System Description for Context Hierarchy .................................................. 50
   3.3 Design Specification on Processing Clusters ............................................... 55
   3.4 Global Architecture and Overall Controlling of DADS ............................... 60
      3.4.1 Initialization of the System Configuration .............................................. 62
      3.4.2 Global Memory Management ............................................................. 63
3.4.3 Overall Process Control within a DADS-Based Application System ............................................. 66

4 Concluding Comments and Discussions ................................................................. 73

II A DADS Application for the Interpretation of SLAR Images .............................................. 77

5 Introduction ................................................................................................................. 79
  5.1 Background of the SLAR-Image Problem .......................................................... 80
  5.2 Preliminary Steps Towards Knowledge Exploration ......................................... 82
  5.3 Combined Segmentation Approach ................................................................. 85

6 Edgeness Detector: A New Method for Extracting Regional Boundaries ................. 93
  6.1 Introductory Review of the Edge-Detection Problem ...................................... 93
  6.2 Edgeness: a New Concept for Edge Definition ............................................... 98
  6.3 Case Study in the Continuous Space .............................................................. 108
  6.4 Practical Definition and Implementational Aspects .......................................... 118
  6.5 Post-processing of an Edgeness Image ............................................................. 127
    6.5.1 Thresholding an Edgeness Image .............................................................. 128
    6.5.2 Iteration-Based Adaptive Shrinking Algorithm:
       A Smart Approach .............................................................................................. 129

7 Constrained Region Growing ...................................................................................... 137
  7.1 The Split-and-Merge Algorithm ....................................................................... 138
  7.2 Edge-Constrained Region-Growing Based Segmentation ................................ 141

8 High-Level Interpretation through Anomaly Handling ............................................. 149
  8.1 Extraction of Crop Fields Following the Rational Reasoning .......................... 149
  8.2 Refinement of the Extracted Crop Fields through Anomaly Handling ................ 154
    8.2.1 Detection of the Anomalies .................................................................... 157
    8.2.2 Handling of the Detected Anomalies ...................................................... 163
  8.3 Discussion ............................................................................................................ 167

9 Concluding Comments ............................................................................................... 173
Summary

Image understanding by computer is a process which proceeds from raw digital images to a symbolic description of the contents of the image scene. It is, for instance, easy, even for a child, to discover the presence of a house on a usual screen. Nonetheless, even an experienced adult will feel it awkward if he\(^1\) is asked why that part of the screen represents a house. Moreover, he is going to be even more embarrassed when that particular part of the screen is scaled out so that all constituent points (i.e., pixels) are individually visible. Unfortunately, such an out-scaled version is precisely what a computer has at its disposal and the why here is necessary for us (as human beings) to explain to the computer in one or another way if the computer is asked to understand the house in the image. Basically, image understanding by computer involves two extremely difficult issues. One is the way to express or represent our reasoning with perception in a logical way. For instance, a particular combination of a roof, walls, windows and doors suggests the presence of a house. The other is the reconciliation between primitive concepts which are familiar to a human perceiver and the numerical features which are the only things a computer knows to manipulate. For instance, what constitution of discrete points stands for a line segment?

In the first part of this thesis, a general system design, i.e., the Distributed and Anomaly-Driven System (DADS), is proposed for the purpose of building up a general image-understanding system. Within this framework, two primary notions are introduced, i.e., the notion of context hierarchy and that of anomaly, respectively. By means of the context hierarchy, the understanding process of a particular problem can be described through intermediate meta-objects. The objective of doing this is to explore maximally the rational part of our thinking which has to do with perception, and yet in such a way that it is in our view friendly to

\(^1\)With he we mean both she and he throughout this thesis.
a human user and manageable by a computer. As noted earlier, describing a perceptual reasoning process along a logical path and characterizing a perceptual primitive in a numerical way are by no means simple tasks even for a rather constrained problem. Impreciseness and insufficiency are unavoidable. Quite often, the impreciseness and insufficiency are discovered only after attempting to understand the image contents by a process under the established context hierarchy. This is particularly true if we note that the numerical characterization of a particular pictorial primitive generally fails to consider all possible variations and unexceptionable irregularities which happen to be common to a digital image of a natural scene. To deal with such phenomena, DADS provides an anomaly-driven mechanism based on the notion of anomaly. By means of this mechanism, one can improve the system performance by assessing various insufficiencies of the current system and accordingly, by adding appropriate anomaly handlers to counter those insufficiencies (i.e., anomalies in the DADS terminology) on a dynamic basis. In this way, we can establish flexible interaction between various processing levels, which are traditionally separated. Assessing a particular class of anomalies involves that part of our thinking of which we are generally unaware or which is uncommon to us (e.g., when the anomalies are caused purely by the inappropriateness of the discrete grid). We call it the irrational part of our thinking with perception. The major merit of the anomaly-driven mechanism is that the assessed anomalies can be effectively dealt with without a significant increase in system complexity and processing time.

In Chapter 2, we start with discussing certain aspects of human visual perception. In relation to image understanding by computer, we then present the modelling of DADS and discuss its various constituent parts in an attempt to show that such a model is natural with respect to the process of understanding pictorial contents and is in a practical sense suitable in respect to image understanding by computer.

In Chapter 3, we give a detailed presentation on the DADS design with emphasis on the system framework. Beside the global architecture and the system control, it involves also data representations, processing specifications and system specifications with respect to any actual application of the DADS framework.

In Chapter 4, concluding comments are given. In particular, various properties of DADS are summarized, which can be observed in or expected from the previous presentations. Also, issues in the future development on DADS are indicated.
In the second part of the thesis, we concentrate on an example of a DADS-based application system for interpreting SLAR images (i.e., Side-Looking Airborne Radar images). The objective is not to give a complete solution to such a complex interpretation problem. Rather, it aims at gaining some practical insights into the DADS architecture and at evaluating its suitability and potential towards this tough problem.

In Chapter 5, the SLAR-image problem is introduced and formulated. The issue of low-level image segmentation which played a dominant role in the past research on the SLAR-image problem is discussed in a formal way and, in particular, by discussing various background aspects a new segmentation definition is given, based on the combined approach. With respect to DADS application, the initial system setup (i.e., the initial context hierarchy) is given.

In Chapter 6, the Edgeness Detector used to detect regional boundaries in an SLAR image is introduced and fully discussed from theoretical foundation to practical design strategy in the discrete domain. As a means to post-process an edgeness image, the Iteration-Based Adaptive Shrinking Algorithm is presented.

In Chapter 7, emphasis is on the issue of extracting tentative regions in an SLAR image. In particular, the Edge-Constrained Region-Growing Based Segmentation approach for achieving a better low-level segmentation is presented.

In Chapter 8, the extraction of crop fields versus non-crop fields in an SLAR image is presented. Especially, the utilization of the anomaly-driven mechanism within DADS is explored by considering two simple types of assessed anomalies. As an illustrative example, a DADS-based application system for interpreting SLAR images is established.

Through our experiment with the current DADS-based application system, we have observed the effectiveness and the flexibility of the anomaly-driven mechanism. These initial results lead us to expect that the overall system performance can be much improved by assessing more classes of anomalies in the subsequent research. We are confident that the DADS framework has a strong potential for use as a sophisticated system for the purpose of image understanding.

Parallel to the realization of the current DADS-based application system, we have also studied some basic image-processing problems and made contributions in that respect. In particular, we propose a new edge detector called Edgeness Detector. Also, the proposals of the Iteration-Based Adaptive Shrinking Algorithm and the Edge-Constrained Region-
Growing Based Segmentation approach are among the outcomes of this research. The experimental results of these new methods applied to various types of heavily noise-corrupted images have proved to be quite good. Undoubtedly, these new methods are generally useful and valuable in many other applications.
Part I

DADS: A Distributed and Anomaly-Driven System for General Image Understanding
Chapter 1

Introduction

1.1 The Issue of Image Understanding

With the rapid development of digital processing capabilities in the last several decades, many methods to acquire, improve and process digital images have been introduced and developed with varying degrees of success. Increasingly, attempts are made to have a machine not only digitize a real scene and subsequently process it as a 2D signal along a specified path, but also to be able automatically to 'interpret' or 'understand' what is present in the scene. Though the step from the conventional processing to such a flexible 'understanding' does not appear to be so much difficult as to become infeasible, for many researchers it turns out to be much more difficult and involve a much broader range of quite different disciplines (like signal processing, computer science, psychology, neurology, perception and cognitive sciences) than many would originally have imagined (see for instance [Fis87, Lev85a]). Many may become frightened by a plausible fact that the more serious research is done, the more the awareness of difficulties grows. Without ever thinking about the infeasibility of creating such a human-like understanding capability in a man-made machine, people will naturally question why the reverse should not be true, given so many exciting scientific successes like the tracks of mankind on the moon, the development of nuclear energy and many others, which would all have been thought of as unrealistic or absurd even as far back as the previous century.

Essentially, image understanding by computer can be considered as the extraction of meaningful entities or the confirmation of the absence of some meaningful entities within the given image(s), as well as giving a description of their interrelationships in the context of the original scene.
which is represented by the image(s). In practice, it is true that the issue of image understanding by computer is mainly investigated by researchers in the fields of image processing and pattern recognition. However, one should be aware of several essential aspects of such an issue. Firstly, an 'improved' image achieved through some magnificent image-processing techniques does not necessarily contribute in any substantial way to the ultimate goal of understanding. Secondly, a well-considered recognition scheme may be based on a wrong modelling assumption. Finally, the most crucial aspect may probably be the fact that the result is often judged inadequately by some seemingly appropriate criteria. For instance, in the example shown in Figure 1.1 we encounter a problem of extracting a convex curve whose true appearance is shown in Figure 1.1a. Figure 1.1b and Figure 1.1c show two possible results from a practical image-understanding operation. In an attempt to justify these results, Figure 1.1b may prove to be 'better' than Figure 1.1c under the usual Least Squared Error criterium (LSE) whereas a convexity-based criterium may tell us the reverse. In reality, it is difficult for us to judge which of the possible numerical criteria is actually suitable for a particular application.

![a. the truth](image1)

![b. result 1](image2)

![c. result 2](image3)

Figure 1.1: A simple illustrative example: *which numerical criterium is suited for judging the results?*

Why is the problem of image understanding basically so tricky and so difficult? We attempt to explain this by discussing two of its aspects.

On the one hand, the output of an image-understanding process is generally desired to be a set of entities in the scene which is represented by the input image(s). These entities may in some cases correspond to some well-defined objects like, for instance, a machine part in the application of industrial robot vision. In other cases, they may correspond to some seemingly well-described objects like a water zone or a forest area in the field of aerial picture analysis. Anyhow, these entities are, in our mind, described or characterized mainly in the following four ways whenever we
1.1 The Issue of Image Understanding

see the image and describe its contents ourselves:

1. By surface areas each with certain individually intrinsic features. For instance, a textured appearance gives rise to a forest area, a steadily luminant portion stands for the surface of a man-made machine part and so on.

2. By transitions due to some systematic local differences in certain luminance features. For instance, the edge of a human face against a rather irregular or noisy background is caused by transitions from regular luminances to irregular ones.

3. By contextual (in)compatibilities among more certain and less certain objects which are close to each other in the spatial domain. For example, an almost rectangularly shaped object on a road will most probably be a car in an aerial image.

4. By known constitutions of expected entities in terms of some more primitive entities. For instance, a personal car consists of a chassis and four wheels in a particular way.

As human beings, we naturally employ terms like edge, texture, (ir)regularity and (in)compatibility as a matter of course to explain our own understanding process whenever we have to communicate with others about our own discoveries made in the given image. The underlying basis here is the assumption that others will grasp the same ideas as to what is meant by terms like edge, texture and so on. As a matter of fact, no one would ever think of the precise numerical aspects of these terms, even though they describe certain features solely represented by the numerical values in the given digital image which is being viewed and interpreted. Moreover, when we look at an image of a real-world scene, objects of various complexities are quickly identified and recognized, but very often without any conscious effort on our part. Indeed, in our interaction with the surrounding physical world many complex visual tasks are accomplished without apparent burden. In other words, we see things from a given image but we are generally not aware of how we have done it.

On the other hand, in the process of image understanding by computer the starting point is always a digital image (or a set of images). The computer can not easily manipulate with our familiar primitives like edges or regions unless sufficient numerical descriptions for these primitives are together supplied to the computer. The gap between numerical and symbolic descriptions is one of the major difficulties in image understand-
ing by computer. In specific applications, a tremendous amount of effort has been put in to bridge this gap through, for instance, applying goal-oriented heuristic or even ad-hoc methods. Many successful examples may be found in the literature (see, for instance, [Nie85, Tur88, Tur85]). However, the situation is also disappointing: a system which was designed to solve a particular problem often can not easily be adapted or generalized to another application, even when both applications are very similar. Even worse, a slightly different problem will very often require a new system design again accompanied by a tedious process of trial-and-error.

The difficulty of image understanding by computer is not only contributed to by vague terms whose characteristics are extremely difficult to describe in a numerical way, it is also considerably contributed to by the fact that the underlying reasoning mechanism of human perception is hard to trace. Very long ago, before the problem of image understanding by computer was even touched on by the research community engaged in image processing and pattern recognition, researchers in various other disciplines were already engaged in the process of understanding and disclosing the mystery of human visual perception. Many assumptions were made and various partial results have been achieved. For example, the notion of context was established and its crucial importance to human perception is now generally recognized. Also, the hierarchical nature of human perception is acknowledged, though views on its dynamic behaviour are still divided and a subject of controversy among researchers even those of a common discipline. Some people favour the view that the perception process follows a data-driven mechanism from simple entities to complex ones. Others insist on the importance of some expected high-level entities influencing the consideration of certain low-level entities. For example, supposing that only square or circular disks may occur in a particular given image, two clearly opposing views will emerge as to the understanding process involving such an image. One view is to insist that the detection of either a square or a circular boundary induces the awareness of the existence of a square or a circular disk respectively. The other view will stress that the expectation of square and circular disks triggers the detection of square and circular boundaries. Nevertheless, by neglecting the cognitive aspect neither side will deny the crucial role played by the context. Another closely related issue is how human perceptual knowledge is represented, in particular, the knowledge which is responsible for the interactions between the various stages of a perception process. While the use of hierarchies, semantic nets, graphs and production rules have been proposed to tackle the tough prob-
1.2 *Past Achievements*

Acknowledging the difficulties of the image-understanding problem, engineering scientists will, however, never distance themselves from dealing with it gradually in an engineering way. Indeed, a tremendous amount of work in the field of image understanding has been done during the past two decades and has been accompanied by a rapid increase in image understanding techniques for various, though specific, applications (see, for instance, [Mat89,Kar89,Ini84,Nev82a,Ger87,Kit84,Bha87,Pau88,Wha82]). Many successful results have been achieved in practice, and many new ideas and design strategies have been proposed which are presumably leading towards the ultimate goal of designing a general-purpose image-understanding system. As we pointed out in the previous section, the problem of image understanding is complex and comprehensive and involves many disciplines of quite distinct interests; it is therefore not surprising at all that many researchers have worked for years just to deal with a particular subproblem varying from edge detection, low-level segmentation, feature selection, search and matching techniques up to designing a heavily constrained special-purpose understanding system. It may even be observed that a significant number of researchers spend much of their effort just on establishing some support systems needed for the research and not actually on the research itself. Neglecting such effort is basically ignoring one particular complexity dimension in the difficulty with an image-understanding system. It is, however, not our purpose here to
review all existing achievements individually as the review will then not only disperse the main ideas behind such a mass development but also induce an undesired and perhaps even misleading feeling that the problem of image understanding by computer is practically only solvable through specific restrictions or that the total problem can be solved by solving some individual subproblems in separate ways (for a broad coverage we refer to [Han78, Dev87]).

In 1977, a workshop on computer-vision systems was held at the University of Massachusetts during which all participants had given their own views on a broad range of questions concerning various aspects of image understanding by computer. The proceedings edited later by Hanson and Riseman (see [Han78]) were considered to give a comprehensive review of the state-of-the-art in the field of image understanding. Many basic ideas which appeared there are still adopted and used in their original forms by researchers of today. At that time it was generally agreed that human perceptual knowledge is, in principle, described by symbolic means on the one side, while the generally huge quantity of data by a raw input image array is certainly not suitable for a direct symbolic description. It was suggested that a division of an understanding system into roughly two main levels would help the coupling between the human knowledge and the direct numerical information presented in the raw image(s). Figure 1.2 symbolizes such a two-level staged system, where \( I \) represents

![Diagram](image)

**Figure 1.2: A depiction of the two-level staged system.**

the symbolic descriptions of the output from the low-level stage which in turn serve as input primitives for the high-level stage. A paradigm governing such a framework is that the low-level stage performs as both a data reducer and a symbolic-primitive generator while the high-level stage introduces the human (usually domain-dependent) knowledge into the processing through symbolic descriptions and symbolic processing at various
1.2 Past Achievements

sublevels. In the various proposed systems, \( I \) usually corresponds to primitive edges and/or regions, either separately or combined, with some spatial relationships among these primitives. It was indeed a practical approach such as what has always been done by engineering scientists to decompose a complex problem into a set of smaller and quite often simpler subproblems.

Along the general line of the two-level system, Hanson and Riseman proposed a system for understanding natural scenes (see [Han78a] and [Han78b]). In their view, the low-level process (or the segmentation process) should, in many instances, be effective without recourse to semantics, i.e., it should be domain independent. Therefore, they designed their low-level process without any explicit use of domain-dependent knowledge of the particular scene. In the context of the scheme in Figure 1.2, one of the main advantages claimed was the completeness of the description of the original image contents by the output of the low-level stage, i.e., the layered directed RSV-graph (a multi-layered graph with its nodes corresponding to Regions, edge-Segments and edge-Vertices). They expected that this description would be sufficient for the high-level process for a variety of applications, with only some specific but non-structural readjustments, like the use of a different gradient operator dependent on the application. Without even considering the possibility of some erroneous (from the viewpoint of the high-level stage) primitives produced by the low-level process, we may expect that such an overflow of low-level output (refer to the claimed completeness above) does not form a manageable input for the high-level stage. In designing their high-level processing part (i.e., the interpretation stage), Hanson and Riseman integrate various kinds of knowledge-representation techniques (such as hierarchies and semantic nets etc.) to describe the scene model. They then employ a simple hypothesis-based control strategy. Indeed, at that time it was unique to have such a general combination of low-level and high-level processes. However, as the authors themselves acknowledged: 1) the need for constructing an internal model describing the major semantic elements in the scene as well as their spatial relationships and 2) the dependence of the system’s power and effectiveness upon the quality of the residing knowledge sources, it is at least questionable whether it is possible to describe the scene properly by only using those symbolic representations. Fischler (see [Fis78]) argued the need for some
form of isomorphic\textsuperscript{1} knowledge representation. In reinforcing this argument, Fischler asked "can you verbally describe your face well enough to a stranger so that he could recognize you if he should ever accidentally and unexpectedly encounter you in person?" It is not our intention here to dig into this matter, our purpose being to point out that the framework as proposed by Hanson and Riseman in [Han78a] and [Han78b] may not even be on the track of a future general-purpose image-understanding system. In particular, it is interesting to note that the authors themselves mention the need for feedback of the information from semantic hypotheses formed at the high-level processing stage back down to the low-level stage, which was originally completely free of semantics! However, they failed to give any indication of how this should be implemented nor did they analyze the suitability of their design for such a feedback loop. Consequently, if there are some local insufficiencies in generating the RSV-graph due to the lack of high-level knowledge, it is doubtful whether it is possible in their design to correct the consequent high-level errors on a dynamic basis, if this can only be accomplished through the guidance of high-level knowledge.

Up to now, quite some approaches have been published towards understanding/interpreting/recognizing various contents in an image (for references we refer to the Bibliography). Without explicitly characterizing each of these approaches, we observe that nearly all obey (a part of) the conceptual architecture as depicted in Figure 1.3, where all levels are, essentially speaking, mutually independent. Although many techniques have successfully been developed using this framework for various specific applications, the framework as such has two inevitable drawbacks. Firstly, when some local insufficiency is resulted at a low level (due to the lack of some necessary higher-level knowledge) which in turn causes an error at a higher-level processing stage, then either we have to swallow some final errors or we have to start all over again without any comfortable assurance of success. Secondly, if we should wish to make the system ready to counter various kinds of unexpected irregularities in the input image, then the system complexity would expand explosively in practice. It would be ideal if we could make our system powerful and yet flexible without a high trade-off in complexity.

In recent years, the rise of rule-based production systems and the development of both various reasoning mechanisms and various advanced

\textsuperscript{1}Isomorphic representations can be interpreted as those which are not directly based on a symbolic or parametrical mechanism. Examples are, for instance, the usual template representations.
1.2 Past Achievements

data-representation techniques (refer to [Dav77,Nie81,Bar81,Coh85]) have caused researchers in the field of image understanding to become greatly attracted to using such methodologies to tackle their own problems. The research on image understanding by computer has hereby experienced a tremendous impact. It is as if a door has opened towards a new era for the entire enterprise. Already, some successful applications have been reported.

Figure 1.3: The framework of traditional image-interpretation systems.

Though most of the published approaches to (partial) image understanding can be characterized by (part of) the scheme in Figure 1.3, there are only very few which deal with the total image-understanding problem for a particular application. Indeed, most of the approaches are restricted to a specific part of the problem such as low-level image segmentation, object-boundary detection or high-level classification. Methodologically, we can roughly group them into the following three categories:
a) *Structural approaches using sophisticated data-representation techniques and/or detailed object models.*

There are a tremendous number of approaches in this category. The main feature of these approaches is that the application problems being undertaken are often partial or quite limited so that sufficient prior knowledge about the image contents is available. Examples here are, for instance, the well-known Split-and-Merge algorithm by Horowitz and Pavlidis (see [Hor76]), dynamic-programming-based object-boundary detection by Gerbrands, Backer and van der Hoeven (see [Ger85]), model-based object-recognition technique by Nevatia and Price (see [Nev82]) and many others (see, for instance, [Vli89,Sed85,Oht85]). The Split-and-Merge algorithm in [Hor76] is based on the assumption that a good uniformity criterium is available and any sufficiently uniform portion in the image domain is considered to be an object. Thus, the algorithm will work satisfactorily if we have succeeded in obtaining such a globally valid uniformity criterium (which happens to be difficult in many practical cases). The dynamic-programming-based object-boundary-detection method in [Ger85] relies heavily on the prior knowledge of the image contents, i.e., an approximate *region-of-interest* within which the desired object boundary is expected. The main advantage of this approach is that it will always find the best possible boundary for the object even under very low signal-to-noise ratio conditions (see [Ger88a]). Thus, the method can be powerful in cases where the knowledge of the *region-of-interest* is indeed available. The model-based object-recognition approach in [Nev82] is similarly heavily based on a selective scene model of the possibly existing objects in the image. This is basically a rather limiting requirement. However, if we can succeed in constructing such models, then the corresponding objects may effectively be found even under distorting surroundings. Such a kind of selectivity is of course quite appreciated if, for instance, we are only interested in certain 'hot-spots' within the image domain. To conclude, the methods in this category can be powerful, each for some particular application areas where sufficient prior knowledge is available.

b) *Production-rule based approaches without dynamic interactions among different levels of processing.*

Here, the production rules are basically used to construct the processing bodies within individual system levels, while the overall system processing is strictly serial from lower to higher levels, in the sense that no backtracking to preceding levels (feedback) is allowed. An advan-
tage of such approach is that the single levels can be constructed more easily and is possibly more effective because the desired function for a single processing level is relatively easily defined. However, certain questions remain as whether we can correctly define the functioning of all separate processing levels and, if so, whether or not the actual processing of the concatenation will yield the required results. Examples here are, for instance, the system ANGY by Stansfield (see [Sta86]) and the low-level image-segmentation system by Nazif and Levine (see [Naz84]).

c) Production-rule based approaches with limited interactions between different levels of processing.
Here, each processing level may contain several basically independent processing units. Some production rules are used to initiate each processing unit and to evaluate the result from each of the units in an attempt to achieve more responsible reprocessing by some lower-level processing units if desired. Examples are, for instance, the systems presented by Nagao and Matsuyama (see [Nag80]) and by McKeown, Harvey and McDermott (see [McK85]).

In the following, we discuss some approaches among the last two categories in somewhat more detail because of their potential contribution to the development of an overall image-understanding system.

In 1980, Nagao and Matsuyama (see [Nag80]) described a unique and successful system for understanding some complex aerial photographs. Apart from its contributions to basic image-processing techniques and the numerical-calculation schemes for various primitive symbolic descriptions, its uniqueness is shown in its overall system design by:

1. A complete blackboard-structure system architecture based on production rules;
2. A focussing mechanism through the characteristic regions, which significantly increases the efficiency of the overall system;
3. The distribution of the available specific knowledge over the different subsystems. In this manner, the overall system performance can be improved to a very high degree thanks to the individual sophistication of the subsystems. In addition, the maintenance of the system becomes much easier, as updating a subsystem will basically not cause any substantial change in the overall system.
A remarkable property of this system is that the global control is kept fairly simple and natural. The system also appears easily extendable to other application areas due to the distribution of the specific knowledge over the different subsystems. Nevertheless, such a subsystem can not always be made immune to the operational insufficiency simply because of the lack of the high-level knowledge. A typical example is the low-level segmentation subsystem. In dealing with this problem, the authors made an attempt to influence the low-level processing through the knowledge of the partial high-level interpretation results already at hand. That attempt was, however, primitive and of limited success. Moreover, as there is no hierarchy in the design of the high-level system, it is doubtful whether it may be extended to applications where some objects are typically described hierarchically with symbolic representations.

In 1984, Nazif and Levine (see [Naz84]) published their pioneering work on an approach to low-level image segmentation. They succeeded in encoding some declarative knowledge in terms of condition-action rules into a rule-based production system. Their main achievements are the construction of the so-called strategy rules (or meta-rules) and the adjustment of a traditional rule-based system towards a new environment, i.e., the image domain, which can be considered as a single entity but more precisely should be seen as a dynamic population of yet undefined entities. Notice that a traditional rule-based production system like the one introduced when building MYCIN (see [Sho75]) is specifically suitable for a static working environment. A consequent advantage of the system by Nazif and Levine is obviously the flexibility, which allows adding to, or adjusting the encoded knowledge by manipulating the rule base. However, the suitability of this system for future development towards an overall understanding system depends essentially on whether or not our overall knowledge is suitable to be encoded in a rule-based system, and if so then whether or not our reasoning strategy can methodologically be simulated by such a reasoning mechanism.

Other more application-oriented approaches have frequently been reported, where attempts are made to describe declarative knowledge by some explicit symbolic representations such as graphs, frames or semantic networks. One of the interesting cases is the system called ANGY by Stansfield (see [Sta86]). It is a 3-stage rule-based system for recognizing coronary vessels. The first stage is just a conventional set of image-processing subroutines providing an initial segmentation in terms of edges and regions. The next two stages are basically pure rule-based production sys-
tems containing respectively the domain-independent knowledge (for the segmentation refinement) and the domain-dependent knowledge (for the final recognition). In the experiments with ANGY, it was discovered that the system performed equally well even without the second stage. Clearly, the separate use of our knowledge is apt to ignore or disturb inter-process co-operations. Moreover, the author also points out that even in this relatively simple application the system can not be efficiently designed and implemented in a purely production-rule based approach if some further improvement is desired. In our view, it is likely that a well-coordinated usage of various knowledge-representation techniques and a proper integration of various control strategies may put us on the right track towards building a future general-purpose image-understanding system. However, few practical achievements have yet been made in this direction.

1.3 The Scope of PART I

From the foregoing discussions, we have a feeling that many among the current achievements in image-understanding systems have some common deficiencies. Firstly, it is the limited capability of various knowledge-representation techniques and the difficulty to combine them properly. Secondly, there is no truly effective mechanism yet to provide a local readjustment on a lower-level processing stage when corresponding lower-level insufficiencies are discovered by some higher-level processes and/or when some indicative information at a higher-level stage is already available. The main focus of the first part of the thesis is on designing a mechanism which will allow proper mutual interactions among the various levels of processing.

In view of the above, we present our Distributed and Anomaly-Driven System architecture (DADS), which is based on two primary notions to be introduced in the sequel, i.e., the notions of context hierarchy and of anomaly. DADS provides an environment in which one considers a particular application problem in terms of the data classes and their mutual relationships (i.e., the associative and productive relationships to be made clear later) and properly distributes specific knowledge over the system in terms of the processing clusters. Moreover, it provides us with the possibility to define the interactions among the various anomaly-driven processing clusters in an efficient way. This framework will result in an application system which is highly sophisticated (i.e., with fewer errors in the final
output) and still efficient (i.e., the software development and the actual processing time will not increase correspondingly). In addition, we are also going to consider some aspects of human visual perception and its relationship with the DADS architecture.
Chapter 2

Image-Understanding Modelling

2.1 Preliminary Remarks on Certain Aspects of Visual Perception

Human visual perception is, generally speaking, a process in which a human being understands a scene, which is presented to him either directly by the circumstantial surrounding or indirectly by a pictorial representation (e.g., a photograph or a display screen). Disregarding certain specific aspects, image understanding by computer can be seen as a process similar to a human perception process, especially when only the goals of the processes are to be considered. In order to make some attempts towards building a general-purpose image-understanding system, it is not only natural but also essential to consider some established achievements in the research on human perception even though some of those achievements may still be rather controversial and subject to criticism. However, we should acknowledge the fact that still very little has been revealed of the very nature of the human perception process, and profound theories or descriptions are simply not available yet. The discussion to be conducted in this section will be brief and practical. It is not the intention of this thesis nor its research scope to consider deeply human perception.

Towards a question such as ‘What is the proper nature of human perception?’ or a challenge such as ‘Give a well-founded description of the human perception process’, there will be either no answers at all for the time being, or a variety of incomplete but mutually competing and con-
tradicory answers. Each would be based on some kinds of experimental evidence and therefore all seem to be reasonably correct. Given the basically logical manipulation capability and the preciseness of a digital computer, engineering scientists would prefer to have a perception process described along a formal deductive and/or inductive reasoning mechanism and have the perceptual knowledge represented with some non-isomorphic tools. However, psychological study tells us that both common-sense reasoning and iconic representation schemes play a far from negligible role in the human perceptual process. As an example of common-sense reasoning we may have *If a block is hanging over too far, it will topple* (see [Fis87], P.286). Examples of iconic representation schemes are, for instance, templates or even a usual digital image itself. The exploration of common-sense reasoning based and/or iconically represented perceptual knowledge about a particular application is often very hard, and so is the subsequent manipulation by a digital computer. As far as the methodology of the human perception process is concerned, there are two quite different points of view, namely the sequential (or logical) paradigm versus the parallel (or Gestalt) paradigm (see [Fis87], for instance). The former stresses an approach in which only a small portion of the available data is considered at any time, while the latter considers the data on a global basis, namely all at once. Researchers from various backgrounds settle for the assumption of the existence of both phenomena, although again there is no consensus as to the mechanism which combines the two in such a successful and yet magical process of human perception.

In the course of the human perception process, it is conceivable that the prior awareness or knowledge of the global context of the expected scene has a significant influence. Bugelski and Alampay (see [Bug61]) showed through their experiments that if a subject (i.e., a human being) is conditioned to expect a given category (or generalization) of stimulus, then the identification time is significantly reduced when the stimulus is actually presented. Interestingly enough, there are circumstances in which the prior awareness of a wrongly assumed global context plays a negatively influencing role. It was observed by Palmer (see [Pal75]) that a misleading context specification clearly leads to a slowdown of the identification process. In his experiments, the experimentee was told he was to be presented with some stimuli all belonging to the same class of concepts (i.e., animal). When some non-animal stimuli were presented to the experimentee, the response time was much longer than if animal stimuli had actually been presented. Although among people of various professions
who are investigating the subject there is again no unified view on measuring the importance of context, they all agree that the notion of context plays an inevitable and important part. However, little is known about the detailed mechanism governing such context-sensitive behaviour in human perception or about the extent of the role played by the prior contextual knowledge.

To explore how human beings extract entities with common sense from a perceived scene and facilitate a workable structure for practical purposes, Uhr (see [Uhr80]) proposed a parallel-serial layered structure of converging variable resolution as shown in Figure 2.1, where \( \{T_i : i = 1, 2, \ldots, n\} \) are sets of transforms and \( \{I_i : i = 1, 2, \ldots, n\} \) are input/output memories for these sets of transforms. The transform layers are sandwiched between memory layers with the output of a lower transform layer serving as the source of input for the next-higher transform layer. The entire process appears as a serial queue of as many subprocesses as there are layers of transforms.

![Figure 2.1: Uhr's overall structure of a parallel-serial pyramid or cone (taken from [Uhr80]).](image)

In his modelling Uhr, in fact, stresses the following three primary aspects:

1. Intra-layer parallelism.
   Within each transform layer, a set of transforms spread about throughout its input domain. Most of these transforms are individually focussed on a relatively local set of input information (like synapses usually receive inputs from relatively nearby neurons) and all of them act at the same time, i.e., in parallel. Moreover, different kinds of transforms may co-configure in such a single transform layer and may even concurrently
operate on the same local input information. For instance, the transforms for different grey-value edges and textured edges may operate simultaneously in the same local neighbourhood.

2. Inter-layer's serial nature.

Contrary to the concurrent processing nature within each transform layer, the overall layered structure globally exhibits a serial character, i.e., each transform layer must wait for and look at the output of its preceding transform layer.


The cardinality of the output information from a particular transform layer will usually be smaller than that of its input. This in turn effectively reflects the degree of information abstraction or reduction by the transform layer and gives a converging pyramid/cone shape which results in an overall structure of hierarchically organized layers of transforms and memories.

In order to give enough convincing weight to his modelling by the parallel-serial structure, Uhr gave a particular description of the living perceptual system based on some neuro-biological and psychological material. In particular he reported two experiments, the result of one of the experiments contradicted the illusion that the brain is entirely serial in processing while that of the other experiment showed that the brain can not be entirely parallel in processing either (see [Uhr80], P.19).

Of course, it would be difficult even for a neuroscientist to judge Uhr's description of the visual system as especially distorted. Nevertheless, it should be noticed that many systems had already been presented before Uhr proposed this formal structure (see for instance, [Baj78] or [Dav78]), all of them were basically taking a parallel-serial layered approach while usually not being thought of as having this structure. Methodologically speaking, almost similar or even more developed structures are used in other areas. The well-known Hearsay-II model (see [Erm80]) in the area of speech recognition is, for instance, a very similar but more developed approach than Uhr's parallel-serial structure and yet it was introduced much earlier. To compare the Hearsay-II model with Uhr's structure, we can consider each level of hypotheses in the Hearsay-II model as a memory layer (e.g., segments, syllables, words and so on) and the corresponding knowledge sources as the transform layers. By disregarding the processing activities associated with VERIFY and RPOL in Hearsay-II (for details we refer to [Erm80]), we conclude that Uhr's parallel-serial structure is
2.2 DADS Modelling

almost identical to the blackboard-based system hierarchy in the Hearsay-II model. With backtracking facilitated by the processing activities of VERIFY and RPOL, however, the Hearsay-II model obviously has some extra processing capability. Nevertheless, it would be unfair strictly to compare the merits and demerits of these two structures while neglecting their individual underlying backgrounds and interests.

Without being involved in an apparently endless argument on the physiological and psychological basis of Uhr’s model, we assume that the most important messages are the following:

a) The human perception process can be globally characterized by a data-flow scheme.
b) This global data-flow scheme shows a tendency towards becoming more compact as the data flow upwards to more abstract levels.

However, careful reading of [Uhr80] gave us an almost conclusive feeling about Uhr’s very motivation in his proposal. His proposal was, in fact, the outcome achieved by summarizing many of the various systems known at that time. Therefore, his consideration on human perception was purpose-directed and partial. This inevitably leads to many shortcomings in his model.

2.2 DADS Modelling

In digital-image-understanding research we cannot afford to wait for a definite answer to the question posed as to the very nature of the human perception process. Past experience has taught us that such an answer may simply not exist, and even in the future may never be found. Worst of all, it may probably be meaningless in practice to search for such a precise answer. Marr argued (see also [Mar82], P.27): trying to understand perception by studying only neurons is like trying to understand bird flying by studying only feathers; it just cannot be done. In our opinion, a general-purpose image-understanding system is desired not for the purpose of simulating the process of human perception but rather for the purpose of emulating the performance behaviour of the human perception process in a particularly restricted manner. From that point of view, we outline the following three general guidelines for our considerations:
a) The process is restricted to be purely cognitive.
   The need for this limitation is very clear. The cognitive part of human
   perception is mostly unconscious whereas a computer is always built
   under our conscious mind (or logical mind) as an artificial manipulator.
   Therefore, a computer can in principle only handle notions predefined
   by us and proceed in a way which has been prespecified.

b) The process should conduct its main processing purposefully and in an
   efficient manner.

c) The process is expected to handle circumstantial irregularities within
   the actual data in both an effective and an efficient way.

The recognition restriction in a) basically implies the availability of
sufficient prior knowledge of the global context within which our under-
standing machine will be assumed to work. This is certainly the case in
practice, though both the extent and depth in terms of describability of
such knowledge may vary from case to case.

The points in b) and c) are actually our criteria for matching the
performance behaviour of human perception in similar purely cognitive
situations. In other words, these points require an emulation of the human
perceptual ability both to be aware of the underlying cognitive purpose
and to adapt the real processing to match the actual circumstances.

In our modelling towards DADS, we start with the parallel-serial
structure as presented by Uhr. The reason for this choice is straightfor-
ward and practical. Despite the fact that unconsciousness and parallelism
play an important role in human perception, we always use our conscious
mind to think serially when describing our perception process whenever
we are asked to do so. Unfortunately, this describing is always what we
first have to do if we want our understanding system to do the same. For
example, there are a triangle and a circle in both pictures in Figure 2.2
and the appearances of both a triangle and a circle are assumed to be
known to the potential perceiver. If one is asked how one did it after
having recognized the triangle and the circle in both pictures, one will in
most cases answer 'I saw two dark objects, one has a triangular shape and
the other has a circular shape.' When one is further asked why it were
the dark objects in particular which were discovered, one might possibly
reply that it was due to seeing transitions between ('averaged') luminances
while not being aware that the transitions were discovered either serially
or in parallel. However, if one is asked how these luminance transitions
are detected in terms of operations, then the answers will generally become
vague. Our conclusion is that when forced to describe a perception process, a human being is usually keen on employing intermediate meta-objects (or meta-descriptions) to sketch globally the processing route, even though we generally feel it much more difficult to explain, or are even unable to explain, how those (meta-)objects are detected individually. In the previous example, our meta-object based description would then probably take the form shown in Figure 2.3, while it is not clear how those intermediate

meta-objects (e.g., luminance transitions) are detected. On the one hand, we believe that the failure in explaining the how here is because this how largely involves our unconscious (and parallel) perceptual activity and on the other hand that a description by meta-objects is in effect a disclosure of (or an attempt to disclose) our conscious perceptual activity. Again, our
general guidelines indicate that our conscious thinking in perception should be exploited as much as possible, while leaving our success in emulating heuristics and the unconscious thinking to our experience already gained. It is therefore natural that a scheme such as in Figure 2.3 is adopted and constructed to the deepest and broadest possible extent. Such a scheme is, in fact, similar to the structure proposed by Uhr (see [Uhr80]).

If we would succeed in emulating our unconscious thinking in detecting luminance transitions shown in Figure 2.2a (or Figure 2.2b) by using a suitably chosen edge detector, our task would be completed. Uhr's structure seems, therefore, to suffice for that purpose. However, real-world problems are never that simple. Not only will it be hard to find successful numerical emulators for particular unconscious parts of our thinking, but also it will be difficult to construct corresponding data-flow schemes. Further to explain the latter, we can look through the famous Bongard Problems (also abbreviated as BP-problems) described in [Bon69]. In a BP-problem, we do not have to consider some tough parts in our unconscious thinking to do with perception, such as how a line segment or a point is detected. Solving a BP-problem only means giving a sufficiently characteristic description of the given pictorial contents in terms of some logical relationships (such as hierarchies and spatial relationships) and the given primitive entities (such as points, line segments and so on). To those who are engaged in the field of digital image interpretation, it is truly a great relief to take simply these primitive entities for granted. Surprisingly, even in this relatively limited and well-definable situation, it turns out to be extremely difficult and tricky to obtain a sufficiently characteristic description to a BP-problem (for more details we refer to [Bon69]).

Continuing our discussion, suppose now that the recognition of the picture contents in Figure 2.4 is desired. If we ask how the noisy triangle in this picture was successfully recognized, an answer such as 'I have discovered transitions there between 'averaged' luminances instead of that between pure luminances and they happen to form a triangle as well' may probably be given. Well, as a means to emulate such behaviour, the previous approach is too limited. In principle, we can employ two edge detectors in parallel for this situation, one for clear edges and the other for noisy ones. However, in more realistic situations, such a stubborn extension is likely to lead to some infeasibly complex mass parallelism (in terms of computing time or system complexity) and/or unresolvable synchronization issues among simultaneous but related subprocesses. Bearing these considerations in mind, we suggest modelling such process in the following
2.2 DADS Modelling

Figure 2.4: A picture which contains triangles and circles appearing in different ways.

The perceiver first used the previously successful edge detector for clear edges (as the prior context suggested). When having discovered that a portion of the picture domain (occupied by the noisy triangle) could not be assigned to any of the expected object categories (i.e., triangle, circle and background), the perceiver then tried to search for transitions between averaged luminances (a different edge detector) and that happened to yield transitions together forming a triangle. Thus, the originally noisy portion became resolved and understood.

Thus, the perceiver will apply originally unplanned means if meanwhile something unexpected emerges. In the above example, such unexpected entities are a collection of unexplainable small/irregular objects around the noisy triangle. We call such an unexpected entity (or a description of such unexpected entities) an anomaly. In general, an anomaly can also be caused by circumstantial errors in the supplied data. If we use Uhr's structure, the handling of such an anomaly will mean the additions and/or replacements of some transforms within certain transform layer(s), all applied only to the locations where the anomalies emerge. Clearly, the original Uhr's structure does not facilitate such an on-line modifying mechanism. Significant extensions to that structure are therefore unavoidable.

On the basis of the above argument, we propose a Distributed and Anomaly-Driven System (DADS) as globally sketched in Figure 2.5. Bearing in mind that our modelling must be kept parallel with our three general guidelines mentioned earlier in this chapter, we briefly outline the functional characteristics of each block in Figure 2.5 as follows:
1. Processing Hierarchy.
This is basically a combination of the context hierarchy and the corresponding processing blackboxes. The notions of both context hierarchy and processing blackbox will be formally introduced and discussed in detail later. For the present, it is sufficient to realize that the processing hierarchy is in many senses similar to Uhr's parallel-serial structure but with data classes and processing blackboxes instead of memory layers and transform layers, respectively. However, it is much more enriched for reflecting, and much less restrictive towards, the modelling of our conscious thinking during perception. For example, the globally serial nature of Uhr's structure, which may be quite restricting for many practical situations, is not present in the processing hierarchy, and is replaced by a natural and flexible hierarchical structure. Another important difference worthy of note is that the processing hierarchy itself is no longer a static scheme like Uhr's structure. On the contrary, it should be viewed as a dynamic scheme depending on the actual processing.

2. Anomaly Handlers.
This is the collection of all available anomaly handlers. While the term anomaly handler will be introduced and fully covered in detail in the subsequent sections, we here state that the main task of an anomaly handler is to solve detected anomalies of the corresponding kind. The action of such an anomaly handler may mean an explicit updating of some processing blackboxes in the processing hierarchy (e.g., new values for some parameters or even a new operator like a different edge detector) or an implicit updating by taking over part of the functioning
from some processing blackboxes (e.g., the anomaly handler solves the supplied anomalies itself and yields output data which are compatible with some data class(es) in the processing hierarchy).

3. Supervisor.

This is in many senses the system's overall controller in charge of the overall coordination. However, it will not be much involved in the actual problem-specific processing matters. In particular, it contains limited knowledge of the specific application problem being undertaken. Its main task is to activate a processing blackbox or an anomaly handler if some appropriate input for the processing blackbox or anomaly handler actually occurs. In this way, the construction of the supervisor is much immune to a change in the application.

In the following, we discuss another example further to explain our proposal of the anomaly-driven mechanism, with which we intend to emulate the required flexibility and efficiency in handling circumstantial errors or unexpected irregularities among the actual data.

Figure 2.6: A picture containing an English word *feet* which is printed in a rather unusual way (taken from [Fis87]).

Suppose that a picture containing an English word *feet* as shown in Figure 2.6 is to be understood by a human perceiver. This will be trivial when all portions of the picture are presented at once. However, one is unlikely to understand the picture if it is presented by successively showing only a single tiny portion of the picture even in an exhaustive way. Such a phenomenon is used in the argument between the sequential (or logical) paradigm and the parallel (or Gestalt) paradigm with respect to human perception (see also [Fis87], P.13-15). Let us assume in the former case (i.e., when the entire picture was shown at once) that the perceiver also
started looking at individual local portions of the picture and by incorporating the knowledge about the word feet, a global consideration, which led to the final success in understanding the picture, was activated only by the subsequent understanding failure (for the piece-wise bar segments, which constitute an anomaly in our terminology, were not supposed to be the objects for recognition). We then have a feeling that the perceiver is following the anomalies on the one hand and is limited by his capability on the other hand (for if the perceiver had no knowledge of the alphabet at all, success would never have been possible even though the entire picture was presented at once). Motivated by this consideration, we feel strongly that the controlling mechanism which is responsible for the appropriately dynamic behaviour in human perception, may be modelled as an anomaly-driven mechanism. Furthermore, the notion of anomaly can be instantaneous or ultimate. When an anomaly is instantaneous a special (exceptional) part of the background knowledge will be triggered to tackle the anomaly, while an understanding failure will result if an anomaly turns out to be ultimate (a case in which the perceiver is at his wits' end).

The present brief introduction to the DADS modelling is closely guided by the objectives stated in the initial general guidelines. While believing that our system design can provide a number of means to fulfill our wishes in regard to these final aims, we do acknowledge that it may not be easy to realize such a system in practice. For example, within a particular application, it may not be directly clear whether or not it is possible to describe symbolically (with or without thinkable templates) all (meta)-objects with respect to a sufficiently complete processing-hierarchy (cf. how difficult it was even to solve a BP-problem through some meta-descriptors). Also, it will not be easy to define and represent all possible anomalies in advance and develop corresponding anomaly handlers. However, we do not see how otherwise to avoid these difficulties if we want a machine to be flexible and able to detect and deal with possible anomalies. In the following, details of various aspects of the DADS modelling will be discussed.

2.3 Context Hierarchy and Processing Blackboxes

Image understanding by computer can basically be seen as perception by computer, where the actual scene is replaced by a scene ‘representation’,
i.e., a digital image (or a set of related digital images). The dominant paradigm governing such a practical engagement is the \textit{signals-to-symbols} paradigm, in which the raw sensed data in numerical representations are transformed into a meaningful and explicit description of the corresponding scene by a set of inductive steps employing progressively more abstract representations. Fischler and Firschein use a sketch shown in Figure 2.7 to illustrate this paradigm (see also [Fis87]). Like the observed appropri-

![Figure 2.7: The signals-to-symbols paradigm for computational vision (taken from [Fis87]).](image)

ateness of the saying (in the area of Artificial Intelligence) ‘If you are a hammer, everything looks like a nail’, for image understanding by computer it can be similarly felt appropriate to say ‘If you are a digital computer, then everything looks like a number or a defined symbol’. Philosophically, we may question the overall appropriateness of the signals-to-symbols paradigm (refer also to the discussion in [Fis87], on P.287–289). Nevertheless, as far as practical applications are concerned, this assumption is not only essential but also quite inevitable. A machine is an artificial manipulator made by man, and it can therefore in principle only handle notions predefined by man and proceed in a way prespecified as well. To a machine, a digital image is nothing but a conventional form consisting of signals with most primitive meanings. And yet, a set of recognized objects in terms of some defined symbols are generally desired from an image-understanding system. Therefore, we will not be sceptical about the signals-to-symbols paradigm and describe an image-understanding process
as a data-transformation process from the input space $X$ to the output object space $Y$, where $X$ corresponds to the set of all input pixels and $Y$ corresponds to the set of all possible final objects being searched for\textsuperscript{1}.

If at all possible, we would like to encode all of our knowledge into the computer. Unfortunately, this cannot be achieved at present. In practice, we have a particular machine to perform a particular task. In other words, we have to be content with a rather restricted output space $Y$. In particular, $Y$ is to contain an element which will be assigned to a portion of $X$ if the machine fails to understand that portion of $X$, given the built-in knowledge base. By specifying $Y$, we have in fact also established a global context for our understanding system.

The notion of context hierarchy corresponds to a hierarchical graph description, in which the main data-flow chart is represented from $X$ to $Y$ through some appropriate meta-data. We define it in the following way.

**Definition 2.1 Data Class**

A data class is a collection of data elements, which have a common generic property.

**Definition 2.2 Context Hierarchy**

A context hierarchy is a directed graph, where a vertex corresponds to a data class; a branch from vertex $A$ to $B$ indicates that the generation of the data class corresponding to $A$ needs the data class corresponding to $B$. Moreover, it will become a directed tree (thus a flow network) if all bidirectional branches are removed.

Uhr’s original parallel-serial structure (see Figure 2.1) can in fact be converted into such a graph, in which each vertex corresponds to a memory layer while the branches correspond to the transform layers. However, our context hierarchy is allowed to take a much more general and flexible form. For instance, it is no longer globally serial, but a general flow-network-like structure, in which all (input, intermediate and final) data are represented and the possible flow routes for actual data elements starting from the input data classes (i.e., $X$) to the output data classes (i.e., $Y$) are specified as well. The main advantage of this general flow-network-like structure with respect to Uhr’s structure is obvious. For instance, the same low-level data

\textsuperscript{1}Here we do not explicitly state the relationships among the objects since they may be considered as some attributes of the involved objects.
can simultaneously serve as the input for two or more separate higher-level transforms. With such a context hierarchy, we can structure our encodable active knowledge in accordance with the hierarchical relationships among various types of data. In our strict sense, a context hierarchy is a rational description of a particular transformation from \( X \) to \( Y \) in terms of some intermediate data classes and their mutual relationships. The predicate rational here means that we omit all irregularly occurring or unexpected intermediate data classes. More precisely, it is an optimistic data-flow chart from \( X \) to \( Y \), given the machine’s capabilities with respect to the particular application. Figure 2.8 is an example of context hierarchy, where \( Y \) consists of two data classes corresponding respectively to square and circular disks and each vertex represents a certain class of (initial, intermediate or final) data.

![Diagram](image)

Figure 2.8: A context hierarchy for recognizing squares and circles.

Here, we point out that the notion of context hierarchy in our DADS modelling is based on the assumption that an actual image-understanding process can always be explicitly described as a data-transformation process from the input space \( X \) to the output object space \( Y \). However, many practical problems are not initially presented in such a straightforward form. We consider them in two cases.

Firstly, almost all of the image-understanding applications involve the finding of certain meaningful objects which may appear in the input space. Normally, we only have some positive specifications for the characterization of the expected objects and do not explicitly specify the appearances of possibly existing non-object parts in the input space. In general, the objective is considered to be reached if we detect the objects actually present. By introducing a new object class corresponding to the possible non-object
parts in the input space, the *nonsense object* class, we can reformulate such problems as a full transformation problem from $X$ to $Y$ with $Y$ now including the class of nonsense objects. The only consequence here is that we may also be actively involved in locating the nonsense objects instead of solely finding the meaningful objects. This reformulation may require some extra effort for some explicit characterizations of the possible nonsense objects. However, it will not necessarily lead to an extra processing burden as an obviously necessary consequence.

In other cases, problems are aimed at the full input space, i.e., each portion of the input space is desired to be understood with prescribed meanings. Examples here are, for instance, to be found in the field of remote sensing for land-use classification, harvest estimation and so on. Again, we introduce a special object class tagged as *nonsense* as there is no practical guarantee that all portions in the input space will turn out to possess our prescribed meanings.

Altogether, if we actually want to formulate the problem as a full transformation process and make use of this formulation, we need to introduce a special *nonsense object* class and explicitly specify its characteristics. In practice, we may have some difficulties in specifying such a nonsense object, but we will certainly be well paid in terms of system performance. The system design is based on the assumption that an appropriate specifications on the possible nonsense objects can be defined. Therefore, the objective becomes a true recognition on the full input space, a complete transformation from $X$ to $Y$.

Bearing this in mind, the context hierarchy for the previous problem involving square and circular disks should therefore be modified to take the form as shown in Figure 2.9.

Next, we will set out to introduce a binary attribute to each of the links in the context hierarchy. This attribute can take a value indicating either a productive or an associative relationship. Let us suppose that we have a link from vertex $L$ to vertex $H$ as shown in Figure 2.10 within a context hierarchy. The definitions of the associativeness and the productivity for the link from $L$ to $H$ are given below.

**Definition 2.3  Associativeness**

Vertex $L$ is said to be associative to vertex $H$ if the generation of the data elements in the class corresponding to vertex $H$ requires the consultation of some data elements from the class corresponding to vertex $L$. 
2.3 Context Hierarchy and Processing Blackboxes

Figure 2.9: A context hierarchy for recognizing squares and circles.

**Definition 2.4** Productivity

Vertex \( L \) is said to be productive to vertex \( H \) if the generation of the data elements in the class corresponding to vertex \( H \) involves a direct absorbing mapping from some data elements in the class corresponding to vertex \( L \).

Figure 2.10: Vertex \( L \) is directed to vertex \( H \) in a context hierarchy.

The underlying reason behind introducing the associative and productive attributes is the following:

Under the *signals-to-symbols* paradigm we construct a data-flow network by means of a context hierarchy. However, we notice by experience that there are two basic but different paths for the data flowing up from a lower vertex to a next-higher vertex. When some lower-level data elements have succeeded in flowing upwards to a higher level, they will either be explicitly transformed (or absorbed) into some higher-level data elements and thus essentially do not need to flow upwards along other paths, or still need to flow upwards along other possible paths. In our terminology, these
two possibilities are called productive and associative, respectively. Moreover, we notice that if a data class has a productive link to its immediate higher-level data classes, all data elements from the class will then need to be absorbed into its productively linked higher-level data classes.

To illustrate this idea, we consider again the example involving square and circular disks. Suppose that the final squares, circles and background are described in terms of regions and that the edges are considered only to ensure that they will not intersect a square or a circle. Then, when constructing the context hierarchy, we will generally require that all data elements from the class region are to be transformed (or absorbed) into either one of the classes circle, square or nonsense object (e.g., the background) in an exhaustive manner. At the same time, we will also implicitly allow that a particular data element from the class edge may be used as input (consultative) information for the generation of both a data element in the class circle and a data element in the class square, or not be used at all as input information for generating data elements in the classes circle, square and nonsense object. Corresponding to our definitions, this phenomenon clearly means that the data class edge is associatively linked to the data class square (or circle or nonsense object), while the data class region is productively linked to the data class square (or circle or nonsense object).

![Diagram](image)

Figure 2.11: A context hierarchy for recognizing squares and circles.

The above definitions for both the productivity and the associativeness are based on the assumption that our practical applications will allow the applicability of these definitions, either directly or via a reformulation with affordable effort. For the example of square and circular disks mentioned earlier, we may have the context hierarchy as depicted in Figure 2.11,
where a productive link is represented by a solid line and an associative one is represented by a dotted line.

Evidently, a context hierarchy itself is also a representation of knowledge. However, a different kind of knowledge which we call active knowledge, is represented by various processing modules. It distinguishes itself from the knowledge in the context hierarchy by its functional character. In our case, it is represented by various *processing blackboxes* which are individually attached to each data class in the context hierarchy. A processing blackbox is defined as follows.

**Definition 2.5  Processing Blackbox**

A processing blackbox is a functional processing module which generates the actual data elements in a corresponding data class by means of the existing data elements of those input data classes as indicated by the context hierarchy.

Thus, a processing blackbox represents the knowledge involving the generation of a certain class of data based on its input data from other lower-level data classes as indicated by the context hierarchy. In other words, it operates on a particular kind of input data and produces a particular kind of output data. In the example of Figure 2.11, the processing from the data class *input image* to the data class *region* is, for instance, an example of this kind of encoded active knowledge.

### 2.4 Anomaly and Anomaly Handlers

The notion of anomaly is essential in our system design. It plays a key role in effectively controlling the system performance. Below, we attempt to give our definition for the notion of anomaly in a strict way.

**Definition 2.6  Anomaly**

An anomaly is basically a data element. However, a class of anomalies does not directly correspond to any vertex in the context hierarchy and its actual occurrence is irregular or even unpredictable.

The set of anomalies can be considered as supplementary to the context hierarchy and obtainable through a somewhat *irrational* survey of the
data transformation from $X$ to $Y$ presented by the context hierarchy. The term *irrational* here suggests that, starting from the existing context hierarchy and the corresponding active knowledge being encoded, we only look for the possible incompleteness and inconsistencies. Generally speaking, there is no guarantee that the context hierarchy together with the corresponding active knowledge will be complete as well as consistent. In our opinion, neglecting the concept of anomaly explains why many existing approaches do not function properly or even fail altogether in practice. In the earlier example of square and circular disks, it may happen that an edge is not closed or cuts right through a region. Such cases will cause either inappropriate functioning or a failure. Clearly, the anomaly of an unclosed edge or of a case in which an edge significantly intersects a region is the major cause behind the system malfunctioning. In this respect, an anomaly may look like a particular description of some entities rather than a solid entity. (Note that a solid entity and a description of some solid entities are both viewed as data elements in our consideration.) To some extent, an anomaly class can also correspond to a certain type of circumstantial data errors which may arise. Moreover, we attach a special processing-blackbox called an *anomaly handler* to each class of anomalies and it will only then be triggered if some corresponding anomalies are actually encountered.

Below, we discuss an example which further explains the practical meaning of the notion of anomaly and the principle of our anomaly-driven mechanism.

Let us take the previous example of square and circular disks. Suppose that our prior knowledge indicates that an occluded square or circle is in principle allowed to occur but such an occlusion will not be commonly encountered among the potential input images. Basically, it is possible to employ a context hierarchy as shown in Figure 2.12. However, if we define a class of anomalies containing all cases, in which a region is not sufficiently convex in the way as a square or circle is, and require the processing blackboxes corresponding to the generation of the data classes *complete square* and *complete circle* to detect such an anomaly, then we can use a context hierarchy as depicted in Figure 2.11. Clearly, this context hierarchy will lead to a much more efficient overall processing than the one in Figure 2.12 as the overall system control will not be disrupted by the occasionally occurring occlusions and yet the occlusions can be efficiently resolved if they actually occur.

In practice, we may fail to consider all possible anomalies and thus
fail to include the corresponding anomaly handlers. However, the performance of the system depends significantly upon its ability in countering the possible anomalies.

2.5 Further Discussion

In our system design, due to the flexibility in the processing hierarchy, it is possible to create operations within one particular data level in such a manner that they correspond with a limited spatial extent in the image domain. In other words, essentially different operations can in effect be simultaneously applied to distinct and/or overlapping data. This is an attractive property with respect to modelling a particular application problem along the signals-to-symbols paradigm. In particular, it facilitates the incorporation of both the hierarchical and the competitive nature, which are believed to be characteristic to the human perception process. Because this parallelism involves operations within one or more data levels and because it involves not only the same operations extended in the global spatial domain but also operations of different natures at different spatial locations, we call this hybrid parallelism.

In most of the earlier systems for image understanding, our notion of context hierarchy can, in fact, implicitly be found. However, little explicit attention has been paid to the possible incompleteness of an actual
context hierarchy and, especially, to the consequent necessity of anomaly handling. In some systems, much attention is paid to issues surrounding data representation and rational reasoning. In other systems the global controller/supervisor is supposed explicitly to solve all kinds of possible anomalies. The results are either hardly successful at all, or partially successful for very particular situations.

In DADS, we essentially split our active knowledge into pieces and distribute these over the processing blackboxes and anomaly handlers. Both a processing blackbox and an anomaly handler are known to the outside world (i.e., the supervisor) only through its input/output description, while relations among all processing blackboxes are characterized by the context hierarchy. Because of their resemblance, we have made the following convention:

**Definition 2.7** Processing Cluster

Both a processing blackbox and an anomaly handler are called a processing cluster.

As far as the supervisor is concerned, its task is quite clear. Towards the processing blackboxes it functions as an administrator with the aid of the context hierarchy and the actual data at various existing vertices. In reference to an anomaly handler, the supervisor is the activator in case there are data available which fit the anomaly handler. However, an important issue as to the overall control of the system should be discussed. This concerns two questions: a) when will the entire processing come to an end? b) is the entire processing guaranteed to end? Obviously, this would be of no concern if no anomalies ever actually occur. When an anomaly does occur, our system description up to now has implicitly allowed the possibility that the output from an anomaly handler may again result in an anomaly of a different kind, which in turn may again yield an anomaly of the original type. If no proper actions were taken, there is a good chance that the system enters into a deadlock situation due to such oscillations. To tackle the problems of possibly recurrent anomalies, one might charge the supervisor with the task to detect and resolve undesired anomaly recurrences, through, for instance, arbitration. On the basis of the following argument, however, we do not consider this to be the right 'solution':

a) If the supervisor has limited knowledge about the actual image understanding problem being undertaken (which is presumably the case),
then not only the detection of anomaly recurrences by the supervisor will be difficult but also the subsequent resolving of the recurrences (e.g., through arbitration) will generally be pointless and thus perhaps wrong itself.

b) If the supervisor is supposed to have sufficiently deep and broad knowledge of the actual understanding problem, then the design of the supervisor itself will become awkward and inflexible. Moreover, a new kind of recurrence, the design recurrence, will be apt to occur, i.e., a supervisor within another supervisor. It is in practice hard to resolve such a recurrence.

Bearing these considerations in mind, we propose to resolve the anomaly-recurrency problem through the actually involved anomaly handlers themselves. The general motivation behind this proposal is the following:

a) An anomaly handler which is involved in such a recurrence is most informed about its particular properties.

b) A recurrence of this kind involves processing tracks at unpredictable moments (as far as the overall system processing is concerned) and its specific features can be any among a huge variety for an actual application. Requiring the supervisor to resolve it will cause an intolerable inefficiency (since anomalies are supposed to occur in a rather irregular manner), whereas it can be incorporated easily in the involved anomaly handler itself.

Since the ultimate aim of our system design is headed towards a general-purpose image-understanding system, anomalies can thus significantly vary from one application problem to another, and we therefore do not definitely specify how this anomaly-recurrency resolving should systematically be carried out in direct detail. However, we do give some generally useful and basic ideas about realizing such a mechanism as follows:

a) An anomaly is itself a kind of data, and in the case of image understanding under the signals-to-symbols paradigm, it always covers a certain portion of the image domain (or the scene domain). We call this the anomaly territory.

b) An anomaly recurrency generally occurs in an anomaly handler because there is a significant overlap of anomaly territories between the current
anomaly and previously resolved anomalies under the same anomaly handler.

To illustrate these two general ideas we give a short example. Assume that the following three items are among the existing data classes and processing clusters:

a) *primitive region*: This is a principal data class which contains regions at a primitive level.

b) *acute shape*: This is a class of anomalies containing regions at a more abstract level, each of them has an undesired acute shape.

c) *acute splitter*: This is the anomaly handler corresponding to the anomaly class *acute shape*. Its function is to resolve the anomalies in *acute shape* by splitting an input anomaly (i.e., an acute region) from *acute shape* into smaller but less acutely shaped primitive regions corresponding to the data class *primitive region*.

Figure 2.13 shows two of the possible anomalies from *acute shape*, which are resolved by *acute splitter* and transformed into some new primitive regions corresponding to the principal data class *primitive region*.

![Diagram](image)

Figure 2.13: Examples of anomalies which are resolved by *acute splitter*.

A recurrence may occur due to some anomalies in *acute shape* if the system, for instance, first encounters an anomaly like anomaly ⟨a⟩ in Figure 2.13, and some time later, after *acute splitter* has resolved the anomaly, it encounters again an anomaly like anomaly ⟨b⟩ in Figure 2.13, whose territory happens to overlap significantly that of the previous one as shown in Figure 2.14.
2.5 Further Discussion

Figure 2.14: An illustrative example of anomaly recurrency.

To detect such a recurrency, \textit{acute splitter} can set up a \textit{frequency map} of the same format as the image domain (or the scene domain), where pixel values corresponding to the territory of an anomaly will be increased whenever the anomaly actually occurs. An anomaly recurrency can thus be detected through inspecting, for instance, the averaged or maximal value among the pixels' frequencies within the anomaly territory involved. Subsequently, this value can be used by \textit{acute splitter} to select an anomaly with undesired high-degree recurrency and avoid it by just assigning the anomaly as a nonsense object in the final object space (this is the case where our encoded knowledge is at its wits' end) or flagging it as a new anomaly of another anomaly class. In this way, long or even infinite processing time due to possible anomaly recurrences of high degree can be effectively ruled out by the anomaly handlers themselves.

To conclude, our notions of anomaly and context hierarchy result in a structure in which the active knowledge is distributed over the processing clusters, in accordance with the data classes to be produced and anomalies to be tackled. An anomaly handler as an implicit part of this structure will only then be triggered if some corresponding anomalies have arisen. The result of an anomaly handler may again be an anomaly or a general data class appearing in the context hierarchy. In the former case, the new anomaly will be sent back (or made known) to the supervisor to replace the old one while in the latter case the output of the anomaly handler will be transferred to the relevant processing-blackboxes in the context hierarchy. In this way, a feedback link among the processing blackboxes within the context hierarchy is established through the anomaly handlers. We are convinced that our \textbf{DADS} design, based on the notions of context hierarchy and anomaly, may be expected to be quite powerful.
Chapter 3
System Design for DADS

In the previous chapter we only covered the issues of background motivation, general modelling and global description of the Distributed and Anomaly-Driven System (DADS). To realize such a system will, however, require a much more detailed design specification and architecture description involving representations and organizations of various data types and functional blocks. These topics are to be presented here.

Contrary to the system modelling, the system design will not be restricted to the realization of a system according to its modelling, but also carefully consider various practical issues, such as software flexibility, modularity, ease of future expansion and so on (refer for instance to [Som89], P.188-190 and P.240). In designing DADS towards a practical realization, we keep on following the system modelling and its general guidelines (see Page 21) on the one hand, while being actually aware of various implementational aspects and their practical implications on the other hand. Additional implementation guidelines on the system design are the following:

1. The system realization should provide a flexible environment for embedding the knowledge about (or solution to) a particular practical problem.
2. Once a realization has been made, it should be flexible with respect to any possible future expansion or shift of the application area.
3. The system realization should not unnecessarily hamper the system testing, on the contrary, it should make the testing as easy as possible.

The implications of the first two points are straightforward. To catch
the meaning of the third point we should note that DADS is aimed at achieving the capability of a general-purpose image-understanding system. To some extent, it resembles a general rule-based production system. A general production system satisfies the requirement as stated in point 3 above since the knowledge about (or solution to) a particular problem can easily be embedded in a flexible way by means of a rule base containing individual production rules. In DADS, such kinds of problem-specific knowledge are embedded through the context hierarchy and the processing clusters. It is therefore essential that this embedding can be accomplished in a sufficiently easy and flexible way.

3.1 Representational Specification of Data Classes

The main purpose here is to define a uniform representation mechanism for all kinds of data classes in order to approach the desired general-purpose nature for our DADS. There are basically two kinds of data, i.e., the classes of principal data and those of anomalies. By principal data we mean those which are to appear directly in the context hierarchy, like the input images, edge/region primitives, intermediate edge/region elements, high-level objects (or descriptions) and so on. Anomalies are either a principal data element with a certain strange property (e.g., a region which is too small) or an undesired composition of several principal data elements (e.g., a case in which an edge intersects a region). All data elements are classified into classes according to their generic properties. As there are many similarities among principal data and anomaly data, we will discuss their representations jointly and meanwhile make some individual remarks if necessary.

Any class of data in DADS is designed to be represented by its data descriptor and data storage\(^1\). Firstly, we briefly discuss the distinction between the notion of data descriptor and that of data storage. A data descriptor for a particular class of data is defined to be the description for the entire class such as class name, record type for individual member elements, various flags (e.g., an existence flag to indicate that at least one member element of the class has actually occurred at a certain moment)

\(^1\)Note that the terms data descriptor and data storage here are attached meanings which are different and extended with respect to their common meanings in computer science such as that within the context of compiler design.
3.1 Representational Specification of Data Classes

and so on. In other words, a data descriptor is a somewhat hollow but standing description for the class characteristics as a whole, regardless whether or not some member elements actually exist. Contrary to this, the data storage of a data class involves a significant portion of memory in which all individual descriptions of the actually existing data elements belonging to the class reside. Examples are the storage of an image array, an array of records describing individual primitive regions and so on. The need to make such a division among the description for a particular data class arises from the following consideration.

The precise organization for any class of data is basically problem dependent as far as the application scope of DADS is concerned. For example, in some applications a data class containing primitive edge segments is not needed while others may rely heavily upon such a class of data as some form of intermediate meta-data. Also, the specific form of the member elements in a commonly occurring data class may vary significantly from one application to another. For instance, the solution to a machine-part recognition problem may require a primitive region as some intermediate data to be described by the region's mean grey-value only, while a land-use pattern classification problem may in addition require the sum of squared grey-values or some textural measurements on the region as part of the region's description. Bearing this consideration in mind, a complete and precise specification for each possible data class within the DADS design is clearly impossible if we want the system to be flexible for a variety of applications. A similar example in this aspect is the familiar rule-based production system. From a system point of view, the set of potential production rules in fact constitutes the system data in any production system. Yet, at the design stage for such a production system it is not clear how many conditions or actions each potential rule is to assume. Nonetheless, it is crucial to the design of a particular production system that the design should explicitly specify the following properties for each rule in advance:

1. The maximal numbers of conditions and actions in each production rule.
2. The nature of each possible rule-condition in terms of some numerical, logical or symbolical evaluations in general.
3. The nature of each possible rule-action such as an implicit action by a procedure/function call, or an explicit action described directly within the rule-condition.
4. The usage of *(un)*certainty measures for various conditions within a production rule and the corresponding updating mechanism.

In fact, the specifications for the above rule properties are always part of the overall design for any production system even though they do not explicitly constitute any structural part of the main reasoning body.

Likewise, the data descriptor for a data class in DADS is needed to inform the system's overall controller (i.e., the supervisor) of the specific system status and relevant properties of the corresponding data class in such a way that the system's overall controller has sufficient knowledge about the data class in order to conduct properly the entire processing. The division into data descriptor and data storage has been made to create an efficient environment for the supervisor, since the amount of data within a particular data class is usually very large in applications of image understanding. This is mainly why we have decided to split the description of a data class into a permanent part (i.e., the data descriptor) and an incidental part (i.e., the actual data storage).

From the above argument, it is clear that the data descriptor for a particular data class is only to contain information which is of use for the overall coordination by the supervisor. In the current system design, a data descriptor for any data class is defined in the following way.

**Definition 3.1** Construction of a Data Descriptor

The data descriptor of a particular data class is a record with seven fields as follows:

- **name** a unique name for the data class.
- **rank** the rank of the data class.
- **address** indicates the location of the data storage corresponding to the data class if nonzero; otherwise indicates the absence of any actual member elements in the data class.
- **cluster** name of the corresponding processing cluster.
- **file** name of the temporary deposit file for the corresponding data storage.
- **size** size of the currently available memory in longwords for the corresponding data storage.
- **eff size** size of the actually occupied memory in longwords in the corresponding data storage.
3.1 Representational Specification of Data Classes

In the above definition, the notion of rank for a data class is crucial to the overall system controlling; its precise definition and usage will be discussed in the subsequent sections. Also, the meanings of the remaining fields will be explained as our discussion continues. At present, it is essential to know that the supervisor is hereby informed about the global status of any data class, though not directly about the detailed and individual descriptions of its possibly existing member elements.

Unlike the data descriptor for a data class, the data storage of a data class is to contain the actual descriptions of all individual existing member elements. It is mainly used by the individual processing clusters as their input and/or output source. However, it also serves as a communication medium between the supervisor and any potential processing cluster, which uses the existing data in the corresponding data storage as (part of) its input or output data. For instance, the supervisor may want to know how many elements in a data class an activated processing cluster has failed to process.

From the above considerations, we conclude that for the sake of overall processing coordination there is a need to specify in a global sense the formation for each data storage. The precise composition for each data storage will depend on the particular data class involved and may differ significantly for different data classes. In the DADS design, the data storage of a data class is assumed to correspond to a designated consecutive portion of the system’s global common memory and is required to consist of two standard parts, i.e., a header and a body. The header part is defined as follows.

**Definition 3.2**  Header of a Data Storage

The header of a data storage contains at least the following seven initial fields regardless of the class of data involved:

- **class id** system-wide identification of the corresponding data class.
- **head len** length of the storage’s header in longwords.
- **number** number of the existing member elements.
- **rec len** record length in longwords for any member element description.
- **size** size of the currently available memory in longwords for the data storage.
- **eff size** size of the actually occupied memory in longwords in the data storage.
- **finish** flag to indicate the number of the remaining unprocessed member elements.
Following the above seven initial fields, the header of a data storage can have some other extra fields whose contents are irrelevant for the system overall processing, but only known as well as useful to those processing clusters which get some input data from, or which put away output data in the data storage.

In the above definition, class id is a unique number assigned to the corresponding data class, with which the supervisor identifies the data class. For instance, if there are \( M \) classes of principal data and \( N \) classes of anomalies, then the numbers \( \{1, \ldots, M\} \) will be reserved for identifying the principal data classes while the numbers \( \{M+1, \ldots, M+N\} \) are to be used for the anomaly classes.

In fact, we require in Definition 3.2 that all member elements of a data class should be described by records of fixed size. It is not directly clear whether this requirement can always be satisfied. However, we do not see this as impossible. We explain this through the following examples with different types of data.

**Example 1.** Images as data.
Here, we can just define the record length to be the total size of a stored image.

**Example 2.** Primitive edge segments as data.
Suppose that the description of an individual edge segment is desired to include the coordinates of its ends and its chain code. Here, we thus encounter a case with individual descriptions of variable lengths. To solve this problem we propose to use the heap structure (refer to Appendix A for a detailed presentation of this structure). We explain this briefly as follows:

Split a single description into a fixed-sized part and a variably-sized part. Here, we can choose, for instance, the coordinates of both ends as the fixed-sized part and the chain code as the variably-sized part. To the fixed-sized part, we further add a new field, i.e., the record pointer in a heap structure. Our solution is to use the fixed-sized part as the record and store its corresponding variably-sized part in the heap structure. It is essential that the starting address of the variably-sized part within the heap structure is kept in the record pointer within the corresponding fixed-sized part. The heap structure here can, for instance, be placed directly following the array of data records in the given data storage and the starting address of the heap structure should then be put in a
predefined field among the remaining unspecified fields of the storage's header part.

The body of a data storage is defined to contain the remaining descriptions about the data elements in the class. Moreover, it should directly follow the corresponding header part and is required to start with the array of description records for the existing data elements. From example 2 above we can clearly see that the effective length (eff size) of the data description is not necessarily equal to the product of rec len and number plus head len as one might originally have thought.

As we mentioned earlier, if a class of principal data is at least productively directed to one other class, then all of its existing member elements should be exhaustively absorbed through its productive path(s). Moreover, a class of anomalies should always be resolved if some corresponding anomalies actually exist. To this purpose, the field finish in Definition 3.2 is used to inform the supervisor whether or not all member elements in a principal data class have been absorbed through its productive path(s) or whether or not all anomalies in an anomaly class have been resolved. If a class of principal data is not productively linked to any other class, then the contents in this field are to be ignored by the supervisor.

In continuation to this discussion, we further specify that the first field of each description record in a data storage should be a flag, called subflag. Because of its dynamic behaviour, a previously existing data element may be modified or even removed at some unpredictable moments, due to, for instance, certain feedback loops caused by anomaly handling. An implication of such a possibly iterative modification is that a previously legal description record for a data element may be declared as no longer valid at a later moment. One of the functions of subflag is to indicate whether or not the corresponding data description record is a valid one. In the current design, we have specified the first bit of subflag to fulfill this function. Also, other flagging functions from subflag are required for the following two cases:

1. If the class is an anomaly class.
   subflag should in this case indicate whether or not the corresponding anomaly has already been resolved.
2. If the class is a principal data class and linked productively to at least one other principal data class.
   Here, subflag is supposed to indicate for each individual productive
path whether or not the corresponding data element has already been absorbed through that productive path.

These functions are accomplished by using the remaining bits in subflag. Note that these functions are to be desired only if the field finish in the header part of the corresponding data storage is significant.

The specifications so far for a data storage are not only of interest for the global system modelling, but also may be profitable for future expansion of the DADS framework. For instance, taking a principal data class which is productively linked to at least two other data classes, we may add an extra field called confidence to each of the individual data description records. The field confidence can be used to assume a confidence value from, for instance, the interval [0,1] and will actually be assigned if the corresponding data element is absorbed through a productive path. It tells then how confident such an absorption is. At a later stage, when the data element is again to be absorbed through another productive path, it can be decided whether or not to accept the second absorption by comparing the previous confidence value and the currently resulting confidence value. The importance of having this kind of contest lies in the very desire to have our system emulate some of the competitive aspect in human perception (which is considered to be one of the strongest points in ensuring success in human perception). In this way, it can happen that the same data elements are transformed (processed) by more than one processing blackbox while the final decision for a minimally overlapping output is to be based on the individual confidence values of the output candidates.

3.2 System Description for Context Hierarchy

A context hierarchy is basically describable through a directed graph, where the vertices correspond to the classes of principal data and a link from vertex A to vertex B tells the system that the generation of any actual data elements in the class corresponding to vertex B needs some data elements in the class corresponding to vertex A as part of its input information. As we mentioned earlier, a binary attribute is attached to any link from one vertex to another, indicating either an associative or a productive relationship.

Since the context hierarchy is meant to represent a data-flow scheme,
3.2 System Description for Context Hierarchy

a vertex which does not point to any other one is called a sink, whereas a vertex which only points to others is called a source. In fact, sinks together correspond to the system’s output object space and the set of sources represent the initial input space which is generally single, namely the input image. From our previous system modelling, we notice that to any vertex other than a source, a processing blackbox will be attached, which optimistically performs the actual generation of the data class corresponding to the vertex based on its associatively and productively linked input data classes.

Though the context hierarchy is not explicitly divided into hierarchical levels in a straightforward manner, it does show a hierarchical tendency from the sources to the sinks. To make this hierarchical nature more explicit, we introduce our concept of rank within such a structure as follows.

Definition 3.3 Rank of a Vertex in a Context Hierarchy

Given a context hierarchy, we define an extra artificial vertex called supersource which links to all original sources in the context hierarchy. Moreover, we define the distance from two directly linked vertices to be one. Then, the rank for an arbitrary vertex in the context hierarchy is defined to be the maximum negative distance from the vertex to the supersource. The maximum negative distance here is the maximum distance from the vertex in question to the supersource measured along the opposite path directions, which do not include any paths between two bidirectionally linked vertices.

In Figure 3.1 the rank for each vertex in a possible context hierarchy is shown.

Relating to the notion of context hierarchy, there is another important aspect. A special situation occurs when two vertices in the context hierarchy are mutually linked through a bidirectional path. True, in many practical situations, it can happen that the existence of an object from one class reinforces or weakens the existence of another object from another class and vice versa. For example, in the field of aerial imagery interpretation the discovery of a motorway will enlarge the possibility of a car’s presence on the road and at the same time the confirmation of a car is very often a reason to enlarge the certainty about the existence of a motorway. In our terminology, the relationship between the data class of motorways and that of cars in such a case is said to be mutually associative. However, we feel that there will not be such bidirectional paths in a practical
context hierarchy which are of a productive nature. Thus we require all bidirectional paths in a context hierarchy to be purely associative. A possible confusion caused by the allowed existence of such a bidirectional associative path may arise in the calculation of the vertex rank following the above definition. We therefore require the path with maximal negative length not to include any bidirectional associative paths.

In fact, the rank of a vertex is to tell how 'far' a vertex lies from the supersource. Having defined the rank for a vertex in the context hierarchy, we are now ready to define the rank for a principal data class and an anomaly data class (or a processing blackbox and an anomaly handler) in the next.

**Definition 3.4** Rank of Principal Data Class and Processing Blackbox

a) The rank of a principal data class is that of its corresponding vertex in the context hierarchy.

b) The rank of a processing blackbox is that of its corresponding principal data class, i.e., the class whose member elements are to be produced by the processing blackbox.

To define the rank of an anomaly class, we first outline some additional design specifications on the anomaly classes and anomaly handlers as follows.
Additional Specifications on Anomaly Classes and Handlers
a) An anomaly handler can only solve one class of anomalies.
b) Each anomaly class has exactly one anomaly handler to solve the corresponding anomalies.
c) No two different processing clusters can produce anomalies of the same class.

Based on the above specifications, it is clear that we can make a one-to-one correspondence between the set of anomaly classes and the set of anomaly handlers, i.e., an anomaly class corresponds to the anomaly handler, which solves the anomalies from the class. By this correspondence we also define the rank of an anomaly handler to be that of its corresponding anomaly class. As in DADS an initial anomaly is always detected by a processing blackbox, we define the rank of an anomaly class in an inductive way as follows.

Definition 3.5  Ranks of an Anomaly Class and of an Anomaly Handler

a) If the anomalies from a particular anomaly class are detected by a processing blackbox, then the rank of the anomaly class is that of the processing blackbox.
b) Otherwise, if the anomalies from a particular anomaly class are detected by an anomaly handler, then the rank of the anomaly class is one plus the rank of that anomaly handler.

Our aim in introducing the notion of rank is, in fact, to establish some sort of ordering among the processing clusters in order to let the supervisor properly coordinate the processing by the various processing clusters. Through the above definitions on the notion of rank we can easily show the following proposition.

Proposition 3.1
Suppose vertex A is linked to vertex B and not vice versa, then the rank of A will always be smaller than that of B by at least 1.

We assume that if two vertices are bidirectionally linked through an associative path, then the data generation process for the principal data class corresponding to either one of the vertices can be started independent of
whether there are already some actual principal data elements corresponding to the other vertex. Under this assumption, we can, furthermore, prove the following important proposition.

**Proposition 3.2**

The data generation process for a principal data class through its corresponding processing blackbox can always be started if all principal data classes with smaller ranks contain at least some member elements.

Later when dealing with the overall system control, it will become clear that the property implied in Proposition 3.2 plays a crucial role.

In **DADS**, a context hierarchy is supposed to correspond to a particular application problem. Its representation is therefore a description containing global information about the existing principal data classes, their mutual relationships in terms of productivity and associativeness, corresponding ranks and processing blackboxes. Below, we give the constitution of such a descriptor.

**Definition 3.6** Descriptor of a Context Hierarchy

A given context hierarchy is described by the following entities:

- **context** a unique name for the context hierarchy.
- **dat num** number of principal data classes.
- **anm num** number of anomaly data classes.
- **dat nam** names of the principal data classes.
- **anm nam** names of the anomaly data classes.
- **box nam** names of the corresponding processing blackboxes.
- **ah nam** names of the corresponding anomaly handlers.
- **dat mat** co-incidency matrix among principal data classes defined as follows:

  \[
  \text{dat mat}(i, j) = \begin{cases} 
  0 & \text{if } i \geq j \text{ or } i \text{ is not linked to } j \\
  1 & \text{if } i \text{ is productively linked to } j \\
  2 & \text{if } i \text{ is associatively linked to } j 
  \end{cases}
  \]

- **cor mat** correspondence matrix among principal data classes and anomaly data classes defined as follows:

  \[
  \text{cor mat}(i, j) = \begin{cases} 
  1 & \text{if anomaly class } i \text{ corresponds to principal data class } j \\
  0 & \text{otherwise}
  \end{cases}
  \]

- **dat rank** ranks of the principal data classes.
- **anm rank** ranks of the anomaly data classes.
3.3 Design Specification on Processing Clusters

A processing blackbox and an anomaly handler can be considered to be similar in the sense that they both operate on some given input and yield some output data in a prescribed manner. This is why we call them both processing clusters of active knowledge. In the specification of the DADS design, a processing cluster is known to the outside mainly through its input/output specifications. A major characteristic of processing clusters is their highly dedicated and possibly sophisticated nature. For instance, an edge-growing production system as a particular processing cluster is governed by several specific production rules working solely on edge primitives. The local controller for such a production system can thus be kept simple and straightforward. Functionally, a processing cluster can be described as shown in Figure 3.2, where the parameter input is designed to make the processing cluster controllable to some extent by the outside world. In this way, a processing cluster can be both more flexible and more adaptive towards various applications.

![Diagram of Processing Cluster](image)

**Figure 3.2:** Functional description of a processing cluster.

In order to make a processing cluster known to the system, a descriptor for each processing cluster is kept by the supervisor. Such a descriptor for...
a processing cluster is defined as follows.

**Definition 3.7** Descriptor of a Processing Cluster

The descriptor for a particular processing cluster is defined to constitute the following eight fields:

- `name` a unique name for the processing cluster.
- `inp num` number of input data classes.
- `inp name` the list of the input data class names.
- `out num` number of output data classes.
- `out name` the list of the output data class names.
- `parameter` name of the parameter file.
- `status` on-line status of the processing cluster.
- `rank` rank of the processing cluster.

In the above definition, the field `name` is defined to contain the name of the processing cluster and the field `parameter` is required to hold the name of the corresponding parameter file, which contains the actual values for some parameters needed by the processing cluster. This file will be read in by the processing cluster whenever it is activated by the supervisor. It should be pointed out here that the flexibility of a processing cluster relies significantly on the contents of this parameter file. Except for the field `status`, the meanings of the remaining fields in the descriptor for a processing cluster are straightforward and need no further explanation.

Since a processing cluster in a DADS-based application system is not simply a one-time active functional block, it is necessary for the supervisor to know at run time the status of each processing cluster. This status value is kept within the field `status` of the descriptor for the corresponding processing cluster throughout the entire processing and it is defined in the following way.

**Definition 3.8** On-Line Status of a Processing Cluster

The on-line status of a processing cluster can assume four different values as described in the following:

1. **active.**
   This status means that the processing cluster is ready to be invoked. Such a status can be reached if some data elements in some input data classes for the processing cluster have appeared.

2. **inactive.**
The processing cluster is not ready for any invoking. A processing cluster will keep this status if no data elements in any of its input data classes appear. This status is in fact also the initial status for all processing clusters except for a processing blackbox corresponding to a source.

3. *ready.*

This status indicates that the processing cluster does not need to be invoked at present. Such a status can be reached after the processing cluster has successfully processed all of its current input data.

4. *pause.*

This is a waiting status for the corresponding processing cluster. A processing cluster at this status is interpreted in the sense that the processing cluster requires a temporary processing suspension until some additional input data come available. Such a situation may appear for a processing blackbox whose corresponding principal data class has a bidirectional associative link in the context hierarchy. Also, such a status may be reached, for instance, by an anomaly handler which waits to handle the present anomalies until there is some change in other data classes.

From a system point of view, it will usually be sufficient to have the foregoing specifications for a functional system block. Within a **DADS**-based application system, a processing cluster may, however, exhibit an unpredictably dynamic behaviour throughout the overall processing. The underlying idea for the system's overall controlling in **DADS** is based on a flow-network-like principle involving various data classes. In other words, the supervisor coordinates its controlling mainly through the current status of various existing data classes while the contents of these data class can basically only be changed by the problem-specific processing clusters. Bearing this in mind, it is necessary to specify how a processing cluster should communicate with the supervisor about some categorical issues concerning the system data (i.e., the various data classes). In order to accomplish such a specification, we need to study what specific characterizations these issues may have. In the current **DADS** implementation, we made a number of choices in this respect by considering the specific natures of a potential application problem in the field of image understanding. We call them the *categorical actions* by a processing cluster from the viewpoint of the supervisor. In the following, we state and specify these actions separately for a processing blackbox and an anomaly handler.
Specification of Categorical Actions by a Processing Blackbox

1. New elements in the corresponding principal data class among the output data classes have been generated by the processing blackbox.

2. The processing blackbox requires to wait until some new data elements among its input data classes emerge. Note that this action may result from a processing blackbox with bidirectional associative links at its corresponding principal data class as more consultative data may become available at a later stage.

3. Some previously existing data elements among the output data classes have been declared as no longer valid by the processing blackbox. In this case, the processing blackbox is required to notify the supervisor of the data class involved. This situation may occur since the action of an anomaly handler may, for instance, declare that some previously existing principal data elements in some of the input data classes for the processing blackbox are no longer valid while the processing blackbox had already used those currently removed input data elements to produce some output data elements at an earlier stage.

4. New anomalies have been generated by the processing blackbox. In this case, the processing blackbox is required to notify the supervisor of the anomaly class involved.

Specification of Categorical Actions by an Anomaly Handler

1. New anomalies in another anomaly class have been generated by the anomaly handler. In this case, the anomaly handler is required to notify the supervisor of the anomaly class involved.

2. New elements in a principal data class have been generated by the anomaly handler. Here, the anomaly handler is required to notify the supervisor of the principal data class involved.

3. The anomaly handler requires a processing blackbox to restart processing on its entire input data with newly established parameter values, which may include a different operator (e.g., gradient operator) within the corresponding processing cluster. The anomaly handler is in this case required to notify the supervisor of the data class which corresponds to the processing blackbox involved.

4. The anomaly handler requires a processing cluster to process any of its currently remaining and future input data with newly established parameter values. Here, the anomaly handler is required to notify the supervisor of the principal data class which corresponds to the process-
ing blackbox involved.
5. The anomaly handler has declared some existing data elements as no longer valid. The anomaly handler is required to notify the supervisor of the data class involved.
6. The anomaly handler requires to wait until some change has occurred in some data classes.

It is easy to see that, in fact, the above specifies which actions a processing cluster can categorically take and how it should inform the supervisor of its actions. This is also why we call these specifications the categorical actions by a processing cluster. In our view, these actions are sufficient to allow a dynamic processing behaviour, which is badly needed for solving any real problems.

Now, we discuss some specifications on a processing cluster, which are implied by the discussions above. These implied specifications are called processing specifications and are outlined in the following.

**Processing Specifications On a Processing Cluster**

1. An activated processing cluster should afterwards appropriately update the initial fields head len, number, rec len, eff size and finish (see also Definition 3.2) in the headers of the data storages corresponding to its input/output data classes.
2. If a processing cluster is restarted, either on its remaining input data (this may happen in the case of a bidirectional associative path), or on some newly produced input data (such as may happen when an anomaly handler requires some existing data to be reprocessed), it is the processing cluster's own responsibility (with the help of the field subflag in the individual data description records) to search for the desired data records in the corresponding data storage.

Finally, we explicitly point out that the task of detecting possible anomalies is basically the responsibility of the processing blackboxes. It is, however, not necessary that each processing blackbox be able to do so, or to actually produce some anomalies (hopefully none of them needs to). Also, as we mentioned earlier, an anomaly handler should bear the responsibility of detecting possible anomaly recurrences and of preventing any consequent processing oscillation to occur.
3.4 Global Architecture and Overall Controlling of DADS

As we pointed out in Chapter 2, DADS functionally resembles a general rule-based production system. In other words, DADS itself is only a framework. Figure 3.3 gives the overall architecture of an actual DADS-based application system. Here, the contents in the constitutional part of processing blackboxes and anomaly handlers are considered as external to the DADS framework itself. They depend on the actual application problem, which the DADS based system is assumed to solve. It is thus clear that the issue of constructing these contents is only then to arise when we are actually facing a practical problem and we intend to employ DADS for building up a real system to solve that particular problem.

![Diagram](image)

Figure 3.3: Global architecture of a DADS-based application system.

We can compare this with the construction of a set of production rules within a given production system. Most of the presentations so far in
this chapter are aimed at the specifications, which the actual filling of
the DADS framework for a particular DADS-based application system
should strictly follow. In this section, our presentation concentrates on
the direct design of the DADS framework itself and in particular the
supervisor.

In the following, we give a brief description for each of the constitu-
tional parts in the DADS framework.

*system status* is a block of system-wide physical memory where the
overall system status is constantly being kept. Among the system status
are, for instance, the descriptor for the current context hierarchy, the de-
scriptors for all processing clusters and the data descriptors for all possible
data classes. Thus, once the actual system has started processing with a
supplied context hierarchy, the supervisor will first create *system status*
as a standing and formation-fixed reference memory block. It is also ac-
cessible to all processing clusters. Throughout the entire processing, the
contents in *system status* will be updated by the supervisor and possibly
by some anomaly handlers.

*global memory* is also a block of system-wide physical memory created
by the supervisor right at the beginning of the actual processing. Here, all
data storages for the currently needed data classes (both principal data and
anomaly data) are to be stationed. In other words, it is in *global memory*
that an activated processing cluster is supposed to acquire its input data
and store its possible output data. Clearly, *global memory* should be made
accessible not only just to the supervisor, but, more importantly, also to
any activated processing cluster. Unlike *system status*, the size of *global
memory* will be quite large, since the actual data in an image understanding
process is usually huge. Also, due to the dynamic data behaviour in such
a process the actual formation within *global memory* may undergo many
changes during the processing. To prevent the necessity for a practically
unaffordably huge size of *global memory* and yet at the same time to ensure
that each activated processing cluster can get its input and dispatch its
output in *global memory*, we have introduced the concept of using deposit
data file in our current design. It assumes the availability of some disk files
(see disk storage in Figure 3.3), where the supervisor can temporarily store
some existing processing data which are not needed by or are irrelevant
to the currently activated processing cluster. In this way, the requirement
for a possibly huge size of *global memory* can be greatly reduced. Details
about the concept of the data deposit file will be described in the sequel.

As mentioned before, *supervisor* in Figure 3.3 is the system’s overall
controller for the DADS framework as well as for an actual DADS-based application system. Among the tasks for the supervisor are, roughly, the following:

1. Set up the initial system configuration such as the acquiring of the context hierarchy, initializing system status and global memory based on the given description of the context hierarchy, and establishing the communication channels required for on-line message exchanging with activated processing clusters.
2. Carry out the memory management task on global memory throughout the entire processing.
3. Activate processing clusters in a proper order.
4. Terminate the system processing at the right moment.

Before we set out to present the details about the functioning of the supervisor within a DADS-based application system, we will first point out that, all though based on our general system modelling, it is, in principle, allowed (and in fact very much desired) to have some processing clusters running in parallel, in our experimental system design for DADS no attempts have been made to explore such a practice.

In the following, we discuss the tasks for the supervisor in combination with a further presentation on the overall design of the DADS framework.

3.4.1 Initialization of the System Configuration

First, the supervisor will acquire the description of the context hierarchy, for instance by means of a given input file describing the context hierarchy. With the supplied description of the context hierarchy, the supervisor creates a sufficiently large global common memory for global memory and initializes it as a heap structure (see Appendix A for details about this structure). It is then ready for subsequent data deposits. Similarly, the supervisor creates system status containing proper initial values. This includes setting all processing blackboxes to the status inactive except those processing blackboxes which correspond to the sources in the context hierarchy which are set to the status active. Subsequently, two communication channels are set up by the supervisor, one for the on-line message traffic from the supervisor to any activated processing cluster and the other for the message traffic from an activated processing cluster back to the supervisor. (All processing clusters in our experimental DADS implementation
are to be separate programs in order to facilitate flexibility in the software development.)

3.4.2 Global Memory Management

An arbitrary data class may be related to more than one processing cluster, either as input or output information. It is therefore necessary to make the data storages of relevant data classes available in advance to any of the processing clusters. The introduction of global memory for this kind of data access basically aims at minimizing any undesired delay caused by the data transfer (we must realize that a problem in the field of image understanding will generally involve a huge amount of data). However, it is hardly feasible in practice to create a common memory which is so large that individual memory portions can be assigned to each data class as indicated by the current context hierarchy, with the assurance that these memory portions will be large enough to hold all actually occurring data elements during the entire subsequent processing. The global memory management task for the supervisor is thus to provide each activated processing cluster with the data storages of those data classes which are to be involved in the due-to-be-activated processing cluster.

We note that much of the processing within the framework is basically serial. A processing cluster will not need access to the existing data belonging to all possible data classes (otherwise, the effort to distribute the entire processing over various processing clusters does not serve any useful purpose). It will be sufficient if only those data storages are present in global memory which correspond to the input and output data classes of the processing clusters which are due to be activated. It is this remark, under which the use of global common memory becomes a practically feasible (though the size of global memory will generally still be large even though it is not unaffordably large).

The above implies the dynamic organization of the contents in global memory. By considering the set of data storages for all possible data classes as a set of variably-sized records (though the size of each such record may be very large), we have solved this memory-management issue through the heap structure in combination with temporary deposit files as explained in the sequel (for a more general discussion on dynamic memory management we refer to [Aho77]).

Suppose that global memory is divided into data storages as shown in Figure 3.4 at a given moment, where $M_1$, $M_2$ and $M_4$ correspond
Figure 3.4: A division of *global memory* at a given moment.

to the data storages of three previously involved data classes, *RM* is the remaining free space in *global memory* and *M3* represents a *released* portion of free memory. A released portion of memory is a portion of memory which was previously occupied by a data storage but the contents have been subsequently saved by the supervisor in a designated deposit file. Suppose further that the processing cluster, which is due to be activated, involves three data classes with their data storages corresponding to *M2*, *N1* and *N2*. The memory management task for the supervisor is then to keep *M2* and create *N1* and *N2* in *global memory* in a proper way so that the processing cluster can locate them when activated. We discuss such a task for the supervisor according to different possible situations as follows.

1. *RM* is larger than *N1+N2*.
   
   In this case, the supervisor will just create *N1* and *N2* within the current *RM* and the formation of *global memory* will become as shown in Figure 3.5a with *RM* now reduced.

2. *RM* is smaller than *N1+N2* but *RM+M3* exceeds *N1+N2*.
   
   Here, since the current RM can not simultaneously hold *N1* and *N2* while the total free space in *global memory* can still hold *N1* and *N2*, a *garbage collection* operation will first be carried out by the supervisor on *global memory* in order to combine *M3* with current *RM* into an enlarged *RM*. Subsequently, *N1* and *N2* are properly created within the newly established *RM*. The resulting configuration of *global memory* is shown in Figure 3.5b.

3. *RM+M3* is smaller than *N1+N2* but *M1+M3+M4+RM* exceeds *N1+N2*.
   
   In this somewhat unfortunate situation, the supervisor will first save the contents of *M1* and *M4* within corresponding temporary deposit files and subsequently release the memory portions occupied by *M1*
and $M2$. Then, the previously mentioned garbage collection operation is conducted to combine the current $RM$ with the released memory portions. Finally, the supervisor creates $N1$ and $N2$ resulting in a new configuration as shown in Figure 3.5c.

![Diagram](image)

(a) $M1$ $M2$ $M3$ $M4$ $N1$ $N2$ $RM$

(b) $M1$ $M2$ $M4$ $N1$ $N2$ $RM$

(c) $M2$ $N1$ $N2$ $RM$

Figure 3.5: Modified formations of global memory.

Here, we have to point out that it could in principle happen for the above example that $M2+N1+N2$ exceeds global memory itself. This would unfortunately result in a fatal error. However, such a situation will have a low probability of occurrence if global memory can be made sufficiently large. Remember, the basic issue is to distribute problem-specific active knowledge into separate processing clusters and any processing cluster involving an excessive number of data classes is likely to be the result of poor use of DADS.

Two remaining aspects with respect to the memory management task for the supervisor still need further clarification. They are the following:

a) After creating $N1$ and $N2$ in the previous example, the supervisor should check the current status of $N1$ and $N2$ in their corresponding data descriptors to see if the data storages corresponding to $N1$ and $N2$ previously received any data. If this is the case, the supervisor should retrieve these data from the corresponding temporary deposit files and transfer them into the newly created $N1$ and $N2$, respectively.

b) Directly after the processing by the activated processing cluster involving $M2$, $N1$ and $N2$, the supervisor should check whether the assigned data storages (i.e., $M2$, $N1$ and $N2$) still contain some actual data or
have actually received any. If any of them contain no actual data, then the corresponding memory portions in *global memory* will be released.

With the above descriptions, we conclude our discussion on the global memory management task for the supervisor.

### 3.4.3 Overall Process Control within a DADS-Based Application System

In this section, our main emphasis is on the major task for the supervisor, i.e., the overall on-line processing control. Roughly speaking, this includes proper activation of the various processing clusters and, eventually, termination of the entire processing.

To understand the overall control, we may view the DADS framework represented by the supervisor as a general rule-based production system (e.g., the arbitrator in a primitive purely rule-based production system) and similarly consider the various processing clusters as the set of production rules within such a system. We observe the following major controlling issues for the supervisor from a system’s point of view:

1. Which of the existing processing clusters should be chosen as the next candidate for activation?
2. How to react to and handle the possibly received messages from a currently activated processing cluster?
3. Should the whole processing be terminated at a particular moment?

Remember that the system modelling of DADS (refer to Chapter 2) is based on a data-flow-like scheme (i.e., for the principal data as indicated in the context hierarchy) as well as on the anomaly-driven mechanism (note that anomalies are basically also system data themselves though they are somewhat special). This yields the following observations:

a) A processing blackbox should always be activated if it does not need any input data. In the DADS terminology, such a processing blackbox corresponds to a *source* in the context hierarchy and its function is generally to read the input image(s) into the system.

b) Any of the occurring principal data should be made to flow upwards to some higher-ranked principal data classes along the available paths as indicated in the context hierarchy. This upward flow means either
an actual absorption by some higher-ranked principal data or a reference by the generation process of some higher ranked data according to the attributes of the involved paths (i.e., the productive and associative attributes). In particular, if a principal data class has one or more productively attributed upward paths, each of its existing member elements should be absorbed along at least one such path.

c) Anomaly data are supposed to appear in an irregular and unpredictable way. They must all be resolved by their corresponding anomaly handlers, should they actually occur during the processing.

d) The entire processing can be terminated if no unresolved anomalies are present and the currently existing principal data have no way to flow upwards under the given data-flow scheme. These principal data classes correspond in fact to the sinks in the context hierarchy.

To accomplish such a data-flow based and anomaly-driven processing, the supervisor is designed to deal with it by means of the current status of all possible data classes (i.e., both principal data and anomaly data). Before presenting the details of the overall controlling by the supervisor, we describe how the supervisor should respond to the various categorical actions of an activated processing cluster (see also Section 3.3). We distinguish separate cases for a processing blackbox and an anomaly handler.

Responses to Various Categorical Actions of a Processing Blackbox

1. New elements have been produced in the output principal data class of the activated processing blackbox.

Suppose that vertex A corresponds to the principal data class, which has received the new member elements. Upon notification by the processing blackbox, the supervisor will update the data descriptor of the involved principal data class and look for all existing parent vertices of A and set the status of the corresponding processing blackboxes to active. Also, all anomaly handlers with the status of pause are set to active.

2. The currently running processing blackbox requires to wait until some new input data become available.

Upon notification by the processing blackbox, the supervisor will set the status of the processing blackbox to pause.

3. Some previously existing data elements among the output data classes of the processing blackbox have been declared no longer valid.
Upon notification by the processing blackbox, the supervisor will update the data descriptor of the involved data class. If the involved data class is a principal data class corresponding to vertex $A$ in the context hierarchy, then the supervisor will look for all parent vertices of $A$ and set the status of their corresponding processing blackboxes to *active*. Otherwise, if the involved data class is an anomaly class, then the supervisor will check its data storage to see if there are any remaining anomalies. In the case of remaining anomalies, the corresponding anomaly handler will be assigned the status of *active* and otherwise it will be assigned the status of *inactive*. Also, all anomaly handlers with the status of *pause* are set to the status of *active*.

4. New anomalies have been generated.

Upon notification by the processing blackbox, the supervisor will change the status of the corresponding anomaly handler(s) to *active* and update the data descriptor corresponding to the involved anomaly data class. Also, all other anomaly handlers with the status of *pause* are turned over by the supervisor to the status of *active*.

For an activated processing blackbox, the supervisor will update the data descriptors for all data classes, which appear among the input and output of the activated processing blackbox. Moreover, if situation 2 above does not meanwhile occur, the status of the activated processing blackbox will afterwards be switched to *ready*.

**Responses to Various Categorical Actions of an Anomaly Handler**

1. New anomalies of another anomaly class have been generated.

Upon notification by the anomaly handler, the supervisor will update the data descriptor of the involved anomaly data class and set the status of the corresponding anomaly handler to *active*. Also, all other existing anomaly handlers with the status of *pause* are switched to the status of *active*.

2. New elements in a principal data class have been generated.

Upon notification by the anomaly handler, the supervisor will update the data descriptor of the involved principal data class. Suppose that the involved principal data class corresponds to vertex $A$ in the context hierarchy, the supervisor will look for all parent vertices of $A$ and set the status of the corresponding processing blackboxes to *active*. Also, all existing anomaly handlers with the status of *pause* are turned to
3.4 Global Architecture and Overall Controlling of DADS

Active.

3. A processing blackbox is required to restart processing on its entire input data with newly established parameter values.
Suppose that vertex A in the context hierarchy corresponds to the involved processing blackbox. Upon notification by the anomaly handler, the supervisor will look for all vertices higher ranked than A to which there is a path from A. If any such vertex is found, the supervisor will discard the possibly existing data in the corresponding principal data class and properly update the corresponding data descriptor as well as the descriptor of the corresponding processing blackbox. Also, similar action is applied to vertex A. When discarding entire existing member elements of a principal data class, the supervisor will look for its productively linked lower-ranked vertices and properly update the individual fields subflag in the data storages corresponding to such a child vertex. Finally, the processing blackbox, which is required to restart, will be assigned the status of active. Note that it is the task of the anomaly handler, not the supervisor, to appropriately update the parameter file of the involved processing blackbox.

4. A processing cluster is required to process its remaining and future input data with newly established parameter values.
Upon notification by the anomaly handler, the supervisor will check the status of the involved processing cluster. If its status is pause, it will be turned to active. No further actions will be taken by the supervisor. It is again up to the anomaly handler, but not the supervisor, to update appropriately the parameter file of the involved processing cluster.

5. Some existing data elements have been declared as no longer valid.
Upon notification by the anomaly handler, the supervisor will update the data descriptor of the involved data class. If the involved data class is a principal data class corresponding to vertex A in the context hierarchy, then the supervisor will look for all parent vertices of A and set the status of their corresponding processing blackboxes to active. Otherwise, if the involved data class is an anomaly class, then the supervisor will check its data storage. In the case of no remaining anomalies, the corresponding anomaly handler will be assigned the status of inactive. Also, all anomaly handlers with the status of pause are turned to active.

6. The currently running anomaly handler requires to wait for further handling of its remaining anomaly data until some change has been observed in some data classes.
Upon notification by the anomaly handler, the supervisor will properly
update the data descriptor of the corresponding anomaly class and set the status of the anomaly handler to *pause*.

Similarly to the case with a processing blackbox, the supervisor will update the data descriptors for all those data classes which appear among the input and output of the activated anomaly handler. Moreover, if situation 6 above does not meanwhile occur, the status of the activated anomaly handler is afterwards converted to *inactive*.

From the above presentation, we observe that when a next processing cluster is due to be chosen for activation the overall state of the entire system conforms to the following description:

a) Any of the existing processing blackboxes will assume one of the four previously defined statuses according to the following situations:

- **active**: The processing blackbox has some existing input data to process.
- **inactive**: None of the input data classes for the processing blackbox ever received any member elements.
- **pause**: The processing blackbox has some existing input data to process but wishes to wait until some change has been observed in some data classes.
- **ready**: The processing blackbox has currently processed all of its available input data in respect to generating some elements in its corresponding principal data class.

b) Any of the existing anomaly handlers will assume one of the three previously defined statuses according to the following situations:

- **active**: The anomaly handler has some corresponding anomalies to resolve.
- **inactive**: The anomaly handler has no or no longer has corresponding anomalies to resolve.
- **pause**: The anomaly handler has some corresponding anomalies to process but wishes to wait until some change has been observed in some data classes.

Moreover, we can see from the response descriptions for the supervisor that all possible data classes will maintain compatibility for their respective member elements with respect to the current status of all processing clusters. This means that the following situations cannot occur:
3.4 Global Architecture and Overall Controlling of DADS

a) A principal data class has some unabsorbed member elements while none of the existing processing clusters, which correspond to its productively or associatively linked vertices with a higher rank, has the status of active or pause.
b) An anomaly class has some unresolved member anomalies while its corresponding anomaly handler is at the status of inactive.

Thus, we conclude that the activation of a next processing cluster should only ensure the following two aspects in view of the data-flow model and the anomaly-driven concept:

a) Only those processing blackboxes can be activated whose input data classes actually contain some member elements, with the possibly exception of an input data class which is linked through a bidirectional associative path. This consideration is a direct consequence of the data-flow model.
b) An anomaly handler can always be seen as among the currently activatable processing clusters as long as it has some actual corresponding anomalies. Here, the anomaly-driven concept is reflected.

Based on the above considerations, the general strategy for activating a processing cluster is outlined below:

General Strategy for Processing Cluster Activation by the Supervisor.

If there are some anomaly handlers with the status active, then choose the one with the lowest rank for next activation. Otherwise, choose that processing blackbox which has the lowest rank among the processing blackboxes currently at the status active. If none of such processing blackboxes can be found, then start to terminate the entire processing.

As we can see from Proposition 3.2, the strategy to select the lowest-ranking processing blackbox for activation will ensure that an activated processing blackbox will indeed have its necessary input data elements. In this way, the concept of data flow is strictly followed.

By allowing anomalies to occur and preferring the activation of anomaly handlers, we observe that a backtracking possibility over quite distinct processing levels is provided in an effective and efficient way. In this way, a
DADS-based application system can allow a quite dynamic processing behaviour.

As far as the overall controlling within DADS is concerned, the only remaining issue is how the entire processing is to be terminated. In the above general strategy for processing cluster activation, a description was given of when this will take place. In terminating the entire process, the supervisor will perform the following:

Search for processing clusters which are at the status of pause. If any are found, the supervisor will check the input data classes for these processing clusters. If such a processing cluster is a processing blackbox and it has a productively linked input data class still with some unabsorbed member elements, then these unabsorbed member elements will be transfered to the non-sense object data class. Otherwise, if such a processing cluster is an anomaly handler still with some unresolved corresponding anomalies, then the supervisor will issue an error description about this situation.

This completes our presentation of the overall system design and its specifications. In Appendix B, additional implementational aspects are discussed.
Chapter 4

Concluding Comments and Discussions

Thus far, we have presented our Distributed and Anomaly-Driven System (DADS) for the purpose of general image understanding by computer. Applying DADS to a particular practical problem, however, implies the tasks of constructing the context hierarchy and defining the corresponding anomalies as well as programming the various processing blackboxes and anomaly handlers. This is basically similar to the need for encoding the particular problem-specific knowledge into a set of production rules when applying a general rule-based production system.

For more than a decade, there has been a trend to view almost any particular system as an expert system if production rules and/or some advanced data structures (such as frames, semantic networks using for instance part-of, is-a and other relationships) are used. In our opinion, however, processing based on a production system is just another way of processing, and the above-mentioned advanced data structures are likewise simply yet other ways of expressing knowledge. More traditional techniques of processing and expressing are still potential candidates for reflecting human knowledge. In our opinion, a system in a particular application truly deserves the status of expert system only if it appropriately employs and combines various techniques of processing and expressing in accordance with the performance behaviour of a human expert (refer also to [Her88] for a broad discussion). For example, we feel reluctant to call the system ANGY (see [Sta86]) an expert system. We do acknowledge that production-rule based processing can reflect some of the human knowledge in the field involved. However, we doubt (as ANGY’s author implicitly
acknowledged) whether production rules are appropriate, or sufficient, or even significant with respect to reflecting human knowledge within that particular problem field.

In addition to the above, we therefore avoid questions such as whether or not a particular application system based on the DADS framework may be considered to be an expert system. In fact, we view our DADS basically as a new way of processing and expressing, designed specifically for the purpose of image understanding by computer. The significance and the potential status of expert system for a DADS-based application system depend essentially on whether or not the proposed techniques for processing and expressing are appropriate and sufficiently effective to reflect human perceptual knowledge, and, in particular, whether or not the application will yield significantly better results or at least imply the possibility of eventually obtaining better results through further extensions (e.g., the introduction of more data classes, the modification of some existing processing clusters and so on).

In practice, to require a machine to behave fully in accordance with human expert behaviour will in many cases turn out to be infeasible, or at least be so for the near future. Unlike the issue of image understanding by computer, however, there are many areas such as natural-language processing, speech recognition, planning and medical consultation, in which such an expert level may successfully be approximated by a production system which is basically rule based. We think that the main reason behind such potential success is the fact that these problems are comparatively much better understood and better defined than those in the area of image understanding. Computationally, a marked difficulty in image understanding by computer compared to, for instance, speech recognition is the complexity of mutual relationships among entities in the spatial domain. For example, the relationships among entities of the same abstraction level are, in principle, only to contain the precedence and post-concatenation for speech recognition whereas such a relationship for image understanding by computer will, in principle, have to involve every possible spatial orientation. Also, as far as the methodology behind a general rule-based production system is concerned, a tough difficulty typically arising in image understanding is the difficulty of understanding, defining and extracting meaningful primitives for the reasoning process. As noticed in [Tso84], such a difficulty will be much more limited in the area of natural-language processing. For instance, words as primitives are well defined, well understood and relatively simple to extract. We do not claim that the difficulties
as mentioned above will vanish with the DADS framework. However, we do feel that DADS provides a better environment specifically for the purpose of image understanding, within which we may have a better chance of successfully encoding our problem-specific knowledge. In particular, we can encode the knowledge about our conscious thinking in a more logical and yet distributive way through the mechanism of context hierarchy and anomalies, while effectively distributing our mostly unconscious knowledge over the processing blackboxes and anomaly handlers. We believe that our DADS framework will, in the end, make a significant contribution towards tackling the above-mentioned difficulties, though not directly solving them.

When applying the DADS framework to a particular practical problem, the major issue is how appropriately to choose and define the context hierarchy and the corresponding anomalies and how successfully to encode various processing blackboxes and anomaly handlers. Obviously, these issues are crucial to any successful DADS application. Like a general rule-based production system, DADS only provides a potentially better system framework but does not directly provide the solution to any particular problem undertaken.

To summarize, we observe the following about the DADS architecture:

a) By using a context hierarchy the main system emphasis will not be disrupted or misguided by possibly unexpected local irregularities, and the global attention can focus on the specific global context of the problem undertaken.

b) With the notion of anomaly, measures against local irregularities are only then to be taken (as exceptional actions) if the irregularities are actually encountered. Moreover, irregularities of quite different natures can hereby be tackled consistently, without a proportional increase in the complexity of the main system architecture.

In addition to its conformity with our perception modelling, DADS has four outstanding features as summarized below:

1. Powerful Capability:
   Many special cases or irregularities can be dealt with by defining corresponding anomalies and adding appropriate anomaly handlers.
2. High Efficiency:
   Rarely occurring irregularities will have a minimal chance to deviate
   the main processing attention under the anomaly-driven mechanism.

3. Flexibility and Effectiveness:
   The developments of various processing clusters can be undertaken in-
   dependently. They can be made quite sophisticated and dedicated to
   individual subproblems. Computationally, they can individually em-
   ploy the most suited method (e.g., reasoning with production rules,
   searching in a semantic network structure or matching with graphs) as
   long as they produce the required output. This clearly makes DADS
   both flexible and effective.

4. Reliability:
   By using a suitable mechanism to scale the confidence values (see re-
   marks on Page 50 of Chapter 3), the DADS framework can support the
   establishment of some competitiveness among various processing clus-
   ters. Undoubtedly, this leads to an increase of the processing reliability,
   even to a significant extent.

At last, we state with emphasis that the justification of our DADS
design relies heavily on both our perception modelling and the signals-to-
symbols paradigm for machine perception.

There are some issues as stated below, which will be of interest to any
future research and development on the current DADS design:

1. To what extent is the working environment provided by our DADS
   framework adequate in respect to our problem-specific knowledge?
2. How should a successful mechanism for the confidence values be devised
   in order to establish some competitive feature?
3. What kind of human expert knowledge about image understanding can
   essentially not be incorporated within the framework as presented in
   our current design for DADS?
Part II

A DADS Application for the Interpretation of SLAR Images
Chapter 5

Introduction

In Part I of this thesis we introduced and discussed the Distributed and Anomaly-Driven System (DADS), designed particularly for the purpose of building up a practical image understanding system for a broad range of applications. As we pointed out there, DADS is basically only an overall system framework for a solid practical system, similar to the processing shell of a general rule-based production system. To employ a given rule-based production system in solving a particular application problem, one is always required to develop separately a particular set of problem-specific production rules. Moreover, such a development of problem-specific production rules is in many senses irrespective of the existing production system. In a similar way, the development of a DADS-based application system also involves a separate task, given our DADS framework. This separate task includes the construction of a suitable context hierarchy, the definition of various anomaly classes and, more importantly, the programming of the corresponding processing blackboxes and anomaly handlers. We emphasize again that the employment of the DADS framework in the construction of a real application system by no means implies that DADS itself provides a solution to the problem undertaken. Rather, DADS provides a particularly suitable working environment within which our interpretation knowledge of the particularly undertaken problem may be more easily explored, stored and successfully deployed.

In Part II of the thesis, we are going to discuss an example which yields a DADS-based application system for interpreting SLAR images, i.e., Side-Looking Airborne Radar images. An SLAR is an active imaging radar based on microwave reflectances (for a detailed presentation of SLAR we refer to [Ula81], P.42–47). Our purpose here is to show that the

79
DADS framework is indeed suited to disseminate as well as to encode our knowledge about a particular but complex problem in the area of image understanding.

The development of such an application system is very much irrespective of the given DADS framework. It is therefore quite natural that a significant portion of the presentation in this part of the thesis is only indirectly related to the DADS framework itself. Among the difficult problems here is the construction of effective and successful processing clusters under a chosen context hierarchy and a set of observed anomalies. It should be noted that the construction of the context hierarchy and the definition of anomalies should not be separated from that of the corresponding processing blackboxes and anomaly handlers, respectively. We should realize that a processing cluster usually attempts to emulate some part of our unconscious thinking (e.g., an edge detector). Therefore, it is not guaranteed that a particular processing cluster, which is going to produce what we desire of it (such as indicated in the context hierarchy), can be constructed. In particular, this reminds us not to define any data classes for which we are currently unable to construct a corresponding processing cluster (i.e., a digital transformer) because of our currently limited capability in digital emulation.

5.1 Background of the SLAR-Image Problem

Analyzing SLAR images is part of the investigation on the use of remote-sensing techniques for vegetation mapping in agriculture and forestry. Research in this field has already been going on in the Information Theory Group for several years under the national cooperative research group ROVE (Radar Observation of Vegetation) in the Netherlands as part of the National Remote Sensing Programme (refer to [Loo82]).

A typical SLAR image is shown in Figure 5.1. Roughly speaking, our ultimate purpose in SLAR-image analysis is to have the computer automatically give a full interpretation of the given image(s) in terms of crop fields of different types (such as wheat, potatoes and so on), canals, motorways, roads, villages, farms, and others. It will then become possible to obtain the current status of different crop fields, which can in turn serve the purpose of, for instance, the harvest estimation of various crops within the observed area. Many fruitful results connected to this research have
already been achieved (see [Koo81,Kon82,LeLee84,Kle88,Ger88b]). However, the complete problem is far from being solved. Some of the basic difficulties are among the following:

1. Image registration.
   This is essential if we want to make use of some available a priori knowledge such as existing maps, where the locations of existing motorways, country roads and villages are approximately specified, or if multiple images of the scene must be conditioned.

2. Strong variations in grey-value characteristics.
   Under the same ground truth, the individually received reflectances by the radar, which are ultimately converted into grey values of the pixels, can be quite different. Such effects are, for instance, observed with different grazing angles of the radar pulses. In practice, it is extremely difficult to model the radar reflection of crops accurately. Also, the weather and aircraft conditions may contribute to the radar reflectance variations in an unpredictably complex manner.

3. Temporal variations of image data.
   The radar reflectances of a particular crop area depend not only on the crop type but also on the current growing stage of the crop involved. The modelling of the temporal variation for each type of crop is crucial for the final classification of crop fields, but is again hard.
Parallel to the research activities directed towards solving the above individual issues, our present approach will focus on a limited interpretation problem, which is formulated as follows.

**Problem of SLAR-Image Analysis**

Given an SLAR image which has already been preprocessed to correct some imaging conditions (such as the radar conditions, geometric distortions), find the locations of existing crop fields (not necessarily to classify them into different crop types).

According to our previous discussion on DADS, we observe from the above problem formulation that the corresponding context hierarchy will have one source vertex, which corresponds to the actual input image and two sink vertices, which stand for the desired crop fields and all other regions, respectively. In the DADS terminology, the possibly unclassifiable regions correspond to the *nonsense* objects. Of course, to build up such a complex image interpretation system within the DADS framework, we need to deeply explore our current (theoretical or empirical) knowledge of the problem in question. In the next, we concentrate on this issue and gradually extend our exploration effort.

### 5.2 Preliminary Steps Towards Knowledge Exploration

As has been done in practice, if we want to build a machine to solve a particular application problem, we always start by looking for an *expert* who knows how to solve the problem at hand. If we find such an expert, our primary task will then be to acquire the relevant knowledge from the expert and, subsequently, to try to encode the acquired knowledge into the machine. For example, in aircraft on-board monitoring, an established theory on aerodynamics can be such an expert (or part of such an expert) for a machine which is designated to control the aircraft's flying balance under different aerodynamic conditions; in medical diagnostics, an experienced doctor can be such an expert in the building of a consultation machine in the relevant medical field; in speech recognition, a dictionary or a context-specified vocabulary together with an associated grammar will be such an expert in the building of a machine which is to understand supplied speech passages within the prescribed context. For our current
problem of SLAR-image analysis, it would thus be natural to first find such an expert (note that an expert is in our consideration a collection of established theories and techniques together with experienced manpower), even though our eventual expert may only have limited knowledge which can be written down. Experience has taught us that such a collective expert does not exist in a usual way. In an effort to find such an expert, we try to explore our own past experience and visual knowledge and hope that they may turn out to constitute our desired expert. Thus, instead of starting with knowledge acquisition, we will start with knowledge exploration.

Below, we give some examples of the related knowledge based on our own expertise, which is largely based on common-sense reasoning:

$R_1$: Individual crop fields can be seen as individual regions, each of them is sufficiently uniform with respect to a particular uniformity measure.

$R_2$: A region distinguishes itself from its neighbouring regions by observable differences between some within-region mean measurements.

$R_3$: A crop field normally has a near-polygonal shape with quite smooth sides.

Currently, we have much more of such common-sense knowledge about the contents of an SLAR image, which will be individually explored as our presentation proceeds. For the present, it is enough to know that our written or expressible knowledge based on common-sense reasoning shows the following properties:

a) It always involves some data or data descriptions.
b) It only describes some intrinsic properties of the individually occurring data/data descriptions or the mutual relationships among the occurring data/data descriptions.
c) All involved data or data descriptions originate from the data type region, which is roughly described by $R_1$ and $R_2$.

Bearing in mind the imprecise nature of common-sense reasoning, we conclude that our immediate major tasks in exploring and encoding such knowledge are the following:

1. Formally define the notion of region and correctly quantify the intrinsic properties of the data type region.
2. Formally define and correctly quantify the mutual relationships among all occurring data types.

In the following we discuss the first task and gradually come to the second task.

Because of the common-sense reasoning, we note that the meanings of sufficiently uniform in $R_1$ and observable differences in $R_2$ are imprecise and so are the meanings of uniformity measure and within-region mean measurements. Psychologically speaking, $R_1$ and $R_2$ only reveal some of our conscious part of the thinking, whereas the exact meanings of sufficiently uniform, observable differences, uniformity measure and within-region mean measurements there are related mainly to our unconscious thinking during the process of our perception. We may say that it is these imprecise terms where our conscious and unconscious thinking meet (see also for instance [Hof79], P.647). In a normal verbal manner, it is hardly possible to further explore our common-sense based knowledge in order to capture the exact meanings of these terms. As we mentioned earlier, our job will be done if we can emulate the behaviour of a human expert through our machine. Here, we will do exactly the same to overcome the difficulties related to our unconscious thinking.

As we have already observed, our common-sense based knowledge currently offers no intermediate meta-data (or meta-descriptions) between the data type region and the input image pixels. Obviously, our first task will be to emulate as closely as possible our mental performance in order to transform the input image pixels into those regions. In the usual terminology, this transformation corresponds to the low-level image segmentation. An initial context hierarchy of our SLAR-image analysis system will thus take the form shown in Figure 5.2, where the class region corresponds to our desired regions.

---

![Diagram](image-url)  

**Figure 5.2:** The initial context hierarchy for SLAR-image interpretation.
5.3 Combined Segmentation Approach

In much of our previous work the emphasis has been on region-based segmentation approaches (see, for instance, [Kon82, Lee84, Che85, Kle88]). However, the achieved results do not fully satisfy our desire to have a sufficiently accurate segmentation. Evidences of false region boundaries and regions comprising multiple subregions with different characteristics remain to appear. If one is familiar with the issue of image segmentation, one will undoubtedly acknowledge that none of the traditionally existing region-based segmentation approaches will be able to satisfactorily segment such complicated images as we have. In our view, such a phenomenon is basically inherent in the methodology of the traditional region-based segmentation approach on the one hand, and on the other hand, to require an almost fully satisfactory result in solving an image understanding problem, the incorporation of some high-level knowledge is indispensible and, in particular, the guidance by the encoded high-level knowledge towards the low-level segmentation on a dynamic basis. To tackle this difficult problem, we explore the use of the DADS framework in order to establish such a dynamical interaction between the high-level and low-level processing. However, some inherent drawbacks in the region-based low-level segmentation approach are still to be eliminated, which in our view are not essentially an outcome, due to the lack of high-level guidance. In the next section, we introduce our proposal for a combined approach with both region-based and edge-based methodologies for the purpose of low-level image segmentation.

5.3 Combined Segmentation Approach

The issue of image segmentation has played a crucial role in nearly all of the problems arising in the field of image understanding. A tremendous number of techniques have been developed, each of them appears individually to have some specific merits in some specific application problems (refer to [Har85, Zuc76] for a review on this topic). However, very few of these approaches can comfortably satisfy us in practice, even within one particular application. A familiar but awkward difficulty to do with this issue is the decision as to which of the existing methodologies should be chosen when one is faced with a new problem. In many practical situations, this can really present a serious bottleneck in solving an overall image understanding problem. We observe that the fundamental reasons behind such difficulties are caused by the following two aspects:
1. A confident evaluation of a segmentation result still relies principally on human visual inspection without a reliable criterion which can be expressed numerically.

2. A practical segmentation approach is always (whether explicitly or implicitly) governed by an underlying symbolically well-defined or essentially numerical criterion simply because a digital computer is a logically precise and principally serial manipulator.

To make the above aspects clear, we start with defining an image segmentation in a formal way. Mathematically speaking, defining an entity basically specifies the generation and evaluation scheme of the involved entity at the same time.

In plain terms, image segmentation is meant to divide the image domain into meaningful and non-overlapping regions. For computational reasons, the following definition is very often adopted either explicitly or implicitly:

**Definition 5.1** Low-Level Image Segmentation (Region-Based)

A low-level image segmentation is a partition \( \{ R_1, R_2, \ldots, R_n \} \) of the full image domain \( X \) based on a prescribed uniformity predicate \( U \) such that the following conditions are all met:

a) \( X = \bigcup_{i=1}^{n} R_i \).

b) \( R_i \cap R_j = \emptyset \) for any \( i \neq j \) from \( \{ 1, 2, \ldots, n \} \).

c) \( R_i \) is connected for \( i = 1, 2, \ldots, n \).

d) \( U(R_i) = true \) for \( i = 1, 2, \ldots, n \).

e) \( U(\bigcup_{i \in V} R_i) = false \) for each subset \( V \) of \( \{ 1, 2, \ldots, n \} \) such that \( R_i \)'s with \( i \in V \) are mutually adjacent and \( |V| > 1 \).

In the above definition, the meaning of connectivity depends on the particular application. It can mean the usual 4-connectivity or 8-connectivity (see P.335 in [Ros76a] for these meanings). The **uniformity predicate** is roughly speaking a logical attributor which assigns a *true* value to a region if it is uniform in some sense and a *false* value otherwise. In a presumably strict sense, however, there appear to be some natural constraints which should be formally obeyed by a uniformity predicate. In particular, a uniformity predicate \( U \) should comply to the following constraint for any particular subset \( Y \) of connected pixels:
5.3 Combined Segmentation Approach

\[ U(Y) = \text{true} \implies U(Z) = \text{true} \quad \forall \ Z \subseteq Y \quad (5.1) \]

We call such a uniformity predicate a formal uniformity predicate. For a strict definition of a formal uniformity predicate we refer to Page 68 in [Pav77].

The condition in Eq. 5.1 appears to be quite natural and Definition 5.1 under a formal uniformity predicate appears to be precise and naturally covers our intuitive wishes in connection with an image segmentation. Unfortunately, it is hard to find a practically useful uniformity predicate which satisfies this condition. Moreover, a careful study will disclose some serious practical limitations caused by such a straightforward definition. Noting that the sequential or logical nature is fundamental to any man-made machine, we realize that the machine can not measure the uniformity within a region before it has actually captured the region. Bearing this consideration in mind, Definition 5.1 is thus to imply that we can find a uniformity predicate, which is capable not only of positively characterizing a true region but also of faithfully denying any wrongly composed region. Computationally, this further implies that the machine should try all possible regional formations in order to locate the true regions, assuming that such a powerful uniformity predicate does exist for a particular application. Practical experience has taught us that such a uniformity predicate can scarcely be found except in some trivial situations (see for instance an illustrative example on Page 69 in [Pav77]). In practice, this definition will thus inevitably lead to various kinds of suboptimal or simply erroneous segmentation results.

To avoid the fundamental drawbacks of the region-based segmentation approach, the edge-based segmentation approach has been parallelly developed and employed by basically adopting the following segmentation definition:

**Definition 5.2** Low-Level Image Segmentation (Boundary-Based)

A low-level image segmentation is a partition of the full image domain \( X \) by a boundary mask \( M \) based on a prescribed transition predicate \( T \) such that the following conditions are all met:

a) \( M \) is a subset of \( \bar{X} \), where \( \bar{X} \) is the union of \( X \) and its exterior boundary.

b) The exterior boundary of \( X \) belongs to \( M \).

c) \( T(x)=\text{true} \) for all \( x \) from \( M \).

d) \( \hat{M} \) is closed without ending points\(^1\) where \( \hat{M} \) is a skeleton of \( M \).
In the above definition, the *transition predicate* is also a logical attrib-
utor but it assigns a *true*-value to a pixel if it is in the transitional area
between adjacent regions and otherwise a *false*-value is assigned.

The effect of Definition 5.2 is basically complementary to that of Defi-
nition 5.1. Clearly, Definition 5.2 will cause the machine to suffer less from
its fundamentally sequential nature since the transition predicate here is
assumed to work only on the individual pixel-based locations and can there-
fore be applied in a parallel manner to all individual locations. However,
our difficulty in image segmentation has hereby in fact been transferred to
finding a correct transition predicate, which should yield a segmentation
result satisfying all the requirements stated in Definition 5.2 and in par-
ticular, requirement d) there. Again, practical experience has taught us
that these requirements are difficult to be simultaneously satisfied, par-
ticularly for noisy images where regional boundaries are not everywhere
locally evident.

We observe the following (dis)advantages of the region-based segmen-
tation approach and the edge-based approach, respectively:

**Region-Based Approaches:**
1. Direct extraction of regions.
2. Insensitivity to locally noisy regional transitions.
3. Unfortunate necessity for regional initialization.
4. Dynamic variation of the strengths of regional transitions often leads
to false regions by overmerging and/or oversplitting.

**Edge-Based Approaches:**
1. Accurate localization of locally evident regional transitions.
2. Insensitivity to region’s geometrical shapes.
3. Confusion under locally noisy pixels.
4. No guarantee for correctly closed regional boundaries.

In image segmentation research, there have already been some ap-
proaches which attempt to combine the above categories of segmentation
approaches in an attempt to utilize their individual merits jointly while

---

1 The condition of $\tilde{M}$ being *closed without ending points* can be defined as to say that
the removal of any pixel from $\tilde{M}$ is to cause some topological change in $X \setminus \tilde{M}$. 
avoiding their individual drawbacks (see [Yok88,Sta86]). Here, we discuss this issue in a formal way.

As we mentioned previously, image segmentation is a process in which the image domain is divided into meaningful and non-overlapping regions. Computationally speaking, we are allowed to declare that directly extracting pixels belonging to individual regions is equivalent to indirectly locating all regional boundaries as far as our segmentation purpose is concerned. Then, why are we sometimes forced simultaneously to employ both methodologies in a cooperative way in order to achieve a satisfactory result?

When a human perceiver describes his own perception process in a particular case, the terms of region and edge are almost always used at low levels even as their precise descriptions are not clear to the perceiver himself. In the study of primitive human visual capability, the notion of the just noticeable difference (see for instance [Ros76a], P.46) discloses the fact that the human visual system is only sensitive to changes of luminance (i.e., discontinuity of luminance). In our view, that disclosure can also be used to derive a complementary fact, namely, the human visual system is able to detect non-changes of luminance (i.e., uniformity of luminance). Thinking in a similar way, we observe that the human perceptual system at low levels simultaneously possesses the following two basic features in such a manner that either one of the two features in effect implies the other one.

**Two Basic Low-Level Perceptual Features**

1. The human perceptual system is sensitive to changes in existing visual patterns.

2. Complementarily, the human perceptual system is sensitive to non-changes in existing visual patterns.

In our view, the functioning of both these features is basically responsible for the fact that the terms edge and region are employed by a human perceiver. Although these two features are functionally complementary to each other, we can not simply state that they can be replaced by each other in practical situations. We explain this in the following way:

1. Under the first perceptual feature, a regional boundary is observed which is based on some changes in pictorial patterns across the local transitions of the resulting regions. Thus, a region is detected without
emphasized on its interior visual characteristics.

2. Under the second perceptual feature, a region is observed based on non-
changes in pictorial patterns within the region. Therefore, all visual
characteristics within a region are responsible for the detection of the
corresponding region.

The above two conclusions are clearly contradictory to each other. We
therefore have no other alternative but to acknowledge the following:

A region is detected by the human perceptual system if non-
changes in pictorial patterns within the region have been ob-
served and/or changes of pictorial patterns across its boundary
have been observed.

Following this conclusion, we have come to a new definition for a low-
level image segmentation as given in the next.

Definition 5.3 Low-Level Image Segmentation (Combined-Based)

A low-level image segmentation is a partition \( \{R_1, R_2, \ldots, R_n\} \) of the full
image domain \( X \) based on a prescribed uniformity predicate \( U \) and a pre-
scribed supplementary transition predicate \( ST \) such that the following
conditions are all met:

a) \( X = \bigcup_{i=1}^{n} R_i \).
b) \( R_i \cap R_j = \emptyset \) for any \( i \neq j \) from \( \{1, 2, \ldots, n\} \).
c) \( R_i \) is connected for \( i = 1, 2, \ldots, n \).
d) \( U(R_i) = \text{true} \) for \( i = 1, 2, \ldots, n \).
e) \( U(\bigcup_{i \in V} R_i) = \text{false} \) for each subset \( V \) of \( \{1, 2, \ldots, n\} \) such that \( R_i \)'s
   with \( i \in V \) are mutually adjacent and \( |V| > 1 \).
f) \( ST(\tilde{R}_i) = \text{true} \) for \( i = 1, 2, \ldots, n \) with \( \tilde{R}_i \) denoting the boundary of \( R_i \).

Some remarks on the above definition should be made:

1. According to our previous discussion on low-level perceptual features,
   we do not think a uniformity predicate should be applicable to a tiny
   set of pixels or especially to a single pixel. In other words, it is mean-
   ingless to talk about the uniformity of such a tiny set of pixels. Thus,
   requirement d) above implies that a region should at least contain one
   primitive pictorial pattern. For a noisy image, a single pixel will there-
   fore not in general be considered as uniform. In this sense, the condition
   in Eq. 5.1 should be modified into the following one:
\[ U(Y) = \text{true} \implies U(Z) = \text{true} \quad \forall \ Z \subseteq Y \quad (5.2) \]

where \( Y \) is an arbitrary set of connected pixels and \( Z \) contains at least one primitive pictorial pattern. In fact, this is the condition which is usually followed in practice rather than the excessively formal and practically useless requirement in Eq. 5.1.

2. A supplementary transition predicate \( ST \) is a logical attributor based on a transition predicate \( T \) as \( ST(\tilde{R}) = F(P_T, \tilde{R}) \), where \( P_T = \{ x \in \tilde{R} : T(x) = \text{true} \} \). Thus, \( ST \) is a logical function of \( P_T \) and \( \tilde{R} \). To give an example, we may precisely define a particular \( ST \) as follows:

\[
ST(\tilde{R}) = \begin{cases} 
    \text{true} & \text{if } |P_T|/\tilde{R} \geq \varepsilon \\
    \text{false} & \text{otherwise}
\end{cases} \quad (5.3)
\]

where \( \varepsilon \) is a positive constant.

Our new definition for low-level image segmentation is not meant to dismiss previously existing definitions. We only claim that this new definition is more complete and is more explicitly conform to the human perceptual system. In practice, there are situations in which strictly accurate boundaries for the existing regions are not necessary or even possible (e.g., in the case of large transitional areas) while in other cases a uniformity predicate can be devised, which can be logically reversed to take over the functioning of the transition predicate in detecting the regional boundaries. In these cases, Definition 5.1 will clearly be sufficiently satisfactory.

In analyzing our SLAR images during the past years, we have in fact carried out our research basically in confirmation with Definition 5.1 for image segmentation. Though the obtained results were encouraging, we observe that all of the employed uniformity predicates functioned insufficiently well in detecting the accurate regional boundaries. Without sufficient prior knowledge, it is hardly possible to obtain a reliable segmentation result without significantly deviated region boundaries and/or erroneously merged regions. Our current methodology will be based on our new definition for image segmentation and we expect that we may hereby obtain significantly better results.

To summarize, we note that our task for the low-level image segmentation on SLAR images will have to focus on the choice for a uniformity predicate, a transition predicate as well as on the cooperative functioning among the two predicates. Our new segmentation definition will thus serve as the global optimization criterium for the segmentation process. In doing this, we propose an \textit{Edge-Constrained Region-Growing Based Segmentation} approach. The main strategy is to guide a region-based segmentation ap-
proach by means of a sufficiently reliable (though not necessarily complete) boundary map, which is previously obtained through a suitable transition predicate. In this way, we hope that the resulting segmentation may approximately satisfy Definition 5.3 and therefore we may expect better results from our low-level image-segmentation process. Schematically, our new segmentation strategy is explained in an extended context hierarchy of our SLAR-image-analysis problem as shown in Figure 5.3, where

![Diagram](image)

Figure 5.3: An extended context hierarchy for SLAR-image interpretation.

the data class *edge* corresponds to regional boundaries detected under the transition predicate and the data class *region* corresponds to primitive regions detected by our uniformity predicate under the obtained boundary constraints as required by our new segmentation definition. Chapter 6 focusses on the design and implementation of the transition predicate, where we propose a new and quite effective transition predicate called *Edgeness Detector*. In Chapter 7, we actually come to the presentation of the employed uniformity predicate and, in particular, the *Edge-Constrained Region-Growing Based Segmentation* approach which combines the two predicates in satisfaction of Definition 5.3. In Chapter 8, we will come to the issue of how to make use of the **DADS** framework for establishing some (either implicitly or explicitly) dynamic interactions between the higher-level and low-level processing in order to further improve the low-level segmentation result towards the final interpretation aim.
Chapter 6

Edgeness Detector: A New Method for Extracting Regional Boundaries

6.1 Introductory Review of the Edge-Detection Problem

In this chapter, our main purpose is to extract the regional boundaries in a given digital image and, in particular, those arising in an SLAR image. In line with our previous discussions, it is the purpose here to find a sufficiently reliable transition predicate and establish a practical implementation. In the DADS terminology, this corresponds to the construction of an effective processing cluster, which accepts an input image and generates the possible edge pixels within the input image.

Basically, we encounter here a traditional edge-detection problem, which has been investigated extensively both theoretically and experimentally by many researchers in the field of image processing and pattern recognition. Many techniques for this purpose have already been developed in the past (see [Mar80,Hue71,Tor86,Suk84]). However, it is not simply a matter of selecting which of the existing methods is particularly suitable for SLAR images. Most of the existing edge-detection methods can work satisfactorily only if the edges happen to possess certain underlying characteristics while others depend heavily on the availability of some prior knowledge about the possible edge presence. In our attempt towards extracting regional boundaries in a given digital image, we propose a new
edge-detection scheme, which we call the *Edgeness Detector*. Throughout our experiments with SLAR images, this new edge-detection scheme proves to be quite satisfactory. Our main emphasis in this chapter is thus on discussing various aspects of this new edge-detection scheme. However, before we actually present our new *Edgeness Detector*, the existing edge-detection methods will be discussed in general.

Among the currently available edge-detection techniques, a significant number are essentially based on the notion of *gradient* and therefore inevitably suffer from the following two conflicting observations:

1. An edge pixel generally has an outspoken response to a gradient-like operator.
2. A noisy non-edge pixel may quite often also have an outspoken response to a gradient-like operator. In other words, an observed outspoken response to a gradient-like operator does not necessarily originate from an edge pixel.

In the past, a tremendous effort was made to resolve these two conflicting observations in the hope that a gradient-like operator would become sufficiently reliable. One of the common methods of doing this is the application of a particular low-pass filter to reduce the noise before a gradient-like operator for edge detection is actually applied. However, blindly smoothing a raw input image may often cause the existing edges to be dispersed undesirably over in a wider area.

Among the earliest edge-detection techniques are those which basically only focus on designing a suitable digital version of the continuous gradient operator. Examples are, for instance, the Roberts Cross operator and the Sobel operator (see for instance [Ros76a]). With reference to the two previously mentioned observations, the shortcoming of such a purely gradient-based operator was soon widely acknowledged and this has in turn led to other approaches, which concentrate on removing the image noise while preserving the actual edge presence. Examples here are, for instance, the Kuwahara filter (see [Kuw76]) or the Nagao-Matsuyama filter, which is more general but similar in principle and in design motivation (see [Nag80]).

Still, such a widely used traditional paradigm of smoothing followed by gradient enhancement does not satisfy our practical aim of edge detection for extracting regional transitions in many practical circumstances. One of the main obstacles under this paradigm is that the resulting edge pixels
generally fail to form some closed edges as required for representing regional boundaries.

In 1980, Marr and Hildreth proposed a new method of edge detection, which is based on a Gaussian filter followed by a Laplacian filter (see [Mar80]). The function of the Gaussian filter is to smooth the image noise while keeping the intensity changes at a certain spatial scale intact and the Laplacian operator subsequently acts as an enhancement operator to disclose the local gradient maxima, which are in turn assumed to be the potential edge pixel locations. Through their heuristic arguments, Marr and Hildreth showed that their approach can optimally minimize both the local spatial response and the range of spatial frequencies. By subsequently combining the zero crossings of the resultant output multichannel edge-enhanced images, they claimed that the true edges can always be found and are guaranteed to be closed. In this way, they claimed to have found the ‘optimal’ edge detector. Their arguments were mainly based on some perceptual consideration of the edge phenomena rather than on a rigorous mathematical edge modelling and can be summarized as follows.

On the one hand, edges as intensity changes can occur over a wide range of scales or, in other words, spatial frequencies. Taking ideal ramp edges as an example, this means that there can be edges of different ramp widths in the image. In such a case, it will be impossible for a single band-limited filter simultaneously to keep all these differently scaled edges intact while removing others (note that a noisy environment is essentially also a situation with intensity changes). One way to deal with this kind of situation is separately to consider edges at different scales (i.e., different spatial frequencies). In order to emphasize only edges corresponding to a certain edge resolution, the filter should thus have a restricted bandwidth in the spatial frequency domain. This may be expressed by requiring that its variance in the spatial frequency domain, i.e., $\Delta \omega$, should be small. On the other hand, edges as intensity changes are spatially localized (remember edges are considered here as local phenomena) rather than extended or wavelike at their individual spatial scales. To locate the edge positions accurately the filter’s spatial influence must therefore be concentrated about those positions and in particular it should possess a minimal standard deviation $\Delta z$ in the spatial domain. These two conflicting requirements on $\Delta \omega$ and $\Delta z$ respectively can be brought down to a single one, i.e., minimizing the product $\Delta \omega \Delta z$. As shown by Leipnik (see [Lei60]), only a filter with a Gaussian envelope minimizes this product.

Besides, edges as intensity changes at a particular spatial scale can
be seen as clusters of local maxima of some gradient magnitude or zero crossings of the second derivatives for the corresponding spatial scale. According to the (essentially heuristic) arguments of Marr and Hildreth, one can effectively detect all edges in the image in an optimal way by combining zero crossings obtained from a set of Gaussian-like filters corresponding to various spatial frequency bands (see the spatial coincidence assumption of Marr and Hildreth in [Mar80]).

In fact, the Laplacian in the Marr-Hildreth edge-detection scheme can be replaced by the gradient operator (thus, instead of detecting zero crossings we should then detect local maxima of some gradient magnitude). However, the Laplacian has some advantages over the gradient operator. Firstly, it is orientation independent. Secondly, and perhaps more importantly, the zero crossings in a Laplacian filtered image (similar to local maxima of a gradient enhanced image) will automatically form closed curves for the edge phenomena totally enclosed by the image if the locally linear variation condition of Marr-Hildreth is satisfied.

In practice, the application of the Marr-Hildreth edge-detection scheme means the necessity of employing more than one Laplacian-Gaussian filter corresponding to various spatial frequency bands. This will often pose a heavy computational burden, though otherwise the ‘optimality’ may lost. However, in cases where edges all occur approximately at one particular frequency band, it is possible to achieve the Marr-Hildreth’s optimality with only one Laplacian-Gaussian filter provided that the variance of the Gaussian can be chosen to correspond correctly to the scale of the existing edges. Still then, the tasks of selecting a suitable standard deviation of the Gaussian and sampling the continuous Laplacian-Gaussian filter into an approximating digital version are not trivial. Subsequently, the expected performance as argued by Marr and Hildreth may deteriorate. In an attempt to make the Marr-Hildreth filter more practical, Lunscher and Beddoes studied the design of such a filter by assuming some appropriate prior knowledge about the edge features under investigation (see [Lun86a,Lun86b]). Also, van Vliet, Young and Beckers have proposed another related approach, in which a digital nonlinear ‘Laplacian’ operator is used instead of a usual digital version of the continuous Laplacian (see [Vli89]). Their nonlinear ‘Laplacian’ has a similar behaviour as a filter with second-order derivatives while producing only one output value as the usual Laplacian does. However, its advantage is that it adapts its direction of computing second-order ‘differentials’ to the most relevant direction, i.e., perpendicular to the local edge direction. In this way, it can
be expected that their 'Laplacian' is to produce more reliable output than
a usual digital Laplacian. Moreover, it permits the development of ap-
approximately isotropic digital filters. The significant achievement of their
approach is the practical applicability even under conditions of a very low
signal-to-noise ratio.

It is not our purpose here to review all currently existing edge-detection
approaches. In fact, we only want to cover purposefully the essence of
some basic approaches, where high-level knowledge about the image edges
is not assumed to be known. To conclude, we would like to mention two
other edge-detection techniques, one is the best-fit method by Hueckel
(see [Hue71,Hue73]) and the other is the texture-edge-detection scheme
presented by Rosenfeld and Kak (see [Ros76a], P.290-296). Compared to
the previously mentioned approaches, the basic idea behind both of these
two approaches is to view an edge pixel as among the interior boundary of
a local structuring element for two adjacent regions. Note that this idea
is essentially different from the point-concentrated nature induced by the
notion of gradient.

In Hueckel's method, a set of prescribed ideal edge models are estab-
lished in advance by means of a uniform parametrical representation. At
each given image pixel, the edge model corresponding to the best match
based on the metric of the chosen Hilbert function space is considered as
the potential local edge model for the pixel in question. Subsequently,
the best-matched edge model is actually confirmed if it passes a chosen
correlation test, which uses only the information within the original local
image data and the best-matched edge model, in particular the grey-value
step size in the model. Thus, the edge strength of an actual edge pixel
will be based essentially on its local grey-value step size. In the Rosenfeld-
Kak scheme, a pixel gets its edge strength based on the maximum of the
differences of some average local property between every possible pair of
neighbourhoods, which are adjacent to each other at the involved pixel.

For extracting regional boundaries in SLAR images, we propose a new
method for edge detection. Part of the idea originates from the above two
approaches. In our experiments, it has shown good results based on some
qualitative evaluations. However, we have not yet made any attempt to
compare thoroughly our new edge-detection scheme with other existing
techniques. Nevertheless, for certain practical applications we are very
much confident in expecting that our new edge-detection scheme will be
more effective. In the next section, we formally introduce this new method
in full detail.
6.2 Edgeness: a New Concept for Edge Definition

We start with the definition of an edge pixel. Intuitively, we would expect that any pixel will be considered or assigned either as an edge pixel or as a non-edge pixel. Usually, this can not be achieved at once. Some intermediate stage is generally required, which assigns a temporary value to each pixel expressing the confidence that the pixel is a true edge pixel. A typical example is the magnitude of some gradient-like operator, which may be used to represent such confidence for each pixel. We call this confidence value the edgeness of the pixel. The underlying definition of pixel edgeness for many existing edge-detection techniques does not explicitly involve the notion of edgeness in terms of, for instance, the probability mechanism. From the discussions in Section 6.1, we observe three main types of edgeness definitions among the currently available edge-detection methods. Below, we individually discuss them in relation to their practical implications.

1. Gradient based edgeness.

Here, the edgeness is roughly defined to be some gradient based value (e.g., the usual gradient magnitude) of the pixel obtained through a suitably chosen digital version of the continuous gradient operator. The main reason behind such a motive is quite understandable when noting the following:

a) The notion of continuous gradient has been mathematically well defined and well understood for a very long time.

b) Our visual impression of an edge presence (e.g., through a photograph) resembles our visual impression of a part with high gradient magnitudes of a continuous 2D function displayed on a digital screen.

We note, however, that one should also be aware of the incompleteness of the above two remarks, which can lead to severe consequences in practical circumstances. This incompleteness is represented by the following two observations:

a) The gradient such as in the continuous sense is basically undefined on a discrete grid.

b) A purely noise-caused local non-edge environment may quite often also give rise to a significant value of the gradient magnitude.

Moreover, a digital version of the gradient operator usually assigns
a small value of edgeness to an actual edge pixel at a low intensity change compared to an actual edge pixel at a high intensity change. This uneven edgeness assignment to an edge pixel basically conflicts with our intuitive meaning of edgeness. When we perceive two edges corresponding to two different intensity changes, we will generally view the pixels from both edges as equally legitimate edge pixels. This should be reflected in the confidence measure.

Should a global thresholding be subsequently applied to extract edge pixels from a previously gradient-filtered image, an unavoidable consequence is then that the employed threshold should be chosen on the basis of the smallest intensity change at the true edges. Otherwise the pixels at the existing edges corresponding to small intensity changes will not be detected at all. Such a requirement may not really be an obstacle if the image is, for instance, a result of a binary image corrupted by some noise, which is relatively low with respect to the intensity change in the original binary image (note that in such cases the intensity changes corresponding to the true edges are roughly the same). However, for images like our SLAR images this consequent requirement can be awkward. In SLAR images, there are intensity changes among neighbouring regions such that these intensity changes vary dynamically over the image domain. Furthermore, the intensity changes are in the same order as the noise level.

2. Zero-crossing based edgeness.

Here, edgeness is defined to correspond to the zero crossings in the Laplacian-filtered image. Basically, the idea here is similar to that in the gradient based scheme since zero crossings correspond in fact to local gradient maxima. The advantage here is, however, supposed to be that it suffers virtually no negative effect by the dynamic variation of intensity changes at the existing edges. Nevertheless, this method will only then work in a reliable way if we can be sure that the noise in the image is not going to cause any undesired zero crossings. This is in fact a hard requirement for many practical applications. Note that a noise-corrupted non-edge environment may locally turn out to be a visually misleading edge environment by, for instance, some accidental micro-edges. The Marr-Hildreth edge-detection approach provides an ‘optimal’ scheme in an attempt to satisfy this requirement. In practice, a discrete version of the Laplacian-Gaussian filter will in general still yield spurious edges. Very often, a gradient-based cleaning operation is needed to remove the artifact zero crossings (see for instance [Vli89]).
3. Matching based edgeness.

Here, the edgeness for a pixel is defined to be proportional to some measurement on the best match out of a set of prescribed edge models. These models can be prepared either explicitly (e.g., those in Hueckel’s method) or implicitly (e.g., the texture-edge-detection scheme by Rosenfeld and Kak). Intuitively, this method should suffer less from the digital approximation and the image noise since usually only near-iconic edge models are used and the noise is smoothed out largely by the averaging mechanism within the near-iconic models. However, detailed analysis leads to the following observations:

a) At an edge location, the edgeness is still assigned in some way proportional to the intensity change around the local edge segment, although not just proportional to the intensity change around the involved pixel such as with a gradient based edgeness. In the Rosenfeld-Kak scheme, the edgeness is supposed to correspond to the difference of some average local property under the best-matched edge model and this difference is in its turn in some way (depending on the correlation scheme in the matching) proportional to the intensity change of the local edge segment. In the Hueckel approach, the edgeness corresponds to the result of the chosen correlation test. It is easy to derive that the larger the grey-value step size of the best-matched edge model is, the higher the probability is that the edge pixel will pass the correlation test (see also [Hue71], P.120-121). Thus, the edgeness is in principle also in some way proportional to the intensity change of the local edge segment.

b) At a noisy non-edge location there is no local edge segment at all, so none of the edge models is in fact meaningful for the matching process. In a forced attempt to match the existing edge models at such a location, the obtained result is of course quite absurd when one of edge models turns out to be the ‘best match’. Moreover, it is our view that such a ‘best match’ essentially yields the worst measurement on the non-edge fact with respect to the set of all employed edge models.

Similar to the case with gradient based techniques, when a global thresholding is applied to extract edge pixels reliably, the threshold should be chosen so that it is both not too high for the best match at true edge segments with the smallest intensity change and yet too high for the best matches caused by some noisy non-edge pixels. Under heavy noise conditions relative to the smallest intensity change at true
edges, such an optimal threshold may even not exist. This consequence is obvious for the Rosenfeld-Kak scheme. For the Hueckel approach, however, to extract edge pixels reliably while avoiding non-edge pixels, it is necessary that the correlation test should be designed to block the worst measurement on the noisy non-edge locations while accepting the best measurement on the edge locations corresponding to the smallest intensity change. The consequent difficulty in designing this test is similar to the other approaches in this category.

Summarizing the above we propose to define the notion of edgeness under the following guidelines:

1. The edgeness of a pixel should be related to the presence of a locally systematic intensity change along a line, which intersects the involved pixel.
2. The edgeness of a pixel should not be exclusively related to the local intensity change of a potential underlying edge-segment, which intersects the involved pixel.
3. Noisy non-edge pixels should be suppressed if no locally systematic intensity changes are observed.

Before we formally define the notion of an edge pixel or that of edgeness based on the above guidelines, we introduce some necessary concepts. In defining those concepts below, we do not explicitly distinguish between entities in continuous space and their counterparts on a discrete grid. Thus, a point may just be a point in the continuous space or be a pixel on a discrete grid.

**Definition 6.1** Familiarity of Line Segments

A pair of line segments are said to be familiar with each other if they are spatially close to each other in a sense that they tend to coincide with each other.

**Definition 6.2** Family of Line Segments

A family of line segments is a set of line segments such that each of the line segments is familiar with every other one in the set and they all cross at a common point.
These definitions are illustrated in Figure 6.1 for straight line segments, where the set of line segments in Figure 6.1a may be considered as one family while the line segments in Figure 6.1b do not constitute a single family of line segments.

Figure 6.1: Examples for illustrating the familiarity among straight line segments.

**Definition 6.3** Average Grey-value Property

An average grey-value property is an overall property of a given portion of the image domain obtained by a total measurement of all grey values within the portion of domain.

Examples of an average grey-value property are, for instance, the usual grey-value mean and the mean value of some textural measurements such as the grey-level difference mean (GLD-mean) measure.

**Definition 6.4** Intersecting Lines and Generating Point

Given a point, an intersecting line is a line segment which goes through the point. The point is in this case called a generating point to the line segment.

**Definition 6.5** Pair of Neighbourhoods

Given a generating point and an intersecting line, a corresponding pair of neighbourhoods is a pair of small neighbourhoods local to the generating point so that one of the neighbourhoods lies along one side of the intersecting line and the other along the opposite side.
Definition 6.6 Neighbourhood Difference

Given a pair of neighbourhoods, the neighbourhood difference is the difference between the value of some chosen average grey-value property within one of the neighbourhoods and that within the other neighbourhood.

Figure 6.2 shows an illustrative example with a generating point $p$, an intersecting line $L$ and a corresponding pair of neighbourhoods $(N_1, N_2)$. Supposing that the grey-value mean is considered as the measure for the average grey-value property, then $|M_1 - M_2|$ will be the neighbourhood difference corresponding to $p$ and $L$ where $M_1$ and $M_2$ are the grey-value means within neighbourhood $N_1$ and neighbourhood $N_2$, respectively.

Figure 6.2: An illustrative example: $p$ is the generating point; $L$ is an intersecting line; $N_1$ and $N_2$ together form a corresponding pair of neighbourhoods.

Definition 6.7a Circular Transformation: Continuous Case

Let $F$ be an arbitrary function on a real interval $[a, b]$ with $F(a) = F(b)$ and let $t$ be an arbitrary point from $[a, b]$. The circular transformation $E$ of $F$ at the point $t$ is a function on $[a, b]$ defined as follows:

$$E(x) = \begin{cases} 
F(x + t - a) & x \in [a, a + b - t) \\
F(x + t - b) & x \in [a + b - t, b]
\end{cases}$$

(6.1)
**Definition 6.7b** Circular Transformation: Discrete Case

Let $F$ be an arbitrary discrete function on an integer interval $[1, n]$ and let $t$ be an arbitrary integer from $[1, n]$. The circular transformation $E$ of $F$ at the point $t$ is a discrete function on the integer interval $[1, n+1]$ defined as follows:

$$E(i) = \begin{cases} 
F(i-1+t) & i \in [1, n-t+1] \\
F(i-1+t-n) & i \in [n-t+2, n+1]
\end{cases}$$  \hspace{1cm} (6.2)

Figure 6.3 shows two examples of the circular transformation, one for the continuous case and the other for the discrete case.

![Diagram](image1)

Figure 6.3: Examples of circular transformation: a) continuous case; b) a discrete case.)

Now, conform to our intuitive feeling of an edge appearance, we formally define an *edgeness point* (or an edge point) as follows:
Definition 6.8  Edgeness Point

An edgeness point is a point through which we can locally draw one and only one family of intersecting lines such that the neighbourhood differences corresponding to the lines in the family are significantly larger than the neighbourhood differences corresponding to other possible intersecting lines.

In a more direct manner, Definition 6.8 suggests that the meaning of edgeness is coupled to the meaning of some average grey-value property. Or more precisely, at an edgeness point there is always an intersecting line such that the corresponding neighbourhood difference will be significantly large compared to the neighbourhood differences corresponding to other spatially quite distinct intersecting lines, i.e., non-familiar intersecting lines. In other words, the neighbourhood differences corresponding to intersecting lines close to the local edge segment are to be significantly large with respect to those corresponding to other possible intersecting lines. In Figure 6.4 we give two illustrative examples of Definition 6.8 for an edge point and a non-edge point, respectively.

(a) A noisy edge point.  (b) A noisy non-edge point.

Figure 6.4: Illustrative examples of Definition 6.8: a) only one family of intersecting lines have near-maximal neighbourhood differences, thus an edge point is implied; b) several families of intersecting lines achieve near-maximal neighbourhood differences, thus a non-edge point is observed.

The essence of this new definition for edgeness may be clarified as follows:
1. Noisy non-edge points will hardly ever satisfy Definition 6.8. Let us consider a non-edge point in a noisy environment. According to Definition 6.8, we measure the neighbourhood differences corresponding to all possible intersecting lines under some prescribed average grey-value property. Let us assume that the involved local noise is sufficiently random though its influence may be quite significant. Then, we will usually obtain a set of near-maximal neighbourhood differences corresponding to several intersecting lines, which are mutually quite distinct in the spatial domain (such as in Figure 6.4b), unless the noise effect is by chance so locally systematic that a significant local edge segment has been effectuated. Clearly, our new definition is thus able to suppress noisy non-edge points in an effective way since a locally systematic noise effect (i.e., a visible edge segment caused purely by the local noise effect) rarely occurs in practice. If such a phenomenon does occur, it should be dealt with at higher levels of processing and not at a low-level edge-detection stage. For a low-level edge-detection scheme, the task should be to detect all edges being 'seen', even when some of them are by chance caused by some locally systematic behaviour of the noise.

2. Accurate edge locations.
Since in Definition 6.8 the noise effect is greatly suppressed by computing values of an average grey-value property, it is not necessary first to apply one of the traditional smoothing operations for cleaning the noisy points. Generally speaking, a separate smoothing operation causes the original edge locations to be deviated or widened. In Definition 6.8, such an effect is not explicitly present due to the fact that for each point we actually apply a set of averaging (or smoothing) operations corresponding to different intersecting lines before the neighbourhood differences are to be compared. Thus, the final decision on a particular point depends on all results from a set of different averaging operations. In a word, our new definition implies a smoothing effect but not explicitly a smoothing operation in the usual sense and therefore the undesired distortions around existing edges under a traditional smoothing operation should have no reason to occur with Definition 6.8.

A unique characteristic of our definition of an edge pixel is that it is based on the mutual relationships between measurements on a prescribed characteristic feature (i.e., the neighbourhood differences in our case) with respect to all possible local edge models. Traditional low-level
edge-detection approaches do not consider such an 'inter-model' behaviour. Instead, they let the consideration solely rely on the measured characteristic value for the most prominent edge model. For instance, the Hueckel approach uses an almost complete set of local edge models as we do, but it does not consider the 'inter-model' behaviour. In the terminology of Definition 6.8, the Hueckel approach will accept a pixel as an edge pixel when the maximal neighbourhood difference is large enough, regardless of whether this maximal neighbourhood difference may be obtained or approximated by more than one family of intersecting lines. This is clearly a wrong decision and fortunately our new definition explicitly prevents such a wrong decision.

Theoretically, our new definition for edgeness itself already expresses the essence of how to devise a corresponding edge detector. However, since we ultimately have to work on a discrete grid, terms like line and neighbourhood appearing in Definition 6.8 should all be translated into some corresponding entities in the discrete domain and such a translation is not trivial. In practice, the main implementational tasks will be as follows:

1. For each possible intersecting line, a corresponding filter (or filtering scheme) should be devised, which measures the neighbourhood difference corresponding to the intersecting line. The design of such a filter includes the design of a pair of neighbourhoods and that of a numerical computational scheme for measuring the chosen average grey-value property within either one of the neighbourhoods.
2. Given the designed filters for each intersecting line, an effective numerical scheme should be designed to detect those situations where all of the maximal filter responses (i.e., neighbourhood differences) are originating from filters corresponding to intersecting lines from one and only one family. Those situations correspond to a true edge presence at the involved location.

Up to now we have, in fact, introduced and discussed our new definition for an edgeness point only on some heuristic grounds rather than on the basis of some rigorous mathematical models for edge phenomena. In the following, we are going further to explore the meaning of Definition 6.8 by conducting a more rigorous study, in an attempt to realize a corresponding edge detector in the continuous space. Because of the accessibility to a thorough mathematical treatment provided by the continuous
space, the discussion in the next section will lead to a much more precise explanation of Definition 6.8. Clearly, this is very much to be desired for any practical application on a digital computer. In the end, the design of the final digital version will follow the main strategy discovered there and be required to behave as closely as possible to the continuous counterpart.

6.3 Case Study in the Continuous Space

In this section, our main aim is to devise an edge detector according to Definition 6.8 for the continuous case. The reason that we specifically devote an entire section to discuss the construction of a continuous edge detector based on Definition 6.8 is to develop some specific insights into and to obtain some firm and rigorous support for our new idea of edge detection.

First, we define the set of filters which measure the differences of values of some average grey-value property within two small neighbourhoods along an intersecting line. Clearly, we cannot in practice afford to consider all possible lines as their descriptions will then depend on an infinite number of parameters. We observe that most of the edges in an image are locally smooth edges except, for instance, edge crossings or corners. By disregarding these special cases we can locally approximate each edge segment by a straight edge segment. Our first restriction is thus that the meaning of intersecting line in Definition 6.8 is restricted to straight line segments (see also Definition 6.4).

Parameterization of the set of straight lines all intersecting a particular common point is easy and requires only one parameter, i.e., the angle between a line and the horizontal axis. The next problem is then the design of a pair of neighbourhoods corresponding to each intersecting line and each such neighbourhood is to be used for calculating the value of some average grey-value property within the neighbourhood and subsequently the neighbourhood differences. For each straight intersecting line, we have chosen a pair of neighbourhoods which are half-circles of an equal radius \( R \) on both sides of the line segment which together form a complete circle as shown in Figure 6.5. The filters to compute the neighbourhood differences are then defined as follows if the usual grey-value mean is considered as a good characterization of the underlying average grey-value property and is chosen for that purpose:
\[ h_\alpha(r, \theta) = \begin{cases} 
1 & \theta \in [\alpha, \alpha + \pi), r \leq R \\
-1 & \theta \in [\alpha + \pi, \alpha + 2\pi), r \leq R \\
0 & r > R 
\end{cases} \tag{6.3} \]

where \( \alpha \in [0, 2\pi] \) is the angle of the corresponding intersecting line and \( R \) is the prescribed radius of the two half-circular neighbourhoods (see also Figure 6.5).

![Diagram](6.3)

Figure 6.5: Example of a filter \( h_\alpha \) from the chosen set of rotationally equivalent filters \( \{h_\alpha : \alpha \in [0, 2\pi]\} \) with \( a = 2/\pi R^2 \).

The advantages of using this set of filters are the following:

a) The filters are rotationally equivalent. Computationally, this property will prove to be convenient.

b) The filters have an equal total weight for all involved neighbourhoods, within which the average grey-value property (i.e., the grey-value mean in this case) is measured. Such an isotropical nature will directly provide us with a solid base subsequently to compare the responses for determining the maximum or more precisely, the near-maxima\(^1\), as implied in Definition 6.8.

Now, we come to the problem of how to distinguish cases where the near-maxima among the filter responses correspond to only one family of intersecting lines and those cases where the near-maxima among the

\(^1\)The use of near-maxima instead of maximum is aimed at eliminating those small fluctuations in an edge presence, which are actually insensitive even to an human perceiver.
filter responses correspond to more than one family of intersecting lines. To investigate this, we compute the responses of the chosen filters \( \{ h_\alpha : \alpha \in [0, 2\pi] \} \) with a prescribed radius \( R \) for an ideal straight step edge in the noise-free case. Without loss of generality (remember the rotational equivalency among the filters), we assume that the given edge is vertical and positioned at \( x = d \) with \( d \) being an arbitrary value. The image function \( f \) is then expressed as follows:

\[
f(x, y) = \begin{cases} 
  u + v & x \geq d \\
  u & x < d 
\end{cases}
\]  

(6.4)

where \( v \) is assumed to be non-negative.

Given an arbitrary point, we assume without loss of generality that it corresponds to the point \((0, 0)\). Then, \(|d|\) in Eq. 6.4 is the distance of the filter center to the edge. We define \( F(\alpha) \) to be the filter response of \( h_\alpha \) at the given point. According to our definition, \( F(\alpha) \) will then be determined by the following expression:

\[
F(\alpha) = \int \int f(x, y) h_\alpha(x, y) dx dy
\]

(6.5)

For various values of \( d \), we now calculate \( \{ F(\alpha) : \alpha \in [0, 2\pi] \} \).

a) \(|d| > R\).

Situations in this category are shown in Figure 6.6 and from there we can simply conclude the following:

\[
F(\alpha) = 0 \quad \alpha \in [0, 2\pi]
\]

(6.6)

b) \(|d| \in [0, R]\).

Situations in this category for either \( d < 0 \) or \( d \geq 0 \) are shown in Figure 6.7, where we have

\[
c \overset{\text{def}}{=} \arccos(|d|/R)
\]

(6.7)

The filter responses here are as follows:
6.3 Case Study in the Continuous Space

\[ F(\alpha) = \frac{2v}{\pi} \begin{cases} 
\alpha - \cos^2(\alpha) \tan(\alpha) & \alpha \in [0, c) \\
\cos(\alpha) \sin(\alpha) & \alpha \in [c, \pi - c) \\
\pi - \alpha + \cos^2(\alpha) \tan(\alpha) & \alpha \in [\pi - c, \pi] 
\end{cases} \quad (6.8) \]

\[ F(\alpha) = -F(\alpha - \pi) \quad \alpha \in (\pi, 2\pi] \quad (6.9) \]

For details of the derivation we refer to Appendix C.

Figure 6.6: Sketch of the filtering situations with \(|d| > R\).

\[ f(x, y) = u \quad x = 0 \quad f(x, y) = u + v \]

\[ h_\alpha \quad \alpha \quad d \quad h_\alpha \quad \alpha \]

\textit{horizontal line}

\[ f(x, y) = u \quad f(x, y) = u + v \quad f(x, y) = u \quad f(x, y) = u + v \]

\[ \text{horizontal line} \]

\[ (a) \quad d \in [0, R]. \quad (b) \quad d \in [-R, 0). \]

Figure 6.7: Sketch of the filtering situations with \(|d| \leq R\).
Let us now consider the filter responses \( \{ F(\alpha) : \alpha \in [0, 2\pi] \} \). What we observe from the above calculations is that for each value of \( d \) the circular transformation \( E : [0, 2\pi] \rightarrow \mathbb{R} \) of \( F \) at the central minimum point (i.e., \( \alpha = 3\pi/2 \)) of \( F \) has the following expression:

\[
E(\beta) = \begin{cases} 
-M & \beta \in [0, \pi/2 - c) \\
G(\beta + c + \pi/2) & \beta \in [\pi/2 - c, \pi/2 + c) \\
M & \beta \in [\pi/2 + c, 3\pi/2 - c) \\
G(-\beta + c + 3\pi/2) & \beta \in [3\pi/2 - c, 3\pi/2 + c) \\
-M & \beta \in [3\pi/2 + c, 2\pi] 
\end{cases}
\]

(6.10)

where the constant \( M \) is the maximum of \( F \) and the function \( G : [0, 2c] \rightarrow \mathbb{R} \) is a strictly increasing function expressed as follows:

\[
G(x) = x - c + \cos^2(c) \tan(c - x) \quad x \in [0, 2c] 
\]

(6.11)

In Figure 6.8 we have shown such a circular transformation of \( F \) for different values of \( d \), where \( w \overset{\text{def}}{=} \pi - 2c \) is the width of the top part. From there we conclude the behaviour of the circular transformation \( E \) of \( F \) at the central minimum point of \( F \) in accordance with \( d \) as follows:

1. \( w \) reaches the maximum of \( \pi \) if \( |d| \) exceeds \( R \) and in that case all filter responses become equal, i.e., zero.
2. \( w \) is in some way (though not strictly linearly) proportional to \( |d| \) if \( |d| \) does not exceed \( R \).
3. \( E \) has a unimodal behaviour as described by Eqs. 6.10–6.11 with \( G : [0, 2c] \rightarrow \mathbb{R} \) being a strictly increasing function.

In the above derivation, we have assumed a vertical ideal step edge. Since only the circular transformation of \( F \) at the central minimum point of \( F \) was used to arrive at the above conclusions and the central minimum point of \( F \) is always unique except for the case when \( F \) is everywhere zero, it is easy to see that the above conclusions will still hold if the edge is replaced by an arbitrarily oriented ideal straight step edge.

In our heuristic definition of an edgeness point (see Definition 6.8) we only mentioned the familiarity among intersecting lines corresponding to near-maximal neighbourhood differences and considered this aspect as the
characteristic feature. In the current ideal case, this may correspond to the width \( w \) depending on our particular wish concerning the familiarity among intersecting lines. Suppose that two intersecting lines are only then familiar with each other if they actually coincide with each other. Then, we encounter an edge point if and only if we obtain the maximal neighbourhood difference corresponding to precisely one intersecting line, i.e., \( w = 0 \). However, due to our special choice of the set of filters we discover another unique property owned by edge and near-edge points. This is the strictly monotonic nature of the non-extremal filter responses if considered as a function of \( \alpha \) (see also the conclusion 3 above). Edge detection based on Definition 6.8 becomes thus equivalent to detecting the above behaviour of \( F \). In other words, starting from Definition 6.8 and choosing the rotationally equivalent filters \( \{ h_\alpha : \alpha \in [0, 2\pi] \} \), in the case of an ideal straight step edge an edgeness point is defined by the following alternative definition:

**Definition 6.9** Edgeness Point for the Continuous Case

An edgeness point is a point at which the responses by the filters \( \{ h_\alpha : \)
α ∈ [0, 2π]} under a circular transformation at the central minimum point of the filter responses jointly show a unimodal behaviour with a sufficiently small width for the top part.

In reality, an edge is corrupted by noise. In the case of digital images, the noise can even be contributed by the quantization procedures. For a noisy situation, the circular transformation of an actual realization of F at the central minimum point of F may generally be quite different from that in Figure 6.8. Thanks to the averaging mechanism within the filters \{h_α : α ∈ [0, 2π]\}, we may, however, expect that it will ‘on the average’ have a similar behaviour as in the noise-free cases. Let us assume in the continuous case that the ideal straight step edge is corrupted by an additive generalized zero-mean white noise. Then according to Eq. 6.5 above, the expected values of the filter response F will show exactly the same behaviour as the original filter responses in the noise-free case. Thus, Definition 6.9 is indeed valid ‘on the average’ even under white noise corrupted situations.

Since we have already chosen the set of filters and the average grey-value property, we can define certain issues related to Definition 6.8 more precisely.

**Definition 6.10** Familiarity Measure and Familiarity of Intersecting Lines

A familiarity measure \(M_f\) is a non-negative value such that an arbitrary pair of intersecting lines \(L_1\) and \(L_2\) are said to be familiar with each other if and only if the smallest angle between \(L_1\) and \(L_2\) is not larger than \(M_f\).

**Definition 6.11** Approximation Measure and Approximate Equality for Neighbourhood Differences

An approximation measure \(M_a\) is a non-negative value such that an arbitrary pair of neighbourhood differences \(F_1\) and \(F_2\) are said to be approximately equal to each other if and only if \(|F_1 - F_2| ≤ M_a\).

Summarizing the above discussions, we come to the following strategy to test the unimodal behaviour of the filter responses in order to detect possible edge pixels in a given noisy image.
Algorithm I: Edge Detection in the Continuous Case

a) Extract the minimal value from \( \{ F(\alpha) : \alpha \in [0, 2\pi] \} \), say it is \( F(t) \) with \( t \) from \([0, 2\pi]\).

b) Circularly transform \( \{ F(\alpha) : \alpha \in [0, 2\pi] \} \) at the point \( t \), say it yields \( \{ E(\beta) : \beta \in [0, 2\pi] \} \). Thus, \( E(0) = E(2\pi) = F(t) \). And moreover, \( E(\pi) \) is the maximum of \( \{ F(\alpha) : \alpha \in [0, 2\pi] \} \).

c) Letting \( A_M \) and \( A_m \) be two prescribed approximation measures, define the following subsets of \([0, \pi]\):
\[
X_M = \{ \beta \in [0, \pi] : E(\pi) - E(\beta) \leq A_M \} \\
X_m = \{ \beta \in [0, \pi] : E(\beta) - E(0) \leq A_m \} \\
D = [0, \pi] \setminus X_M \setminus X_m
\]

d) The test of unimodality always fails if the closure of \( D \) fails to be described by a form of \([m, M]\) with \( m < M \).

e) Suppose that the unimodality test did not fail in the above step. Thus, the closure of \( D \) is equal to \([m, M]\). Then, the test of unimodality will still fail if any of the following conditions is violated:

1) \( \text{closure}(X_m) = [0, m] \).

2) \( \text{closure}(X_M) = [M, \pi] \).

3) \( m \leq L_1 \) and \( \pi - M \leq L_2 \).

f) If the test of unimodality still does not fail, define then a function \( G \) as follows:

\[
G(x) = E(x + m) \quad x \in [0, M - m]
\] (6.12)

The test of unimodality is then finally to be passed if \( G \) satisfies a certain \textit{near-increasing} criterium.

We briefly illustrate each step in Algorithm I as follows:

In step a) the point \( t \) at which \( F \) is minimal is located, which is then used in step b) to get the circular transformation \( E \) at that point. Since \( F(\alpha) = -F(\alpha - \pi) \) for \( \alpha \in [\pi, 2\pi] \), \( E(\pi) \) is the maximum of both \( E \) and \( F \). In step c), \( X_M \) and \( X_m \) correspond to respectively the near-maxima and near-minima of the restriction of \( E \) on \([0, \pi]\) based on given approximation measures \( A_M \) and \( A_m \). The extraction of \( X_M \) and \( X_m \) is necessary since in a noise corrupted image it is unrealistic to assume that \( E \) assumes its minimal value at only one point if an edge situation is actually involved. \( D \) is the remaining domain of the restriction of \( E \), which should be a consecutive interval where \( E \) is near-increasing according to Definition 6.9. Step e) is used to ensure that \( X_M \) is only around the point 0, \( X_m \) is only
located around the point $\pi$ and the near-increasing domain $D$ is sufficiently large (refer also to the requirement of a sufficiently small width of the top part of the unimodality as indicated in Definition 6.9). All these properties are required to effectuate a unimodal behaviour. Finally in step f), the actual test of unimodality of $E$ is carried out.

We illustrate Algorithm I through an example in Figure 6.9a.

After step b) the response function $F$ is circularly transformed into $E : [0,2\pi] \rightarrow \mathbb{R}$ as shown in Figure 6.9b. To have a unimodal behaviour for $E$, it is of course necessary that $X_M$ is centered around the point $\pi$ and $X_m$ is separately centered around both ends of the function $E$ as indicated in Figure 6.9c. Moreover, there should be a sufficient number of points with

\[ F(\alpha) \quad \text{and} \quad E(x) \]

\(\text{(a) Original } F \quad \text{and} \quad \text{(b) Circular transformation.}\)

\[ E(x) \quad \text{and} \quad G(x) \]

\(\text{(c) Sets of near-extremes.} \quad \text{and} \quad \text{(d) Corresponding } G \text{.}\)

Figure 6.9: An illustrative example for Algorithm I, where $\tilde{M} = 2\pi - M$ and $\tilde{m} = 2\pi - m$.

some intermediate $E$-values between the near-maxima and near-minima. From this point of view, it is easy to see that steps d) and e) just carry out such a test. The function $G$ shown in Figure 6.9d is one of the two extracted parts with intermediate $E$-values. In order to confirm finally the
unimodality of \( E \), step f) carries the near-increasing test on \( G \). It is easy
to see that the example in Figure 6.9a will pass the test of unimodality as
expected.

The actual performance of \textbf{Algorithm I} depends on a number of
parameters, i.e., \( A_M, A_m, L_1 \) and \( L_2 \). The meanings of these parameters
are related to the desired accuracy for edge locations as well as the degree
of the noise effect. Similarly, the employed \textit{near-increasing} criterium is
related in this way. Based on our experimental results, \( L_1 \) and \( L_2 \) can
be chosen as some fixed fractions of the total interval length, i.e., \( \pi \). \( A_M \)
and \( A_m \) may not be kept fixed throughout the processing. To let them be
both less data-dependent and still have a dynamic behaviour, we suggest
to fix \( A_M \) and \( A_m \) as some fractions of \( \max \{ F(\alpha) \} \) (i.e., \( E(\pi) \)). A precise
definition for a \textit{near-increasing} criterium is difficult. Below, we attempt to
give such a definition.

\textbf{Definition 6.12 Near-Increasing Criterium.}

Let \( c \) be a given familiarity measure and let \( g \) be a local averaging filter. A function \( E : [a, b] \rightarrow \mathbb{R} \) is called to satisfy the near-increasing criterium if we can find an equidistant sampling \( \{ a_1, a_2, \ldots, a_n \} \) of the
interval \([a, b]\) with sampling distance \( c \) such that the finite series \( \{ G(a_i) : i = 1, 2, \ldots, n \} \) is an increasing series, where \( G(a_i) \) for \( i = 1, 2, \ldots, n \) is
defined as follows:

\[
G(a_i) = \int_a^b g(x - a_i) E(x) dx
\]

(6.13)

Now, we state two important propositions on the edge-detection scheme
as presented in \textbf{Algorithm I} by means of the following numerical definition
of edgeness at an arbitrary point.

\textbf{Definition 6.13 Edgeness Based on Algorithm I}

Given a particular point \( p \), we perform \textbf{Algorithm I} on \( p \). The edgeness
of \( p \) is then defined as follows:

\[
\text{edgeness}(p) = \begin{cases} 
\max \{ F(\alpha) \} & \text{if unimodality test passed} \\
0 & \text{otherwise}
\end{cases}
\]

(6.14)
Proposition 6.1 Sufficiency of Algorithm I

Assume that we have an ideal local edge segment at a particular point $p$, which is locally corrupted by an additive generalized zero-mean white noise. Then, $p$ has a significantly large chance of passing the unimodality test in Algorithm I. Moreover, the expected value of edgeness at $p$ such as defined in Definition 6.13 is quite close to the involved edge step size, if the unimodality test in Algorithm I is passed.

Proposition 6.2 Reliability of Algorithm I

Assume that we have an ideal non-edge local environment at a particular point $p$, which is locally corrupted by an additive generalized zero-mean white noise. Then, $p$ will have little chance of passing the unimodality test in Algorithm I. Thus, the edgeness at $p$ such as defined in Definition 6.13 is quite likely to be zero.

The implication of the above two propositions is of course very much desired. Hereby, we are quite confident that our new definition of edgeness is much less dependent on the actual edge step size than is the case for most of the existing edge-detection schemes. However, rigorous justifications for the above two propositions require extensive mathematical treatments and may distract from the current discussion. Therefore, we present these justifications in Appendix D.

Finally, we point out that the strategy in Algorithm I is still far from our ultimate goal of finding discrete edges on a discrete grid. Nevertheless, the importance here is that the discussion in the continuous domain has given us some convincing arguments in support of our new edge detector (consider Proposition 6.1 and Proposition 6.2) and moreover, some strategic ideas on how to implement an actual detector on a digital computer. It is now our task to convert these ideas into a discrete environment. In the next, we will discuss this issue in detail.

6.4 Practical Definition and Implementational Aspects

The main purpose in this section is to design a digital version of an edge detector according to our new definition. In addition to exploring the basic ideas presented in Section 6.2, we will also try to make use of the discovery
(e.g., the unimodal behaviour of neighbourhoods differences) arising in the previous discussion for the continuous case.

The idea in Definition 6.8 translated in a discrete grid is as follows:

If a pixel is part of an edge segment on the boundary of two adjacent regions, then we can always locally draw one and only one family of intersecting lines such that each of the lines roughly represents the local boundary. Moreover, we can find two small neighbourhoods along both sides of each such a line so that the difference of values of some average grey-value property within the neighbourhoods individually will be larger than any such differences obtained by intersecting lines outside the above family.

The essential and unique merit of our new definition for an edge pixel compared to, for instance, the traditional gradient-based definitions is its expected sufficiency towards an edge pixel on the one hand and its expected exclusiveness towards a non-edge pixel on the other hand (refer also to Proposition 6.1 and Proposition 6.2). In other words, a true edge pixel is likely to satisfy our definition for an edgeness pixel whereas a true non-edge pixel is not likely to do so (of course this is only in the sense of low-level visual processing). Clearly, definitions for an edge pixel essentially based on the notion of gradient at least do not possess the exclusiveness as mentioned here.

In the case of a discrete grid, the number of discrete intersecting lines within a small region-of-interest of a particular pixel will be rather limited. Also, the number of pairs of small neighbourhoods along both sides of one specific intersecting line is relatively limited. To continue our discussion further into the details of our design, we limit ourselves to a 5×5-sized region-of-interest for each pixel to be considered. However, the basic design ideas and implementational strategies to be given in the following are similar for any other case.

Suppose that we are to investigate whether or not the central pixel of a 5×5-sized window is an edge pixel. In Figure 6.10, a possible intersecting line is shown if the pixel is actually an edge pixel with the line as part of the boundary for the two assumed adjacent regions. Here, we have implicitly assumed that we are only interested in getting edges on some pixels rather than edges between pixels. To measure the value of some average grey-value property individually within two small neighbourhoods each lying within either one of the two regions, a number of choices can be made for
the neighbourhoods. Three possibilities are shown in Figure 6.11, where
Figure 6.11a and Figure 6.11b correspond to respectively overlapping and
nonoverlapping rectangular neighbourhoods, and Figure 6.11c corresponds
to overlapping non-rectangular neighbourhoods.

\* : the central pixel in question
\* : pixels on an intersecting line
\* : pixels in the 5x5-window
\* : other unimportant pixels

Figure 6.10: An intersecting line as part of the boundary between two
assumed adjacent regions.

\(\text{\begin{tabular}{c|c|c}
\hline
\text{\(a\)} & \text{\(b\)} & \text{\(c\)} \\
\hline
\end{tabular}}\)
an edge), it is quite natural that a neighbourhood should be chosen as large as possible and neither of the two neighbourhoods in a pair may cross over the involved intersecting line. In fact, this requirement may play a dominant role in assuring that the measurement of some average grey-value property within a neighbourhood is actually reliable.

3. Locality.
Naturally, one will agree that an edge pixel at a low level of visual processing is to be locally characterized and, in particular, it is characterized by two mutually touching parts from two adjacent regions, which cause the edge presence. Following this point of view, it is then obviously necessary that both of the two neighbourhoods in a pair should be chosen as close as possible to the central pixel as well as to the involved intersecting line. Moreover, such a neighbourhood should not be chosen excessively large for fear of crossing over the underlying region.

A major step towards the actual design for a digital edge detector based on our new definition involves a set of intersecting lines and their corresponding pairs of neighbourhoods so that the chosen intersecting lines are sufficiently representative for possible local edge segments (refer also to the filters \( h_\alpha : \alpha \in [0, 2\pi] \)) in the previous section). Before we present our choices for this purpose, we outline some essential underlying considerations as follows:

a) All chosen intersecting lines and their corresponding pairs of neighbourhoods should be mutually balanced.

The meaning of being balanced is that all intersecting lines should be approximately of equal length in the sense of a Euclidean-like metric and the corresponding neighbourhoods in each pair should be approximately of equal area. In view of an isotropical behaviour of our final edge detector, such a requirement is crucial.

b) The number of the chosen intersecting lines should be kept as small as possible.

We note that our aim here is to investigate whether or not the central pixel is an edge pixel and not the detection of an accurate small edge segment if the pixel happens to be a true edge pixel. Thus, two intersecting lines, which are mutually familiar, do not both have to be considered. For practical reasons, many unnecessary computations can thus be avoided (note that each extra intersecting line means at least an additional computation load for measuring the property within the
two corresponding neighbourhoods and the neighbourhood difference).
c) The chosen intersecting lines should be sufficiently representative to all possible intersecting lines.

This requirement means that for each possible intersecting line there should be at least one line out of the set of chosen intersecting lines such that both lines are mutually familiar. The need for such a requirement is quite evident. In Definition 6.8, it was in fact explicitly specified that all possible intersecting lines should be involved, though such an involvement may be considered either directly or indirectly. For an edge pixel, if all chosen intersecting lines are spatially quite distinct to the true local edge segment, then the measured maximal difference of values of some average grey-value property will not be characteristic to the actual edge presence. For a non-edge pixel on the other hand, the poor representativity of the chosen intersecting lines may enlarge the probability of still passing the unimodality test and getting an outstanding maximal difference of values of some average grey-value property, leading to the detection of an erroneous edge pixel.

Bearing the above considerations in mind, we give several possible choices for the intersecting lines and their corresponding pairs of neighbourhoods in Figure 6.12, Figure 6.13 and Figure 6.14. Each of them corresponds to an initial choice in Figure 6.11.

![Intersecting lines and corresponding pairs of neighbourhoods](image)

Figure 6.12: Choice 1 for intersecting lines and their corresponding pairs of neighbourhoods.

Having made the choices for the set of intersecting lines and their corresponding pairs of neighbourhoods, we now come to the step of trying
reliably to detect significantly large near-maxima among the differences between the values of some average grey-value property corresponding to one and only one family of mutually close intersecting lines (see also Definition 6.8). At the same time, we will also try to make sure that such a detection is going to fail at a non-edge pixel.

Figure 6.13: Choice 2 for intersecting lines and their corresponding pairs of neighbourhoods.

Figure 6.14: Choice 3 for intersecting lines and their corresponding pairs of neighbourhoods.

Further to conduct our description, we assume from now on that the usual grey-value mean is used for the average grey-value property. Under this assumption, it is easy to see that the previous choices for intersecting lines and corresponding pairs of neighbourhoods in Figure 6.12 and
Figure 6.13 become effectively equivalent. However, these two choices will remain distinct for general cases with a chosen average grey-value property other than the usual grey-value mean.

Under the above assumption, we obtain two sets of masking filters from the choices in Figure 6.12, Figure 6.13 and Figure 6.14 to measure the neighbourhood differences. Each of the two sets contains 8 filters with only 4 independent filters. An independent subset from each of two sets is shown in Figure 6.15.

\[
\begin{array}{cccc}
a & a & 0 & -a-a \\
a & a & 0 & -a-a \\
a & a & 0 & -a-a \\
a & a & 0 & -a-a \\
a & a & 0 & -a-a \\
a & a & 0 & -a-a \\
\end{array}
\begin{array}{cccc}
0-a-a-a & 0-a-a-a & a-a-a-a & a-a-a-a \\
a & 0 & -a-a & 0 & 0 & 0 & 0 \\
a & a & 0 & -a-a & 0 & 0 & 0 & 0 \\
a & a & 0 & -a-a & 0 & 0 & 0 & 0 \\
a & a & 0 & -a-a & 0 & 0 & 0 & 0 \\
\end{array}
\]

(a) Filters corresponding to the choices in Figure 6.12 and Figure 6.13.

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & b & 0 & -b \\
b & 0 & -b & 0 \\
0 & b & 0 & -b \\
0 & 0 & 0 & 0 \\
\end{array}
\begin{array}{cccc}
0 & 0 & 0 & 0-b \\
0 & 0 & -b-b & 0 \\
b & 0 & -b & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

(b) Filters corresponding to the choice in Figure 6.14.

Figure 6.15: Independent subsets of the two filter sets to measure neighbourhood differences, where \(a = 1/10\) and \(b = 1/4\).

We can see that these two sets of filters are similar to the set of rotationally equivalent filters \(\{h_\alpha : \alpha \in [0, 2\pi]\}\) for the continuous case defined in the previous section. In fact, they can be viewed as digital versions of \(\{h_\alpha : \alpha \in [0, 2\pi]\}\). It is therefore obvious that we can expect a similar behaviour here by these two sets of filters. As a matter of fact, it is precisely our underlying strategy to emulate the continuous edge detector in Section 6.3 and borrow from the ideas there in order to design a successful digital edge detector. In the following, we continue our presentation by following the ideas as presented Section 6.3 and, in particular, those in Algorithm I.

The design of a general discrete scheme similar to Algorithm I should be closely related to the chosen average grey-value property as implied in Definition 6.8 and the set of chosen intersecting lines as well as their
corresponding pairs of neighbourhoods. Moreover, care should be taken in noticing the actual noise level in the image, the behaviour of existing edges (such as the width of an edge) and the desired accuracy on edge locations.

In our current implementations, we have only used sets of filters which resemble digital versions of the rotationally equivalent filters \( \{ h_\alpha : \alpha \in [0, 2\pi] \} \) defined in Section 6.3. The main reason is simply that we want to employ the ideas presented in Definition 6.9. Below is our general detection strategy following Algorithm I in Section 6.3, where \( \{ F(i) : i = 1, \ldots, 2n \} \) stand for the filter responses at the pixel in question:

**Algorithm II: Edge Detection in the Discrete Case**

a) Search for the minimum in \( \{ F(i) : i = 1, \ldots, 2n \} \), say it is \( F(t) \).

b) Circularly transform \( \{ F(i) : i = 1, \ldots, 2n \} \) at the point \( t \), say the result is \( \{ E(i) : i = 1, \ldots, 2n+1 \} \) with \( E(1) = E(2n+1) = F(t) \). And moreover, \( E(n+1) \) is the maximum of \( \{ F(i) : i = 1, \ldots, 2n \} \).

c) Search for the set of near-maxima in \( \{ E(i) : i = 1, \ldots, n+1 \} \) under a given approximation measure \( A_M \), say it yields \( Y_M \). Define \( X_M = \{ i : E(i) \in Y_M, i = 1, \ldots, n+1 \} \).

d) Search for the set of near-minima in \( \{ E(i) : i = 1, \ldots, n+1 \} \) under a given approximation measure \( A_m \), say it yields \( Y_m \). Define \( X_m = \{ i : E(i) \in Y_m, i = 1, \ldots, n+1 \} \).

e) Define \( D = \{ 1, \ldots, n+1 \} \setminus X_M \setminus X_m \). If \( D \) can be described by a form of \( [m, M] \) with \( m \leq M \), continue the next step; otherwise, exit by stating the failure of the unimodality test.

f) \( D \) is thus equal to \( [m, M] \). Continue the next step if none of the following conditions is violated:

f.1) \( X_m = [1, m-1] \).

f.2) \( X_M = [M+1, n+1] \).

f.3) \( |X_M| \leq N_1 \) and \( |X_m| \leq N_2 \).

Otherwise, exit by stating the failure of the unimodality test.

g) Define a function \( G \) as follows:

\[
G(i) = E(i+m-2), \quad i = 1, 2, \ldots, M-m+3
\]

The test of unimodality is finally to be passed if \( G \) satisfies the near-increasing criterium defined as follows:

\[
G(i+1) - G(i) \geq M, \quad \forall i \in [1, M-m+2]
\]
where $M_e$ is a prescribed measure.

Based on the discussion in the previous section, we can expect that Algorithm II as described above will yield much more sufficiently reliable results compared to many existing methods. Nevertheless, we must still be aware of some possibly strong noise influence, which may in very special cases still cause a non-edge pixel to pass the test of unimodality in Algorithm II. Motivated by Proposition 6.1 and Proposition 6.2 in Section 6.3, we propose to assign a special measurement value, i.e., the edgeness, to each of the tested pixels instead of a binary attribute indicating only the presence of either an edge pixel or a non-edge pixel. This value is the current case defined as follows:

$$\text{edgeness} = \begin{cases} \max\{F(i)\} & \text{if unimodality confirmed} \\ 0 & \text{otherwise} \end{cases} \quad (6.17)$$

According to the above assignment, each pixel will be assigned a non-negative value as its edgeness. When assigned to a pixel which passed the test of unimodality, the function of the value of edgeness defined here is somewhat similar to a usual gradient-based measurement (such as the gradient magnitude). However, the essential difference of our new edgeness assignment is that it assigns a zero value of edgeness to a pixel which failed the test of unimodality even if the maximum of the filter responses may be quite large. Recalling the implications of Proposition 6.1 and Proposition 6.2, we may claim that our new edge detector with our edgeness assignment is to be more powerful and reliable than many existing low-level edge-detection methods. We call it Edgeness Detector.

From the theoretical results in Appendix D, we conclude that even under conditions of a very low signal-to-noise ratio, the Edgeness Detector can quite reliably work provided that the employed filters have sufficiently large spatial influence (refer to Table D.1 and Table D.2). Throughout our experiments on various heavily noise-corrupted images (including non-SLAR images), we have separately applied various types of filters working on either a $5 \times 5$ region-of-interest or a $7 \times 7$ region-of-interest. The choices for the remaining parameters, i.e., $A_M$, $A_m$, $N_1$, $N_2$ and $M_e$ in Algorithm II (refer also to the parameter $\epsilon$ in Definition D.5), are made quite easily in view of the theoretical results in Appendix D. In particular, $A_M$, $A_m$ and $M_e$ were chosen to be effectively zero and remain fixed for all
types of the filters and on all of the processed images. The results are all remarkably well except the fact that there are quite some resulting edgeness pixels which are actually no edge pixels at all. However, the obtained edgeness values for those erroneous pixels are indeed quite small as we have predicted in our theoretical investigation (refer to Appendix D). In Figure 6.16, we give an example of the result of the Edgeness Detector when applied to an SLAR image.

Figure 6.16: An experimental result on an SLAR image with the Edgeness Detector. (a) The original SLAR image; (b) The output edgeness image (scaled for display purpose).

6.5 Post-processing of an Edgeness Image

As we observed in the previous discussion, an edgeness image has a rather conservative nature. In other words, it is obtained under the underlying consideration of better having more non-edge pixels as edgeness pixels than losing a key edge-pixel. So, it is in general still not suitable for direct utilization by subsequent higher-level processing. Moreover, in the high-level processing, an edge primitive is often desired to have a width of a single pixel and to have a significant length without too many gaps,
even when the local pixels do not show such an evidence solely by their grey values. These kinds of requirements can be considered as being in correspondence with the sensational nature of human perceptual ability and it is therefore clearly understandable that such requirements were not explicitly imposed in the previous low-level processing for generating an edgeness image. For applications in which the edges are meant to represent regional boundaries, it may be furthermore required, or at least desired, that the edge segments are actually closed. In the current section we propose some successful methods for post-processing an edgeness image. By applying these post-processing techniques, it will be possible to fulfil the above-mentioned desires to a large extent.

6.5.1 Thresholding an Edgeness Image

The aim of thresholding an edgeness image obtained by means of the Edgeness Detector is to eliminate those edgeness pixels which do not correspond to an underlying edge segment.

The theoretical results in Table D.1 and Table D.2 suggest that in an edgeness image a correctly obtained edgeness pixel is likely to have its edgeness value close to the size of the local regional transition, whereas an erroneously obtained edgeness pixel is likely to have its edgeness value very small with respect to the standard deviation of the local noise. Throughout our experiments, we have indeed observed the true implication of this theoretical result.

Let us define the signal-to-noise ratio to be that of the smallest step-size of adjacent regions and the existing noise strength. The above-mentioned table results provide us with an effective means of how to choose a conservative but sufficiently critical threshold for the edgeness image if we have some indication of the existing value of signal-to-noise ratio. For different SLAR images as well as some other heavily noise-corrupted images, we have followed the strategy of selecting the threshold to be a bit less than or equal to the expected value of the smallest step-size of the regional transitions. In this way, the threshold can be easily chosen and remain fixed for one category of images (i.e., those images which have approximately the same noise effect and the same expected value of the smallest step size of regional transitions). In Figure 6.17, an experimental result on an SLAR image is shown, where very few of the edge pixels are lost. By using the same set of parameters on other SLAR images among the same category, we have obtained similarly good results.
6.5 Post-processing of an Edgeness Image

Figure 6.17: An experimental result by thresholding the edgeness image shown in Figure 6.16b with a threshold comparable to the smallest step size of regional transitions.

6.5.2 Iteration-Based Adaptive Shrinking Algorithm: A Smart Approach

A thresholded edgeness image gives a good indication of the possible edge pixels in a sense that most of them indeed come from actual edge segments. In practice, however, an edge segment may not only be corrupted by noise but also be dispersed over a wide transitional area. Even in a thresholded edgeness image, such an edge segment will yield a wide strip with positive edgeness values. For the sake of simplicity, we call a pixel an edgeness pixel if its edgeness value is positive and a connected component of edgeness pixels is in turn called an edgeness component. In this section, we focus on obtaining single-pixel wide edgeness components from a thresholded edgeness image. To achieve this aim, it is thus almost always necessary to have a sort of operation which can appropriately shrink the width of each edgeness component while preserving the underlying edge segment. In a more precise way we outline the following general guidelines with respect to the very aim of the shrinking operation:

a) Shrinking requirement 1: Topology conservation.

Under this requirement it is necessary that all removed edgeness pix-
els should not change the topology of the original underlying edgeness components, i.e., relations between components and the connectivity within each edgeness component should be preserved.

b) Shrinking requirement 2: Edge length conservation.

This requirement demands that any remaining edgeness component should be a smoothly thinned version of its original underlying edgeness component but not a shortened version in 'length'.

c) Shrinking requirement 3: High representativity.

Hereby it is required that only those edgeness pixels may be removed whose values of edgeness are locally insignificant within the original underlying edgeness components.

d) Shrinking irrelevance at non-edge edgeness components.

This irrelevance means that if an edgeness component is a purely noise-effected one (thus it does not actually constitute an edge segment), then it does not matter whether or not some of its edgeness pixels are removed. Thus, this is in essence an offered freedom rather than a requirement.

There are quite a number of existing methods which can be used to perform a similar shrinking operation. Examples are, for instance, the local maxima finding or skeletonization techniques. However, there are really few methods available which take all of the above three somewhat mutually conflicting requirements simultaneously into account in a harmonized way. The local maxima finding techniques essentially only take the third requirement into account while simply leaving the satisfaction of the other two requirements to the circumstantial luck. Complementary, the skeletonization techniques strictly obey the first requirement and limitedly follow the second requirement, whereas the third requirement is hereby simply forgotten.

To achieve our aim of shrinking satisfactorily, we propose a new adaptive shrinking algorithm on an iterative basis, which carefully takes all of the above mentioned requirements into account while making use of the shrinking irrelevance as stated in d) above. Globally, it is described as follows:

Adaptive Shrinking Algorithm

step 1. Scan all remaining edgeness pixels one by one. If a scanned edgeness pixel satisfies the conditions for removing, then remove it immediately.
6.5 Post-processing of an Edgeness Image

step 2. If some edgeness pixels were removed during the above scan, then go to step 1; otherwise, terminate the operation.

Apart from defining the conditions for removing, the functioning of the above algorithm is quite straightforward. Indeed, it will successfully work whenever these conditions can be chosen in a critically correct way.

Next, we set out to define some necessary concepts and meanwhile the construction of the conditions for removing will be gradually explored. Because of the shrinking irrelevance stated previously, we consider each encountered edgeness component as if it has an underlying edge segment.

**Definition 6.14 8-Simplicity for an Edgeness Pixel**
Let $p$ be an edgeness pixel and let $N$ be the set of neighbouring edgeness pixels within its $3 \times 3$-neighbourhood. $p$ is called an 8-simplicial pixel if $N$ consists exclusively of a single 8-connected component.

**Definition 6.15 4-Exposedness for an Edgeness Pixel**
An edgeness pixel is called a 4-exposed pixel if it is 4-connected to at least one non-edgeness pixel.

In Figure 6.18 some examples are shown involving the 8-simplicity and 4-exposedness. A major property with the 8-simplicity is that the removal of an edgeness pixel will certainly cause a change in the topology of the involved edgeness component if the pixel is not an 8-simplicial pixel before its removal (see Figure 6.18e). In other words, possible edgeness pixels as candidates for removal are only among the 8-simplicial pixels. Within this category of pixels, we note that the removal of an edgeness pixel will not cause any change in the topology of the underlying edgeness component if and only if the pixel is 4-exposed (see Figure 6.18a–d). In a word, the topology of the edgeness components will remain intact if and only if the removed pixels are both 8-simplicial and 4-exposed.

**Definition 6.16 Degree of Exposedness for an Edgeness Pixel**
Given an edgeness pixel, the size of the largest 4-connected non-edgeness component in its $3 \times 3$-neighbourhood is called the degree of exposedness for the pixel. It is clear that this value will lie in the interval $[0, 8]$. 
Figure 6.18: Illustrative examples for the 8-simplicity and 4-exposedness with a 'o' for a neighbouring edgeness pixel and a 'o' for a non-edgeness pixel: \( \langle a, b, c, d, f \rangle \) are 8-simplicial; \( \langle b, c, d, e \rangle \) are 4-exposed.

For the examples in Figure 6.18, the degree of exposedness is thus 1, 1, 5, 7, 1 and 0 respectively. Within the category of 8-simplicial as well as 4-exposed pixels, the value of this degree approximately tells us the extent to which the pixel as a boundary pixel of the underlying edgeness component is exposed to the non-edgeness surrounding. Concerning our second shrinking requirements, we feel reluctant to remove an edgeness pixel if its degree of exposedness is too small or too large for fear of yielding an unsmoothly thinned edgeness component (see Figure 6.18b) or an edgeness component thinned in 'length' (see Figure 6.18d). To conclude, an edgeness pixel should not be removed if its degree of exposedness is either too large or too small.

Combining the foregoing discussions, we summarize our conditions for removing in parametrical form with parameter \( P = \{ S_1, S_2, T_1, T_2, T_3 \} \) as follows:

1. The pixel is an 8-simplicial pixel as well as a 4-exposed pixel.
2. The degree of exposedness lies within a prescribed interval \([S_1, S_2]\).
3. There should be at least one edgeness pixel in its \(3 \times 3\)-neighbourhood with an edgeness value which exceeds that of the central pixel by a value \( T_1 \).
4. The minimal edgeness value among the edgeness pixels from its \(3 \times 3\)-neighbourhood should exceed that of the central pixel by a value \( T_2 \).
5. The averaged edgeness value of the edgeness pixels within its \(3 \times 3\)-neighbourhood should exceed \( T_3 \) percentage of that of the central pixel.

In the above, condition 1 is used to satisfy the first shrinking requirement, condition 2 for the second shrinking requirement and the remaining conditions for the third shrinking requirement.

Since the above construction has simultaneously taken the three shrinking requirements into account, it can be expected that the performance
behaviour tends to be rather conservative, i.e., it is better to have some redundant remaining edgeness pixels than to throw away a key edgeness pixel accidentally.

In fact, we would like to have a critical shrinking result, i.e., a shrunk edgeness image which satisfies all three shrinking requirements and contains edgeness pixels with a width of one pixel only. To obtain such critical behaviour from the shrinking algorithm, there is no practical guarantee that a corresponding quintuplet \( P \) of parameters will exist. Thanks to the conservative nature of our shrinking algorithm, however, we propose to apply the shrinking algorithm iteratively under different values for the parameter \( P \). The background motivation of the methodology is as follows:

Let \( A \) denote the set of remaining edgeness pixels corresponding to the ideal result. For an arbitrary \( i \), let \( A_i \) be the set of remaining edgeness pixels after applying the shrinking operation under parameter \( P_i \). The conservative behaviour of our shrinking algorithm means that \( \{P_i : i = 1, \ldots, n\} \) can be found such that the successive application of \( P_i \) yield \( A \subset A_i \) for \( i = 1, \ldots, n \). Thus, instead of looking for a particular \( \tilde{P} \) with \( \tilde{A} \simeq A \), we may succeed by looking for several \( P_i \)'s such that the successive application of the corresponding shrinking operation may yield a result quite near to \( A \).

For various images, we have applied our adaptive shrinking algorithm on such an iterative basis. An example is shown in Figure 6.19 by using the thresholded edgeness image in Figure 6.17.

Supported by our experiments on various types of images, we observe the following advantages of our new edge-detection method if conservative choices are made for the parameters in various stages (i.e., edgeness generation, thresholding and adaptive shrinking on an iterative basis):

1. Sufficiency.
   Most of the true edge segments (i.e., those being seen solely by local grey values) are actually detected. In particular, this is the case even for corners or crossings (this may be explained by our conservative choices for various parameters).

2. Reliability.
   The amount of detected edge pixels which are in fact no edge pixels at all, is negligibly small.
Figure 6.19: Experimental results by applying the Iteration-Based Adaptive Shrinking Algorithm to the thresholded edgeness image shown in Figure 6.17 (scaled for display purpose).
3. Accuracy.
   The locations of the detected edge segments correspond remarkably well to the true locations.
Chapter 7

Constrained Region Growing

In continuation of the discussion in Chapter 5, the presentation here is directed towards the issue of extracting tentative regions, which hopefully correspond to either a crop field or a non-crop area in an SLAR image.

When we refer to a region in an image, we usually mean a collection of connected pixels, which all appear similar with respect to certain features such as similar grey values. One of the key issues is, therefore, how to confirm or discard any given set of connected pixels as a legitimate region. The notion of uniformity predicate as mentioned in Chapter 5 can provide an important means of tackling this issue.

Suppose that we have luckily discovered a uniformity predicate for some particular application such that it can not only reliably confirm any 'true' region but also faithfully reject any 'improper' region. We call this an ideal uniformity predicate. And yet, even in such an unlikely, ideal situation we are still going to face a severe computational problem. This is caused by the unfortunate necessity of the regional initialization as mentioned in Chapter 5. In actual applications, this step can really have unpredictable implications on and deteriorate the effectiveness of the subsequent processing. If a preselected region fails to satisfy the uniformity test, how should we then reconfigure the region (possibly together with some other neighbouring improper regions) into several other regions for subsequent attempts? Bearing in mind that a uniformity predicate does not supply any indication of how this reconfiguration should be achieved in practical cases, there will be a prohibitively large number of possible reconfigurations. In 1976, Horowitz and Pavlidis introduced a famous scheme called the Split-and-Merge algorithm, which offered a computationally quite affordable scheme to generate an often reasonably reliable
regional initialization (see [Hor76]). Indeed, thanks to those attractive features, the Split-and-Merge algorithm is widely used in many region-based or combined segmentation approaches even up to today. Here, we also employ this algorithm in establishing our region extraction approach. Next, we introduce the Split-and-Merge algorithm in brief.

7.1 The Split-and-Merge Algorithm

The original Split-and-Merge algorithm by Horowitz and Pavlidis consists mainly of two subsequent stages. The first stage generates a partial quadtree (to be defined later), which serves as a reliable regional initialization in terms of quadtree blocks for the next stage. We call it the split-merge stage. The second one is the grouping stage (possibly including some post-processing steps such as small region elimination), which tries to group existing adjacent blocks into larger regions, such that the region based segmentation definition as given in Definition 5.1 of Chapter 5 is satisfied by the final regions.

![Quadtree Blocks](image)

Figure 7.1: Successive partitioning of a $2^2 \times 2^2$ picture into four square subblocks.

Any digital image can be considered to be of size $2^N \times 2^N$ with $N$ being some positive integer. By partitioning the image domain successively into $4^L$ smaller square blocks of size $2^{N-L} \times 2^{N-L}$ up to $L = N$, we obtain a unique multiresolutional representation of the original image domain. In Figure 7.1, a simple example of such a successive partitioning with $N = 2$ is shown, where each block is represented by a unique index (see also [Pav77], P.104). Since each block at level $L$ corresponds to 4 blocks at level $L+1$ by their common coverage in the original image domain, we can establish a parent-child relationship between blocks from immediate levels and
7.1 The Split-and-Merge Algorithm

therefore build up a tree structure as shown in Figure 7.2. Horowitz and Pavlidis called this the *Quartic Picture Tree* (QPT) of the image domain.

![Quartic Picture Tree](image)

\[
\begin{align*}
L &= 0 \\
L &= 1 \\
L &= 2
\end{align*}
\]

Figure 7.2: Corresponding quartic picture tree (QPT) of the successive partitioning in Figure 7.1.

The aim at the split-merge stage is to partition the image domain into a set of QPT blocks with the largest possible sizes under the given uniformity predicate. Obviously, such a set of blocks uniquely corresponds to the vertices of a particular cut-set of the QPT from a graph-theoretical point of view. In Figure 7.3 we have show an example of such partitions together with its corresponding cut-set in the QPT. We call this a *partial quadtree* of the QPT. Bearing our segmentation aim in mind, each block in a final partial quadtree should accordingly satisfy the uniformity predicate.

![Partial Quadtree](image)

Figure 7.3: A partial quadtree from the *Quartic Picture Tree*, where the dotted line connects the vertices of the corresponding cut-set.

The basic strategy to obtain a partial quadtree as desired for regional initialization is as follows:
1. From the chosen initial mid-level in the QPT, successively merge the quadruplets of blocks upwards into the next level according to the employed uniformity predicate until no more merging is possible.

2. From the chosen initial mid-level in the QPT, successively split the remaining blocks downwards into the next level in favour of the employed uniformity predicate until no more splitting is necessary or possible.

A major merit of the above strategy is its operational parallelism. This is crucial, as far as the regional initialization for segmentation is concerned. Also, the resulting blocks in the output partial quadtree are quite easy to describe completely by numerical means, even including their mutual relationships. For instance, it is possible to define a unique order among the QPT blocks so that each ordering number will not only tell how large the involved QPT block is but also tell which of the other QPT blocks it is adjacent to (refer to [Sam82, Sam85] for a more detailed discussion). Since a QPT block corresponds to a square portion in the image domain, it is easy completely to describe all intrinsic properties (such as the location, internal grey-value properties and so on) of a QPT block under such an ordering. Due to these particular features, a regional initialization in terms of QPT blocks enables a suitable and attractive environment for any kind of subsequent higher-level manipulations such as grouping. In our view, the above-mentioned operational parallelism and attractive data organization explain the continued wide popularity of the split-merge stage even up to today.

It is easy to see that any partial quadtree can only assume its constituent square blocks from a quite specific subset of the entire power set of the original image domain whereas a region in a real image will in general rarely correspond to such a set of square blocks. However, we realize that an output partial quadtree is only intended to yield an initial segmentation in terms of those square blocks (possibly including individual pixels) such that each of them is guaranteed to be within a ‘true’ region though not necessarily to constitute precisely a ‘true’ region. Bearing this in mind, the subsequent stage is therefore to group neighbouring blocks into actual regions under a chosen similarity criterium. According to the region-based segmentation definition such as also adopted by Horowitz and Pavlidis, the similarity criterium in the grouping stage should be similar to the uniformity predicate at the previous split-merge stage, i.e., a set of neighbouring blocks may be grouped together into a single region only if they jointly satisfy the uniformity predicate. In practice, however, an ideal uniformity
predicate can hardly be found. Very often, a non-ideal uniformity predicate is actually used, such as the variance criterium or submeans based min-max criterium as given below:


Let \( R \) be an arbitrary region in the image domain. \( R \) is said to satisfy the variance criterium if the sample variance of the grey values in \( R \) does not exceed a prescribed threshold.

2. Submeans based min-max criterium.

Let \( \mu_1, \mu_2, \mu_3 \) and \( \mu_4 \) be the sample means of the grey values in the four child blocks, which all correspond to a common parent block. Moreover, let \( M \) and \( m \) be the maximum and minimum among \( \{\mu_1, \mu_2, \mu_3, \mu_4\} \), respectively. Then, the common block is said to satisfy the submeans based min-max criterium if \( M - m \) does not exceed a prescribed threshold.

As a consequent measure, similarity criteria other than the employed uniformity predicates are often used in practice in order to restrict undesired grouping behaviour such as is caused by a non-ideal uniformity predicate and/or the sequential processing nature of the grouping stage. Horowitz and Pavlidis use the difference of mean values within the neighbouring regions for the measurement of similarity. For more details, we refer to [Hor76, Pav77].

It should be noted that the Split-and-Merge algorithm, especially the split-merge stage, has become a framework for tackling a wide class of practical problems rather than being just a fixed approach. Both the uniformity predicate at the split-merge stage and the similarity criterium at the subsequent grouping stage can be freely chosen according to the practical feasibility and actual performance with respect to the application. It is even so that a categorically completely different strategy can be used to replace the primitive grouping method such as proposed by Horowits and Pavlidis.

7.2 Edge-Constrained Region-Growing Based Segmentation

In the foregoing discussion, we noted that the issue of finding an ideal uniformity predicate is, in practice, virtually unsolvable. Under the gen-
eral framework of the Split-and-Merge algorithm, what are then the consequences of using a non-ideal but practical uniformity predicate and/or similarity criterium? For our SLAR images, both the theoretical consideration and practical experience have shown that the variance criterium and the mean-value-difference similarity are quite applicable to the uniformity predicate and the similarity criterium respectively (see [Ger84]). In the following, we discuss the consequences of these choices and present an improved approach to counter the negative effects in order to base our segmentation approach on our new definition as stated in Definition 5.3.

The variance criterium \( \text{pred}_v \) adopted as a uniformity predicate is formally defined as follows:

\[
\text{pred}_v(X) = \text{true} \quad \text{if} \quad \text{sample\_variance}(X) \leq \sigma^2
\]  

(7.1)

where \( X \) is an arbitrary region and \( \sigma^2 \) is a prescribed positive threshold.

Under the above definition of the variance criterium, it can be shown easily that a crucial constraint for a formal uniformity predicate as shown in Eq. 5.1 is not to be obeyed whereas the modified condition in Eq. 5.2 can be argued to have been satisfied. This also explains the applicability of the variance criterium as an informal uniformity predicate.

Consider the examples in Figure 7.4, where \( X \) and \( Y \) represent two equally-sized QPT blocks and both have nearly-binary grey values but the dark pixels in \( X \) have a quite uniform distribution while that in \( Y \) have a rather selectively populated distribution. Moreover, the sample variances in \( X \) and \( Y \) are assumed to be both equal to \( \nu^2 \).

![Figure 7.4: Examples of QPT blocks with X being fairly uniformly valued and Y containing two individually rather uniform but mutually distinct regional subcomponents R1 and R2.](image)

Now, let \( \{x_1, x_2, x_3, x_4\} \) and \( \{y_1, y_2, y_3, y_4\} \) denote the sample variances in the four child blocks of \( X \) and \( Y \) respectively. Under the above condi-
tions (see also Figure 7.4), it can easily be derived that \(\max\{x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4\}\) will be approximately equal to \(v^2\) or smaller than \(v^2\) whereas \(y_2\) may significantly exceed \(v^2\). Considering Figure 7.4, it is obvious that we would like \(X\) to become one of the QPT blocks in the final partial quadtree and \(Y\) to be divided into seven subblocks as indicated in Figure 7.4. Under the usual variance criterium, there are only two possibilities of achieving this result as indicated below:

1. **Dynamic assignment of the threshold \(\sigma^2\).**
   Here, the actual value for \(\sigma^2\) is to be determined dynamically at each spatial location of the image domain. In particular, it will be assigned a value to satisfy \(\sigma^2 \geq v^2\) when operating on \(X\) and to satisfy \(\sigma^2 < v^2\) when operating on \(Y\), respectively.

2. **Suitable choice for the initial mid-level of the QPT.**
   Suppose that \(\sigma^2\) is fixed in advance to be equal to \(v^2\). If the initial mid-level is luckily chosen to be corresponding to the child blocks of \(X\) and \(Y\), then it is quite conceivable that the desire to have \(X\) become one block and \(Y\) become seven subblocks as indicated in Figure 7.4 can be achieved without any dynamic assignment for \(\sigma^2\).

The first possibility above will generally require the availability of knowledge about the dynamic behaviour of the regional transitions over the entire image domain. In practice, such knowledge is hardly acquirable and certainly infeasible to obtain for complex images. The second possibility in fact implies nothing but a requirement to choose the initial mid-level of the QPT to correspond to sufficiently small blocks. However, the basic foundation behind the validity of the variance criterium as an informal uniformity predicate comes from the assumption that it will not be used on small blocks. Therefore, this possibility is quite limiting in practice.

We are thus in a situation where the suitability of the variance criterium will be circumstantial. An error, such as \(Y\) in Figure 7.4 turning out to be one block in the final partial quadtree, cannot be ruled out in advance. In our view, the fundamental reasons behind such an error are the following.

Firstly, very few true regions in a real image will have boundaries coinciding with the boundaries of some relatively large (e.g., larger than \(2 \times 2\)) blocks in the QPT. Secondly, the sample variance measurement (or other similar global measurements) needed in the variance criterium inherently has a somewhat averaging feature, by which the influence of the boundary
artifacts can be dispersed over the entire block under consideration and thus becomes hidden.

In the grouping stage (or any other subsequent stage), the final result is directly dependent on the reliability of the partial quadtree. Under the mean-value-difference similarity criterium, however, even a reliable partial quadtree may very well be ill-treated. Strictly speaking, two regions (which are initially QPT blocks) are to be grouped under this criterium into a new region if the following condition is met:

$$|\mu_1 - \mu_2| \leq \varepsilon,$$

where $\mu_1$ and $\mu_2$ are the sample means of the two regions, respectively. To avoid any misgrouping, it is clear that $\varepsilon$ should be either chosen smaller than every possible regional transition or adapted dynamically. Similarly to choosing $\sigma^2$, a dynamically adapted $\varepsilon$ will require knowledge about the dynamic behaviour of the existing regional transitions. However, in a global approach without a dynamically adapted $\varepsilon$, to require $\varepsilon$ to be smaller than any possible regional transition may equally mean to require a high uniformity (i.e., a small sample variance) within each true region.

Summarizing the above discussions, we come to a conclusion as follows:

In order to reach an acceptably correct segmentation result within the framework of the Split-and-Merge algorithm we should possess some knowledge about the behaviour of the existing regional transitions over the entire image domain and use this knowledge to adjust dynamically the thresholding parameters $\sigma^2$ and $\varepsilon$. Otherwise, our segmentation success relies heavily on the circumstances.

For complex images, the above-mentioned knowledge can hardly be acquired in practice. In an effort to compensate the lack of such prior knowledge, a method has been developed (see [Kle88,Ger88b]) in which some prior knowledge was used to guide a Split-and-Merge algorithm based segmentation process. Used as prior knowledge were existing map information, previous segmentations over the same ground area, and so on. The result improved remarkably. Here, instead of assuming the availability of such kinds of off-line knowledge for our current low-level segmentation issue, we try to use the information contained within the raw image(s) to guide the low-level segmentation process as much as possible.
The source of our guiding information comes from the existing edge information. However, it should be made clear that we do not assume the information of a completed edge map (otherwise there would no longer be a segmentation problem!). Instead, we only assume the availability of a rather incomplete edge map (i.e., edge gaps caused by the missing of edge pixels are in principle allowed). The basic strategy here is that any region or block should not cross over an established edge pixel in the given edge map, or, almost equivalently, the region or block should have none of the established edge pixels within its interior. To illustrate this idea, we consider again the examples in Figure 7.4 as follows.

Suppose that we established only one edge pixel around the boundary between $R_1$ and $R_2$ in block $Y$ as shown in Figure 7.5. Now, even when $v^2$ is smaller than $\sigma^2$, we are always to divide $Y$ into seven subblocks as indicated in Figure 7.4, regardless of which initial mid-level was chosen. Similarly, we may also establish such an influence from established edge pixels on the subsequent grouping operation. In particular, the edge pixels at weak regional transitions can prevent a misgrouping even when $\varepsilon$ is chosen not sufficiently small.

![Figure 7.5: An established edge pixel marked by 'x' on the QPT-block Y from Figure 7.4.](image)

What we observe from the above illustration is that an established edge component can guide the regional segmentation process quite significantly, even if it is quite incomplete. The essential aspect here is that the established edge pixels should be reliable themselves. We call our new approach the *Edge-Constrained Region-Growing Based Segmentation*. Since it uses the partial result from an edge detection operation, it is a combined approach of both edge and region based segmentation methods.

In implementing this new approach, we should note several side-effects caused by the imposed edge-constraints. At first, the set of QPT blocks, from which any partial quadtree assumes its constituent QPT blocks, is a
Figure 7.6: An experimental result on an SLAR image with the Edge-Constrained Region-Growing Based Segmentation approach.
rather specific subset of the power set of the original image domain. Due to the inclusion of single-pixel blocks a partial quadtree can, in principle, approximate any irregularly shaped region boundary. However, the introduction of edge-constraints may significantly enlarge the cardinality of the final output partial quadtree and, in particular, an excessive number of very small QPT blocks may result. Such a phenomenon may become misleading in the subsequent grouping stage.

To summarize, we propose a scheme to accomplish the Edge-Constrained Region-Growing Based Segmentation by redefining the variance criterium into the following:

**Definition 7.1** Edge-Sensitive Variance Criterium

\[
pred_{ev}(X) = \begin{cases} \text{true} & \text{if } \left\{ \begin{array}{l} \text{sample}\_\text{variance}(X) \leq \sigma^2 \text{ and} \\ X \text{ contains no interior edge pixels} \end{array} \right. \\
\end{cases}
\]  

(7.3)

where \( X \) is an arbitrary QPT block from the QPT.

At the same time the mean-value-difference similarity criterium is in this approach modified into the following:

**Definition 7.2** Edge-Sensitive Mean-Value-Difference Similarity Criterium

\[
|\mu_1 - \mu_2| \leq \epsilon \quad \land \quad X \cup Y \text{ contains no interior edge pixels}
\]  

(7.4)

where \( \mu_1 \) and \( \mu_2 \) are the sample means of the two neighbouring regions \( X \) and \( Y \) under consideration.

From our discussions so far, we can see that our new segmentation approach is indeed based on the definition as given in Definition 5.3. We conclude this chapter by showing an experimental result in Figure 7.6.
Chapter 8

High-Level Interpretation through Anomaly Handling

In the previous chapters, the main objective has been to constitute a sophisticated scheme to produce regions optimally which are subsequently used for high-level interpretation/analysis. In this chapter, we deal with the problem of such high-level processing in connection with our current application requirements. As stated before, our current problem on interpreting SLAR images involves the extraction of crop fields versus non-crop areas only. However, even for such a two-class high-level problem the accompanying complexity can be severe and has many general aspects. The powerful mechanism, i.e., the anomaly-driven feedback mechanism of DADS, will be utilized, explored in detail and its effectiveness will be examined. We stated in Part I that the anomaly handling mechanism is designed to cope with the irrational part of our thinking. Before we actually come to the topic of utilizing this mechanism, we try first to explore maximally our rational thinking involving the current high-level interpretation problem.

8.1 Extraction of Crop Fields Following the Rational Reasoning

Intuitively, we will accept a primitive region as a possible crop field if, for instance, its size is not too small and its shape can be approximately described as a convex blob. In contrast to this, a road- or river-like area, which is usually long, extends only along its main axis though its size can
also be large. Of course, there can be many characteristics other than the size and the shape which help a perceiver to recognize a primitive region as a possible crop field. In our current implementation, we employ only the size and the shape as the characteristic features for a possible crop field since our present aim is only to show the suitability and the effectiveness of the DADS framework towards solving such a complex interpretation problem as analyzing an SLAR image in a dynamic way. To summarize, our rational reasoning reveals the following two rules for justifying a primitive region as a possible crop field:

1. The region has a large size.
2. The region has an approximately convex shape.

The encoding of the first rule above is straightforward in contrast to that of the second, as the meaning of approximately convex is much more imprecise and vague in a numerical sense. In our current approach we employ the regional distance for the measurement of this feature as defined below.

**Definition 8.1 Regional Distance**

The regional distance at a pixel from a particular connected set of pixels (such as a primitive region or a crop field) is the distance from the pixel to the exterior of the set.

In Figure 8.1 we show a regional map together with the corresponding regional distance map for all previously generated primitive regions of an SLAR image. We observe that a road- or river-like primitive region can have a large size but its maximal regional distance will remain quite small relative to its size. This implies that the maximal regional distance within a primitive region can reveal quite a lot about whether or not the region has an approximately convex shape. Following this observation, we come to the following attempt to define strictly the vague notion of approximately convex shape.

**Definition 8.2 Regional Approximate Convexity**

The regional approximate convexity of an arbitrary set of connected pixels is the maximum of the regional distances within the set.
8.1 Extraction of Crop Fields Following the Rational Reasoning

Figure 8.1: (a) Regional map from an SLAR image, which contains 5302 primitive regions. (b) Corresponding distance map with brightness representing distances.

Combining the observation in Figure 8.1 and the above definitions, we reformulate the two foregoing rules characterizing a crop field as follows:

1. The size of the region must exceed a pre-set threshold (crop size).
2. The regional approximate convexity of the region should exceed a preset threshold (crop extent).

After application of these rules, the resulting satisfactory crop fields are shown in Figure 8.2. We observe from these results that nearly all of the presumably existing crop fields have been extracted, though some of the extracted crop fields should actually belong to non-crop areas while many others still have undesired properties such as holes, strong boundary acuity or even mixtures of different crop fields. Since it is up to the subsequent anomaly handling process to improve such insufficiencies of the extracted crop fields, we do not need to be bothered by them at the moment. What is important to note at this stage is the fact that none of the possible crop fields has failed to be extracted. The number of the previously generated primitive regions is quite huge (there are more than 5000
primitive regions in Figure 8.1) and this uncomfortably large quantity will undoubtedly present a heavy computational burden on the subsequent operations such as the anomaly handling. To avoid this likely unnecessary obstacle, we introduce an extra intermediate data class between the class of primitive regions and the classes of crop fields and non-crop areas: the class of *candidate crop fields*, formally defined as follows.

![Image](image.png)

Figure 8.2: The 87 preliminarily satisfactory crop fields out of the original 5302 primitive regions in Figure 8.1.

**Definition 8.3  Candidate Crop Field**

A candidate crop field is either a primitive region which satisfies the criteria for a crop field, or a set of mutually connected primitive regions such that none of them satisfies the criteria for a crop field.

Thanks to the optimistic nature of our preliminary crop field extraction operation (refer to the result in Figure 8.2), the transformation from the data class of primitive regions to that of candidate crop fields is not likely to discard any clues to the possible crop fields which have already appeared while at the same time the number of data elements involved in the subsequent processing is drastically reduced. In Figure 8.3, the result of this transformation is shown. It is clear that none of the clues to the possibly existing crop fields is lost and yet the number of resulting candidate
crop fields is only approximately 700, which is much smaller compared to the original number of more than 5000 primitive regions before the transformation.

![Map of the 690 resulting candidate crop fields out of the 5302 primitive regions in Figure 8.1.](image)

Figure 8.3: Map of the 690 resulting candidate crop fields out of the 5302 primitive regions in Figure 8.1.

Up to now, we have established a quite optimistic processing path from a raw input image through a number of intermediate data classes up to the final data classes of crop fields and non-crop areas. In the DADS terminology, we have constructed a specific context hierarchy as shown in Figure 8.4. Methodologically speaking, there is actually no essential difference yet when compared with many traditional cascade-like schemes. Indeed, as with those schemes, we are not always happy with the final overall output result even if we succeed in optimally adjusting all processing parameters towards the given input image(s). The essential feature of DADS is its ability in effectively correcting the output result in a dynamic manner. In particular, it provides means to readjust certain parts of the processing on the basis of some individual local circumstances while preserving the other processing parts in a globally compatible fashion. In the next section, we discuss this aspect in detail and show how powerful and suitable the DADS framework is in realizing such dynamic processing towards an overall satisfactory interpretation result.
8.2 Refinement of the Extracted Crop Fields through Anomaly Handling

As we mentioned early in Part I, the DADS architecture offers two major frameworks within which we can utilize our knowledge about a particular category of images. On the one hand, within the framework of context hierarchy our rational interpretation knowledge can be encoded through logical emulation, while on the other hand the framework of anomaly handling mechanism can be used to deal with our knowledge of an irrational nature. The previous presentations in the current thesis part all concentrated on exploring our rational knowledge and subsequent encoding of such knowledge within the framework of context hierarchy. The presentation hereafter will concentrate on the usage of the anomaly-handling based framework, which, in fact, forms the outspoken distinguishing feature of DADS.

When we look at the result obtained under the context hierarchy (as shown in Figure 8.4) based processing, we do, unfortunately, very often observe some errors. Such errors can locally occur in the final output, like an ill-formed crop field or a missed crop field. Similarly, they can also appear at a much earlier processing stage such as a very overmerged regional map or an excessively fragmented regional map. One of the major causes for these errors is what we call the functional inadequacy of digital emulation. This is in fact a fundamental and common source of errors in the area of image processing in a broad sense. Taking an example of edge detection, for instance, none of the existing approaches can be guaranteed
to yield all true edges in a complex image, even though we apparently know where the edges lie if we have to locate them manually. Within our own DADS-based application system, the establishment of the context hierarchy and the corresponding processing blackboxes are essentially based on quite particular background thinkings. In constructing the context hierarchy, we define a number of intermediate data classes as if we know exactly what their member elements are like. When programming the various processing blackboxes in accordance with the established context hierarchy, we either implicitly or explicitly adjust our thinking on an emulative basis. (Note that all involved data classes are considered to be defined). In other words, we concentrate our effort only on establishing some workable digital emulating schemes for the extraction of various data classes ‘known’ to us. Apparently, such emulations can only then be successful and without significant errors if our definitions for the involved data classes can also be numerically and/or logically defined. As we noted before, even for a simple basic concept such as a digital edge segment there is no way appropriately to express or define it by purely numerical and/or logical means.

Among other sources of occurred errors is, for instance, the circumstantially unexpected irregularities. An SLAR image such as we have is itself an outcome of complex preprocessing steps, which transform the direct radar reflectances into image pixels through, for instance, calibration, geometric correction, averaging and resampling. Due to the large variety and circumstantially unpredictable nature of the scene and imaging conditions, unexpected irregularities may occur within an SLAR image. A simple example is, for instance, the occurrence of black bursts, i.e., a portion of pixels with no data at all. Such kinds of irregularities may cause the malfunctioning of the various processing stages under our currently established context hierarchy, though they are not supposed to occur too often.

To counter these errors by means of the anomaly handling mechanism provided by DADS, we first have to find out where (i.e., at which of the vertices in the context hierarchy) the undesired or unpredictable errors may occur and subsequently try to define these errors numerically and/or logically and construct some suitable algorithms to correct these errors. In the terminology of DADS, these errors are called anomalies and their definition and correction are called respectively the anomaly detection and anomaly handling.

Looking at the constructed context hierarchy as shown in Figure 8.4, possible anomalies may in principle occur at every vertex other than the
source vertex. For example, at the vertex edge, it may happen that the resulting edge map contains virtually no edge pixels or conversely contains very few non-edge pixels because the employed operational parameters or even the operations themself were inadequate towards the particular input image. Also, at the vertex region some of the resulting primitive regions may be overmerged due to the unexpected weakness of the relevant local regional transitions, which is beyond the detection capability of the currently employed generation scheme for primitive regions. Unlike the construction of the context hierarchy and its corresponding processing blackboxes, searching and subsequent handling of the possible anomalies are actually carried out without the direct concern of solving the actual interpretation problem on an SLAR image but with the concern of having the established processing scheme work properly. In other words, it aims at various individual vertices of the context hierarchy, which in turn directly aim at solving the interpretation problem. Because of its particular nature, the process of searching and handling possible anomalies will in addition also heavily depend on how lucky we are in having sucessfully encoded our rational interpretation knowledge through the context hierarchy and its corresponding processing blackboxes. This may further explain why we call this process an irrational one.

With our current implementation, we have encountered few significant anomalies in various experiments on SLAR images except at the vertex crop field. Below, we briefly describe two of those classes of anomalies called hollow crop and irregular crop, respectively:

1. **hollow crop:** non-massive crop fields.
   An anomaly of this class is a field from the class candidate, which satisfies our criteria for a crop field but contains unwanted small holes.

2. **irregular crop:** irregularly shaped crop fields.
   An anomaly of this class is also a field from the class candidate, which satisfies our criteria for a crop field. However, its boundary has a rather irregular shape, which is unlikely for a usual crop field. In Figure 8.5 we show a number of these examples.

In the above, we have only described two classes of anomalies, which we will try to solve in the sequel. Of course, there are still more anomalies of other types. For example, we encountered some crop fields which were in fact relatively wide motorways. This is because relatively wide motorways may satisfy our extraction criteria for a crop field under our current
8.2 Refinement of the Crop Fields through Anomaly Handling

Figure 8.5: Examples of irregularly shaped candidate crop fields as anomalies of the class *irregular crop* caused by: \(a\) a locally ragged boundary; \(b\) an inward dent; \(c\) a mixture of different crop fields.

parameter setup, though in our current limited problem specification for interpreting SLAR images, motorways are not required to be extracted. However, we point out with emphasis that it is not within the scope of the present thesis work to provide a completed solution to the complex problem of interpreting SLAR images. Our current aim is to show that the **DADS** architecture is suitable and, especially, it has the potential to tackle this tough problem fully in the immediate future. In particular, we want to show in detail how the anomaly handling mechanism works within the **DADS** framework. Under these considerations, we have chosen these two relatively simple types of anomalies only.

### 8.2.1 Detection of the Anomalies

As a general rule, once we have succeeded in defining certain entities through numerical and/or logical means, we have essentially also accomplished the task of actually detecting these entities. Here, we are going to do exactly the same thing for detecting the anomalies from the two classes mentioned previously.

According to our previous description, an anomaly from the class *hollow crop* is a candidate crop field containing one or more holes (for a definition of discrete holes refer to [Ros76a], P.337). It is clear that a strict definition for the anomaly class *hollow crop* is automatically obtained once the issue of connectivity is solved. Among the possible choices, we favour the usual 4-connectivity, which was also used previously for the within-connectivity of the data elements from the classes *region*, *candidate* and *crop field* respectively. In Figure 8.6, we show several of such anomalies.
We clearly see that the anomaly in Figure 8.6b would be omitted in the case of 8-connectivity.

\[ \begin{array}{c}
\text{\textbullet}\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\end{array} \quad \begin{array}{c}
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\textbullet\textbullet\textbullet\textbullet\textbullet \\
\end{array} \]

(a) \quad (b)

Figure 8.6: Examples of anomalies of the class hollow crop with \textbullet for a pixel in an anomaly, \textbullet for a pixel in a hole and \textcirc for other pixels.

A more complicated issue is the definition of the class irregular crop. The meaning of an irregularly shaped boundary in the foregoing description is essentially subjective and vague. In fact, we encounter here a general shape-description problem, which has long been a difficult issue in quantitative image analysis. The fact that the human visual system is remarkably capable of associating and recognizing various pictorial shapes does not mean that we also have the corresponding quantitative means at our disposal to characterize these shapes. Among the existing methods for quantifying discrete shapes, we encounter tremendous difficulties when applying them towards our current aim of representing anomalies from the class irregular crop. To apply the parametrical representation techniques, the immense variation of the possible anomalies prevents us, for instance, from constructing some applicable functional kernels. Also, the application of shape-primitive based techniques was quickly abandoned due to similar causes. For example, we can assume that all crop fields have an approximately polygonal and convex shape (thus, an anomaly from the class irregular crop does not have such a shape). When given a candidate crop field we, however, have no any idea about which of the possible prototype convex polygons (i.e., 3-sided, 4-sided or other \(n\)-sided ones) we should employ in a fitting scheme to justify the candidate crop field as a legitimate crop field (thus, to falsify it as an anomaly in the class irregular crop). Moreover, even if we can afford to try all possible prototype polygons in an exhaustive way, we will still be unable to establish a computationally feasible fitting scheme unless we would be able first to find out where the vertices (or sides) lie in the given discrete contour of the given candidate crop field.

After many attempts, we eventually constructed a distance-histogram
based justification scheme for recognizing anomalies from the class *irregular crop*, which is also computationally quite affordable. In the following, we are going to illustrate this scheme in detail.

Instead of directly defining an irregularly shaped crop field, we first try to define a regularly shaped crop field. We assume that a regularly shaped crop field is approximately a convex polygon with \( n \) vertices, though the exact value for \( n \) is not known in advance. In Figure 8.7, we depict such a convex polygon with thick line segments, as if working in the continuous space. Within such a shape, we define a distance value for each interior point similar to the regional distance defined previously. In other words, the distance value for each interior point is the Euclidean distance from the involved point to the boundary of the polygon. In this way, we can obtain a function \( G : [0, D] \rightarrow \mathbb{R}^+ \) for each given convex polygon such that \( G(d) \) is the area of the interior points whose distance values are less than or equal to \( d \). Here, \( D \) stands for the maximal distance value within the given polygon.

Figure 8.7: A given convex polygon and some of its inner equidistance polygons.

Related to the function \( G \) above, we define the concept of *inner equidistance polygon* for any given convex polygon.

**Definition 8.4** Inner Equidistance Polygon

An inner equidistance polygon corresponding to distance \( d \) for a given convex polygon is the set of interior points within the given polygon such that the distances of all these points are equal to \( d \).

In particular, the inner equidistance polygon corresponding to distance zero is the given convex polygon itself. In Figure 8.7, the thin line
segments represent some of the inner equidistance polygons corresponding to increasing values of $d$.

By subdividing the interval $[0, D]$ into $m$ equally sized intervals $[d_0, d_1), [d_1, d_2), \ldots, [d_{m-2}, d_{m-1}), [d_{m-1}, d_m]$ with $d_0 = 0, d_m = D$ and $d_i - d_{i-1} = h$, we can define another discrete function $H$ on $\{1, 2, \ldots, m\}$ as follows:

$$H(i) = G(d_i) - G(d_{i-1}) \quad \text{for } i = 1, 2, \ldots, m$$

The function $H$ above can be seen as a 'distance histogram' function in the continuous space. Below, an important proposition on the properties of the function $H$ is given.

**Proposition 8.1**

If the inner equidistance polygons corresponding to distances $d_0, d_1, \ldots, d_k$ are all mutually similar for some $k$ from $[2, m]$, then we have

$$H(i) - H(i+1) = 2h^2 \sum_{j=1}^{n} \cot \left( \frac{a_j}{2} \right) \quad \text{for } i = 1, 2, \ldots, k-1$$

where $a_1, a_2, \ldots, a_n$ are the inner angles of the polygons.

To prove the above proposition, we only need to prove that the expression in Eq. 8.2 is valid for an arbitrary $i$ from the interval $[1, k-1]$ since this expression does not explicitly depend on any particular value for $i$. We show this as follows.

![Figure 8.8: A portion of three inner equidistance polygons corresponding respectively to distances $d_{i-1}$, $d_i$ and $d_{i+1}$ at the $j$th side.](image)


8.2 Refinement of the Crop Fields through Anomaly Handling

In Figure 8.8, we have depicted a corresponding part of three consecutive inner equidistance polygons corresponding respectively to distances $d_{i-1}$, $d_i$ and $d_{i+1}$ at the $j$th side. Since the polygons represent equidistant points, the sets of points $\{A_j, B_j, C_j\}$ and $\{A_{j+1}, B_{j+1}, C_{j+1}\}$ are thus respectively linear and the angles $\alpha$ and $\beta$ are respectively equal to $a_j/2$ and $a_{j+1}/2$. Moreover, since the polygons are mutually similar, $C_j$ and $C_{j+1}$ will therefore not overlap each other. We encounter here thus two trapezia with one sitting upon the other such that both of them have the same height $h$ and have the same base angles $\alpha$ and $\beta$. Let the lengths of the line segments $A_jA_{j+1}$, $B_jB_{j+1}$, and $C_jC_{j+1}$ be $X$, $Y$ and $Z$. From planar geometry, we know that the area of each of these two trapezia satisfies the following:

$$\text{area}_{upper_j} = \frac{h(Y+Z)}{2} = \frac{h(2Y - h\cot(\alpha) - h\cot(\beta))}{2} \quad (8.3)$$

$$\text{area}_{lower_j} = \frac{h(X+Y)}{2} = \frac{h(2Y + h\cot(\alpha) + h\cot(\beta))}{2} \quad (8.4)$$

Combining the above two expressions, we obtain

$$\text{area}_{lower_j} - \text{area}_{upper_j} = h^2(\cot(\alpha) + \cot(\beta)) = h^2(\cot(\frac{a_j}{2}) + \cot(\frac{a_{j+1}}{2})) \quad (8.5)$$

Under the definition in Eq. 8.1, we can also express $H(i)$ and $H(i+1)$ in the following way:

$$H(i) = \sum_{j=1}^{n} \text{area}_{lower_j} \quad (8.6)$$

$$H(i+1) = \sum_{j=1}^{n} \text{area}_{upper_j} \quad (8.7)$$

Hereby, it is easy to see that the expression in Eq. 8.2 follows directly from combining the previously obtained expressions in Eqs. 8.5–8.7 which completes the proof for Proposition 8.1.

What the above proposition tells us is actually the following:
Given an arbitrary shape, we denote its maximal regional distance by $D$ and divide the interval $[0, D]$ into $m$ equally sized intervals. If the given shape is approximately a convex polygon, then there should be a sufficiently large number $k$ from $[2, m]$ such that the following expression is valid,

$$H(i) - H(i+1) = \text{constant} \quad \text{for } i = 1, 2, \ldots, k - 1 \quad (8.8)$$

Moreover, the larger $k$ is, the more certain that the given shape is approximately a convex polygon. If no such sufficiently large number $k$ can be found, then the given shape is discarded to be approximately a convex polygon.

In a strict sense, the above conclusion is, of course, only valid for a continuous shape. To deal with a discrete shape such as the anomalies from the class \textit{irregular crop}, a similar conclusion will, however, be much more complex and it may even not exist in a rigorously mathematical sense since the notion of \textit{polygon} is simply undefined in any discrete space. Nevertheless, we have experienced through experiments that a straightforward generalization of the above conclusion for the discrete case does give sufficiently reliable results. Therefore, we use this generalization to characterize a crop field with an approximately convex polygonal shape or in other words, to justify an anomaly from the class \textit{irregular crop}. Below, we give this generalized criterium.

\textbf{Criterium for an Irregularly Shaped Crop Field}

Given an arbitrary element from the data class \textit{candidate}, we denote its maximal regional distance by $D$ and divide the interval $[0, D]$ into $m$ equally sized intervals. Subsequently, we define a function $HH$ on $\{1, 2, \ldots, m-1\}$ as follows:

$$HH(i) = H(i) - H(i+1) \quad \text{for } i = 1, 2, \ldots, m-1 \quad (8.9)$$

If a sufficiently large number $k$ from $[2, m]$ such that the following expression is valid can not be found,
then the candidate crop field is justified to represent an anomaly from the class *irregular crop*. Here, \( \varepsilon \) is a small positive number which can be experimentally chosen.

In the above, the issue of detecting anomalies has been covered. Next, we discuss how the detected anomalies should be processed by an anomaly handler.

### 8.2.2 Handling of the Detected Anomalies

In our current implementation, the handling operations for the two anomaly classes described previously have been kept relatively simple. We briefly discuss them in the following.

An anomaly from the class *hollow crop* is handled in a straight manner, i.e., by absorbing the gaps into the candidate crop field, which constitutes the anomaly, if certain conditions are met. These conditions are as follows:

a) The gap should not have a size exceeding a preset global threshold.

b) The size of the gap should not exceed a certain percentage of the size of the candidate crop field, which constitutes the involved anomaly.

The handling of an anomaly from the class *irregular crop* is more complicated. There are quite a few factors which may lead to such an anomaly. For example, a locally ragged boundary or an inward dent can cause such an anomaly (see Figure 8.5a-b).

Thus far, we have used the conventional closing operation to handle the anomalies from the class *irregular crop*. The experiments undertaken show satisfactory results, particularly for those anomalies as shown in Figure 8.5a-b. Before we begin on further discussions, we must point out that a simple closing operation does not always resolve all the detected anomalies. Especially, an anomaly such as shown in Figure 8.5c will normally not be resolved. However, this failure should not in any way discourage us from tackling the present interpretation problem within the DADS framework. It should be our immediate future task to incorporate more sophisticated operations for an enlarged anomaly handling capability. Again, we emphasize that a completed, successful system for interpreting SLAR images is beyond the scope of the present study.
Apart from the sole handling of the detected anomalies, there are a number of quite critical issues accompanying such an anomaly handling. These issues are typical of a problem such as image understanding on the one hand and inherent within the DADS architecture on the other hand. Below, we discuss them individually.

a) An existing non-anomaly data element may be affected.
Resolving an anomaly will generally change the territory of the anomaly. This is certainly so in the case of gap absorption and closing operations. For example, the closing operation on an anomaly from the class \textit{irregular crop} may claim extra pixels from some other existing data elements such as an already established crop field which is adjacent to the anomaly; the absorption of a gap by an anomaly from the class \textit{hollow crop} will surely claim the entire territory of other existing data elements within the gap. Since we are interpreting each individual part of the image domain separately (though all interpreted parts should be mutually compatible and collectively yield the final global interpretation), these parts should not overlap in the image domain. When handling a particular anomaly, we should thus be aware of the possible undesired disruption of some other existing non-anomaly data elements caused by the territory change of the involved anomaly.

b) A yet-to-be-handled anomaly may meanwhile be changed.
The reason for this phenomenon is twofold. Firstly, a candidate crop field may be tagged as an anomaly from both the class \textit{hollow crop} and the class \textit{irregular crop}. If it is first handled as an anomaly from the class \textit{hollow crop}, then it may absorb some other pixels and thus become changed itself. Thereby, it may no longer satisfy the anomaly criteria of the class \textit{irregular crop}. Similarly, this may happen the other way round. Secondly, it is quite possible that two or more anomalies from a common class are mutually adjacent in the spatial domain. In such a situation, an anomaly may turn out to be no longer such when it is actually handled, since the handling of its adjacent anomalies which occurred previously may have claimed some significant portion of its territory.
Bearing the above phenomena in mind, we should always check the current validity of an anomaly before the anomaly is actually handled.

The two categories of phenomena described above in fact reveal the dynamic character of a general image-understanding process. In plain words,
8.2 Refinement of the Crop Fields through Anomaly Handling

This means that a data item once emerged may be significantly affected or even destroyed before it can actually initiate a subsequent operation itself. Within the DADS design, such phenomena are carefully dealt with through various specified categorical actions by an anomaly handler (refer to Chapter 3).

Just as noted previously on Page 49 of Section 3.1, we attach a special field called subflag to each of the data elements from those data classes which may be involved in the phenomena described above. Currently, these data classes are candidate and crop field among the principal data classes, and irregular crop and hollow crop among the anomaly data classes. Each of the bits in the field subflag has been assigned a separately specified meaning. For example, the field subflag for a data element from the class candidate has five meaningful bits called new, remove, crop, irregular and hollow. The meanings of these flagging bits are given as follows:

- new: the candidate crop field is newly generated.
- remove: the candidate crop field was previously generated but currently no longer exists.
- crop: the candidate crop field has been accepted as a crop field in the data class crop field.
- irregular: the candidate crop field is currently labelled as an anomaly in the anomaly class irregular crop.
- hollow: the candidate crop field is currently labelled as an anomaly in the anomaly class hollow crop.

Below, we give an illustration of how to deal smoothly with the two previously mentioned phenomena by means of the specified categorical actions for an anomaly handler. We discuss this only for the anomaly handler corresponding to the anomaly class irregular crop. The discussion is basically similar for any other anomaly handlers.

Suppose that anomaly A from the class irregular crop is now to be handled. The corresponding anomaly handler will first check the field subflag of A to see whether its remove-bit is set. If this is the case, then A will be skipped for actual handling. Otherwise, the handler will subsequently check the field subflag of the seed data element of A. The seed data element is the candidate crop field, which was previously mapped to A. If the irregular-bit there is not set, then A will still be rejected for further actual handling. Otherwise, the closing operation will be applied to A in an attempt to resolve it. Since the seed data class is in this case the prin-
principal data class candidate, the handler will search for the possibly affected candidate crop fields after the closing operation. (Note that in this case the data class candidate also serves as the input to the anomaly handler and the supervisor learns of such a relationship through the descriptor of the anomaly handler). In the existing description records for those affected candidate crop fields, the remove-bits then all become set. At the same time, the corresponding updated description data records are added to the current data storage for the data class candidate with only their new-bits being set as if they are newly generated candidate crop fields. If the data storage for the class candidate has actually undergone any mutations, the anomaly handler will properly issue the corresponding return messages to the supervisor of the DADS framework, telling it that some existing elements from the principal data class candidate have been declared as no longer valid and/or some new elements (i.e., updated elements) in this class have been generated.

Recalling the presentation in Section 3.4.3 of Part I, we know that the supervisor will, upon reception of these responses, accordingly activate the relevant processing blackboxes again, which accept the data class candidate as part of the input information. Within our current implementation, the relevant processing blackbox is the one which generates the data elements in the class crop field. In this way, the possibly affected existing crop fields will be reexamined and new anomalies may emerge again from the newly born data elements in the class candidate.

We see from the above that both previously described phenomena can be dealt with within DADS. The only remaining issue around the system's dynamic behaviour is the possible iteration phenomena, i.e., what we called the anomaly recurrency in Section 2.5 of Part I. Such a recurrency is almost certain to occur at an anomaly such as shown in Figure 8.5c, since our closing operation is known to be incapable towards an anomaly like that. Here, we have employed a technique similar to that pointed out on Page 41 of Section 2.5 for detecting anomaly recurrences with an undesired high degree. However, we have not assigned a highly recurrent anomaly as a new anomaly of another new class. Instead, we have assigned the seed candidate crop field of those anomalies as member elements in the data class nonsense object (i.e., non-crop areas). The reason for this simplification is again that the aim of the current study is only to show that the DADS framework is potentially suitable for building up a general image understanding system on a dynamic basis. This simplification should be among the first to be removed by incorporating more anomaly classes.
and the corresponding anomaly handlers. The DADS framework will not be fundamentally changed.

To conclude, a series of pictures is shown in Figures 8.9–8.11 to explain an experiment on an SLAR image using the methodology presented in this chapter.

8.3 Discussion

From the foregoing presentations, we observe that the anomaly-driven mechanism offered by the DADS framework provides a particularly suitable working environment in which a DADS-based application system can achieve somewhat dynamic behaviour. More precisely, this dynamic behaviour encompasses the following:

a) A previously accomplished part of processing can be reinitiated with different operational parameters or even different operational schemes. For example, we can define an anomaly which indicates the occurrence of an excessively overpopulated edge-pixel map. According to the specific nature of such an anomaly, the corresponding anomaly handler may update the edgeness threshold or even change the filtering scheme for extracting the edgeness values of pixels by manipulating the parameter input of the processing blackbox corresponding to the data class edge. Subsequently, it issues a message to the supervisor requiring the processing blackbox to reprocess its input data entirely.

b) A part of a previously generated data class can individually undergo a deviating treatment if there appears to be such a need. This kind of dynamic behaviour was observed and discussed in the previous presentations.

The important aspect in properly achieving dynamic behaviour is constantly to ensure the overall consistency and compatibility among all occurring data. For those data elements which belong to a common data class, this requirement means that they should not significantly overlap in the spatial domain. For data which belong to different data classes but which are mutually related through some data-flow path, this requirement means that they should jointly conform to the manner of data flow. For example, a data element from a higher ranked data class should lose its status to exist whenever any of its (either directly or indirectly) originat-
Figure 8.9: The initially generated data elements in the class \textit{crop field} and the anomalies detected by the corresponding processing blackbox.
8.3 Discussion

(a) Currently satisfactory crop fields.

(b) Corresponding anomalies in the class irregular crop.

(c) Corresponding anomalies in the class hollow crop.

Figure 8.10: The currently generated data elements in the class crop field and the detected anomalies at an intermediate moment when some anomaly handling operations have been accomplished.
(a) Final satisfactory crop fields.

(b) Final unresolvable anomalies in *irregular crop*.

(c) Final unresolvable anomalies in *hollow crop*.

Figure 8.11: The finally generated data elements in the class *crop field* and the remaining unresolvable anomalies.
ing data elements from some lower ranked data classes vanishes. Towards establishing this kind of consistency and compatibility, DADS makes use of the *categorical actions* for a processing cluster to keep track of each possible step of the dynamic processing. In this way, the individual processing clusters only need to concentrate on their own specific tasks within the entire interpretation process while leaving to the supervisor the overall coordination to control the dynamic behaviour.

To improve the performance of the currently established DADS-based application system, we only need to investigate where (i.e., at which of the vertices of the context hierarchy) a particular malfunctioning is observed and define the corresponding anomalies accordingly. Once a new class of anomalies is defined, we accordingly improve the overall performance by adding the appropriate anomaly handler. In this manner, the number of anomaly classes, and thus the number of anomaly handlers can become large. However, the overall processing efficiency of the resulting system will not deteriorate in a proportional way. Remember, anomalies are not supposed to occur frequently and anomaly handlers are only then to be activated if corresponding anomalies do actually occur.
Chapter 9

Concluding Comments

In this part of the thesis, we presented an actual DADS-based application system for the interpretation of SLAR images. The system presented here should be considered as an example only, in the sense that it should serve to illustrate the potential power of DADS. It is in no way intended to be a complete solution towards the problem of SLAR-image interpretation. Thus, it is by no means our intention to judge the DADS framework by means of the interpretation results obtained to date, even within our currently limited problem formulation. Our intention is only to illustrate how a DADS-based application system can actually be built and, more importantly, to show the foreseeable capability of the DADS framework towards overall satisfactory interpretation results.

Regarding the behaviour of the current DADS-based application system, we conclude the following:

1. The system is flexible. Because of the functional nature of each processing cluster, expanding the current application system or changing to another application will only require the addition and/or adjustment of some individual processing clusters and the updating of the description of the corresponding context hierarchy.

2. The processing is efficient since the anomaly-caused reprocessing involves only those data which indeed need to be reprocessed.

3. The dynamic processing behaviour has been observed and, especially, it converges to a final state after a few reprocessing steps caused by the encountered anomalies.

4. The system is effective. Once a particular class of anomalies is defined, anomalies actually encountered will be effectively dealt with by the
corresponding anomaly handler without deviating the overall processing attention.

5. Interactions between different levels of processing can be effectively established through various anomaly handlers.

On examining the final results shown in Figure 8.11, we admit that the current results are still insufficient with respect to our very interpretation aims. In particular, there are still too many portions in the image domain which are assigned to the class of *nonsense object*. These nonsense objects are mainly caused by unresolvable or recurrent anomalies from the class *irregular crop*. The origin of such an undesired outcome is twofold, i.e., the limited resolving capabilities of the currently implemented anomaly handlers as well as the inaccurate and static numerical schemes for detecting anomalies. In our view, they can be solved to a large extent by fully utilizing the features of the *DADS* framework (note, for instance, the full scale of *categorical actions* for a processing cluster presented in Chapter 3). Currently, we are considering the following options for further improvements:

1. Anomaly detection schemes on a dynamic basis.
   Hereby, we mean a dynamic modification to certain anomaly detection schemes in a processing blackbox through the anomaly handlers. For example, when encountering some highly recurrent anomalies the anomaly handler corresponding to the class *irregular crop* should, instead of sending the anomalies to the class of *nonsense object*, adjust certain parameters of the processing blackbox corresponding to the data class *crop* so that the anomaly detection scheme for *irregular crop* is somewhat softened. In this way, some original anomalies may pass the test and become a legal crop field (we observe that a quite large proportion of the recurrent anomalies are, in fact, quite acceptable as crop fields).

2. Adjusting the existing anomaly handlers.
   As we noted before, the current anomaly handler for the class *irregular crop* is too limited in actually resolving a corresponding anomaly. Further improvement is undoubtedly necessary. For instance, we can utilize the available information in the data class *edge* to confirm whether or not an anomaly should be really considered as such. If this confirmation is denied, the corresponding anomaly can be absorbed to the class *crop* directly by the anomaly handler itself. Otherwise, the anomaly
handler may also consider splitting the anomaly into more primitive data elements in the class candidate so that these new elements may have a better chance of being accepted as a crop field.

8. Adding new classes of anomalies and corresponding anomaly handlers. Probably, we are better off defining a new class of anomalies, which stands for recurrent anomalies in the class irregular crop, and adding a corresponding anomaly handler. As we can see from the current experiments, anomalies from the class irregular crop have been caused by a diverse range of causes. The closing operation in the handler can only, in a limited way, solve those anomalies whose boundaries are locally irregular, but fails to solve an anomaly the irregularity of which has a more global cause. Thus, separate treatments for such an irregularly shaped crop field is clearly necessary to obtain an improved performance.

We believe that the above modifications will significantly improve the final interpretation results.

In addition to the development of a DADS-based application system, we have also made contributions to the basic image processing techniques, in particular, the Edgeness Detector, the Iteration-Based Adaptive Shrinking Algorithm and the Edge-Constrained Region-Growing Based Segmentation. These new techniques are, in fact, generally suitable for many other applications. This can also be seen from our experiments (though not directly reported in the current thesis) on heavily noise-corrupted images other than the SLAR images.
Bibliography


Bibliography


Appendix A

Heap Structure

The heap structure is a data structure especially suitable for dynamically manipulating data records of (significantly) variable sizes. Examples are, for instance, the records which describe edge segments by individual chain codes. In such cases, it will be either wasteful or practically unaffordable if an array of fixed-sized records should be defined with sufficiently large record size. The usage of the heap structure will in these cases be a good alternative.

Basically, the heap structure contains two parts, i.e., the memory heap and the array of record pointers as described in the following.

a) Memory heap.
This is normally a large memory chunk with prescribed size. During the processing course, it will be divided into two consecutive portions, i.e., the occupied memory portion and the remaining free space portion as shown in Figure A.1. Two important parameters here are the total size of the memory heap and the starting address of the remaining free space or the heap pointer.
All existing data records of variable sizes are in principle orderlessly stored within the occupied portion of the memory heap. The formation of such a record is shown in Figure A.2, where \( N \) is the record number, \( S \) is the record size and \( \text{body} \) is the actual record data.

b) Array of record pointers.
This is basically a one-dimensional array and each of its components contains the starting address of the corresponding data record in the memory heap.

185
occupied portion  remaining free space

memory heap

heap pointer

Figure A.1: Composition of the memory heap.

\[
\begin{array}{c}
N \\
S
\end{array}
\]

Figure A.2: The formation of a data record in the memory heap.

From the above description it is easy to see that the access to a particular data record is easy and fast just by means of the corresponding record pointer to locate the actual data record in the memory heap. In the sequel, we discuss the dynamical manipulation on such a heap structure caused by adding or changing a record.

a) Adding a new data record.
The procedure will normally contain three steps as given below:
1. Place the new data record at the start of the remaining free space of the memory heap, which is currently indicated by the heap pointer.
2. Assign the current value of the heap pointer to the record pointer corresponding to the new data record.
3. Modify the heap pointer accordingly, by adding the size of the new data record to indicate the reduced remaining free space of the memory heap.

b) Updating an existing data record.
As the updated data record may not have the same length as the original data record, the updating proceeds by first dismissing the existing data record and then considering the updated data record as a new record. The dismissing of an existing data record is accomplished by negating for instance the second field in the record, i.e., the record size field. In this way the originally occupied memory by the data record becomes
then released.

By the gradual build-up and repeated modification of the memory heap in the above manner, we can imagine that two erroneous or undesired situations may occur as described below:

1. There are many of released data records within the occupied portion of the memory heap. In the usual terminology, the memory portion occupied by such released data records is called garbage.
2. The remaining free space of the memory heap is smaller than the size of a new data record, which is due to be added.

The situation 1 above is clearly undesired and wasteful while the other situation is to cause a processing error. One of the usual assumptions when employing the heap structure is that situation 2 will rarely occur and if it does occur, situation 1 will then be supposed to co-occur. The garbage collection operation, which may be tedious, is an operation which will be activated to deal with situation 1 only if situation 2 occurs. By the above assumption, the garbage collection operation is thus an exceptional processing. The task of the garbage collection operation is to squeeze the data records in the occupied memory of the memory heap and thus enlarge the remaining free space. If after a garbage collection operation we still encounter situation 2, then a fatal error occurs. The main features of a heap structure are:

a) Direct and fast access to individual data records.
b) Easy memory allocation, i.e., no principal worry about the quite distinct sizes for expected data records.
c) The extra load due to exceptional garbage collection operations will not cost us more than what we may gain from such a data structure.

In a more general usage of the heap structure, the array of record pointers does not need to be a one-dimensional array. It can be any multi-dimensional array provided that the individual record pointers are specified in a prescribed field of the array components.
Appendix B

Implementation Aspects for the DADS Framework

As we can see from the discussion on system design in Chapter 3, the implementation of the DADS framework involves the supervisor mainly, i.e., the system’s overall controller. It contains, in principle, none of the actual processing clusters. Their individual implementations are to come forwards only when the DADS framework is actually employed for solving a particular application problem (or a particular category of application problems). However, the implementation of the DADS framework itself will impose some strict implementational specifications on the potential processing clusters in any future applications. The discussion here is not meant to give a full and detailed description of the current DADS implementation. Instead, we only give a brief description in order to illustrate this implementation on a global basis.

In the specification of the DADS design, a processing cluster is known to the outside world mainly through its input/output specifications. Out of concern about the efficiency as well as the flexibility in the system development and processing, all processing clusters in the current DADS implementation are separate programs instead of a set of subroutines. In reality, some of them may just be a straightforward program (e.g., a program which generates an output quadtree from an input image based on the variance criterium) or even be a dedicated rule-based production system (e.g., an edge grower on a set of input edge primitives based on a particular set of production rules).

From the above considerations, we come to the following basic implementational requirements on the computer facility:
1. Multitasking environment.
   This implies the possibility of simultaneous activation of several mutually independent processes (or programs).
2. Global memory facility.
   This will enable mutually independent processes to share a common global memory block, which is created by one of these involved processes.
3. Asynchronous message exchange.
   For a flexible well-coordination among mutually independent processes it is often necessary that some direct communication between the running processes is possible at any desired intermediate moment of processing. With the help of an asynchronous message exchange facility such a form of process communication can be established.
4. Status control over other processes.
   This can be seen as an extension to the previous requirement and implies that one process can alter the status of another independent process such as activation, suspension or even dismissing. In this way, we can establish a sort of master-slave relationship among independent processes.

The VMS-based VAX11/750 minicomputer, which was available for this project, has the capability basically to support all of the above requirements. In view of this, the current DADS implementation has been developed on this machine, even though the only resident high-level programming language, i.e., VAX/VMS FORTRAN 77, is somewhat awkward in the context of flexible data structures and processing control. In order to improve the environment for developing the system, a lot of auxiliary facilities have been developed so that the native inflexibility of FORTRAN has been greatly reduced.

In the current implementation, which is essentially a serial one, the supervisor is considered as the master process for any future DADS-based application system and all other processes (i.e., the individual processing clusters) are its slave processes.

According to the tasks of the supervisor such as described in Chapter 3, its main processing strategy is roughly as follows:

1. Acquire for the descriptions about the current context hierarchy and all processing clusters.
2. Set the status of all anomaly handlers to inactive.
3. Set the status of all processing blackboxes to inactive except those corresponding to the sources, which are set to active.
4.1 Look for lowest ranked processing clusters with the status of active. If none has been found, then go to step 5.1.
4.2 If more than one processing cluster have been found, then select an anomaly handler among them if possible, or otherwise select an arbitrary processing blackbox among them.
4.3 Prepare the memory management on global memory for the selected processing cluster and subsequently, activate the processing cluster.
4.4 Respond to various return messages concerning the categorical actions taken by the activated processing cluster; go to step 4.1.
5.1 Look for a principal data class, which is productively linked at least to one higher ranked data class and its finish flag still shows the false-value; If none has been found, then go to step 6.1.
5.2 Transfer all remaining unabsorbed member elements of the data class to the class of nonsense objects; Go to step 5.1.
6.1 Look for a processing cluster with the status of pause; if none has been found, then goto step 7.
6.2 Transfer all remaining unprocessed input data of the processing cluster to the class of nonsense objects; Go to step 6.1.
7. Output the system log file end exit.

In Figure B.1, the main processing flow-chart for the supervisor is shown. In the following, these modules will be individually discussed.

1. **INIT**: Initialization of the System Configuration.
   This is the only subroutine, where the system input information is gathered in and properly handled to set up the initial system configuration. The main functions of this subroutine are:

   a) Gathering the detailed description about the context hierarchy and related descriptions about such as anomaly classes and anomaly handlers.
   b) Create common global memory blocks for system status and global memory (see also Chapter 3).
   c) Establish the asynchronical communication channels for subsequent message exchanges with other slave processes (i.e., processing clusters).
Currently, we have chosen to feed all the system input information through a special system input file with the prespecified formation. It contains in turn the following information:

a) Name of the context hierarchy involved.
b) Number of the principal data classes.
c) Number of the anomaly data classes.
d) Individual descriptions of all involved data classes, containing the following information:
   - Identification number of the data class.
   - Name of the data class.
   - Rank of the data class.
   - Name of the corresponding temporary deposit file.
   - Name of the corresponding processing cluster.
   - Name of the parameter file to the processing cluster.
e) Co-incidence matrix among classes of principal data.
f) Correspondence matrix between principal data classes and anomaly data classes.
Specified size for the common global memory global memory.

On the basis of the above information, INIT will create global memory and system status as global memory blocks by means of the VAX/VMS global memory facility and assign unique names for the subsequent identification by a slave process. It can easily be seen that the above information in the system input file is sufficient for INIT to compute the initial values of all fields in system status, which contains the descriptor for the context hierarchy, the descriptors for all processing clusters and the data descriptors of all data classes (refer also to Chapter 3).

Because of the essentially sequential nature of the current implementation, only one processing cluster will be activated at any moment. This means that the supervisor is to have only one running slave process at any time. Thus, two communication channels for the message exchange will be sufficient, with one for the outgoing messages to the activated slave process and the other for the incoming messages from the activated slave process. INIT creates these two channels by using the VAX/VMS Mailbox Facility and keeps their channel numbers as their identifications for the subsequent usage by any activated slave process.

2. SELECTOR: Selection of a Processing Cluster.

The function of this subroutine is straightforward, i.e., properly selecting one among the processing clusters with the status of active. If such a processing cluster can be found, the control is then to be transferred to the next subroutine MM. Otherwise, the control will be transferred to TERMINATOR. The selection criterium for a processing cluster with the status of active was already given in the foregoing presentations (see for instance Page 191).

3. TERMINATOR: System Termination.

The main task of this subroutine is to perform the termination of the entire processing. The steps to be taken are given in Page 191. However, before it actually terminates the entire processing, the contents in the data storage of each existing data class are transferred to the corresponding deposit file. So, if the system log file contains no error message, all desired system output data can then be retrieved from the corresponding deposit files.

The system log file currently contain the following information:
a) Final value of *system error status*, which is a global variable in the system and constantly tells whether or not a specific error has been encountered.

b) Number of non-fully processed data classes, which are either principal data serving as input to a processing blackbox through a productive path or anomalies.

c) Names of all above mentioned data classes.

4. MM: Memory Management.

The subroutine MM will be invoked if a processing cluster has successfully been selected by SELECTOR for the subsequent activation. The main task of MM is to put the data storages of all input/output data classes of the selected processing cluster properly within *global memory* and, if necessary, remove some other existing data storages from *global memory* by saving their contents into the corresponding temporary deposit files.

The main strategy of MM has, in fact, already been described in Section 3.4.2. Here, we will discuss one remaining implementational issue only, i.e., how MM calculates the desired size of a data storage involved. Because of the dynamic nature of all data classes, it should not be assumed that the required size of each data storage can be determined at the system-initialization stage and they will remain unchanged throughout the entire processing. Thus, MM will face the problem of determining the required size of a data storage and in particular when the data storage is to be used for the first time (as may happen with an output data class).

To overcome this problem, we have made some additional implementation specification on the processing clusters as stated below:

**Additional Implementation Specification on a Processing Cluster**

Each input or output data class of a processing cluster should be described in the corresponding parameter file by a specific data line as shown below:

\[(\text{DATA})\langle C \rangle \quad \text{class name} \quad \text{def}\]

where *class name* is the name of the involved data class and the values of *C* and *def* will together specify how to calculate the desired size for the corresponding data storage according to the following implementational conventions:

*C = 0*: size equal to *def*.
*C = 1*: size equal to current effective size of the storage.
$C = 2$: size equal to current effective size of the storage plus $\text{def}$.
$C = 3$: size equal to current effective size of the storage times $\text{def}$.

What the above specification means is that the potential designer of a processing cluster should specify the sizes for the input and output data storages through the corresponding parameter file in the above manner. Moreover, a parameter file may not contain a data line starting with
\{'(DATA)\}' for a purpose other than the above one.

By reading the corresponding parameter file, MM can then calculate the actually desired sizes for the involved input and/or output data storages.

5. **ACTIVATOR**: Activation of a Processing Cluster.

This is the subroutine, which is responsible for activating a prepared processing cluster as a slave process and responding to various *categorical actions* taken by the activated processing cluster. In Figure B.2 a schematic processing flow-chart for ACTIVATOR is given. We briefly explain each part as follows.

At the beginning, the parameter file of the processing cluster is to be updated. This updating includes, for instance, the placement of the actual sizes of all involved data storages.

Once its parameter file has been prepared, the processing cluster will be activated as a slave process and certain messages will be sent to the slave process. These messages include, for instance, those which enable the slave process to map its own working memory onto *global memory* and possibly also onto *system status*. Examples are such as the identity and size of *global memory*.

After activating the processing cluster, ACTIVATOR suspend its current processing and wait for return messages from the slave process.

There are two kinds of return messages, which may come from a slave process, one for the notification of a *categorical action* taken by the slave process and the other one for indicating the exit sign from the slave process. By convention, the last message from a slave process is always the exit sign and all other previous messages are for the purpose of notifying an undertaken categorical action.

Any message from a slave process is defined to consist of a *message code* followed by the actual message. If the message code is zero-valued, then the message is considered as the exit sign and moreover, the contents of the subsequent message should then indicate whether or not any error
has occurred in the slave process. A nonzero message code always means the notification of an undertaken categorical action and at the same time, it indicates which of the allowed categorical actions has actually been undertaken by the slave process (refer to Section 3.3 for a summary of allowed categorical actions to be taken by a processing cluster).

Upon reception of a message notifying an undertaken categorical action, ACTIVATOR will immediately respond according to the specifications given in Section 3.3.

Upon reception of a message notifying the exit sign, the error code will be put to the system error status and the control is then transferred to SELECTOR.

Figure B.2: Processing flow chart of the subroutine ACTIVATOR.
Appendix C

Calculation of Continuous Filter Responses

Here, we will give a detailed derivation of the continuous filter responses $F(\alpha)$ out of the set of rotationally equivalent filters $\{h_{\alpha} : \alpha \in [0, 2\pi]\}$ under an ideal straight step edge. The results were already stated in Eqs. 6.8–6.9.

Letting $\alpha$ be an arbitrarily fixed point from the interval $[0, \pi]$, we replace $h_{\alpha}$ by $h(x, y)$ and $h(r, \theta)$ in the Cartesian coordinate system and in the polar coordinate system, respectively.

By the convention that $h_{\alpha}(r, \theta + 2n\pi) = h_{\alpha}(r, \theta)$ for all $\theta \in [\alpha, \alpha + 2\pi]$ and $n \in \mathbb{Z}$, we can easily conclude from the definition in Eq. 6.3 that the function $h_{\alpha}$ possesses the following properties:

\[
    h(r, \theta) = -h(r, \theta - \pi) \quad \forall r \in [0, \infty), \theta \in [-\pi, \pi] \quad \text{(C.1)}
\]

\[
    \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) \, dx \, dy = \int_{-\pi}^{\pi} \int_{0}^{\infty} h(r, \theta) \, r \, dr \, d\theta = 0 \quad \text{(C.2)}
\]

\[
    \int_{r \cos(\theta) \leq d} h(r, \theta) \, r \, dr \, d\theta = -\int_{r \cos(\theta) \geq d} h(r, \theta) \, r \, dr \, d\theta \quad \forall d \in (-\infty, \infty) \quad \text{(C.3)}
\]

According to the expression in Eq. 6.5, the filter response $F(\alpha)$ can be derived in the following way:

\[
    F(\alpha) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) h(x, y) \, dx \, dy
\]
\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{d} u h(x, y) \, dx \, dy + \int_{-\infty}^{\infty} \int_{d}^{\infty} (u + v) h(x, y) \, dx \, dy \]
\[ = u \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) \, dx \, dy + v \int_{-\infty}^{\infty} \int_{d}^{\infty} h(x, y) \, dx \, dy \]
\[ = v \int_{-\infty}^{\infty} \int_{d}^{\infty} h(x, y) \, dx \, dy = v \int_{x \geq d} h(x, y) \, dx \, dy \]
\[ = v \int_{r \cos(\theta) \geq d} h(r, \theta) \, r \, dr \, d\theta \quad (C.4) \]

and substituting \( \theta \) by \( \theta - \pi \) yields furthermore,

\[ F(\alpha) = v \int \int_{r \cos(\theta) \leq -d} h(r, \theta - \pi) \, r \, dr \, d\theta \quad (C.5) \]

With the properties of \( h_\alpha \) in Eq. C.1 and Eq. C.3 the expression in Eq. C.5 can be rewritten as follows:

\[ F(\alpha) = -v \int \int_{r \cos(\theta) \leq -d} h(r, \theta) \, r \, dr \, d\theta = v \int \int_{r \cos(\theta) \geq -d} h(r, \theta) \, r \, dr \, d\theta \quad (C.6) \]

Letting \( D = |d| \), we conclude therefore from the expressions in Eq. C.4 and Eq. C.6 the following:

\[ F(\alpha) = v \int \int_{r \cos(\theta) \geq D} h(r, \theta) \, r \, dr \, d\theta \quad (C.7) \]

or equivalently,

\[ F(\alpha) = v \int_{-\pi}^{\pi} \int_{r \cos(\theta) \geq D} h(r, \theta) \, r \, dr \, d\theta + v \int_{0}^{\pi} \int_{r \cos(\theta) \geq D} h(r, \theta) \, r \, dr \, d\theta \]
\[ = v \int_{0}^{\pi} \int_{r \cos(\theta) \geq D} h(r, -\theta) \, r \, dr \, d\theta + v \int_{0}^{\pi} \int_{r \cos(\theta) \geq D} h(r, \theta) \, r \, dr \, d\theta \]
\[ = v \int_{0}^{\pi} \int_{r \cos(\theta) \geq D} \{h(r, -\theta) + h(r, \theta)\} \, r \, dr \, d\theta \]
\[ = v \int_{0}^{\pi/2} \int_{r \cos(\theta) \geq D} \{h(r, -\theta) + h(r, \theta)\} \, r \, dr \, d\theta \quad (C.8) \]

Now, we define a function \( H(r, \theta) \) and a constant \( c \) as follows:
Appendix C Calculation of Continuous Filter Responses

\[ H(r, \theta) = h(r, -\theta) + h(r, \theta) \quad r \in [0, \infty), \theta \in [0, \pi/2] \quad \text{(C.9)} \]

\[ c = \arccos(D/R) \quad \text{(C.10)} \]

It is easy to see that \( c \in [0, \pi/2] \). To carry on the further derivations of \( F(\alpha) \) we separately consider different values of \( \alpha \) in the interval \([0, \pi]\).

a) \( \alpha \in [0, c] \).
Since \( c \in [0, \pi/2] \), we have thus also \( \alpha \in [0, \pi/2] \). The function \( H(r, \theta) \) will then have the following expression:

\[ H(r, \theta) = \begin{cases} 
4/\pi R^2 & \theta \in [0, \alpha], r \leq R \\
0 & \text{elsewhere} 
\end{cases} \quad \text{(C.11)} \]

Filling this in Eq. C.8 yields,

\[ F(\alpha) = \frac{4v}{\pi R^2} \int_0^\alpha \int_{r \cos(\theta) \geq D, r \leq R} r \, dr \, d\theta = \frac{4v}{\pi R^2} \int_0^\alpha \int_{D/\cos(\theta)}^R r \, dr \, d\theta \]

\[ = \frac{2v}{\pi R^2} \int_0^\alpha (R^2 - \frac{D^2}{\cos^2(\theta)}) \, d\theta = \frac{2v}{\pi R^2} (R^2 \theta - D^2 \tan(\theta)) \bigg|_0^\alpha \]

\[ = \frac{2v}{\pi R^2} (R^2 \alpha - D^2 \tan(\alpha)) = \frac{2v}{\pi} (\alpha - \cos^2(c) \tan(\alpha)) \quad \text{(C.12)} \]

b) \( \alpha \in [c, \pi/2] \).

Similar to case a) above, the function \( H(r, \theta) \) will appear as shown by the expression in Eq. C.11. And, we have therefore the following:

\[ F(\alpha) = \frac{4v}{\pi R^2} \int_0^\alpha \int_{r \cos(\theta) \geq D, r \leq R} r \, dr \, d\theta \]

\[ = \frac{4v}{\pi R^2} \left\{ \int_0^c \int_{r \cos(\theta) \geq D, r \leq R} r \, dr \, d\theta + \int_c^\alpha \int_{r \cos(\theta) \geq D, r \leq R} r \, dr \, d\theta \right\} \quad \text{(C.13)} \]

The second integration term in Eq. C.13 is zero since the condition \( r \cos(\theta) \geq D \) with \( \theta \) from the interval \([c, \alpha]\) is conflicting with the condition \( r \leq R \). \( F(\alpha) \) becomes in this case then,

\[ F(\alpha) = \frac{4v}{\pi R^2} \int_0^c \int_{D/\cos(\theta)}^R r \, dr \, d\theta = \frac{2v}{\pi R^2} (R^2 c - D^2 \tan(c)) \]

\[ = \frac{2v}{\pi} (c - \cos(c) \sin(c)) \quad \text{(C.14)} \]
c) $\alpha \in [\pi/2, \pi - c]$.

The function $H(r, \theta)$ becomes in this case as follows:

$$H(r, \theta) = \begin{cases} \frac{4}{\pi R^2} & \theta \in [0, \pi - \alpha], r \leq R \\ 0 & \text{elsewhere} \end{cases}$$  \hfill (C.15)

Filling this in Eq. C.7 yields,

$$F(\alpha) = \frac{4v}{\pi R^2} \int_0^{\pi - \alpha} \int_{r \cos(\theta) \geq D, r \leq R} r dr d\theta$$  \hfill (C.16)

By defining $\alpha' = \pi - \alpha$, we obtain

$$F(\alpha) = \frac{4v}{\pi R^2} \int_0^{\alpha'} \int_{r \cos(\theta) \geq D, r \leq R} r dr d\theta$$  \hfill (C.17)

Since $\alpha' \in [c, \pi/2]$, we have here thus a similar expression as that in Eq. C.13 with $\alpha'$ instead of $\alpha$. According to the result in Eq. C.14, we conclude then,

$$F(\alpha) = \frac{2v}{\pi} (c - \cos(c) \sin(c))$$  \hfill (C.18)

d) $\alpha \in [\pi - c, \pi]$.

The function $H(r, \theta)$ is represented in this case similar as the previous case by the expression in Eq. C.15. By defining $\alpha' = \pi - \alpha$, we obtain under Eq. C.8 the following:

$$F(\alpha) = \frac{4v}{\pi R^2} \int_0^{\alpha'} \int_{r \cos(\theta) \geq D, r \leq R} r dr d\theta$$  \hfill (C.19)

Since $\alpha' \in [0, c]$, the expression in Eq. C.19 above is thus similar as the first expression in Eq. C.12 with $\alpha'$ instead of $\alpha$. According to the calculation in Eq. C.12, we obtain therefore,

$$F(\alpha) = \frac{2v}{\pi} [\alpha' - \cos^2(c) \tan(\alpha') - \frac{2v}{\pi} [\pi - \alpha + \cos^2(c) \tan(\alpha)]]$$  \hfill (C.20)

Combining all above results, we finally conclude the following expression for $F(\alpha)$:
Appendix C Calculation of Continuous Filter Responses

\[ F(\alpha) = \frac{2v}{\pi} \begin{cases} 
\alpha - \cos^2(c) \tan(\alpha) & \alpha \in [0, c] \\
\c - \cos(c) \sin(c) & \alpha \in [c, \pi - c] \\
\pi - \alpha + \cos^2(c) \tan(\alpha) & \alpha \in [\pi - c, \pi] 
\end{cases} \] (C.21)

Since \( h_\alpha = -h_{\alpha - \pi} \) for \( \alpha \in [\pi, 2\pi] \), we have also \( F_\alpha = -F_{\alpha - \pi} \) for \( \alpha \in [\pi, 2\pi] \). Combining this with the above result yield precisely the same expressions as that in Eqs. 6.8–6.9. Hereby is our derivation completed.
Appendix D

Formal Justification of Propositions 6.1–2

Here, we will try to give a formal justification to the statements made in Propositions 6.1–2. Towards this aim, we need to reformulate these statements in a rigorously mathematical way and make use of some formal mathematical means. We start with exploring some necessary mathematical means.

D.1 Notations and Properties One

Definition D.1
1. \( n \in \mathbb{N}^+ \) and \( a, \sigma \in \mathbb{R}^+ \) are arbitrarily fixed numbers.
2. \( x_1, x_2, \ldots, x_n \) is an i.i.d. set of random variables, all having a \( \mathcal{N}(0, \sigma^2) \)-distribution.
3. \( y_1, y_2, \ldots, y_n \) is another set of random variables defined by
   \[
   y_i \overset{\text{def}}{=} \begin{cases} 
   x_i & \text{if } x_1 \geq a, x_2 \geq a, \ldots, x_n \geq a \\
   0 & \text{otherwise}
   \end{cases} \quad i = 1, 2, \ldots, n.
   \]
4. \( s = y_1 + y_2 + \cdots + y_n \).
5. \( p \overset{\text{def}}{=} \mathbb{P}\{x_i \geq a\} = \int_{a}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2} dx \) for \( i = 1, 2, \ldots, n \).
6. \( \Phi(s, a, \sigma) \overset{\text{def}}{=} \mathbb{P}\{x_1 + x_2 + \cdots + x_n \leq s, x_1 \geq a, x_2 \geq a, \ldots, x_n \geq a\} \) for all \( s, a, \sigma \in \mathbb{R}^+ \) satisfying \( s \geq na \).
7. \( \phi(s, a, \sigma) = \frac{\partial \Phi}{\partial s}(s, a, \sigma) \) for all \( s, a, \sigma \in \mathbb{R}^+ \) satisfying \( s \geq na \).
Lemma D.1

\[ f_{y_i}(y) = (1-p^n)\delta(y) + \frac{p^{n-1}}{\sqrt{2\pi\sigma}} e^{-y^2/2\sigma^2} I_{\{y \geq a\}} \quad \forall i=1,2,\ldots,n \quad (D.1) \]

where \( f_{y_i} \) is the probability density function of \( y_i \), \( I \) is the indicator function on \( \mathbb{R} \) and \( \delta \) is the Dirac function.

**Proof.** Let \( i \in \{1,2,\ldots,n\} \) be an arbitrarily fixed number. From the given definition of \( y_i \) we can derive its probabilities as follows:

\[ P\{y_i < 0\} = 0 \quad \text{and} \quad P\{y_i \in (0,a)\} = 0 \quad (D.2) \]

\[ P\{y_i = 0\} = 1 - P\{y_i \geq a\} = 1 - P\{x_1 \geq a, x_2 \geq a, \ldots, x_n \geq a\} \]
\[ = 1 - \prod_{j=1}^{n} P\{x_j \geq a\} = 1 - p^n \quad (D.3) \]

\[ P\{y_i \in [a,y]\} = P\{x_i \in [a,y], x_1 \geq a, x_2 \geq a, \ldots, x_n \geq a\} \]
\[ = P\{x_i \in [a,y], x_j \geq a, j = 1,2,\ldots,n, j \neq i\} \]
\[ = p^{n-1} \int_{a}^{y} \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} dx \quad y \geq a \]

Taking the derivative of the above expression and combining with the previous results yields Eq. D.1.

Using the the above lemma we can further obtain the following results:

\[ E\{y_i\} = \int_{-\infty}^{\infty} f_{y_i}(y) dy = p^{n-1} \int_{a}^{\infty} \frac{y}{\sqrt{2\pi\sigma}} e^{-y^2/2\sigma^2} dy \]
\[ = \frac{p^{n-1}\sigma}{\sqrt{2\pi}} e^{-a^2/2\sigma^2} \quad i = 1,2,\ldots,n \quad (D.4) \]

\[ E\{s\} = \sum_{i=1}^{n} E\{y_i\} = \frac{np^{n-1}\sigma}{\sqrt{2\pi}} e^{-a^2/2\sigma^2} \quad (D.5) \]
Lemma D.2

\[ \int_{-\infty}^{\infty} s \phi(s, a, \sigma) \, ds = \frac{np^{n-1} \sigma}{\sqrt{2\pi}} e^{-a^2/2 \sigma^2} \quad \forall a, \sigma \in \mathbb{R}^+ \] (D.6)

Proof. Let \( a, \sigma \in \mathbb{R}^+ \) be arbitrarily fixed numbers. From the given definition of \( s \), we can derive its probabilities as follows:

\[ P\{s < 0\} = 0 \quad \text{and} \quad P\{s \in (0, na)\} = 0 \] (D.7)

\[ P\{s = 0\} = 1 - P\{s \geq na\} = 1 - P\{y_i \geq a, i = 1, 2, \ldots, n\} \\
= 1 - P\{x_i \geq a, i = 1, 2, \ldots, n\} = 1 - \prod_{i=1}^{n} P\{x_i \geq a\} = 1 - p^n \] (D.8)

\[ P\{s \in [na, s]\} = P\{y_1 + \cdots + y_n \leq s, y_1 \geq a, \ldots, y_n \geq a\} \\
= P\{x_1 + \cdots + x_n \leq s, x_1 \geq a, \ldots, x_n \geq a\} = \Phi(s, a, \sigma) \quad \forall \ s \geq na \] (D.9)

Combining the above results yields the probability density function of \( s \) as follows:

\[ f_s(s) = (1 - p^n) \delta(s) + \phi(s, a, \sigma) I_{s \geq na} \] (D.10)

Thus, the expectation of \( s \) becomes

\[ E\{s\} = \int_{-\infty}^{\infty} s f_s(s) \, ds = \int_{na}^{\infty} s \phi(s, a, \sigma) \, ds \] (D.11)

Since \( E\{s\} \) was also given in Eq. D.5, we conclude therefore,

\[ \int_{na}^{\infty} s \phi(s, a, \sigma) \, ds = \frac{np^{n-1} \sigma}{\sqrt{2\pi}} e^{-a^2/2 \sigma^2} \] (D.12)

which completes the proof.

\[ \# \]
D.2 Notations and Properties Two

Definition D.2

1. \( n, k \in \mathbb{N} \) and \( \mu, a, \sigma \in \mathbb{R}^+ \) are arbitrarily fixed numbers with \( k \leq n \) and \( n \geq 2 \).

2. \( \{x_i^{(k)} : i = 1, 2, \ldots, n\} \) is a set of independently distributed random variables, where \( x_i^{(k)} \) has a \( N(\mu, \sigma^2) \)-distribution for \( i = 1, 2, \ldots, k \) and a \( N(-\mu, \sigma^2) \)-distribution for \( i = k+1, k+2, \ldots, n \).

3. \( \{y_i^{(k)} : i = 1, 2, \ldots, n\} \) is another set of random variables defined by

\[
y_i^{(k)} = \begin{cases} 
  x_i^{(k)} & \text{if } x_i^{(k)} \geq a, x_2^{(k)} \geq a, \ldots, x_n^{(k)} \geq a, \\
  0 & \text{otherwise}
\end{cases} \quad i = 1, 2, \ldots, n.
\]

4. \( s = y_1^{(k)} + y_2^{(k)} + \ldots + y_n^{(k)} \).

5. \( p \overset{\text{def}}{=} \mathbb{P}\{x_i^{(k)} \geq a\} = \int_a^\infty \frac{1}{\sqrt{2\pi \sigma}} e^{-(x-\mu)^2/2\sigma^2} \, dx \) for \( i = 1, 2, \ldots, k \).

6. \( q \overset{\text{def}}{=} \mathbb{P}\{x_i^{(k)} \geq a\} = \int_a^\infty \frac{1}{\sqrt{2\pi \sigma}} e^{-(x+\mu)^2/2\sigma^2} \, dx \) for \( i = k+1, k+2, \ldots, n \).

7. \( \Phi_k(s, a, \mu, \sigma) \overset{\text{def}}{=} \mathbb{P}\{x_1^{(k)} + x_2^{(k)} + \ldots + x_n^{(k)} \leq s, x_1^{(k)} \geq a, x_2^{(k)} \geq a, \ldots, x_n^{(k)} \geq a\} \) for all \( s, a, \mu, \sigma \in \mathbb{R}^+ \) satisfying \( s \geq na \).

8. \( \phi_k(s, a, \mu, \sigma) = \frac{\partial \Phi_k}{\partial s}(s, a, \mu, \sigma) \) for all \( s, a, \mu, \sigma \in \mathbb{R}^+ \) satisfying \( s \geq na \).

Lemma D.3

\[
f_{y_i^{(k)}}(y) = (1 - p^k q^{n-k}) \delta(y) + \begin{cases} 
  \frac{p^{k-1} q^{n-k}}{\sqrt{2\pi \sigma}} e^{-(y-\mu)^2/2\sigma^2} I_{y \geq a} & y \geq a, i = 1, \ldots, k \\
  \frac{p^k q^{n-1-k}}{\sqrt{2\pi \sigma}} e^{-(y+\mu)^2/2\sigma^2} I_{y \geq a} & y \geq a, i = k+1, \ldots, n
\end{cases}
\]  \hspace{1cm} (D.13)

where \( f_{y_i^{(k)}} \) is the probability density function of \( y_i^{(k)} \).

Proof. Let \( i \in \{1, 2, \ldots, n\} \) be an arbitrarily fixed number. From the given definition of \( y_i^{(k)} \) we can derive its probabilities as follows:

\[
\mathbb{P}\{y_i^{(k)} < 0\} = 0 \quad \text{and} \quad \mathbb{P}\{y_i^{(k)} \in (0, a)\} = 0 \hspace{1cm} (D.14)
\]

\[
\mathbb{P}\{y_i^{(k)} = 0\} = 1 - \mathbb{P}\{y_i^{(k)} \geq a\} = 1 - \mathbb{P}\{x_j^{(k)} \geq a, j = 1, 2, \ldots, n\}
\]
D.2 Notations and Properties Two

\[ 1 - \prod_{j=1}^{n} P\{x_j^{(k)} \geq a\} = 1 - p^k q^{n-k} \] (D.15)

\[ P\{y_i^{(k)} \in [a, y]\} = P\{x_i^{(k)} \in [a, y], x_j^{(k)} \geq a, j = 1, 2, \ldots, n\} \]
\[ = P\{x_i^{(k)} \in [a, y], x_j^{(k)} \geq a, j = 1, 2, \ldots, n, j \neq i\} \]
\[ = \begin{cases} 
  p^{k-1} q^{n-k} \int_a^{y} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx & y \geq a, i = 1, \ldots, k \\
  p^k q^{n-1-k} \int_a^{y} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x+\mu)^2}{2\sigma^2}} dx & y \geq a, i = k+1, \ldots, n
\end{cases} \]

Taking the derivative of the above expression and combining with the previous results yields Eq. D.13.

By applying the above lemma, we can further obtain,

\[ E\{y_i^{(k)}\} = \begin{cases} 
  p^{k-1} q^{n-k} \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(a-\mu)^2}{2\sigma^2}} + \mu p \right) & i = 1, 2, \ldots, k \\
  p^k q^{n-1-k} \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(a+\mu)^2}{2\sigma^2}} + \mu q \right) & i = k+1, \ldots, n
\end{cases} \] (D.16)

\[ E\{s\} = \sum_{i=1}^{n} E\{y_i^{(k)}\} = kp^{k-1} q^{n-k} \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(a-\mu)^2}{2\sigma^2}} + \mu p \right) \]
\[ + (n-k)p^k q^{n-1-k} \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(a+\mu)^2}{2\sigma^2}} + \mu q \right) \] (D.17)

Lemma D.4

\[ \int_{na}^{\infty} s \phi_k(s, a, \mu, \sigma) ds = kp^{k-1} q^{n-k} \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(a-\mu)^2}{2\sigma^2}} + \mu p \right) \]
\[ + (n-k)p^k q^{n-1-k} \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(a+\mu)^2}{2\sigma^2}} + \mu q \right) \quad \forall \ a, \mu, \sigma \in \mathbb{R}^+ \] (D.18)

Proof. Let \( a, \mu, \sigma \in \mathbb{R}^+ \) be arbitrarily fixed numbers. From the given definition of \( s \), we can derive its probabilities as follows:

\[ P\{s < 0\} = 0 \quad \text{and} \quad P\{s \in (0, na)\} = 0 \] (D.19)
\[ P\{s=0\} = 1 - P\{s \geq na\} = 1 - P\{y^{(k)}_i \geq a, i = 1, 2, \ldots, n\} \]
\[ = 1 - P\{x^{(k)}_i \geq a, i = 1, 2, \ldots, n\} \]
\[ = 1 - \prod_{i=1}^{n} P\{x^{(k)}_i \geq a\} = 1 - p^k q^{n-k} \quad (D.20) \]

\[ P\{s \leq na, s\} = P\{y^{(k)}_1 + \ldots + y^{(k)}_n \leq s, y^{(k)}_1 \geq a, \ldots, y^{(k)}_n \geq a\} \]
\[ = P\{x^{(k)}_1 + \ldots + x^{(k)}_n \leq s, x^{(k)}_1 \geq a, \ldots, x^{(k)}_n \geq a\} \]
\[ = \Phi_k(s, a, \mu, \sigma) \quad \forall s \geq na \quad (D.21) \]

Combining the above results yields the probability density function of \( s \) as follows:

\[ f_s(s) = (1 - p^k q^{n-k}) \delta(s) + \phi_k(s, a, \mu, \sigma) I_{\{s \geq na\}} \quad (D.22) \]

Thus, the expectation of \( s \) becomes

\[ E\{s\} = \int_{-\infty}^{\infty} sf_s(s)ds = \int_{na}^{\infty} s\phi_k(s, a, \mu, \sigma)ds \quad (D.23) \]

Since \( E\{s\} \) was already given in Eq. D.17, we conclude therefore,

\[ \int_{na}^{\infty} s\phi_k(s, a, \mu, \sigma)ds = kp^{k-1}q^{n-k}\left(\frac{\sigma}{\sqrt{2\pi}}e^{-\frac{(a-\mu)^2}{2\sigma^2}} + \mu p\right) + \\
(n-k)p^{k}q^{n-1-k}\left(\frac{\sigma}{\sqrt{2\pi}}e^{-\frac{(a+\mu)^2}{2\sigma^2}} + \mu q \right) \quad (D.24) \]

which completes the proof. 

#
D.3 Definition and Properties of White Noise

Definition D.3 Two-Dimensional White Noise

Two-dimensional white noise \( \{g(x, y) : (x, y) \in \mathbb{R}^2 \} \) with time-set \( \mathbb{R}^2 \) and variance parameter \( \sigma^2 \) is a generalized random field which is physically only measurable through a specified set \( \mathcal{T} \) of test functions as follows:

\[
g(t) = \iint_{\mathbb{R}^2} t(x, y)g(x, y)\,dx\,dy \quad \forall \, t \in \mathcal{T}
\]  
(D.25)

where \( \mathcal{T} \) should also be a subspace of the usual Hilbert function space \( L^2(\mathbb{R}^2) \) defined as follows:

\[
\langle t_1, t_2 \rangle_{L^2} = \iint_{\mathbb{R}^2} t_1(x, y)t_2(x, y)\,dx\,dy \quad \forall t_1, t_2 \in \mathcal{T}
\]  
(D.26)

Moreover, \( \{g(t) : t \in \mathcal{T} \} \) should possess the following properties:

1. \( g(t) \) has a centralized Gaussian distribution for each \( t \in \mathcal{T} \).
2. \( \mathbb{E}\{g(t_1)g(t_2)\} = \sigma^2 \langle t_1, t_2 \rangle_{L^2} \) for every pair of \( t_1, t_2 \in \mathcal{T} \).

In particular if \( \sigma^2 = 1 \), then the random field \( \{g(x, y) : (x, y) \in \mathbb{R}^2 \} \) is called the standard two-dimensional white noise.

Let \( n \in \mathbb{N}^+ \) and \( R \in \mathbb{R}^+ \) be arbitrarily fixed numbers. A set of mutually orthogonal indicator functions on \( \mathbb{R}^2 \) is defined in the polar coordinate system as follows:

\[
I_i(r, \theta) = \begin{cases} 
1 & \theta \in [(i-1)\pi/n, i\pi/n), \, r \leq R \\
0 & \text{elsewhere}
\end{cases} \quad i = 1, \ldots, 2n 
\]  
(D.27)

As a notational convention we specify the following:

\[
I_{i+2nk} = I_i \quad \forall \, i = 1, 2, \ldots, 2n, \, k \in \mathbb{Z}
\]  
(D.28)

Subsequently, a set of random variables is defined as follows:

\[
\nu_i = g(I_i - I_{i+n}) \quad i \in \mathbb{Z}
\]  
(D.29)
Let $\mu \in \mathbb{R}^+$ and $i \in \{1, 2, \ldots, n\}$ be arbitrarily fixed numbers. We define another set of random variables as follows:

$$
\begin{align*}
\nu_{j}^{(i)} &= \begin{cases} 
\pi R^2 \mu / 2n + \nu_{j+i-1} & j = 1, 2, \ldots, n-i+1 \\
-\pi R^2 \mu / 2n + \nu_{j+i-1} & j = n-i+2, n-i+3, \ldots, n
\end{cases}
\end{align*}
$$

(D.30)

Also, as notational conventions we specify the following:

$$
\nu_{j}^{(i+n)} = -\nu_{j}^{(i)} \forall j, i = 1, \ldots, n 
$$

(D.31)

$$
\nu_{j}^{(i+2nk)} = \nu_{j}^{(i)} \forall j = 1, \ldots, n, i = 1, 2, \ldots, 2n, k \in \mathbb{Z} 
$$

(D.32)

**Lemma D.5**

1. $\nu_{i+2nk} = \nu_i$ and $\nu_{i+n+2nk} = -\nu_i$ for all $i, k \in \mathbb{Z}$.

2. For each $k \in \mathbb{Z}$, $\{\nu_i : i = k, k+1, \ldots, k+n-1\}$ is an i.i.d. set of random variables, all having a $\mathcal{N}(0, \pi R^2 \sigma^2 / n)$-distribution.

3. For each $i \in \{1, 2, \ldots, n\}$, $\{\nu_j^{(i)} : j = 1, 2, \ldots, n\}$ is a set of independent random variables. Moreover, $\nu_j^{(i)}$ has a $\mathcal{N}(\pi R^2 \mu / 2n, \pi R^2 \sigma^2 / n)$-distribution for $j = 1, 2, \ldots, n-i+1$ and a $\mathcal{N}(-\pi R^2 \mu / 2n, \pi R^2 \sigma^2 / n)$-distribution for $j = n-i+2, n-i+3, \ldots, n$.

4. For each $i \in \{n+1, n+2, \ldots, 2n\}$, $\{\nu_j^{(i)} : j = 1, 2, \ldots, n\}$ is a set of independent random variables. Moreover, $\nu_j^{(i)}$ has a $\mathcal{N}(-\pi R^2 \mu / 2n, \pi R^2 \sigma^2 / n)$-distribution for $j = 1, 2, \ldots, n-i+1$ and a $\mathcal{N}(\pi R^2 \mu / 2n, \pi R^2 \sigma^2 / n)$-distribution for $j = n-i+2, n-i+3, \ldots, n$.

5. For each $i \in \mathbb{Z}$, $\{\nu_1^{(i)} : j = i, i+1, \ldots, i+n-1\}$ is a set of independent random variables.

6. $\nu_1^{(i)}$ has a $\mathcal{N}(\pi R^2 \mu / 2n, \pi R^2 \sigma^2 / n)$-distribution for $i = 1, 2, \ldots, n$ and a $\mathcal{N}(-\pi R^2 \mu / 2n, \pi R^2 \sigma^2 / n)$-distribution for $i = n+1, n+2, \ldots, 2n$.

7. For each $i \in \mathbb{Z}$, $\{\nu_1^{(i)} : j = i, i+1, \ldots, i+n-1\}$ represents the same set of random variables as that represented by $\{\nu_j^{(i)} : j = 1, 2, \ldots, n\}$.

The proof of this lemma will be omitted because it involves only some boring finger-exercises rather than essentially complex manipulations.
D.4 Reformulation and Justification

Now, we come to the stage to reformulate the statements made in Propositions 6.1–2 in a rigorously mathematical way. We start with the assumption that the set of rotationally equivalent filters is limited to \( \{h_\alpha : \alpha = (i - 1)\pi/n, i = 1, \ldots, 2n\} \), where \( n \in \mathbb{N}^+ \) is an arbitrarily fixed number. Below is a notational convention under this assumption:

\[
\begin{align*}
    &h_i = h_{(i-1)\pi/n} & i = 1, \ldots, 2n & (D.33) \\
    &h_{i+2nk} = h_i & i = 1, 2, \ldots, 2n, k \in \mathbb{Z} & (D.34)
\end{align*}
\]

It is easy to see that \( h_i \) possesses the following property:

\[
\begin{align*}
    &h_i = 2 \left( \sum_{j=i}^{n-1+i} I_j - \sum_{j=n+i}^{2n-1+i} I_j \right) / \pi R^2 = 2 \sum_{j=i}^{n-1+i} (I_j - I_{j+n}) / \pi R^2 & \forall i \in \mathbb{Z} & (D.35) \\
    &\int_{-\infty}^{\infty} h_i(x, y) dx dy = 0 & \forall i \in \mathbb{Z} & (D.36) \\
    &h_i = -h_{i+n} & \forall i \in \mathbb{Z} & (D.37)
\end{align*}
\]

The image function \( f(x, y) \) is considered as the sum of an ideal step edge and an additive white noise \( g(x, y) \) with variance parameter \( \sigma^2 \). Without loss of generality, we can simply assume that it is represented as follows:

\[
\begin{align*}
    f(x, y) = \begin{cases} 
    \mu + g(x, y) & y \geq d \\
    g(x, y) & y < d
\end{cases} & (D.38)
\end{align*}
\]

where \( d \) is the distance from the filter center to the edge and \( \mu \in \mathbb{R}^+ \) is the size of the edge step.

Let \( F_i \) denote the filter response obtained by applying the filter \( h_i \) for any \( i \in \mathbb{Z} \). We now try formally to define the property of unimodality involving \( \{F_i : i = 1, 2, \ldots, 2n\} \) as follows.
Definition D.4  Unimodality

\{F_i : i = 1, 2, \ldots, 2n\} is said to possess the property of unimodality if and only if the following condition is satisfied:

\[ F_i = \max\{F_1, \ldots, F_{2n}\} \implies F_j - F_{j+1} \geq \epsilon \quad \forall \quad j = i, \ldots, i+n-1 \quad (D.39) \]

where \(\epsilon \in \mathbb{R}\) is a prescribed non-negative number and \(F_j\) is considered to be \(F_{j-2n}\) if \(j > 2n\).

According to our definition of edgeness, the edgeness \(e\) of an arbitrary point can be expressed as follows:

\[
e = \begin{cases} 
\max\{F_1, F_2, \ldots, F_{2n}\} & \text{if unimodality test passed} \\
0 & \text{otherwise} \end{cases} \quad (D.40)
\]

Below are some notational conventions, which will be assumed throughout the subsequent discussions:

1. \(G(x) = \int_{-\infty}^{\infty} e^{-t^2/2} \sqrt{2\pi} dt \) for all \(x \in [0, \infty)\).
2. \(a = \pi R^2 \epsilon / 4\), \(\tilde{\sigma}^2 = \pi R^2 \sigma^2 / n\) and \(\tilde{\mu} = \pi R^2 \mu / 2n\).
3. \(p = G[(n\epsilon - 2\mu) R \sqrt{\pi} / 4\sigma \sqrt{n}] \) and \(q = G[(n\epsilon + 2\mu) R \sqrt{\pi} / 4\sigma \sqrt{n}] \).
4. \(K_p = \frac{\tilde{\sigma}}{\sqrt{2\pi}} e^{-(a-\tilde{\mu})^2/2\tilde{\sigma}^2} + \tilde{\mu} \) and \(K_q = \frac{\tilde{\sigma}}{\sqrt{2\pi}} e^{-(a+\tilde{\mu})^2/2\tilde{\sigma}^2} + \tilde{\mu} \).

D.4.1 Probability and Expectation at a Non-Edge Location

Here, we consider the case in which the filtering center does not involve an edge presence at all. This corresponds to \(|d| \geq R\). For this case, the filter response \(F_i\) can be derived in the following way:

\[
F_i = \int \int \mathbb{R}^2 f(x, y) h_i(x, y) dx dy = \int \int \mathbb{R}^2 g(x, y) h_i(x, y) dx dy
\]

\[
= \int \int \mathbb{R}^2 2 \sum_{j=i}^{n-1+i} (I_i(x, y) - I_{j+n}(x, y)) g(x, y) / \pi R^2 dx dy
\]
D.4 Reformulation and Justification

\[
\begin{align*}
&= \frac{2}{\pi R^2} \sum_{j=i}^{n-1+i} \int \int_{\mathbb{R}^2} \{I_j(x,y) - I_{j+n}(x,y)\}g(x,y)\,dx\,dy \\
&= \frac{2}{\pi R^2} \sum_{j=i}^{n-1+i} \nu_j \quad i \in \mathbb{Z} \\
\end{align*}
\]

Thus, we can further conclude,

\[
F_i - F_{i+1} = 2(\nu_i - \nu_{i+n})/\pi R^2 = 4\nu_i/\pi R^2 \quad i \in \mathbb{Z} \\
\]

Theorem D.1

\[
P\{e = 0\} = 1 - 2nG^n\left(\frac{R\varepsilon \sqrt{n\pi}}{4\sigma}\right) \quad \forall R, \varepsilon, \sigma \in \mathbb{R}^+, n \geq 2. \\
\]

Proof. Let $R, \varepsilon, \sigma \in \mathbb{R}^+$ and $n \geq 2$ be arbitrarily fixed numbers. According to the definition of $e$, we have

\[
P\{e = 0\} = 1 - \sum_{i=1}^{2n} P\{unimodality, F_i is the maximum\} \\
= 1 - 2nP\{unimodality, F_1 is the maximum\} \\
= 1 - 2nP\{F_j - F_{j+1} \geq \varepsilon, j = 1, \ldots, n\} \\
= 1 - 2n\prod_{j=1}^{n} P\{v_j \geq \pi R^2 \varepsilon / 4, j = 1, \ldots, n\} \\
= 1 - 2n \prod_{j=1}^{n} P\{v_j \geq \pi R^2 \varepsilon / 4\} = 1 - 2nG^n\left(\frac{R\varepsilon \sqrt{n\pi}}{4\sigma}\right) \\
\]

Theorem D.2

\[
f_e(m) = (1 - 2nG^n\left(\frac{R\varepsilon \sqrt{n\pi}}{4\sigma}\right))\delta(m) + n\pi R^2 \phi(\pi R^2 m / 2, a, \tilde{\sigma})I_{\{m \geq n\varepsilon / 2\}} \\
\]

where $f_e$ is the probability density function of $e$.

Proof. Let $m \in \mathbb{R}^+$ be an arbitrarily fixed number.
\[ P \{ e \in (0, m] \} = P \{ \text{unimodality, } \max \{ F_1, F_2, \ldots, F_{2n} \} \leq m \} \]
\[ = \sum_{i=1}^{2n} P \{ \text{unimodality, } F_i \leq m, F_i \text{ is the maximum} \} \]
\[ = 2n P \{ \text{unimodality, } F_i \leq m, F_i \text{ is the maximum} \} \]
\[ = 2n P \{ F_i \leq m, F_j - F_{j+1} \geq \varepsilon, j = 1, 2, \ldots, n \} \]
\[ = 2n P \{ v_1 + \cdots + v_n \leq \pi R^2 m/2, v_j \geq \pi R^2 \varepsilon / 4, j = 1, \ldots, n \} \]
\[ = \begin{cases} \frac{2n \Phi(\pi R^2 m/2, a, \bar{\sigma})}{\sigma} & m \geq n \varepsilon / 2 \\ 0 & \text{otherwise} \end{cases} \] (D.46)

By taking the derivative of the above result and combining with the expression in Eq. D.43, we obtain thus the desired expression for \( f_e \) in Eq. D.45.

**Theorem D.3**

\[ E \{ e \} = \frac{4n^2 \bar{\sigma} G^{n-1} (R \varepsilon / \sqrt{n \pi / 4 \sigma})}{\pi R^2 \sqrt{2 \pi}} e^{-a^2 / 2 \bar{\sigma}^2} \] (D.47)

**Proof.** According to Eq. D.45, we have

\[ E \{ e \} = \int_{-\infty}^{\infty} m f_e(m) dm = \int_{-\infty}^{\infty} mn \pi R^2 \phi(\pi R^2 m/2, a, \bar{\sigma}) dm \]
\[ = \frac{4n}{\pi R^2} \int_{\pi \varepsilon}^{\infty} s \phi(s, a, \bar{\sigma}) ds \] (D.48)

Applying the equivalence relation in Lemma D.2 immediately yields the expression in Eq. D.47 for \( E \{ e \} \).

**D.4.2 Probability and Expectation at an Edge Location**

Here, we consider the case in which the filtering center lies precisely on the edge segment. In other words, \( d = 0 \). For this case, the filter response \( F_i \) can be derived in the following way:
\[ F_i = \int \int_{\mathbb{R}^2} h_i(x, y) f(x, y) \, dx \, dy \]

\[ = \frac{2}{\pi R^2} \sum_{j=i}^{n-1+i} \int \int_{\mathbb{R}^2} I_j(x, y) f(x, y) \, dx \, dy \]

\[ - \frac{2}{\pi R^2} \sum_{j=n+i}^{2n-1+i} \int \int_{\mathbb{R}^2} I_j(x, y) f(x, y) \, dx \, dy \]

\[ = \frac{2}{\pi R^2} \left\{ \sum_{j=i}^{n} \int \int_{\mathbb{R}^2} I_j(x, y) (\mu + g(x, y)) \, dx \, dy \right\} \]

\[ + \sum_{j=n+i}^{n-1+i} \int \int_{\mathbb{R}^2} I_j(x, y) g(x, y) \, dx \, dy \]

\[ - \frac{2}{\pi R^2} \left\{ \sum_{j=n+i}^{2n} \int \int_{\mathbb{R}^2} I_j(x, y) g(x, y) \, dx \, dy \right\} \]

\[ + \sum_{j=2n+i}^{2n-1+i} \int \int_{\mathbb{R}^2} I_j(x, y) (\mu + g(x, y)) \, dx \, dy \]

\[ = \frac{2}{\pi R^2} \cdot \frac{(n-2i+2)\pi R^2 \mu}{2n} \]

\[ + \frac{2}{\pi R^2} \sum_{j=i}^{n+i} \int \int_{\mathbb{R}^2} (I_j(x, y) - I_{j+n}(x, y)) g(x, y) \, dx \, dy \]

\[ = \frac{2}{\pi R^2} \left\{ \frac{(n-2i+2)\pi R^2 \mu}{2n} + \sum_{j=i}^{n-1+i} v_j \right\} \]

\[ = \frac{2}{\pi R^2} \sum_{j=1}^{n} v_j^{(i)} \quad \forall \ i = 1, 2, \ldots, n \quad (D.49) \]

Similarly, we can show the following properties:

\[ F_i = \frac{2}{\pi R^2} \sum_{j=1}^{n} v_j^{(i)} \quad \forall \ i = n+1, n+2, \ldots, 2n \quad (D.50) \]

\[ F_i - F_{i+1} = \frac{4}{\pi R^2} v_1^{(i)} \quad \forall \ i \in \mathbb{Z} \quad (D.51) \]
Theorem D.4

$$P\{e > 0\} = \frac{(p+q)(p^n-q^n)}{p-q}. \quad \text{(D.52)}$$

Proof. According to the definition of e, we have

$$P\{e > 0\} = P\{\text{unimodality}\} = \sum_{i=1}^{2n} P\{\text{unimodality, } F_i \text{ is the maximum}\}$$

$$= \sum_{i=1}^{2n} P\{F_i - F_{i+1} \geq \epsilon, j = i, \ldots, i+n-1\}$$

$$= \sum_{i=1}^{2n} P\{v^{(i)}_j \geq \pi R^2\epsilon/4, j = i, \ldots, i+n-1\}$$

$$= \sum_{i=1}^{2n} P\{v^{(i)}_j \geq \pi R^2\epsilon/4, j = 1, 2, \ldots, n\}$$

$$= \sum_{i=1}^{2n} \prod_{i=1}^{n} P\{v^{(i)}_j \geq \pi R^2\epsilon/4\} = \sum_{i=1}^{n} \left\{ \prod_{i=1}^{n} P\{v^{(i)}_j \geq a\} + \prod_{j=1}^{n} P\{-v^{(i)}_j \geq a\} \right\}$$

$$= \sum_{i=1}^{n} \left\{ p^{n-i+1}q^{i-1} + q^{n-i+1}p^{i-1} \right\} = \frac{(p+q)(p^n-q^n)}{p-q} \quad \text{(D.53)}$$

Theorem D.5

$$f_e(m) = (1 - \frac{(p+q)(p^n-q^n)}{p-q})\delta(m) + \frac{\pi R^2}{2} \sum_{i=1}^{n} \{\phi_i(\pi R^2m/2, a, \mu, \sigma)$$

$$+ \phi_{n-i}(\pi R^2m/2, a, \mu, \sigma)\} I_{m \geq n\epsilon/2} \quad \text{(D.54)}$$

where $f_e$ is the probability density function of e.

Proof. Let $m \in \mathbb{R}^+$ be an arbitrary number.

$$P\{e \in (0, m]\} = P\{\text{unimodality, } \max\{F_1, F_2, \ldots, F_{2n}\} \leq m\}$$
\[\sum_{i=1}^{2n} P\{unimodality, F_i \leq m, F_i \text{ is the maximum}\}\]

\[= \sum_{i=1}^{2n} P\{F_i \leq m, F_j - F_{j+1} \geq \varepsilon, j = i, i+1, \ldots, i+n-1\}\]

\[= \sum_{i=1}^{2n} P\{v_1^{(i)} + \cdots + v_n^{(i)} \leq \pi R^2 m/2, v_j^{(i)} \geq \pi R^2 \varepsilon/4, j = i, \ldots, i+n-1\}\]

\[= \sum_{i=1}^{2n} P\{v_1^{(i)} + \cdots + v_n^{(i)} \leq \pi R^2 m/2, v_j^{(i)} \geq a, j = 1, 2, \ldots, n\}\]

\[= \sum_{i=1}^{n} P\{v_1^{(i)} + \cdots + v_n^{(i)} \leq \pi R^2 m/2, v_j^{(i)} \geq a, j = 1, 2, \ldots, n\}\]

\[+ \sum_{i=1}^{n} P\{-v_1^{(i)} - \cdots - v_n^{(i)} \leq \pi R^2 m/2, -v_j^{(i)} \geq a, j = 1, 2, \ldots, n\}\]

\[= \left\{ \begin{array}{ll}
\sum_{i=1}^{n} \{\Phi_i(\pi R^2 m/2, a, \bar{\mu}, \bar{\sigma}) + \Phi_{n-i}(\pi R^2 m/2, a, \bar{\mu}, \bar{\sigma})\} & \text{if } m \geq n\varepsilon/2 \\
0 & \text{otherwise}
\end{array} \right.\]

By taking the derivative of the above result and combining with the expression in Eq. D.52, we obtain thus the desired expression for \(f_e\) in Eq. D.54.

# Theorem D.6

\[\textbf{E}\{e\} = \frac{2}{\pi R^2} \left\{ K_p \frac{np^{n+1} - 2p^n q + 2q^{n+1} - np^{n-1} q^2}{(p-q)^2} + K_q \frac{nq^{n+1} - 2q^n p + 2p^{n+1} - nq^{n-1} p^2}{(q-p)^2} \right\} \]  \hspace{1cm} (D.55)

\textit{proof.} Let \(m \in \mathbb{R}^+\) be an arbitrarily fixed number.

\[\textbf{E}\{e\} = \int_{-\infty}^{\infty} m f_e(m) dm \]

\[= \int_{n\varepsilon/2}^{\infty} \frac{m \pi R^2}{2} \sum_{i=1}^{n} \{\Phi_i(\pi R^2 m/2, a, \bar{\mu}, \bar{\sigma}) + \Phi_{n-i}(\pi R^2 m/2, a, \bar{\mu}, \bar{\sigma})\} dm \]

\[= \frac{2}{\pi R^2} \int_{na}^{\infty} s \sum_{i=1}^{n} \{\Phi_i(s, a, \bar{\mu}, \bar{\sigma}) + \Phi_{n-i}(s, a, \bar{\mu}, \bar{\sigma})\} ds\]
\[
\begin{align*}
E(e) &= \frac{2}{\pi R^2} \sum_{i=1}^{n} \left\{ \int_{na}^{\infty} s \phi_i(s, a, \mu, \sigma) ds + \int_{na}^{\infty} s \phi_{n-i}(s, a, \mu, \sigma) ds \right\} \quad (D.56) \\
&= \frac{2}{\pi R^2} \left\{ \sum_{i=1}^{n} ip^{i-1}q^{n-i}K_p + (n-i)p^i q^{n-1-i}K_q \right\} \\
&\quad + (n-i)p^{n-i-1}q^iK_p + ip^{n-i}q^{i-1}K_q \right\} \\
&= \frac{2}{\pi R^2} \left\{ \sum_{i=1}^{n} ip^{i-1}q^{n-i}K_p + \sum_{i=0}^{n-1} ip^{n-i}q^{i-1}K_q \right\} \\
&\quad + \sum_{i=0}^{n-1} ip^{i-1}q^{n-i}K_p + \sum_{i=1}^{n} ip^{n-i}q^{i-1}K_q \right\} \\
&= \frac{2}{\pi R^2} \left\{ 2 \sum_{i=1}^{n} ip^{i-1}q^{n-i}K_p - np^{n-1}K_p \right\} \\
&\quad + \sum_{i=1}^{n} ip^{n-i}q^{i-1}K_q - nq^{n-1}K_q \right\} \\
&= \frac{2}{\pi R^2} \left\{ 2q^{n-1}K_p \sum_{i=1}^{n} i(p/q)^{i-1} - np^{n-1}K_p \right\} \\
&\quad + 2p^{n-1}K_q \sum_{i=1}^{n} i(q/p)^{i-1} - nq^{n-1}K_q \right\} \quad (D.57) \\

\text{Since } \sum_{i=1}^{n} ix^{i-1} = \frac{nx^{n+1} - (n+1)x^{n+1}}{(x-1)^2} \text{ for all } x \in \mathbb{R} \text{ with } x \neq 1, \text{ we obtain then,}
\end{align*}
\]
\[
E(e) = \frac{2}{\pi R^2} \left\{ 2q^{n-1}K_p \frac{n(p/q)^{n+1} - (n+1)(p/q)^{n+1}}{(p/q-1)^2} - np^{n-1}K_p \right\}
\]
\[
+ 2p^{n-1}K_q \frac{n(q/p)^{n+1} - (n+1)(q/p)^{n+1}}{(q/p-1)^2} - nq^{n-1}K_q \right\}
\]
\[
= \frac{2}{\pi R^2} \left\{ 2K_p \frac{np^{n+1} - (n+1)p^n q + q^{n+1}}{(p-q)^2} - np^{n-1}K_p \right\}
\]
\[
+ 2K_q \frac{np^{n+1} - (n+1)q^n p + p^{n+1}}{(q-p)^2} - nq^{n-1}K_q \right\}
\]
\[
= \frac{2}{\pi R^2} \left\{ K_p \frac{np^{n+1} - 2p^n q + 2q^{n+1} - np^{n-1}q^2}{(p-q)^2} \right\}
\]
D.4 Reformulation and Justification

\[ + K_q \frac{nq^{n+1} - 2q^n p + 2p^{n+1} - nq^{n-1} p^2}{(q-p)^2} \]  \hspace{1cm} (D.58)

D.4.3 Conclusions

In the previous sections, we derived the expressions for the probability density function and the expectation of the edgeness \( e \) either not on an edge segment at all or precisely on an edge segment. To make use of those results towards a rigorous justification for the statements made in Propositions 6.1–2, we need to reformulate those statements in a similarly rigorous way. According to the previous presentation, Propositions 6.1–2 can be strictly reformulated as follows:

**Reformulation of Propositions 6.1–2**

Under reasonable choices for the parameters \( n, \varepsilon, R \) with respect to the existing *signal-to-noise* ratio \( S = \mu/\sigma \), the following statements are true:

1. If the filtering center lies within a purely non-edge noisy location, then \( \mathbb{P}\{e_1 = 0\} \) is sufficiently large. Moreover, \( \mathbb{E}\{e_1\} \) is sufficiently small with respect to \( \mu \).

2. If the filtering center lies exactly on a local edge segment, then \( \mathbb{P}\{e_2 > 0\} \) is sufficiently large. Moreover, \( \mathbb{E}\{e_2\} \) is sufficiently large with respect to \( \mu \).

The term of *reasonable choices* in the above reformulation applies to those choices which satisfy some basic constraints in view of the very principle of our new edge-detection scheme. Next, we try to make such choices and see if the two assertions in the above reformulation can be justified in such a reasonable way.

We first introduce a parameter \( r \), which is defined by the following:

\[ r = n\varepsilon/\mu \]  \hspace{1cm} (D.59)

By substituting \( S \) (i.e., the local signal-to-noise ratio) and \( r \), the constants \( p, q, K_p \) and \( K_q \) defined previously will take the following forms: (see also Page 212):
\[ p = G((r - 2)RS\sqrt{\pi/4\sqrt{n}}) \quad (D.60) \]
\[ q = G((r + 2)RS\sqrt{\pi/4\sqrt{n}}) \quad (D.61) \]
\[ K_p = \left( \frac{R}{S\sqrt{2n}} e^{-\pi R^2 S^2 (r - 2)^2/32n} + \frac{\pi R^2 p}{2n} \right) \mu \quad (D.62) \]
\[ K_q = \left( \frac{R}{S\sqrt{2n}} e^{-\pi R^2 S^2 (r + 2)^2/32n} + \frac{\pi R^2 q}{2n} \right) \mu \quad (D.63) \]

Similarly, we can rewritten the obtained results in the previous section as follows: (refer to Eqs. D.43, D.47, D.52 and D.55):

\[ P\{e_1 = 0\} = 1 - 2nG^n\left( \frac{RrS\sqrt{n\pi}}{4n} \right) \quad (D.64) \]
\[ E\{e_1\} = \frac{4n^2 \mu}{\pi RS\sqrt{2n}} G^{n-1}(Rr S\sqrt{n\pi}/4n)e^{-\pi R^2 r^2 S^2/32n} \quad (D.65) \]
\[ P\{e_2 > 0\} = \frac{(p + q)(p^n - q^n)}{p - q} \quad (D.66) \]
\[ E\{e_2\} = \frac{2}{\pi R^2} \left\{ K_p \frac{np^{n+1} - 2p^nq + 2q^{n+1} - np^{n-1}q^2}{(p - q)^2} \right. \]
\[ + K_q \frac{nq^{n+1} - 2q^n p + 2p^{n+1} - nq^{n-1}p^2}{(q - p)^2} \} \quad (D.67) \]

1. Basic considerations on the choice for \( n \).

Firstly, since we are essentially only interested in whether or not the given point is an edge point and, if so, how evident it is. In other words, it is not necessary to have a particular filter \( h_i \) so that it corresponds precisely to the local edge segment, should the given point be a true edge point. Thus, it is not necessary to choose an excessively large value for \( n \) even if the local noise occurrence happens to be rather heavy. Secondly, an excessively large value for \( n \) is certain to enlarge the computational burden in a proportional way. Thus, \( n \) should be chosen as small as possible. However, there is an essential requirement on \( n \). It says that
should be large enough to ensure the credibility of the *unimodality* test. To conclude, \( n \) should be chosen as small as possible provided that it can ensure the credibility of the unimodality test. In our view, a value between 4 and 8 for \( n \) will be practically sufficient to ensure this credibility, though the value 4 sounds computationally most attractive.

2. Basic considerations on the choice for \( R \).

The parameter \( R \) which is the radius of the filters' spatial influence plays an important role in measuring the difference of some average grey-value property of two adjacent regions if an edge point is actually involved. In other words, the value of \( R \) is responsible for the representativity of individual filter responses. In contrast to this, an excessively large value for \( R \) is certain to cause an undesired heavy burden in the discrete computation and may even deteriorate the above-mentioned representativity by crossing over the boundary of an involved neighbouring region. Combining the above considerations, \( R \) should therefore be chosen just large enough to ensure the above-mentioned representativity. Again, in our view, a value from \([3, 5]\) for \( R \) will be practically applicable.

3. Basic constraint on the choice for \( r \).

If the given point lies exactly on an edge segment with a step-size \( \mu \), the expected value of the maximal filter response is clearly \( \mu \). Regarding the definition of *unimodality* in Definition D.4, it is obvious that \( n \varepsilon \) should not exceed twice of the maximal filter response if the given point is actually an edge point. This consideration leads us to a rigid design constraint by requiring \( n \varepsilon \leq 2 \mu \) or, equivalently, \( r \leq 2 \).

In the following, we assume \( n = 4 \), \( R \in [3, 5] \) and \( r \leq 2 \). For cases under different local values of \( S \), we try to assign some reasonable values for \( R \) and \( r \) so that the two previously mentioned assertions can be simultaneously justified. The main design strategy here is to choose \( R \) and \( r \) according to the given value of \( S \) so that \( P\{e_1 = 0\} \) is sufficiently large while keeping \( R \) as small as possible. In Table D.1 and Table D.2 we have shown the results for \( S = 2 \) and \( S = 1 \) respectively.

From the results shown in these tables, we can clearly observe that the two assertions in the previously given reformulation of the statements made in Propositions 6.1–2 can indeed be justified even for heavily noise-corrupted situations (e.g., \( S = 1 \)). In particular, we can also observe that the chosen values for \( n \) and \( R \) are quite affordable for practical computations and the resulting filtering schemes are sufficiently localized such as generally required on a low-level edge detector (note the various values of
Table D.1: Results by reasonable choices for $R$ and $r$ under $S = 2$ and $n = 4$.

<table>
<thead>
<tr>
<th></th>
<th>$P{e_1 = 0}$</th>
<th>$E{e_1}$</th>
<th>$P{e_2 = 0}$</th>
<th>$E{e_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 2.5, r = 0.3$</td>
<td>0.849</td>
<td>0.069$\mu$</td>
<td>0.895</td>
<td>0.926$\mu$</td>
</tr>
<tr>
<td>$R = 3.0, r = 0.3$</td>
<td>0.887</td>
<td>0.045$\mu$</td>
<td>0.955</td>
<td>0.967$\mu$</td>
</tr>
<tr>
<td>$R = 3.5, r = 0.3$</td>
<td>0.901</td>
<td>0.034$\mu$</td>
<td>0.984</td>
<td>0.988$\mu$</td>
</tr>
</tbody>
</table>

Table D.2: Results by reasonable choices for $R$ and $r$ under $S = 1$ and $n = 4$.

<table>
<thead>
<tr>
<th></th>
<th>$P{e_1 = 0}$</th>
<th>$E{e_1}$</th>
<th>$P{e_2 = 0}$</th>
<th>$E{e_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 4.0, r = 0.3$</td>
<td>0.804</td>
<td>0.108$\mu$</td>
<td>0.795</td>
<td>0.868$\mu$</td>
</tr>
<tr>
<td>$R = 4.5, r = 0.3$</td>
<td>0.829</td>
<td>0.085$\mu$</td>
<td>0.851</td>
<td>0.899$\mu$</td>
</tr>
<tr>
<td>$R = 5.0, r = 0.3$</td>
<td>0.849</td>
<td>0.069$\mu$</td>
<td>0.895</td>
<td>0.926$\mu$</td>
</tr>
</tbody>
</table>

$R$ in Table D.1 and Table D.2).
Samenvatting

Beeldinterpretatie per computer is een proces om vanuit ruwe digitale beelden te komen tot een symbolische beschrijving van de door de beelden voorgestelde scène. Het is, bijvoorbeeld, zelfs voor een onervaren kind eenvoudig om de aanwezigheid van een huis op een gewoon beeldscherm te constateren. Toch zal een ervaren volwassene in verwarring raken wanneer hem gevraagd wordt expliciet te maken waarom juist dat gedeelte van het beeldscherm een huis voorstelt. Bovendien zal hij nog meer in verwarring gebracht worden wanneer dat specifieke gedeelte van het beeldscherm zodanig is vergroot dat alle betrokken beeldpunten afzonderlijk zichtbaar zijn. Een dergelijk vergrote versie is precies wat een computer tot zijn beschikking heeft en is het voor ons noodzakelijk om op een of andere wijze het waarom hierbij in een programma vast te leggen voor de computer als wij de computer willen vragen om zo’n huis in het beeld te herkennen. Beeldinterpretatie per computer betreft in wezen twee extremel moeilijke vraagstukken. Het ene is hoe ons redeneringsproces met betrekking tot het visuele waarnemen op logische wijze beschreven kan worden. Een voorbeeld hiervan is hoe een bepaalde combinatie van een dak, muren, ramen en deuren de aanwezigheid van een huis suggereert. Het andere vraagstuk is het combineren van enerzijds primitieve begrippen die een menselijke waarnemer gewend is te gebruiken en anderzijds numerieke grootheden die een computer weet te hanteren. Bijvoorbeeld, welke combinatie van discrete punten representeert een lijnsegment?

In het eerste deel van dit proefschrift wordt een algemeen systeemontwerp genaamd het Distributed and Anomaly-Driven System (DADS) voorgesteld teneinde een algemeen beeldinterpretatiesysteem op te bouwen. Binnen dit raamwerk worden twee belangrijke begrippen geïntroduceerd, namelijk, de begrippen context-hierarchie en anomalie. Door middel van de context-hierarchie kan het interpretatieproces rondom een specifiek probleem beschreven worden via tussenliggende meta-objecten. De reden om
zo iets te doen is om het *rationele* gedeelte van ons denken bij een waarne-
mingsproces maximaal te kunnen exploiteren en wel zodanig dat de bij-
behorende werkwijze zowel vriendelijk is voor menselijke gebruikers als hanteerbaar voor een computer. Zoals eerder opgemerkt zijn het logische
beschrijven van een redeneringsproces bij het visuele waarnemen en het
numerieke karakteriseren van een waarnemingsprimitieve geenszins een-
voudige zaken, zelfs voor een tamelijk beperkt probleem. Vaagheid en
onvolledigheid zijn hierbij onontkoombaar. Dikwijls worden zulke vaag-
heden en onvolledigheden pas geconstateerd nadat men geprobeerd heeft
om het beeld te interpreteren met een proces aangegeven door de context-
hierarchie. Dit is in het bijzonder waar als wij opmerken dat het nu-
meriek karakteriseren van een bepaalde beeldprimitieve in het algemeen
niet toelaat om alle mogelijke variaties en onverwachte ongeregelheden in
beschouwing te nemen, terwijl deze juist heel gewone verschijnselen zijn
voor een digitaal beeld van een natuurlijke scène. Om dergelijke verschijns-
elen te kunnen hanteren, voorziet DADS een mechanisme gebaseerd op
het begrip anomalie. Door middel van dit mechanisme kan men de werk-
ing van het systeem verbeteren via het detecteren van verschillende on-
volledigheden binnen het huidige systeem en via het, hiermee in overeen-
stemming, toevoegen van geschikte anomalie-processors om die onvolledig-
heden te bestrijden op dynamische basis. Op deze manier wordt een flexi-
bele wisselwerking tot stand gebracht tussen de verschillende verwerkings-
niveaus, welke traditioneel gesproken gescheiden zijn. Het aanduiden van
een bepaalde klas anomalieën betreft dat gedeelte van ons denken dat in het
algemeen onbewust of ongewoon zijn voor ons (bv., wanneer de anomalieën
puur veroorzaakt zijn door de ongeschiktheid van het discrete raster). Wij
noemen dit het *irrationele* gedeelte van ons denken bij het visuele waarnem-
men. Het belangrijke voordeel van het anomalie-gestuurde mechanisme is
dat de aangeduide anomalieën effectief afgehandeld kunnen worden zonder
een significante toename in de systeem-complexiteit en verwerkingstijd.

In Hoofdstuk 2 beginnen wij met de discussie rondom enige aspecten
van het menselijke visuele waarnemen. In relatie tot beeldinterpretatie per
computer wordt vervolgens ingegaan op het modeleren van DADS en ver-
schillende specifieke onderdelen om te laten zien dat een dergelijk model op
natuurlijke wijze is gerelateerd aan het menselijk beeldinterpretatieproces
en praktisch gesproken geschikt is, vanuit het oogpunt van beeldinterpre-
tatie per computer.

In Hoofdstuk 3 geven wij een gedetailleerde presentatie van het DADS-
ontwerp met nadruk op het systeem-ramawerk. Naast de globale archi-
tectuur en de systeem-controle, bevat dit hoofdstuk ook data represen-
taties, verwerkingspecificaties en systeemspecificaties met betrekking tot
een werkelijke toepassing van het DADS-raamwerk.

In Hoofdstuk 4 worden de slotopmerkingen gegeven. In het bijzon-
der worden verschillende eigenschappen van DADS uiteengezet zoals die
direct waargenomen of verwacht kunnen worden vanuit de voorgaande pre-
sentaties. Tevens worden de betreffende punten aangeduid in relatie tot
de toekomstige ontwikkeling van DADS.

In het tweede deel van dit proefschrift concentreren wij ons op een
voorbeeld van een DADS-gebaseerd applicatie-systeem voor de interpre-
tatie van zogenaamde SLAR-beelden (Side-Looking Airborne Radar). Het
doel ervan is niet om een volledige oplossing te geven voor een dergelijk
complex interpretatieprobleem. Integendeel, het is bedoeld om praktische
inzichten in de DADS-architectuur te verkrijgen en om de geschiktheid
alsmede de mogelijkheden met betrekking tot dit moeilijke probleem te
evaluere.

In Hoofdstuk 5 wordt de problematiek rondom de interpretatie van
SLAR-beelden geïntroduceerd en geformuleerd. Het vraagstuk van beeld-
segmentatie dat een dominante rol heeft gespeeld in voorgaand onderzoek
op het SLAR-probleem, wordt op een formele manier besproken en in
het bijzonder wordt door verschillende aspecten te bespreken een nieuwe
segmentatie-definitie gegeven. Met betrekking tot de toepassing van
DADS wordt de initiële opzet van het systeem (d.w.z. de zogenaamde
context-hierarchie) gegeven.

In Hoofdstuk 6 wordt de Edgeness Detector geïntroduceerd en grondig
besproken, vanaf de theoretische onderbouwing tot de praktische ontwerp-
strategie in het discrete domein. Als een middel om een edgeness beeld
ta te bewerken wordt het Iteration-Based Adaptive Shrinking Algorithm
gepresenteerd.

In Hoofdstuk 7 ligt het zwaartepunt op het vraagstuk rondom het
extraheren van potentiële gebieden in een SLAR-beeld. In het bijzonder
wordt de Edge-Constrained Region-Growing Based Segmentation aanpak
gepresenteerd om een betere segmentatie te verkrijgen.

In Hoofdstuk 8 wordt de extractie van respectievelijk percelen en niet-
perceel gebieden in een SLAR-beeld gepresenteerd. In het bijzonder wordt
het gebruik van het anomalie-gestuurd mechanisme binnen DADS bes-
sproken door twee eenvoudige typen van de geconstateerde anomalieën te
beschouwen. Als een illustratief voorbeeld wordt aldus een op DADS
gebaseerd applicatie-systeem voor de interpretatie van SLAR-beelden tot
stand gebracht.

Via experimenten zijn de effectiviteit en de flexibiliteit van het anomalie-gestuurde mechanisme geëvalueerd. De initiële resultaten rechtvaardigen de verwachting dat het totale systeem nog veel verbeterd kan worden wanneer er meer klassen van anomalieën aangeduid worden. Wij zijn er van overtuigd dat het DADS-raamwerk een sterke potentie bezit om gebruikt te worden als een hoog gekwalificeerd systeem voor beeldinterpretatiedoeleinden.

Parallel aan het realiseren van het huidige op DADS gebaseerde applicatie-systeem is onderzoek gedaan naar de fundamentele problematiek rondom beeldverwerking en ook daaraan worden bijdragen geleverd. In het bijzonder wordt er een nieuwe edge detector genaamd Edgeness Detector geïntroduceerd. Tevens behoren de voorgestelde methoden van het Iteration-Based Adaptive Shrinking Algorithm en de Edge-Constrained Region-Growing Based Segmentation tot de resultaten van dit onderzoek. De experimentele resultaten met deze nieuwe methoden toegespast op verschillende typen zwaar met ruis verontreinigde beelden blijken heel goed te zijn. Ongetwijfeld zijn deze nieuwe methoden algemeen bruikbaar en waardevol voor veel andere toepassingen.
Acknowledgements

In a thesis, the acknowledgements generally appear on one of the last few pages and they are only written when the entire thesis work is almost finished. However, my thanks to all those that have contributed to this thesis in one way or another go far back to the very beginning of this work.

First of all, I shall thank all members of the Information Theory Group for all kinds of support and the pleasant working environment. In particular, special thanks are reserved for Prof. E. Backer and Dr. J.J. Gerbrants for their encouragement and discussions throughout the entire work.

For the final preparation of this thesis, I am obliged to J.C. van der Krogt and J.H. Verschuur for their enthusiastic helping hands in preparing all the photographs. Especially, I am obliged to Mrs. J.B. Zaat-Jones for her readiness and effort in correcting the English text.

Also, the critical remarks from Prof. H.J. van den Herik and the valuable discussions with Prof. C.L. Scheffer are gratefully acknowledged.

Last, but certainly not least, I wish to thank my wife, Wei-Ling, for her constant patience and care which have made it possible for me to concentrate fully on the work for this thesis and which have been indispensable to the success of the entire enterprise.
Curriculum Vitae

Xiang Sheng Cheng was born in Shanghai, China, on May 3 of 1960. In July 1978 he finished his secondary school education and joined the National Qualification Test for University Education in China. After a temporary stay of three months at the Chinese University of Science and Technology he was sent to the Netherlands to receive a university education on February 22 of 1979.

After a short but quite intensive Dutch language course at the Language Center of the Delft University of Technology, he started his university education at the Department of Mathematics of the same university. Because of his great interest in working with images, he later chose to carry out his graduation research within the Information Theory Group at the Department of Electrical Engineering. His Master’s thesis is entitled: ‘Multiresolutional Cluster Segmentation Using Spatial Context.’ He received his M.Sc. degree in Mathematics in 1985 from the Delft University of Technology, Delft, the Netherlands.

After receiving his M.Sc. he stayed within the Information Theory Group as a research assistant and worked towards a Ph.D. degree in the area of image-understanding systems, which led to the current thesis work.

His interests and hopes for the future include model-based image interpretation, image processing, image-processing systems, pattern recognition and artificial intelligence.