Image Representation and Processing

A Recursive Approach

by

V.V. Alexandrov
and
N.D. Gorsky
St Petersburg Institute for Informatics and Automation,
St Petersburg, Russia

KLUWER ACADEMIC PUBLISHERS
DORDRECHT / BOSTON / LONDON

Library of Congress Cataloging-in-Publication Data

Aleksandrov, V. V. (Viktor Vasilévich)
Image representation and processing : a recursive approach / by V.V. Alexandrov and N.D. Gorsky.
p. cm. -- (Mathematics and its applications)
Includes bibliographical references and index.
ISBN 0-7923-2136-7 (hc : acid free paper)
1. Image processing. 2. Image processing--Mathematical models. I. Gorskiĭ, N. D. (Nikolaĭ Dmitrievich). II. Title. III. Series: Mathematics and its applications (Kluwer Academic Publishers)
TA1637.A44 1993
621.36'7'028573--dc20 93-19677

ISBN 0-7923-2136-7

Published by Kluwer Academic Publishers,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Kluwer Academic Publishers incorporates
the publishing programmes of
D. Reidel, Martinus Nijhoff, Dr W. Junk and MTP Press.

Sold and distributed in the U.S.A. and Canada
by Kluwer Academic Publishers,
101 Philip Drive, Norwell, MA 02061, U.S.A.

In all other countries, sold and distributed
by Kluwer Academic Publishers Group,
P.O. Box 322, 3300 AH Dordrecht, The Netherlands.

This is an updated translation by the authors of
Image Representation and Processing. A Recursive Approach,
Nauka 1985 ©

Printed on acid-free paper

Printed in the Netherlands

CONTENTS

PREFACE

Image representation and processing is one of the most important and exciting research areas in computer science, and is the subject of much research in many countries. Some problems of image restoration have been solved; different techniques of image data coding and compression have been worked out; systems and devices of binary, greyscale, and color image processing are in operation, and the first image data bases have been implemented.

Nevertheless, irrespective of their success, the capabilities of computers for visual information analysis are negligible compared with human capabilities. A computer may be compared to a blind person feeling with the tip of a stick things in front of him. The main characteristic of human vision is an adequate perception of the surrounding reality, and the ability to recognize and identify objects in various situations and positions. It is just this that a machine is unable to do. Though we speak of "machine perception" of the surrounding world, in reality only a primitive remembering of input data, and not perception takes place: an element-by-element analysis, and not understanding. Because of this, computers perform more as a means of convenient and powerful representation of image data than for their comprehension, and the human brain remains the "processor" and the device for decision making.

The development of computers as a means of increasing the efficiency, quality and capacity of human activity is an example of the evolutionary way of the development of any instrument. A revolutionary step might be the development of a true "machine mind", which is impossible without its perception of the surrounding world.

In our opinion, some of the reasons for the present impasse are insufficient attention to the interrelations of image data structuring in a computer, the peculiarities of human perception, and the problem orientation of image processing.

It has become clear, for instance, that the efforts to create special processors for fast two-dimensional transforms have not resulted in any qualitative leap in solving the problems of image analysis, because a rather narrow class of images is adequate for a two-dimensional stationary field. By inertia, the task of image analysis has often been replaced by exercises in the fast calculation of approximating coefficients of series such as Fourier, Hadamar, Haar, etc., whilst overlooking the fact that the computer is first and foremost an apparatus for analytical conversions.

After the appearance of powerful processors having large volumes of fast memory, another problem has arisen: the structure of image data flow being input into a computer is converted into a data structure effective for either storage (compression, coding) or fast conversions and processing, but not for both simultaneously. This duality has resulted in computer development which parallels either computational operations which are effective only for tasks of numerical processing, or access to the memory being effective for arranging of storage and data retrieval (for example, in data bases).

From our point of view, the representation and processing of images should be realized as a single process combining two main functions: the parallel control of storage and computations. Therefore, the main content of this book is an investigation of new structural forms of visual information which enables combining memory and computational control. Such structures are being analyzed which, on the one hand, satisfy the specific characteristics of human perception and, on the other hand, are natural for a deterministic device such as a computer.

It should be noted that though the book title contains the terms "representation and processing of images", its contents, the first third, at least, is to be understood more widely, i.e. the study of regular hierarchical structures for the representation of informa-

tion fields of different types; scalar, vector, etc. Such structures, named as pyramids, quad- and oct-trees, etc., are widely used now in image processing and computer graphics. These have attracted particular attention following the works of T. Pavlidis, S. Tanimoto, C. Dyer, H. Samet, and other scientists, published at the end of the 70s and the early 80s.

We have approached these structures from another side – through the investigation of self-similarity and space-filling curves which have provided a natural way of ordering the structure elements. We would like to express our thanks to Prof. J.-C. Simon, University Paris VI, for his attention to our work at the beginning of the 80s and for fruitful discussions in Leningrad and Paris.

We thank V. A. Anisimov for his contributions to the text: he participated in the writing of Section 6.2. We also express our gratitude to G.V. Yepifanov for discussions during the process of manuscript preparation and S.N. Mysko for help in computer experiments.

Chapter 1

INTRODUCTION

For a long time, the basic concepts of image processing were traditionally connected with spectral methods. Any difficulties which arose in the solution of practical problems were considered to be caused by equipment and hardware: low memory volume, low power, the absence of special equipment needed for processing holograms (in the case of optical processing), etc. Computer facilities are now highly developed, making it apparently possible to design special hardware that will evidently help to produce artificial systems able to perceive images. However, this problem is not yet solved, and still remains important and difficult. It is necessary to seek new approaches and methods for image processing which will make it possible to produce artificial systems having perception abilities comparable with those of a human being. The main purpose of this book is to describe the development of the certain new concepts and methods of image representation and processing, relating to the appearance based on "recursiveness", a new set of ideas, techniques and technologies.

The basic terminology of digital image processing is supposed to be familiar to our reader, as well as the basic methods employed, as described, for instance, in the works [23, 102, 106, 148].

1.1 Human and Machine Perception

The most important tasks of computer image analysis are not yet realized in image processing devices. One can understand this situation if we compare the human and machine ways of visual information perception. Let us consider the transformation of image data which takes place when an image is perceived by the eye and transferred to the human memory. This would then enable us possibly to apply in artificial systems some basic principles of human image perception.

It is believed that there are three main forms of image data involved in the process of human perception [105]: (1) image data received by the retina of the eye; (2) problem-oriented signal-features transferred by the optic nerve from the eye to the brain; (3) a responding signal, which is formed in the human brain and used in decision making.

In the first stage of perception (1), external light energy causes certain changes in the sensitive elements of the retina (excites them). The data flow is limited in this step because the number of sensitive elements is finite, and their resolution is limited. The same is true for artificial systems, and thus human and artificial perception have much in common in this stage.

The second stage (2) corresponds to signals and features which are transferred to the visual department of the brain. In this stage, human and artificial ways of perception are radically different. In artificial systems, image data are completely and accurately transmitted from the sensors to the memory, thus requiring powerful data transmitting systems and a large memory. In the natural (human) system, the rate of signal transmission is rather low in this step. This appears to be a few hundred bits per second, although the human eye receives about 10^9 bits of information per second, and the carrying capacity of the eye-brain transmitting channel is very high.

The main feature of human perception in this stage is the high compression of data transmitted from the eye to the memory. Such compression does not usually take place in existing artificial image processing systems. What makes it possible to compress the transmitted data to such a great extent? We consider that such compression is possible because the human visual system, consisting of the eyes and brain, adjusts itself so that, at every given moment, it is *not a universal system but a problem-oriented one.* This phenomenon has very much in common with language communication, where of all the words we know we choose a limited thesaurus needed for a certain situation.

A large number of works about visual illusions exist, and new kinds of illusions are being discovered [62]. Illusions indicate that the mechanism of adjustment really exists in human image perception. The example shown in Fig. 1.1 shows how the illusion causes us to believe that equal circles are of a different size because the straight lines in the picture make us consider the figure to be three- dimensional. Other examples: when we walk in darkness, we may take a bush growing near the road for a hiding stranger; we may recognize certain objects among randomly arranged spots of paint - all this depends upon the subject to which our image analyzer is currently tuned [86]. This tuning ability of human vision decreases greatly the number of situations which can be considered to be different in a

particular context, resulting in the amount of data needed for image analysis to be minimized.

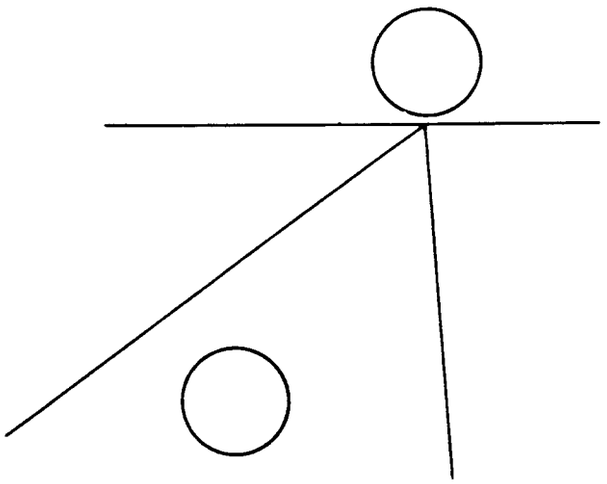


Fig. 1.1. In a context of converging lines, equal circles are perceived differently.

Compression of data is achieved by means of an active adaptation of vision to current conditions and a feedback between the brain and the sensors. The eye adjusts itself according to the current light intensity, contrast, distance, situation and orientation of an object — all these features are reduced to a certain set of invariants, which are much smaller, than all the possible combinations of these features. This kind of adaptation may be called an integral adaptation.

The feedback enables also a local adaptation - when a certain complex image is perceived in a special way: attention is concentrated on salient features, instead of being spread uniformly across the whole picture. This is the so-called "analytical perception procedure", when the primary features of an image are extracted first, then the secondary features, being more complete, are perceived, united and generalized, and, finally, the objects are isolated and recognized. The image data transmitted to the brain become, therefore, not simply compressed, but hierarchically structured. This way of receiving information from the environment can be called induction, because it employs the analysis of particular features (empirical data) in order to lead to a general description.

All these means are used to help to identify objects (situations) observed, thus making it possible to pass to the third stage of perception (3), when the information is admitted by the brain. The information is perceived when the objects observed correspond to known prototypes, according to which a way of behavior is

already known to the perceiver. The third stage of perception is the step from a "pictorial" representation of the image to an abstract model, which describes an object or a situation (Fig. 1.2a), and identifying it in the human brain. Possibly, the identification process involves deduction. Hierarchically, the structured description obtained as a result of the analytical perception procedure is then analyzed level by level, in order to find a relation to some model which already exists in the brain. If some features of the new description differ from those of the existing model they are isolated in order to describe, remember and generalize them.

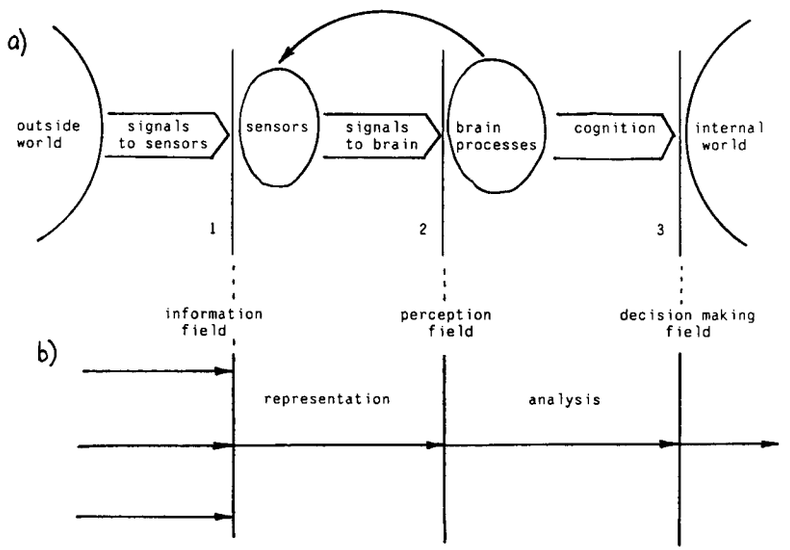


Fig. 1.2. Perception schemes: (a) - by a human, (b) - by a computer.

However, there is another mode of perception, which differs from that described above. It is called simultaneous, or synthetic, recognition with the object observed being perceived at once as a single integral pattern (an example is the recognition of a familiar face). Physiological data show that with synthetic perception there is no time for the analytic procedure, and recognition takes place unconsciously, as a conditional reflex. Hence, it is based on other principles of image data organization or structuring. This mechanism has not been investigated enough and no satisfactory models exist. Much more likely, both analytical and synthetic perception are involved and take place in parallel, supplementing each other, with the former being more oriented to the teaching process, and the latter to

the recognition process. Together, they sequentially form the process of associative reaction of the eye-brain system in interaction with the environment.

Thus, three ways of image data representation correspond to three levels of human perception: representation on the retina, reduced representation in the visual department of the cortex and decisions-identifiers associated with precedents - representation in the memory. The same levels and means of data representation may be form the basis of constructing a model of an artificial image data perception system (Fig. 1.2b). The input part of the system is the information field corresponding to the retina and representing the input flow of image data. Some informative part of the input data flow required for the solution of a particular problem is entered into the perception field (computer memory). The processing of an image represented in the memory should lead to the identification of an image or individual objects which finally is reduced to the process of decision making.

Let us consider the correspondence of image analysis processes in biological and technical systems which are constructed according to Fig. 1.2. The excitation of sensitive elements and the integral adaptation of the eye may be easily modelled in a technical system. At the same time, the feedback between the brain and the eye conditioning an active local adaptation and problem orientation (feedback at Fig. 1.2a) is not usually present in the technical system, because of a data representation technique which does not allow correction by analysis algorithms. Due to this, a large part of the biological mechanisms of data flow reduction is omitted in the technical system and only a few of them remain at the designer's disposal (for example, the accounting of *a priori* data about the class of the problems to be solved or images to be processed).

Hence, the first significant difference between the biological system and the technical one becomes evident even at the stage when the initial data enters the analyzer: the data received are considerably modified and their initial volume is strongly reduced in the former system, whereas the reduction mechanism (if this exists at all) is preset beforehand and remains constant in the latter system. The absence of fast rearrangeability limits the system's capability. Thus, an important task is the development of new approaches to the organization of a computer perception field which takes into account the biological mechanisms of vision and data compression.

1.2. Representation of an Information Field

The perception field of a technical system is that inner representation of data with which the work proceeds. The importance of its correct organization is explained first of all by the following. The complexity and efficiency of image processing algorithms and the range of problems potentially solved just depend on the organization of this structure. This is the result of the above mentioned feedback violation between the algorithms of analysis and the representation algorithms.

The efficiency of problem solving, i.e. the rate of analysis and decision making, and the suitability of the result for the user, greatly depends on the mode of data organization in the computer's memory. Successfully chosen internal data representation makes possible the reduction of information transferred and stored in the system, and the elimination of time-consuming operations of data transformation from one form into another.

For this reason, a choice of the logical and physical structure of the perception field is required, which would consider the specific character of the problem to be solved, as well as make it easier to realize (simulate) the "human" perception procedures, and also to find a structure adequate to the algorithms used and the physical organization (architecture) of the system.

Actually, the structure of the perception fields is determined by the transformations the image data are subject to in the information field. Let us identify three types of transformations (or types of interrelation) of these fields.

(1) The *full perception* is characterized by either the information field being used as the inner information representation or by its transformation which allows to reconstruct simply the initial field. Thus, all the information received is introduced into the system. The main task of the technical systems of full perception is, as a rule, the facsimile transmission of the information field or its storage and reproduction without any distortion. One can conclude that the total volume of input data is to be analyzed only in the case when it is not known beforehand which part of these data is sufficient in order to obtain the information making possible the solution of the problem.

Historically, it was the systems of full perception which appeared first; this emphasises the desire of researchers to enter into the system as much data as possible in the hope of realising a more complete and more qualitative analysis. However, the expediency of this approach is not yet proved. We consider those systems which are more natural (and more similar to human vision) are problem-oriented systems or systems oriented to a given class of objects perceived.

(2) In the case of *problem-oriented perception* it is not a one-to-one but a many-to-one mapping of the information field in the perception field which is involved. It may be said that the information analyzed by this system is a kind of reduced copy of the data received. Examples of reduced internal image representations in systems of this type are a contour image, a limited set of coefficients of a two-dimensional transform, etc.

The basis for the choice of a particular perception field organization should be the range of problems the system is designed to tackle. For instance, it is sufficient to perceive binary, but not greyscale, images for optical character recognition; a contour image of the scene observed is often adequate for an industrial vision system, while a rough representation of a given template with low resolution is successfully used for searching in an image database. The versatility of the human eye is due to its ability to accommodate itself promptly to an enormous range of different conditions and perception tasks. The versatility of a problem-oriented perception system is the larger the greater is the number of tasks the chosen data structure corresponds to. The use of a rearranged perception field (from outside or automatically) promises to enlarge the system capabilities, with the system becoming adaptive in this case.

(3) *Reflex perception* is characterized by preliminarily specifying the procedure of the information field transformation into a set of distinctive features. The number of these features is small compared with the volume of the information field. In fact, further work is carried out in the space of the attributed features with the use of pattern recognition techniques. For instance, to identify an image of a geometric figure, only a small number of characteristics, such as the pattern area and perimeter, the number of its angles, etc., can be used. Systems with such a type of perception can be naturally called specialized, as it is just the particular problem which determines the mode of feature extraction used for decision making. It is this type of system which is most widely used at present. They are, in practice, not rearrangeable and are developed especially for solving a particular problem.

Summarizing, it can be noted that it is the search for data structures for problem-oriented perception systems that is the actual (and difficult) task, as, in full perception systems, this structure is completely determined by the information field itself, and in reflex perception systems - by the problem to be solved .

Many authors have given attention to the importance of studying and searching for new forms of image representation as the fundamental basis for determining achievable results and possible treatments [23, 84, 119, 142]. Conforming to digital image processing, image data representation is actually the data structure (logic or physical) on which processing algorithms are based and

with which hardware or software components of the image processing system interact.

One of the commonly used forms of image data representation in a computer is a two-dimensional matrix of samples, each element of which - the pixel - describes the brightness, or color, of an image element having particular coordinates. The main advantage of such a matrix representation is in preserving the structure of the image processed on a logical (and often physical) level of data representation in a computer, i.e. the spatial organization of brightness and color elements. In this context, the matrix representation is also called the direct representation [102, 119]. The simplicity of this representation enables to easily organize access to, and the processing of, pixels, both sequentially and in parallel. The most often used operation with images presented as matrices of pixels is the row by row scanning of an image and the local processing of each pixel neighborhood; for instance, convolution with a certain nucleus in the vicinity of a 3x3 or 5x5 matrix, edge detection, determination of the local brightness gradient, etc.

Various special image processors have been developed based on matrix representation, such as vector, matrix, and pipeline processors [47, 61]. At the same time, many authors (see [37, 43, 140], for example) have indicated the difficulties of image processing with direct representation. First of all, they result from the rather high expenditures for determining global characteristics, and generalized descriptions of the images processed. Local operations in the pixel vicinity are convenient for the extraction of primary features and for performing operations of pre-processing (noise elimination, local brightness and contrast correction, etc.), but they are not sufficient for image analysis, object identification and recognition. For these tasks, it is necessary to use another type of representation, for example, the description of images using a set of features obtained in the process of performing local operations .

Another approach widely used for image data representation in the computer is an image description using the coefficients of certain (usually orthogonal two-dimensional) transforms, for instance, Fourier, Hadamar, Karhunen-Loev, etc. Instead of brightness samples, coefficients of a brightness function decomposition over a certain basis are stored in the memory in this case. Each spectral coefficient of this kind is an integral image characteristic; it characterizes the whole region of the brightness function determination.

Integral coefficients provide an easy way of accomplishing global operations on images, such as filtration, restoration, distortion correction, and so on [102, 106, 148]. The coefficients of two-dimensional transforms are used as generalized features of images, characterizing them as a whole. They are applied, for instance,

in the case of texture analysis, and recognition of certain image classes. As the spectra of the majority of real images are strongly inhomogeneous, the coefficients of two-dimensional transforms are successfully used for coding and compression of image data [41, 148,152].

At the same time, local operations on an image or its separate parts become difficult if pictorial data are represented by two-dimensional transforms. In many cases, it makes the solution of simple image analysis problems impossible. Global features derived from spectral coefficients adequately characterize those images which can be considered as stationary signals (textures, for example) but are often not suitable for the processing and analysis of other classes of images characterized by local variability, such as real scenes with multiple objects.

Some technological obstacles associated with the hardware required for realizing fast two-dimensional transforms for large volumes of input data arise. As a result, in real systems two-dimensional transforms are often made not for the whole image, but separately for non-overlapping fragments or for separate parts of the image [47, 61, 102]. Though this helps in solving some of the problems encountered, it still often reduces the advantages of global operations with images.

Syntactic image representation methods (also known as structural and/or linguistic methods) [23, 48, 102] describe the images of composite objects as hierarchies of more simple subpatterns and employ the apparatus of formal grammars. Special objects in computer graphics and computer aided design systems are often represented in this way. The advantage of syntactic representation is image description in terms of wholly semantically-meaningful units, which agree well with the particularities of human perception. This representation being developed is in close conformity with the so-called "conceptual hierarchy" or "symbolic" (opposite to "iconic") description of an image, the realisation of which is one of the main purposes of image analysis and computer vision [108].

Within the framework of syntactic representation, linguistic and structural approaches may be identified [48, 106]. In the first case, a composite object (image) is described on the basis of a featured set of non-derivative elements (alphabet) together with rules of their combination (laws of "words" and "phrases" construction). For example, a character image can be described as a combination of line segments of differing form and length. In the second case, features describing different classes of objects are formed on the set of non-derivative elements, but the syntax of the image description language is not defined (as in the first case).

However, any approach is faced with problems concerning the choice and extraction of non-derivative elements. As a rule, an adequate set can be devised only for a sufficiently narrow class of images (or class of objects on them); it is

difficult to find a "semantically complete" basis of non-derivative elements or features which make it possible to describe or to recognize arbitrary images of the chosen class. Problems arise also with greyscale and color image processing; there are difficulties in detecting non-derivative elements in parallel, as their number in real situations can be large. Thus, the main problems here relate to obtaining the syntactic representation itself; in particular, with going from a direct representation to a syntactic one.

Active investigations are in progress in searching for new forms of image representation and new data structures, thus providing further advances in image analysis and image "understanding" by a computer. One promising approach to these problems is the usage of an hierarchy of various representations [23, 84, 68], each being "competent" in its own strict tasks. However, there still remain difficulties concerning their joining with each other, and with the existing hardware.

A matrix of pixels or a matrix of processors does not reflect the essence of image data: a matrix element has only a position (coordinates), while the human perception of image elements is as objects which have both position and a certain form, both color and dimensions. There is an urgent need now to find new types of representation which can account for image contents and which can be effectively realized with computers. This requires the provide construction of processing algorithms having a complexity depending upon the actual complexity of the image being processed and not upon the number of pixels stored.

Alongside the syntactic (structural) description mentioned above, morphological representation based on integral geometry and mathematical morphology [115, 118] and the representation by tree and pyramidal structures using an assembly of images of different resolution and size [92, 113, 119] are being developed. It should be noted that, within the framework of these representations, an elementary object to be treated does not coincide with a brightness element of the initial image, but as well as position and color has some other characteristics as well (size, for example). Operations with these elementary objects are "richer" and more oriented to human perception, than operations with separate pixels or their sets. Let us consider one of these representations in more detail.

1.3. Recursive Approach to Image Representation

What is the reason why the image representation schemes described above fail to solve the problems of machine perception? In brief, one can say: in direct and syntactic representations a computer "is overflowed" with details, while in a spectral representation it "does not see" these details. Where is that intermediate level which would lead to a reasonable compromise between specialization and integration? Let us consider the human experience relating to the permanent search for, and development of, figurative means of spatial and color expression. Its purpose was not only the mapping of the surrounding world, but also the transition of a pattern to the spectator. Therefore, whether it be consciously or not, artists consider in their works the particularities of human perception in order that the impression gained from pictures be more complete, and that the influence power is maximized.

Leonardo da Vinci "analyzed" his pictures taking into due account the laws of perspective, the place from which the picture would be looked at, lighting, etc. The paintings of the masters of the Renaissance can be compared to systems of full perception: the view from a window is just as important for an artist as the hands of a model. Details are carefully worked out, colors are repeatedly verified, the composition is accurately balanced - all is directed to give rise to a full and adequate from the spectator's point of view, impression of the scene he sees.

But as time elapsed, artists began to change the concept of picture construction. They understood that its sense would become more clear and the impression stronger if unimportant details were damped down or even omitted. Rembrandt's and Hals's canvasses contain only the main things - a touch creates shape, details being in shadow, but all this only intensify the perception of the main thought, the integral image. For a spectator, the artist stresses his intention in the problem orientation of the picture, the information flow is compressed, but the perception result is increased; in other words, a greater effect is reached with less means.

And yet, where is the boundary of image integrated perception? To what extent can the same, or an even greater impression, be reached using meagre means compared to that produced by a full reflection of reality? This question was openly posed by artists at the beginning of the 20th century. Impressionists provided a color image perception by limiting the pictorial means - the utilization of pure colors only. Others began to analyze the shape of objects, the interrelation of their parts, and object and phenomena structure. Portraits by Picasso and Braque

comprise an assembly of "cubes" from which the spectator's eye can construct everything. The structural decay of separate elements of an integral image in such cases is on the border of adequate perception by the spectator, but this verge separating talent from quackery is not crossed, and we not only recognize, for example, people so depicted, but also are able to understand their mood and their character.

Irrespective of the shift of details, the substitution of smooth transitions for spasmodic ones, cubistic representation of a man are still integrated images which adequately reflect reality. But what are these images if not their complete structuring? One step further, and we shall have an abstract picture (and this step was made in painting), and then the structure scatters into an assembly of disconnected elements.

We believe that artists, while studying the "transmission" of images to spectators, empirically approach that limit of image data representation which is similar to an intermediate form of their transmission into the human brain. This structural representation is close to the representation on the second level of human perception, in which the analytical process is over whereas the synthetic one has not yet begun.

Thus, compressed and structured representation of an image is still sufficient for adequate perception. An important point is that this structuring can arise not only from the semantics of an image, but also from the shape and brightness of its separate fragments or objects. This makes it possible to device methods of image data representation for a human, or for a computer analysis, in a context-independent class which is not connected with the analysis of the contents of the data processed, but based on a pre-determined processing scheme, and at the same time preserving the image integrity like a two-dimensional or multidimensional field.

Such methods appeared several years ago, and in this book they are called representations based on *pyramidal-recursive structures*. A number of image representation techniques can be united under this name: the corresponding data structures having such common features as hierarchy and regularity [2, 4, 11, 19, 78, 144]. Different authors call them pyramids [1, 35, 37, 43, 71], multiresolution structures [34, 92, 119, 146], quad- and oct-trees [46, 50, 70, 73, 112, 120], cone structures [138-140], recurrent-recursive structures [141, 142], etc.

The first papers describing certain algorithms of image processing based on similar structures appeared in the late seventies [64, 69, 78, 107, 134, 136, 138, 146], then in the period 1985-1987 more than 100 articles were published within this framework.

In the case of a pyramidal-recursive representation, the processed image is described by the ordered sequence of images of different resolution which converge to the original one. These are arranged and drawn usually one under another (hence the term "pyramid" of images). In one of the most widely used alternatives, the initial image is broken into four equal square blocks (possibly intersecting), then the breakage procedure is recursively repeated for each block until its size becomes equal to that of a pixel of the initial image. Each block is ascribed a value called "brightness" or "color", being its generalized characteristics. On finishing the process, we get a set of images each consisting of certain size blocks. These images successively refine each other and converge to the initial one (Fig. 1.3a).

A pyramid construction can also be done in a reverse way, that is, from the lowest levels to the highest ones, by uniting pixels (blocks) of the previous layer. Sometimes pyramids are represented such that each image of the upper level has a size less than that found below with the element size of all images being equal. The structure of data describing the interrelations of brightness elements of all images in the pyramid (Fig. 1.3b) is just an image representation in a computer at a logical level.

Distinctive features of pyramidal structures are their regularity and hierarchy. The regularity predetermines a convenient realization and the effective utilization of such structures in computers, especially in parallel processing. Hierarchy enables, simultaneously or in a consecutive order, work on images of a different degree of generality, and to extract and to use both local (at lower levels) and global (at upper levels) image characteristics. The hierarchy of descriptions having different degrees of generality promotes the context-independent structuring of input images which was attributed above in the representation of works of art.

This structuring can serve the basis for modeling both inductive (in the bottom-up synthesis of data) and deductive (in pyramid top-down analysis) processes of human perception. Simultaneous recognition can also be simulated with pyramidal structures. In fact, if the algorithms of image data representation do not depend on data content and are specified beforehand, then there is the possibility of fast parallel input of image data into the system simultaneously with obtaining the general description. This generalization does not require attributing a semantic structure of images, i.e. it omits the long analytical way of perception. Comparison of an image presented by the pyramid with the search pattern can also be done in parallel, beginning with the utilization of compressed information from the higher levels; that is why a decision about coincidence (recognition) can still be made before the moment of descending to the lower levels of the pyramid.

The natural way to describe the pyramid structure is the use of recursion [2, 11, 32, 73, 78]. Recursion is also a typical characteristic of image processing algorithms using pyramidal representation of images: for many operations it is sufficient to determine only an "elementary cell" of the structure and then to be spread it on all other elements [10, 17, 59, 133, 142]. That is why we use the terms *recursive pyramids* and pyramidal-recursive structures in this book.

Various determinations and ways of describing pyramidal recursive structures are given in the articles [2, 35, 73, 82, 104, 113, 145, 146, 149]. Unfortunately, in most cases they refer to particular cases of certain image dimensions and types.

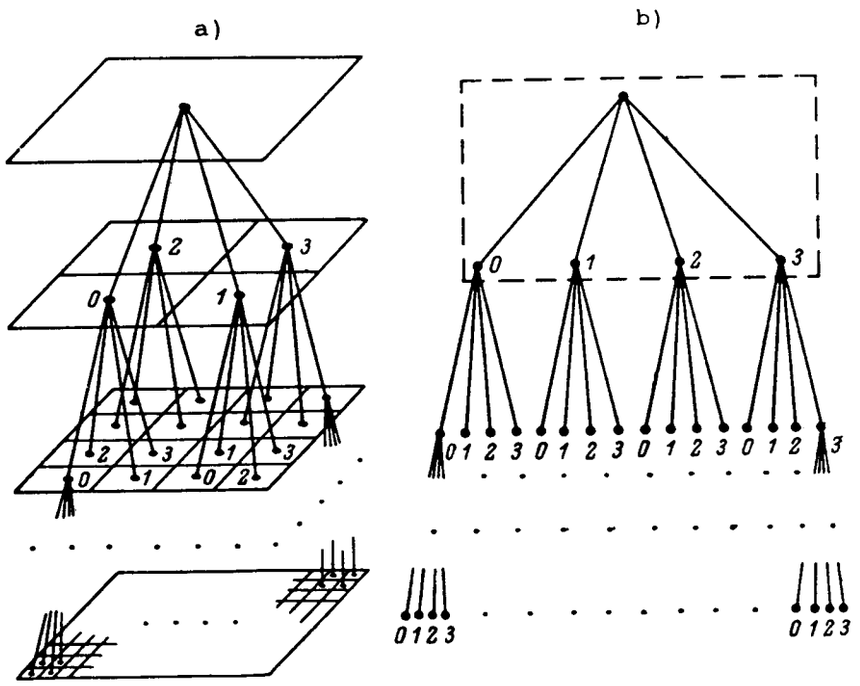


Fig. 1.3. Pyramid of images (a) and corresponding data structure (b).

A number of works describe techniques of identification of pyramidal-recursive structure elements, because the organization of storage and access to these elements, ways of their selection and processing are of great significance for the efficiency of data processing algorithms [39, 49, 51, 73, 109, 111, 149]. The techniques developed also refer to certain types and dimensions of images; they are characterized by some other deficiencies discussed in [11, 15, 53] and in the following chapters of the present monograph.

At present, a large number of algorithms for image representation in the form of different versions of pyramidal-recursive structures have been developed for particular types of images (binary, greyscale) and classes of images (polygonal, contour, segmented). Their main differences are in the ways of structure "filling" by data, i.e. in methods of calculating pixel values at different levels of the pyramid, depending on the brightness (color) of the initial image pixels [123, 128, 142, 146, 147].

Algorithms of image processing with pyramidal-recursive structures are very diverse but, at the same time, often inconsistent. Various methods of coding [1, 44, 76, 114], filtration [34, 80, 98, 127], feature extraction [27, 71, 120, 147], object identification [107, 132, 135, 146], segmentation [22, 31, 63, 64, 69, 99, 124], and synthesis [50, 83, 87, 92] of images are proposed and realized. Many of these algorithms demonstrate a high efficiency with different classes of images; however, some important principles of their work and theoretical models of their functioning are not yet developed, and it is mainly experimental investigations which have been carried out.

Many authors consider that pyramidal structures are a perspective way of image data representation which enables to solve some important problems of storage, search, processing and the analysis of images in digital image processing and computer vision. At the same time, the development of this trend is limited by several factors.

At present, various modifications of pyramidal-recursive structures are used to realize certain algorithms of image processing intended for the solution of particular problems. A common mathematical apparatus of the description of pyramidal-recursive structures, convenient for work with images of different types and dimensions (for example, binary, greyscale, multizonal) has not yet been created. Some researchers describe these structures as trees, others as lists, and yet others as a set of matrices or multidimensional massive. Due to this, different methods of image processing, based on pyramidal-recursive image representation, often duplicate each other and, at the same time, are limited by particular realization or an image type. This results in the creation of heuristic algorithms the transposition and reproducibility of which are difficult because of the use of different data structures and the orientation of particular applications.

Though much experimental material in working with pyramidal-recursive structures has been accumulated, the respective image models both for binary and greyscale images have not yet been developed. The absence of pyramidal-recursive image models does not allow to study algorithms of image processing which exist or are being developed, nor to predict their complexity depending on the

characteristics of input images, to determine data volumes to be stored, transferred, or processed in solving particular problems or in managing with particular types of image.

Some problems concerned with the realization and utilization of pyramidal-recursive structures in computers having a of sequential and parallel architecture are not yet solved. In particular, these are the problems of level-by-level processing of an image pyramid with a gradual refinement of the results in the process of data analysis (similar to the "development" of an image with a level-by-level transmission), and the processing of pyramid structures with the use of nonredundant (compressed) data in systems utilizing pyramidal-recursive representation as the internal data structure.

In this book an attempt is made to answer some of these questions by the formulation and development of the recursive approach to a description and processing of pyramidal-recursive structures with the recursion being both the description technique and the approach for data operation [4, 5, 11, 15, 19, 53, 58].

Let us consider the main ideas assumed as the basis of the approach. Pyramidal representation supposes one and the same law of transition from one level of the pyramid to another with the law being formulated respectively to a group of pixels of the lower level and, as a rule, one pixel of the upper level image. Thus, to construct a pyramid, only an elementary transformation rule of a group of pixels is given which is then spread "in width" to other elements and "in depth" for other levels. This rule describes as well the scheme of the initial data conversion and the resultant data structure (Fig. 1.3b). The structure is completed after a particular image is given as the input information.

Recursion is the natural way to describe the interrelations of pixels of all images from the pyramid: to build a structure, it is sufficient to indicate its *elementary cell* (in Fig. 1.3b this is encircled with a dotted line) and the law of transition to the next level. In other words, to build a structure, only a certain "information gene" - the elementary cell - and the law determining its *development* should be given. Successive utilization provides a stage-by-stage development of the structure description according to the rate of detail.

This approach to the description of the information processes was formulated in [4]. Many phenomena and objects can be found in real life all around us which can be presented in just this way. For instance, Fig. 1.4 shows one of the mathematical objects called fractals [83] built in the following way. Its elementary cell is a pentagon shown in Fig. 1.4a; the development law consists of the construction of similar pentagons on its two short sides at each stage (Fig. 1.4b). By

repeating the process, at the sixth stage one can get an object at Fig. 1.4c, which we denote "cabbage". It turns out that coast lines in geographic maps, the distribution of moon craters, human lungs and many other objects of a different nature can be modelled in a similar way [83]. Thus, the structures and characteristics of real objects can be described and predicted using a recursive "genotype" of minimal volume stored in the memory of the system.

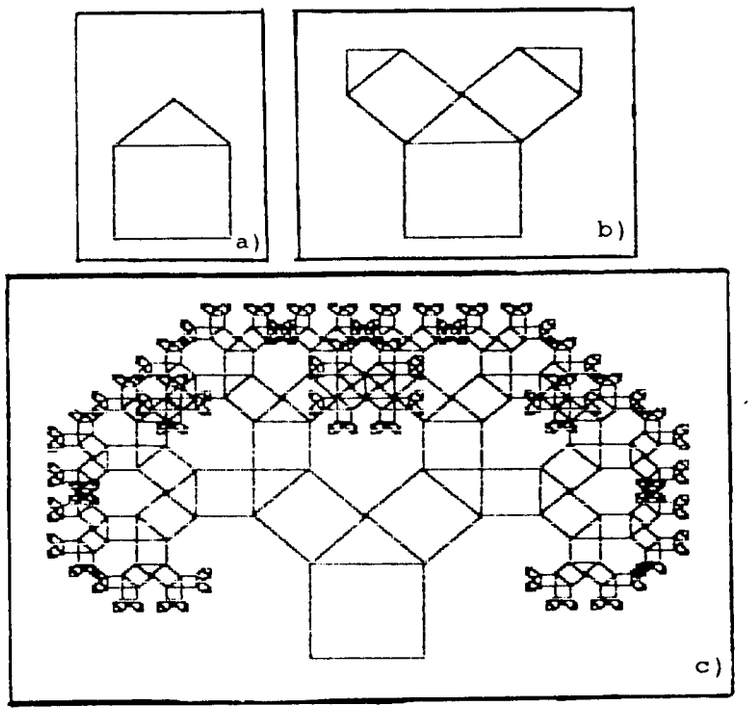


Fig. 1.4. Fractal "the cabbage": (a) basic element; (b) the second stage of development; (c) the sixth stage of development.

It should be noted that objects mentioned in the examples given are not only regular, but hierarchical: each stage of their development gives an object which in some way specifies or refines the preceding one. This hierarchy is predetermined by the description of a law concerning the development of an elementary cell; this is the main difference from the classical description which typically realizes the linear change of the basic element (see Sections 2.1, 2.2).

Another task is how to represent operations with objects (in the context of this book - images) by pyramidal-recursive structures. Two main approaches are

possible. The first is the transformation of the structure as an integral object or the transformation of a set of pixels of an input image. In image processing, this is a traditional operation. Let us recall, for example, filtration or edge detection. In the case of recursive description, mass operations with pixels can be carried out by describing the operation execution only for an elementary cell of the structure and its subsequent expansion on all the cells of the structure.

The second approach, which, in our opinion, is of great significance, refers to operations with separate patterns or with image regions. In essence, at the moment there exists no mathematical apparatus which would provide the operating with arbitrary parts of images as for elementary objects; rather, the region is actually analyzed by a computer as a set of points. A point treated by a machine has no size and shape; man, on the contrary, perceives spots and objects on the image, which simultaneously have position, size and shape.

Pyramidal structures allow image fragments (blocks of pixels which have both size and position) to be considered as elementary objects, these fragments being at the same time elements of a recursive structure (nodes of a representing tree), or pixels of different levels from the image pyramid. The apparatus of their recursive enumeration is given below which provides a specification of the image block position and size in a single description. It enables different operations with images to be made and to refine the results of these operations in the case of image decomposition for a more detailed representation at lower pyramid levels (see Sections 2.3, 3.2, 3.3, 5.2, and 5.3).

However, not only is the size of the blocks important but also is their mutual location. This determines the form of the object in the image. This characteristic not can be deduced from the "pure" hierarchy; "horizontal" interrelations of blocks of different size resulting from the object topology are also significant. The recursive block enumeration allows to preserve these interrelations, or to restore them quickly from the bloŷks' numbers. It makes easier both manipulations with objects in the image and access to the data structure, which we called the pyramidal structure with recursive enumeration of its elements, or, in short, the recursive pyramid (see Sections 2.4, 2.5, 5.1).

One more aspect of importance is the way of representing image data, and their processing, in physical devices. An important problem here is the method of data ordering and access (in the computer memory, or in the case of data transmission or processing) in order to achieve the most effective and rapid handling. The ordering of pixels is practically equivalent to the mapping of a multidimensional array into a one-dimensional sequence. We use such an image data mapping from a pyramid which preserves the topology of a multidimensional

array (in a certain sense) in a one-dimensional sequence. For example, recursive scanning of a Peano curve type, the multidimensional generalizations of which are described in [17], preserve in a one-dimensional neighborhood of a certain pixel on average one half of its neighbors of a multidimensional neighborhood. Thus, image processing can be accomplished using a one-dimensional image representation only. "Uni-dimensionality" of each level of the representing structure turns out to be adequate for the characteristics of those physical devices which are used for processing: sequential performance of some operations and sequential data transmission. At the same time, the ordering of structure elements can significantly facilitate data processing in parallel devices too, as this leads to a unified representation of an image of different dimensions (see Sections 2.2, 4.2, 4.3, 4.4, and 6.4).

In general, the contents of this book is distributed in the following way.

Chapter 2 presents the theoretical basis of the approach. The known scheme of primitive recursion is modified to describe developing hierarchical structures. The definition of the recursive structure is introduced, and methods of the enumeration of its elements are considered. Some particulars of different operations with such structures are given.

Chapter 3 describes the interrelations of a pyramidal-recursive representation with the traditional image representation forms; in particular, with two-dimensional transforms. Pyramidal models of binary and greyscale images are introduced and investigated.

Chapter 4 considers the problem of coding and compression of binary and greyscale images represented by pyramidal-recursive structures. Image progressive transmission with gradual refinement through the communication channels are discussed.

Chapter 5 is devoted to solving some image processing tasks for binary and greyscale images. A pyramidal model of a greyscale image is applied to an hierarchical scene matching and arbitrary oriented object identification.

Chapter 6 contains descriptions of some applications of the approach developed: an optical character recognition algorithm, a mineralogical model, a structure of a specialized computer and some other applications are presented.

Chapter 2

RECURSIVE STRUCTURES AND THEIR PROPERTIES

This chapter contains the basic theoretical results of the book. Its purpose is to describe the mathematical apparatus for the operation with abstract objects of a special type - pyramidal-recursive structures. The use of these structures to represent and process image data is described in subsequent chapters .

In the first section, the meaning of recursion is analyzed and two of its specific features are identified out - recursive definitions and recursive calculations. The first is of importance in the compact description of a process or structure, and the second serves for their transforms. The notion of a recursive structure is introduced, which is later used as the basis for data representation.

The second section considers numbers, the real (numerical) line and multidimensional space as recursive structures represented by k^p-ary regular trees, where each tree node corresponds to some simple space region called a cell.

The techniques of cell identification in multidimensional space is discussed in the third section. Each of the techniques corresponds to some space "scanning" by an ordering of its elements. These scans are compared using the criterion of preservation of topological properties of multidimensional space in an ordered sequence of cells. For further utilization, those scans are chosen which allow spatial processing of data using their one-dimensional representations.

The fourth section describes recursive scans of a multidimensional array which enable cells identifiers to be used not only as symbols or names, but also as values with certain operations defined on them. Based on these operations, manipulation with elements of recursive structures and actually with regions of multidimensional spaces of different dimensions are introduced.

The last section gives the notion of a pyramidal-recursive structure and discusses the usage of these structures for the representation of binary, greyscale

and color images.

The material of this chapter was presented before in [4-7, 9, 11, 17, 19, 59]

2.1. Recursive Descriptions and Recursive Structures

At present, the term *recursive* is often encountered in as applications to different objects and processes. For example, in constructive mathematics and algorithm theory we meet recursive functions, numbers, sequences [66], in signal processing - recursive filters, in computer science and programming - recursive computers, recursive programs, etc. [26, 101]. However, in detailed analysis the "recursivities" mentioned above are different. In this section, two specific recursion types are described, which are proposed by the authors for the compact description and transforms of information [4, 6].

Etymologically, the term "recursion" originates from a Latin *re cursio* - to return. Thus, proceeding from the original meaning of the word, all repeating processes and those with feedback should be called recursive. However, these days the word is mainly used in a narrower sense simply *to describe* objects and processes. Here are some examples of informal definitions of recursion. In [89], one can read: "recursion is used to indicate that the definition of an object contains a reference to the object itself." In [26] it is said: "...in mathematics, recursion is used as a means of describing a function or process in terms of itself." Most clear is the definition of Hofstadter: "...there are recursive definitions. Such a definition can raise an impression that something is determined by itself. This might lead to infinite looping, if not to a paradox. In reality, the recursive definition (being formulated correctly) never determines something in its own terms, but always in terms of simpler versions of itself" [67]. This important remark reveals the main purpose of recursive definitions, that is finding a simpler mode of an object description.

As a formal recursive definition, we shall take the simplest schema of primitive recursion, widely used in constructive mathematics [66][1]:

$$\begin{cases} y(0) = a, \\ y(n) = z(n, y(n-1)). \end{cases} \qquad (2.1.1)$$

1) Usually this schema also includes the parameters:
$y(0, b_1,...,b_n) = a(b_1,...,b_n)$,
$y(n, b_1,...,b_n) = z(n, y(n-1, b_1,...,b_n), b_1,...,b_n)$,
but we omit them for simplification

Here a is a natural number, y and z are functions the values of which are also natural numbers. For instance, it is possible to write:

$$\begin{cases} a*1 = a, \\ a*n = a(n-1) + a; \end{cases} \quad \begin{cases} a^0 = 1, \\ a^n = a^{n-1} a; \end{cases} \quad \begin{cases} 0! = 1, \\ n! = n(n-1)!. \end{cases}$$

The schema (2.1.1) is given only for numerical functions, but objects of another nature can also be defined with it. Further, we interpret this more widely in the following way. Let A be an object from the class A, $A \in A$; Y is a function or an operator, the argument of which is a natural number and the result is an object from A; Z is a function of two arguments: a natural number and an object from A, the result of which is an object from A again, then

$$\begin{cases} Y(0) = A, \\ Y(n) = Z(n, Y(n-1)) \end{cases} \qquad (2.1.2)$$

we call a recursive definition of Y in the class A. Here, and later, on we omit the term "primitive", as only this type of recursion is discussed further.

A change of the number a in (2.1.1) for the object A in (2.1.2) outwardly doesn't change the schema though it significantly extends the sphere of its application. The definition (2.1.1) in the case of $n=1,2,3,...$ generates only a *linear* sequence of numbers, whilst the schema (2.1.2) can produce complex objects of *an hierarchical* structure. This is realised because the object $Y(m)$, though belonging to the same class as $Y(m-1)$, can be more "complex" than the latter since it includes it as a constituent part. For example, the fractal in Fig. 1.4 according to (2.1.2) can be determined in the following way:

$$\begin{cases} \textit{fractal "cabbage"}\ (0) = \ \text{⌂}\ , \\ \textit{fractal "cabbage"}\ (n) = \textit{fractal "cabbage"}\ (n-1), \\ \quad \textit{on every highlighted line of which} \\ \quad \textit{a fractal}\ (0)\ \textit{in a scale}\ 1{:}2^{n/2}\ \textit{is built} \end{cases} \qquad (2.1.3)$$

The decisive factor in obtaining a hierarchical object, which is just the fractal given above, is the possibility to set the operator Z in a way that, in a *one* recursion cycle, at the n-th step *several* elementary changes in the object $Y(n-1)$ might take place. In this example, a pentagon is built at each side highlighted in the preceding cycle, i.e. the object complication in the n-th cycle proceeds in two ways: due to a change of shape of the object obtained in the preceding cycle and due to an increase of the number of possibilities by which these changes can take place.

The main advantage of the recursive definition is the possibility to describe an

object or a class of objects in a compact form. This predetermines the convenience of recursive definitions used as a means of economic ally representing information about a complex phenomenon. However, programmers, for instance, are aware of the fact that recursively defined programs are often not very effective (in the sense of consuming calculation resources) irrespective of their apparent simplicity and compactness. This is due to the way the computer deals with not the recursive definition itself but its somewhat converted form, which needs major means to be achieved.

A human being carrying out the definition of the fractal (2.1.3.) would take an elementary object of 0-th order and then add to it pentagons of the 1st order, 2nd order, etc., until he lost patience. The computer would handle this in quite a different way. Having been given the task to construct a 10th-order fractal, it would determine the action to be performed in the case of a fractal (9) and remember it, than it would repeat the same for fractal (8), etc., until it reached the fractal (0), which is known. This is the stage of problem decomposition. And only now, the composition stage begins: performing of the remembered actions, as it was in the case with the human.

The human proceeded by gradually refining the result obtained at the previous step. As for the machine, however, instead of constructing an object according to a law Z and an elementary description A, it has to bring the initial definition to a form acceptable to perform the operations prescribed. Thus, the schema (2.1.2) when used in practice turns out to be nonconstructive being inconvenient for the automatic design of the object described. Let us therefore change its form.

Note, that to make computations, the machine has to convert the schema of the recursion (2.1.2) by expressing in its right-hand side $Y(m)$ through $Z(m, Y(m-1))$, that is

$$\begin{cases} Y\ (0) = A, \\ Y\ (n) = Z\ (n, Z\ (n-1, Z(\ldots(1, Y(0))\ldots))). \end{cases}$$

Replacing $Y(0) = A$, one can see that the function Y does not occur in the right-hand part, i.e. the recursivity of the definition Y is replaced by using the recursivity of the operator Z (as Z is an argument in addressing Z itself):

$$Y\ (n) = Z\ (n, Z\ (n-1, Z\ (\ldots(1, A)\ldots))). \qquad (2.1.4)$$

In fact, (2.1.4.) describes another type of recursion quite different from the recursive definition. Barron [26] calls this the *recursive utilization* of the function (operator), another way of referring to it is *bottom-up recursive computation* [89]. Recursive definition and recursive computation are often confused. But in reality

their interrelation is strictly determined. As is noted in [137]: "Recursion is a method of problem solution by reducing the problem to one or more subproblems. The subproblem is further reduced in the same way. Finally, subproblems become small enough to be solved immediately. Later, the solutions of the smaller subproblems are brought together for obtaining the solution of a larger subproblem until the solution of the initial problem is obtained".

By performing in (2.1.4) the actions prescribed by the operator Z (beginning with its most inner entry) we obtain successive solutions of subproblems, and at the n-th step - the object sought (solution of the problem). The advantage of computation according to (2.1.4) is in transferring the labour-consuming process of problem decomposition to a person. The description obtained is also actually a computation algorithm. For example, a factorial computation can be described in the following way:

$$n! = MUL\ (n, MUL\ (n-1, MUL\ (\ldots(2, MUL\ (1, 1))\ldots))),$$

where $MUL\ (a,b) = ab$.

By describing the action of the operator Z as a flow-chart, we obtain Fig. 2.1. This is nothing but the feedback schema: A is an input pulse, $Y(n)$ is the response of the operator $Z(n,a)$ depending in the general case upon the conditional time n, and T is a unit delay. Thus, we return again to the initial meaning of the term "recursion", though in reality we have reduced the recursive calculation to one iteration. To summarize, we can note the following.

First, (2.1.4) preserves all the advantages of the recursive definition (2.1.2) because, to obtain an object, it is sufficient to define the elementary object and its transformation law which depends, in the general case, upon the cycle number n, but which does not depend any longer upon the object to be determined, Y.

Second, the construction of a complex object is described by a simple algorithm, i.e. the description obtained is not only compact, but is easily realizable, as only the solution of the elementary problem $Z(n,a)$, a A is to be programmed, while the stages of the object construction are performed automatically.

Third, the schema can be used both for the description of average numerical functions and for the construction of complex hierarchical objects refined at each recursion cycle. We are interested in such a case when some structure describing the data to be processed is used as an elementary object A. The recursive description of this structure in the form of (2.1.4) enables data to be processed in a top-down way, i.e. by refining, at each step, the results obtained earlier.

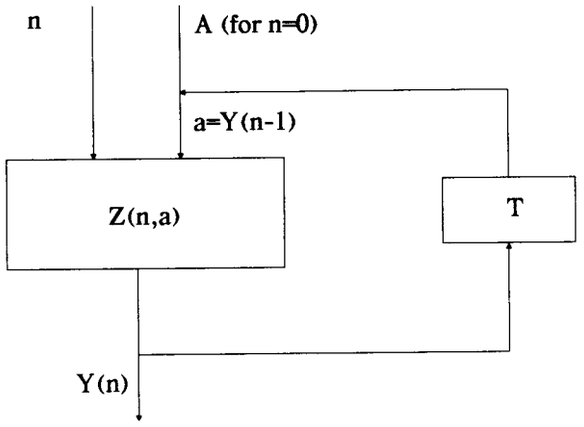


Fig. 2.1. Schema of the recursive calculation.

Let us discuss now the description of recursive structures. Note that using schemas (2.1.2) and (2.1.4), it is possible to describe a set of objects. For example, let us consider the definition of an positive integer number given in the Backus notation: "Writing <digit> : : = 0 | 1 |... | 9 determines the class <digit> as consisting of alternatives 0,1,...,9. Now more complex structures can be defined:

<interger without a sign> : := <digit> |<integer without a sign> <digit>.

This is a recursive description of integer numbers" [26]. Slightly changing this definition, we get the description of an integer number according to the schema (2.1.2):

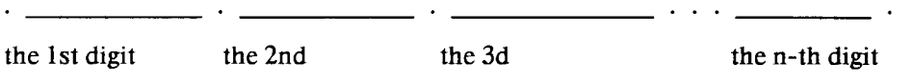
$$\begin{cases} \text{<integer (1)>} = \text{<digit>}, \\ \text{<integer (n)>} = \text{<integer (n-1)><digit>}, \end{cases} \qquad (2.1.5)$$

where digit is a symbol from the set {0,1,2,...,9}.

The description (2.1.5) does not allow the construction of a single particular digit, if at least one object is known which belongs to the set of digits. If it is known what the digits are, for instance, <digit> : : = 0 | 1 | ... | 9, then there is the possibility of substituting different digits into the schema in order to get various numbers. Thus, definition (2.1.5) describes not a single object of type "integer", but any object from the set of integer numbers, therefore it can be interpreted as the definition of the class of objects.

Such a definition describes only the structure of an object from the class to be determined, and the filling of this structure with particular information provides a

particular object as a result. The structure of an object can be presented as a graph $G(S,R)$, where S is a set of nodes; and R is a set of arcs. An arc corresponds to the utilization of the operator Z, every node defines a substitution of one particular object from a set of such objects . This set we denote as **B**. In the above example dealing with integer number, the arc indicates the adding of the next digit and the node one of ten digits, i.e. B={0,1,...,9}. Then (2.1.5) defines the following structure of a number:



If, at each recursion cycle, the operator Z is used several times, then one can obtain more complex hierarchical structures. So, if we suppose that different geometric figures (pentagons or polygons) with two highlighted sides can be assigned as the elementary object A of a fractal (2.1.3), then at the n-th step one obtains different figures having the same structure - binary trees, because it is only on two sides of every figure that the construction of additional objects is done at each step.

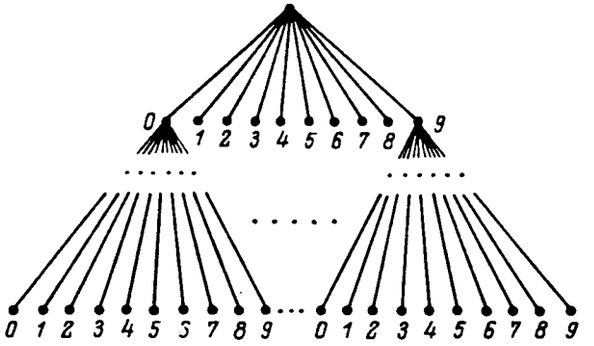
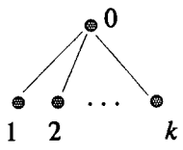


Fig. 2.2. The structure of the class "integer number".

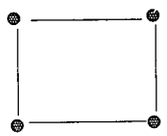
If the set **B** of possible alternatives is finite, then it is possible to speak of a structure which the recursive definition imposes upon the whole class of objects to be defined. In this case, every node of the object structure is split into | **B** | nodes, each being identified and corresponding to one particular alternative from | **B**. Figure 2.2 shows a structure corresponding to the class "integer number" under the condition that 10 different digits exist. This is a regular decimal tree. A separate object (number) is represented in the graph by a subgraph isomorphic to the

structure of a single object. So, the integer number of n digits is a branch of a tree containing n nodes.

The structure of an object or an object class is called recursive if it can be described using the recursive definition corresponding to schemas (2.1.2) or (2.1.4), where A is a set of structures, and A is a graph which will be called the *elementary cell* of a structure. The operator Z can be called the *structure development law*. If Z does not depend on any parameters, then it can be considered as an operator of substitution at the n-th step of all structure nodes identified on the $(n-1)$-th step for the elementary cell A (it is supposed that some nodes of A are identified). For instance, the elementary cell for a regular k-th tree is the graph



in which nodes having the numbers 1,...,k are indicated, and for a rectangular grid a the graph is



all the nodes of which are indicated. We further use mainly recursive structures as trees which are regular and at the same time hierarchical. The structure filled with data describes a particular information field (image), while an abstract unfilled structure describes the interrelation of image elements of a different level from an image pyramid (see Section 2.5).

2.2 Representation of Some Mathematical Objects by Recursive Structures

To process an information field in a recursive form, skill is needed to represent by recursive structures a domain of data (multidimensional space) and the data itself, which can be both scalar and vector. Therefore, this section deals with the technique of the description a number, a numerical axis, and a multidimensional

space as recursive structures. They are presented in [11, 17].

The number. Let us discuss again the definition of a number (2.1.5). Another equivalent form of this definition can be given, if the words "on the right-hand side" are changed to "on the left-hand side". Both definitions give a similar linear structure to a number (and not only to an integer number; the same is valid for numbers having a floating point, if to fix this point and to "build up" the number figure by figure). The only difference of these definitions is that figures involved in the writing of a particular number are written from the left to right in the one case, while from right to left (beginning with low orders) in the other case. This difference, at first glance insignificant, reflects principally different approaches which can exist in data processing.

We shall explain this using a simple example. Let us take a binary number $x=0.x[1]...x[m]$ in the interval [0,1) and suppose that its bits $x[t]$, $t=1,...,m$ can be extracted from the memory in two different ways:

(a) $t = 1,2, \ldots, m$; and (b) $t = m, m-1, \ldots, 1$.

It is easy to see that these ways correspond to the two forms of the recursive definition of a number mentioned above. We shall fix the position of a point representing x in the interval [0,1), to x equal, for instance, to 0.1011 (m=4). After extraction of only one bit, one can see that for approach (a) x is in the right half of the interval [0,1), and for approach (b) x belongs to one of eight intervals of length 1/16. These results are shown in Fig. 2.3, where still unknown bits are denoted with the symbol "*". After extraction of two bits, one can find that: (a) x belongs to one fourth of [0,1), and (b) x belongs to one of four intervals of length 1/16. The process can be repeated for any $t \le m$ (Fig. 2.3).

According to information theory, the extraction of each next bit decreases by a factor of two the uncertainty of the x position in [0,1) irrespective of its position in the string of x bits (under the condition that the probability of the appearance of 0 and 1 are equal), hence, in information theory, the approaches (a) and (b) are equivalent. Really, after extraction of t bits, the sum of the hatched sections on Fig 2.2.1a is equal to those on Fig 2.2.1b for any t and equals 2^{-t}.

However, if it is necessary to indicate the *absolute error* with which a number x is obtained after the extraction of t bits, then approach (b) quickly yields to the approach (a). For the case (b), the error is equal, on average, to $1-2^{t-m}$, and for (a) it decreases exponentially as t grows and equals 2^{-t}. The cause of the difference of bits extraction ways is the positionally of notation. It reflects the inner hierarchy of description something by a number. If the high orders of a number are known, then

the lower-orders only refine the information which is already available due to the primary digits.

Hence, the bits (digits) of a number in the position notation can serve as a natural basis for the organization of bit-serial procedures of data processing - from the most significant to the least significant bits. At the same time, these bits or digits reflect the hierarchy levels of the model described by the number. (It should be remembered, of course, that the figures and notations are arbitrary, and reflect not a "natural", but a "forced" hierarchy of the phenomena with which they are combined. Therefore, all notations should be equally valid for describing real phenomena, while the choice of a particular notation is made only from the point of view of convenience).

Thus, the "right-hand" recursive definition of a number generates a structure of the number which corresponds to the inner hierarchy of the phenomenon described by this number. Progressive algorithms yielding a gradual refinement of the results can be based on this structure.

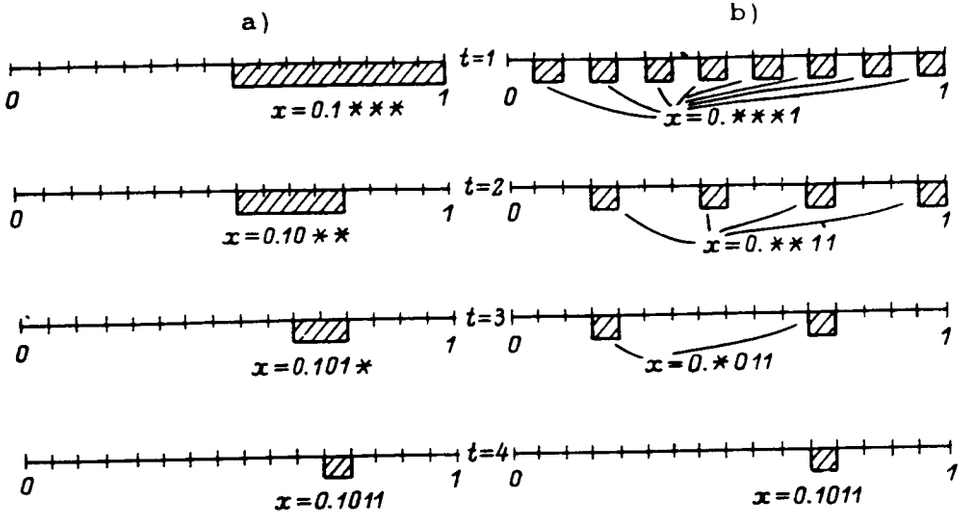


Fig. 2.3. The uncertainty of a point location decreasing with the successive extraction of bits of its coordinate using two different methods.

Numerical axis. In making measurements in practice, researchers are always dealing with a finite amount of significant digits of the parameter measured and strict ranges in which the measurements are made. This means that we use the following model of a measurement process: to find an element from the finite set which is equivalent to the unknown object. When the element is found, the object is

attributed with its all known properties. This is the principle of "identification by indistinguishability" first formulated by Leibnitz.

Most often gradations of some scale appear as identifying elements (ruler points, for instance), with the number of these gradations being finite, as is their minimum and maximum values. If this is not stipulated, then the scale analyzed is considered to be normalized using a unit interval. Let us identify the gradations of this scale by exact fractions using an integer base k notation. Then a recursive structure (one of many possible) in the form of a regular k-ary tree with m levels is brought into correspondence with the scale. Each branch of the tree corresponds to a proper fraction of m digits, which identifies a gradation with the utmost resolution. Limited number of $t<m$ nodes on a branch gives a less accurate value of the same gradation. The number of such "rough" gradations on the scale equals k^t - according to the number of nodes of the t-th level of the tree (Fig. 2.4).

Thus, the recursive structure of a number (positional notation) yields a recursive tree-like structure, each node of which has a one-to-one correspondence with a gradation of a one-dimensional scale of certain size, and each level with a scale of a certain resolution. This set of one-dimensional scales we call a *the dynamic discrete space* [4] of dimensionality 1[1)]. Formally, this can be defined in the following way. Let $D^1 = [0,1)$ be a unit interval and choose the *decomposition base k* (or notation base). Then:

$$\begin{cases} \text{First order cell } q(i_1) = \text{interval } [i_1k^{-1},(i_1+1)k^{-1})D^1, \\ i_1 \text{ is the cell number;} \\ \\ m\text{-th order cell } q(i_1\ldots i_m) = \text{interval } [i_mk^{-m},(i_m+1)k^{-m}] \\ q(i_1\ldots i_{m-1}),\ i_1\ldots 1_m \text{ is the cell number}^{2)}. \end{cases}$$

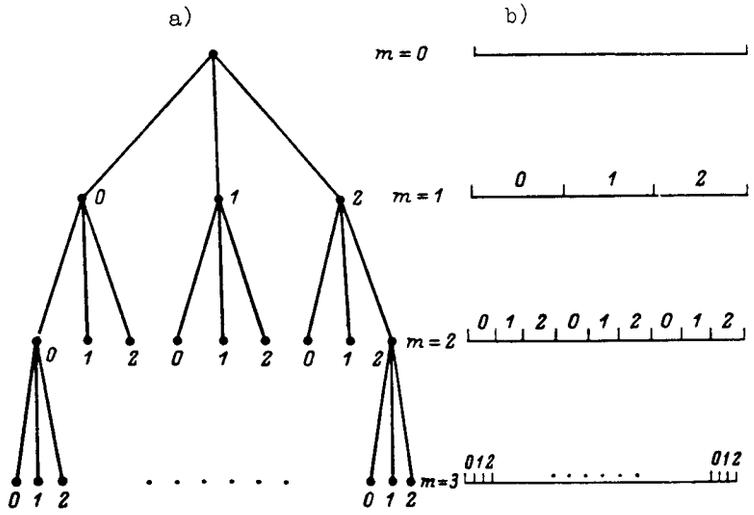
The *discrete space* of the m-th *decomposition* is a set of cells of the m-th order:

$$D_m^1=\{q(i_1\ldots i_m),\ i_t=0,\ldots,k-1;\ t=1,\ldots,m\}$$

The dynamic discrete space is a set of discrete spaces of decompositions from 1 to m. The term "dynamic" means that making m variable or considering the case with m=1,2,3,..., we obtain a refined (developing) object, which at the same time preserves all the properties of the preceding one.

1) Note that these are quantitative scales, hence the node number are not only indentifiers, but also some quantitative characteristics.

2) i_1, ..., i_m are k-ary digits of the cell number.



2.4. The recursive structure of a numerical axis interval (k=3).

Note, that the construction of a discrete space D_m^1 means the simultaneous construction of all discrete spaces D_t^1, 1,...,t, because it is always possible to identify the cells $q(i_1),...,q(i_1...i_{m-1})$ of the $D_1^1,...,D_{m-1}^1$ spaces by the number $i_1...i_m$ of a cell $q(i_1...i_m)$. The most important properties of the one-dimensional dynamic discrete space are:

Property 1. All points of the unit interval D^1 belonging to the same cell of the t-th order $q(i_1...i_t)$ are indistinguishable, if the numbers of higher order cells to which these points belong are not indicated. For certainty, all the points of a given cell are assigned the same coordinate, called *the cell coordinate.* For example, the middle of a respective interval or the left-hand end of the interval can serve as the cell coordinate.

Property 2. Any point $x \in [0,1)$ univocally corresponds to the sequence of nested cells $q(i_1)$, $q(i_1i_2),...,q(i_1...i_m)$ where $i_1,...,i_m$ are the first m digits of the k-ary decomposition of the number x.

Consequence 2.1. The digits $i_1,...,i_m$ determine both the number of a cell (in their listing from left to right), equal to $Q=i_1....i_m$ in k-ary notation and its coordinate in D^1 equal to

$$g = 0.i_1 \ldots i_m = \sum_{t=1}^{m} i_t k^{-t}$$

Consequence 2.2. Cells whose numbers are differ by a unit in the last digit are neighboring cells in D_m^1, and, vice versa, neighboring cell have numbers which

differ by a unit.

Property 3. Let us define the distance between cells as the distance between their centers, and the distance between points D^1 at a given level t as the distance between cells of the t-th order to which these points belong. Then the distance between two points in a one-dimensional discrete space can be determined using both the coordinates of the respective cells and their numbers:

$$|| x_1 - x_2 || = k^{-t} |Q_1 - Q_2| = |g_1 - g_2|.$$

It is not difficult to see that an error in determination of the distance between x_1 and x_2 is not higher than $k^{-t}/2$, i.e., it exponentially decreases with an increase in the decomposition number. This important fact shows that the distance between points in a one-dimensional dynamic discrete space (as well as the coordinates of these points) can be calculated, and with a calculation performed gradually, beginning with a rough estimation and refining this with higher decompositions. Each refinement stage corresponds to passing over to the next level of the recursive structure.

Cells of the dynamic discrete space D^1 have a one-to-one correspondence with the nodes of a regular k-ary tree (Fig. 2.4b). The numbers of nodes of this tree are the last figures of the numbers of the respective cells. The cell number can be restored by reading the numbers of all the nodes from the root to the correspondent leaf. The numbers of the neighboring nodes, as well as the numbers of neighboring cells, differ by a unit (modulo k). This tree with enumerated nodes is the simplest recursive structure which can be used for the representation of a one-dimensional image (signal).

Multidimensional space. The coordinates of a point in a multidimensional space can be determined via its coordinates on each of its axes, as it was done by Descartes. This way predetermines the technique of multidimensional task solving when the initial task is decomposed into more simple subtasks of lower dimension. However, there exists another approach: decomposition not by dimension, but by accuracy. The determination of a point in this way is a successive refinement of its position with respect to the system of neighborhoods of different sizes covering the multidimensional space [6, 17, 51, 129].

In the context of the present book, these neighborhoods are hypercubic cells. The coordinate of a point in a multidimensional space is, in this case, a single number which is defined similarly to an ordinary coordinate in a one-dimensional space with the gradual refinement of its digits. We call it *a positional coordinate.* For a p-dimensional space, we define a positional coordinate of a point recursively by accuracy (or the number of significant digits):

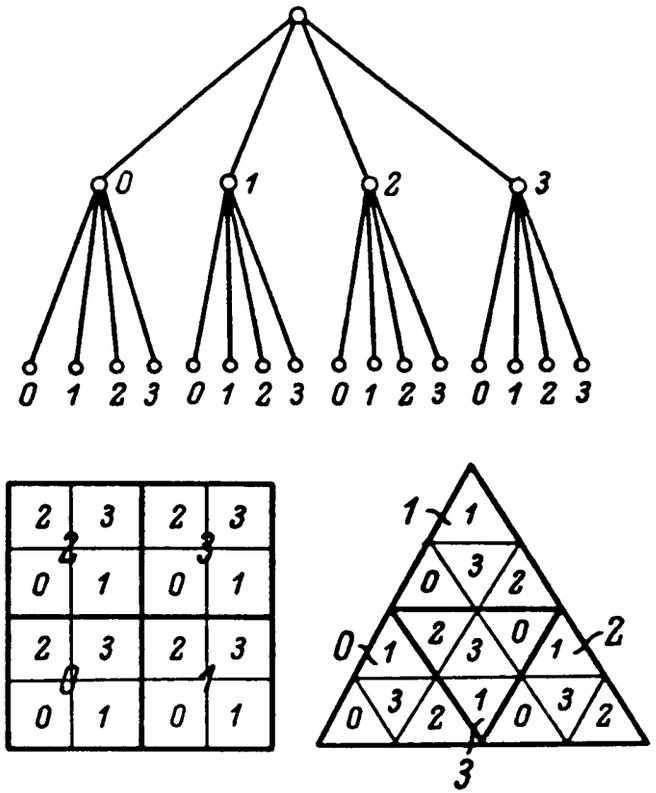


Fig. 2.5. The structure of the interrelations of cell in a two-dimensional discrete

$$\begin{cases} \textit{positional coordinate}\,(1) = .\textit{digit} \\ \\ \textit{positional coordinate}\,(m) = \\ = \langle \textit{positional coordinate}\,(m\text{-}1)\rangle\ \langle \textit{digit}\rangle, \end{cases} \qquad (2.2.1)$$

where <digit> : : = 0 | 1 | 2 | ... | K.

According to this, the positional coordinate is an exact fraction from 0 to 1 in K-ary notation. If we consider all the points of a multidimensional space having the same positional coordinates of given accuracy to be indistinguishable, then the positional coordinates define the space decomposition into cells. In this case, K is the number of cells (quanta, elements) distinguishable in a cell of the previous level.

In the above definition, the dimension p of a space is not presented at all. To *identify* a point or a cell of a p-dimensional space turns out not to be necessary and will appear only in the description of operations with cells.

The definition (2.2.1) overlays the recursive structure as a regular K-ary tree on a multidimensional space. Each branch of the tree identifies a cell of specific discrete space and determines its positional coordinate. Figure 2.5 shows the 4-ary tree with three levels, corresponding to two different structurings of a two-dimensional space. As for the one-dimensional case, each structure level corresponds to the space decomposition into cells of a certain size with a one-to-one correspondence to the structure nodes.

Now let us give a formal definition of a multidimensional dynamic discrete space. Let D^p be a region of a p-dimensional space and choose the notation base K. Then the following recursive definition of discrete space cells can be given:

$$\begin{cases} \textit{First order cell } d(i_1) = i_1\textit{-th element} \\ \textit{a decomposition of } D^p \textit{ into K equal parts;} \\ \\ \textit{m-th order cell } d(i_1 \ldots i_m) = i_m\textit{-th element} \\ \textit{a decomposition of } d(i_1 \ldots i_{m-1}) \textit{ into K equal parts.} \end{cases}$$

The discrete space of the m-th decomposition $D_m{}^p = \{d(i_1 \ldots i_m);\ i_t = 0,\ldots,K\text{-}1;\ t = 1,\ldots,m\}$ is the set of m-th order cells. The dynamic discrete space of dimension p, the D^p, is set of discrete spaces $D_p{}^t$ with $t=1,\ldots,m$. All the spaces $D_{m-1}{}^p,\ldots D_1{}^p$, as in the one-dimensional case, can be considered as progressive modifications of the space $D_m{}^p$.

The definition given differs from that for one-dimensional discrete space only by not mentioning the law (order) of enumeration of each of the decomposition elements. This is the main difference between a multidimensional space and a one-dimensional space: the enumeration order (left to right) in $D_m{}^1$ was "natural", i.e., it originated from the properties of the numerical axis itself; there is no such natural order in $D_m{}^p$, and the cells should be enumerated so as to facilitate the solution of multidimensional problems. These questions will be dealt with in Section 2.3. Here, it should be noted that *the enumeration law* of the m-th order cells within each cell of $(m\text{-}1)$-th order should be known, otherwise the positional coordinate of is "disconnected" from the position of a given cell and, thus, not can serve as its identifier.

If the enumeration laws for all the discrete spaces $D_1{}^p,\ldots,D_m{}^p$ are known, but are taken arbitrarily and differ from each other, then the positional coordinate is suitable for identification, and only for this. Of practical interest remains the case when the positional coordinate serves not only as a cell identifier, but also serves as a means of handling quantitative interrelations between discrete space cells, as is the case for the one-dimensional case (see Consequences 2.1, 2.2 from Property 2

and Property 3). Then the complete "spatial" information is contained in the numbers of the cells or nodes of the recursive structure. It is possible to work with the structure both as a set of multidimensional arrays and as a flat graph, without recourse to the space dimension.

To do this, we choose certain enumeration law of m-th order cells, obtained by the decomposition of a cell of $(m-1)$-th order, ad apply it to all discrete spaces $D_1^p,...,D_m^p$. Only quite definite enumeration laws, out of all possible alternatives, are of practical use, as is shown in the next section.

The main properties of the multidimensional discrete space D_m^p (apart from those related to the enumeration law of the cells) are just the same as those of the one-dimensional space, D_m^1:

Property 4. All the points the of cell $D(i_1...i_m)$ are considered to be indistinguishable until the numbers of cells of higher orders to which these points belong are unknown.

Property 5. Any point $x \in D^p$ uniquivocally corresponds to the sequence of nested cells $d(i_1)$, $d(i_1i_2),...,d(i_1...i_m)$ where $i_1,...,i_m$ are the first m digits of the K-ary decomposition of the number x. The number

$$G = i_1...i_m = \sum_{t=1}^{m} i_t K^{m-t} \qquad (2.2.2)$$

determines the positional coordinate of a cell

$$g = \sum_{t=1}^{m} K^{-t} i_t$$

which can be also written as $g=.i_1...i_m$, where i_t is the t-th K-ary digit of g or G and $0 \le g < 1$.

Note that the region D^p can be of varying shape; in principle, cells of different orders can be dissimilar [2, 40]. For the representation and analysis of images the alternative is convenient when the region D^p is a unit hypercube, i.e. $D^p = [0,1)^p$, with cells of all orders being similar to each other and to the D^p, and also being p-dimensional hypercubes (Fig. 2.5). This case is analyzed below.

If k is the number of equal parts into which a side of a hypercube is divided with each following decomposition, then $K=k^p$ and the positional coordinate of a cell can be written as follows

$$g = \sum_{t=1}^{m} i_t k^{-pt} = .i_1...i_m, \qquad (2.2.3)$$

where i_t is, as before, the t-th k^p-ary digit of g.

An important feature of the positional coordinate (2.2.3) is that it indicates simultaneously both the cell position in the multidimensional space and its

dimension. This property is a general characteristic of numbers written in a position notation; however, it is not of practical use.

The expression $x = 0.5$ means that, in practice, the decimal digits of a number following the digit 5 are unknown (or not needed), and therefore, x is *not a point* on the axis with the coordinate 0.5, but an *interval* of length 0.1. Depending on the way of rounding x, this interval can have the point 0.5000... either as its center or its left-hand end (usually this is the interval [0.45, 0.55)). Similarly, the number of significant digits in the positional coordinate determines the size of the respective cell. Hence, it follows that operations with a positional coordinate of a cell are not those of a point object, but rather those dealing with a region of multidimensional space. They can be considered as operations with a set of higher order cells at once.

The structure of the interrelations of different order cells in the dynamic discrete space $\mathbf{D}^p$ is a recursive structure. It is a graph with enumerated nodes (in the context of this book it is a regular k^p-ary tree); these nodes have a one-to-one correspondence with the cells of $\mathbf{D}^p$, with the arcs of the graph connecting the nodes corresponding to the cells having the numbers $G_1=i_1...i_t$, $G_2=j_1...j_tj_{t+1}$ under the condition that $i_1=j_1,...,i_t=j_t, t=1,...,m$. The enumeration of nodes having a common ancestor is the same as that of t-th order cells obtained by the decomposition of the $(t-1)$-th order cell.

2.3. The Enumeration of Cells in a Discrete Space

Let us consider the dynamic discrete space $\mathbf{D}^p$ and the respective recursive structure to be a k^p-ary tree with m levels, where each level corresponds to the discrete space D_t^p, $t=1,...,m$. The enumeration of cells in the discrete space determines also the enumeration of the structure nodes, i.e. the recursive number G or the positional coordinate $g=k^{-pt}G=.i_1...i_t$ corresponds to each node of the t-th level. The sequence of digits $i_1,...,i_t$ determines the branch of the tree on which the node is to be found. This provides the restoration of hierarchical interrelations (through the upper levels of the tree) of arbitrary structure nodes. However, besides this, the identifiers of the structure nodes should provide for the easy restoration of "horizontal" interrelations of nodes, which for example, enables a fast search of the neighboring cells in D_t^p.

Therefore, a question arises: what additional requirements does the enumeration law of the D_t^p cells have to satisfy in order to solve multidimensional problems (determined for D^p) using identifiers of the structure nodes, i.e. the

positional coordinates of cells. These problems are dealt with in [11, 53]. Note that the enumeration of the t-th order cells determines the means of traversing these cells in p-dimensional space, i.e. it actually determines the method of p-dimensional scanning of the hypercubic array. The task before us is as follows: find the methods of the array element ordering to preserve the information of multidimensional space topology in the k^p-ary tree with enumerated nodes.

In practice, the scans which are of greatest interest are those which either give the highest probability of finding those array elements which neighboring the given one, in its neighborhood after scanning, or which allow the preservation of operations with elements by carrying over the properties and rules of those operations to the positional coordinates of the cells.

Let us consider the p-dimensional discrete space of the m-th decomposition, D_m^p. The set (array) of cells $\{d(i_1...i_m)\}$ of this space is lexicographically ordered with positional coordinates $g_i = .i_1...i_m$, $i = 0,...,k^{pm}-1$, where k is the number of parts into which the side D^p is divided, or the decomposition base. Thus, the scanning technique for an array of $N=k^{pm}$ elements is determined by the enumeration of D_m^p cells, and consists of listing cells according to the change of their positional coordinates. Let the enumeration law for an m-th order cell (obtained by splitting the $(m-1)$-th order cell) remain unchanged with any m and be determined by the 1-st decomposition enumeration law. In this case, the scan belongs to the class of recursive laws [19, 129], examples of which are Hilbert-Peano scans [17, 29, 36, 60, 65, 103].

Formally, it may be assumed that the positional coordinate (2.2.3) determined by the cell numbers is, at the same time, the Cartesian coordinate (in k^p-ary notation) of a cell of some one-dimensional discrete space, for, in one-dimensional space, the position and Cartesian coordinates coincide. Then the recursive scan can be considered as a one-to-one mapping of the multidimensional discrete space to a one-dimensional space:

$$Q_m : D_m^p \rightarrow D_m^1, \qquad (2.3.1)$$

where $Q_m(d(i_1...i_m)) = q(i_1...i_m)$ for $i_t = 0,...,k^p-1$, $t = 1,...,m$. Such mappings have been studied in the papers [5, 7, 9, 12, 13, 15, 17, 19, 29, 59, 60, 79, 81, 96, 103, 117, 121, 122, 128, 129].

Let us state, without explanation, the main properties of recursive scans which concern the interrelations of distances between points in a multidimensional space D_m^p and the distances between corresponding points measured along the scan or, which amounts to the same thing, in a one-dimensional discrete space $D_m^1 = Q_m(D_m^p)$ where Q_m is determined according to (2.3.1) [8,17].

Property 6 Let $x_1=(x_1^1,...,x_1^p)$, $x_2 = (x_2^1,...,x_2^p)$ be two points of D^p. Let two cells, d_1, d_2, from D_m^p, correspond to these. Then, if the first t digits in the k-ary decomposition of x_1^j and x_2^j, $j=1,...,p$ coincide, (that is, $entier(x_1^j k^t) = entier(x_2^j k^t)$ for $j=1,...,p$), than the distance between the cells $q_1=Q_m(d_1)$ and $q_2=Q_m(d_2)$ from D_m^1, or the distance between the points x_1 and x_2, along the scan does not exceed than k^{-pt}. (Suppose that the scan length is normalized to 1).

We call this *convergence by decomposition.* It means that with an unlimited increase of the decomposition number, m coordinates of the images of any points D^p in the space D_m^1 converge to a certain limit. This property is important for the organization of a step-by-step solution of multidimensional problems for which, at each step, the result of the previous step is refined. Note that all recursive scans possess this feature which follows directly from (2.2.1).

Consequence 6.1. The one-dimensional neighborhood of m-th order cell $q(i_1...i_m) \in D_m^1$ - the covering cells of $(m-1)$-th order - always contains cells which, in multidimensional space, neighbor[1)] the cells $d(i_1...i_m) \in D_m^p$ in the direction of each of the coordinate axes.

Property 7. Let the graph of a recursive scan be given in D_m^p. Shift the graph for the k^{-pm} value along all Cartesian coordinate axes, i.e. along the main diagonal of D^p (with part of the scan extending beyond the D^p limits). Denote the discrete space determined by the shifted scan as C^p.

Then, if $x_1 \in Dp$ and $x_2 \in Dp$ are points belonging the neighboring m-th order cells, then in either D_m^p or in C_m^p these points are in the same cell of the $(m-1)$-th order.

Consequence 7.1. Cells of the m-th order, which are neighbors in multidimensional space, have images in the one-dimensional space belonging to one and the same cell of the $(m-1)$-th order either in $D_m^1=Q_m(D_m^p)$ or in $C_m^1 = Q_m(C_m^p)$.

Consequence 7.2. If the point $x \in Dp$ belongs to the cells $d(i_1...i_m) \in D_m^p$ and $d(j_1...j_m) \in C_m^p$, then two cells of the $(m-1)$-th order, $d(i_1...i_m) \in D_{m-1}^p$ and $d(j_1...j_m) \in D_{m-1}^p$, cover all the cells of the m-th order, which neighbor the point containing x. Similarly, the cells $q(i_1...i_{m-1})$ and $q(j_1...j_{m-1})$ cover the images of all the cells which neighbor those containing the point x.

This important consequence reveals the possibility to reconstruct the multidimensional "cross-like" neighborhood of an arbitrary p-dimensional cell (i.e.

t
o
f
i

1) Two p-dimensional cells are neighboring cells if they have a common p-1 order side.

nd all the neighboring cells) using *only two* mappings of the multidimensional discrete space onto one-dimensional space and searching *not more than* k^p elements in each one-dimensional array of cells. This statement was proved in [9].

If any two cells, the number of which differ by a unit, have a common side of order p-1, i.e. are neighbors, then the respective scan is called *quasi-continuous* [17, 36]. This means that each scan "step" is along one of Cartesian coordinate axes and has a length equal to the side of a given cell. The quasi-continuous recursive scan has one more important property:

Property 8. It is possible to describe quantitatively the interdependence of the distance between cells in a multidimensional discrete space and the distance between them measured along the scan, i.e the distance between cell images in a one-dimensional space D_m^1 after the mapping (2.3.1):

$$\| x_1-x_2 \|_V < k(2V+p-1)^{1/V} \| g_1-g_2 \|^{1/p}, \qquad (2.3.2)$$

where $\| x \|_V = \left(\sum_{j=1}^{p} |x^j|^q \right)^{1/q}$;

$x_1=(x_1^1,\ldots,x_1^p)$ and $x_2=(x_2^1,\ldots,x_2^p)$ are integer, or normalized to [0,1), Cartesian coordinates; g_1 and g_2 are integer, or normalized to [0,1), corresponding positional coordinates [17, 36, 129]. Scans which satisfy the relation (2.3.2) enable some properties of information fields to be studied without referring to their multidimensional representation, instead working only with one-dimensional ordered sequences of their elements.

Let us list also some features of the most useful two-dimensional scans (nonrecursive scans included), since the two-dimensional case is the most interesting in the context of image processing. Each of the scans can be characterized by the value of a *neighborhood index* [15, 59]. This integral scan characteristic is calculated in the following way. Take eight elements of an array of two-dimensional cells, surrounding a cell with the number g. After the array is "unfolded" into a one-dimensional sequence, find the number n_g of these elements which form a one-dimensional chain without breaks which also contains the g-th cell. Evidently, $0 \le n_g \le 8$. The value

$$J_m = \frac{1}{N} \sum_{g=0}^{N-1} n_g$$

($N=k^{2m}$ is the total number of cells) is called the neighborhood index of a scan and characterizes the average number of neighboring elements preserved by this scan. With $m \to \infty$, J_m tends to the limit designated below as J_∞; the values of J_∞ have been given in [59].

One more feature of a two-dimensional scan, which is of use in the

investigation of the properties of a two-dimensional information field, is the *symmetry coefficient* C [15, 59]. With $p=2$ this is the ratio of the number of scan "steps" along the x axis to that along the y axis (the x axis is considered to be the axis having the minimum number of steps along it). According to the definition, $0 < C \le 1$.

The main features of various scans are given in Table 2.1. The scans themselves are shown in Fig. 2.6. Table 2.1 shows that recursive scans have significant advantages over traditional ones. Thus, it is reasonable to make the enumeration of cells of each decomposition $t=1,\ldots,m$ in a dynamic discrete space in accordance with one of the recursive scanning laws. Certain conclusions can then be drawn about the spatial interrelations of cells in the multidimensional space and about the hierarchical "subordination" of cells of different order using the recursive numbers of nodes in a recursive structure or, which amounts to the same, the positional coordinates of cells.

At the same time, it should be noted that the positional coordinates constructed on the basis of the enumeration of D_m^p cells according to one of the recursive scans do not yet provide all the necessary operations with structure elements. In particular, they do not enable an extract calculation (nor even approximately as in (2.3.2)) the distances between different D^p cells. Two approaches can be used to eliminate these difficulties: (1) to go from position to Cartesian coordinates, and vice versa, when required; (2) to determine operations with positional coordinates which are analogues of the necessary operations with Cartesian coordinates. Both alternatives are discussed below.

Table 2.1. Characteristics of two-dimensional scans

Scan type (See Fig.2.6)	Convergence by decomposition	Quasi-continuity	J_∞ value	C value	Generalization on any p	Simplicity of description
(a)	-	-	2	0	+	+
(b)	-	+	2	0	+	+
(c)	-	+	2	1	-	-
(d)	+	-	3.5	0	+	+
(e)	+	-	3.6	0.5	+	+
(f)	+	+	4.6	1	+	-
(g)	+	+	4	0.33	+	-

2.4. Cartesian and Positional Coordinates. Operations with Positional Coordinates

Let us consider a set of vectors describing the coordinates of points from $D_m{}^p$. Each vector x can be brought into correspondence with two coordinate representations: The Cartesian coordinates are $x=(x^1,...,x^p)$, and the positional coordinates are $g_x=.i_1,..,.i_m$. In the description $g_x=.i_1,...,i_m$ the line $i_1,...,i_m$ is a single number (is its k^p-ary digits), which can be both the vector's *identifier* (corresponding to be cell's number) and the *value* (positional coordinate) for which various vector operations can be defined [11, 51, 53]. Both representations are equivalent (i.e. the set of Cartesian and position coordinates can be compared via a one-to-one correspondence), which is not difficult to show as follows:

Decompose the component x^q, $q = 1,...,p$ of the vector x into the k-ary fraction:

$$x^q = .x^q[1]...x^q[m] = \sum_{t=1}^{m} x^q[t]k^{-t}, q=1,...p, \qquad (2.4.1)$$

where $x^q[t]$ is the t-th k-ary digit of the q-th coordinate component of the vector x. The value $x^q[t]$ can also be interpreted as the q-th projection of some vector $(x^1[t],...,x^p[t])$ - let us denote this as $x[t]$. Then x can be represented in the following way:

$$x = \sum_{t=1}^{m} x[t]k^{-t} \qquad (2.4.2)$$

Figure 2.7 presents a geometric interpretation of the expression (2.4.2) for the case $p = 2$, $m = 3$, $k = 2$, and $x = (0.75, 0.875) = ((.110)_2, (.111)_2)$. Note that $x[t]$, $t=1,...,m$ are no less than Cartesian coordinates of the t-th order cell[1)] with respect to the covering cell of the $(t-1)$-th order. If the enumeration law of the cells in discrete space does not depend upon the decomposition number, then the vector $x[t]$ is in a one-to-one correspondence with the number i_t of the t-th order cell into which x falls, i.e. i_t can be considered as *one more denotation* of the vector $x[t]$. In fact, each component of $x[t]$ has one of the k values. There are p components in total, i.e. there exists k^p different $x[t]$ vectors. The index i_t also has k^p different values, which we shall consider as names (numbers) of the vectors $x[t]$.

Thus, the representations of (2.4.2) and (2.2.3) are equivalent, and the k^p-ary digits $i_1,...,i_m$ of a cell's positional coordinate can, at the same time, be considered

1) For Cartesian coordinates of a cell, we take those of the point closest to zero, i.e. the "lower left" corner of a cell.

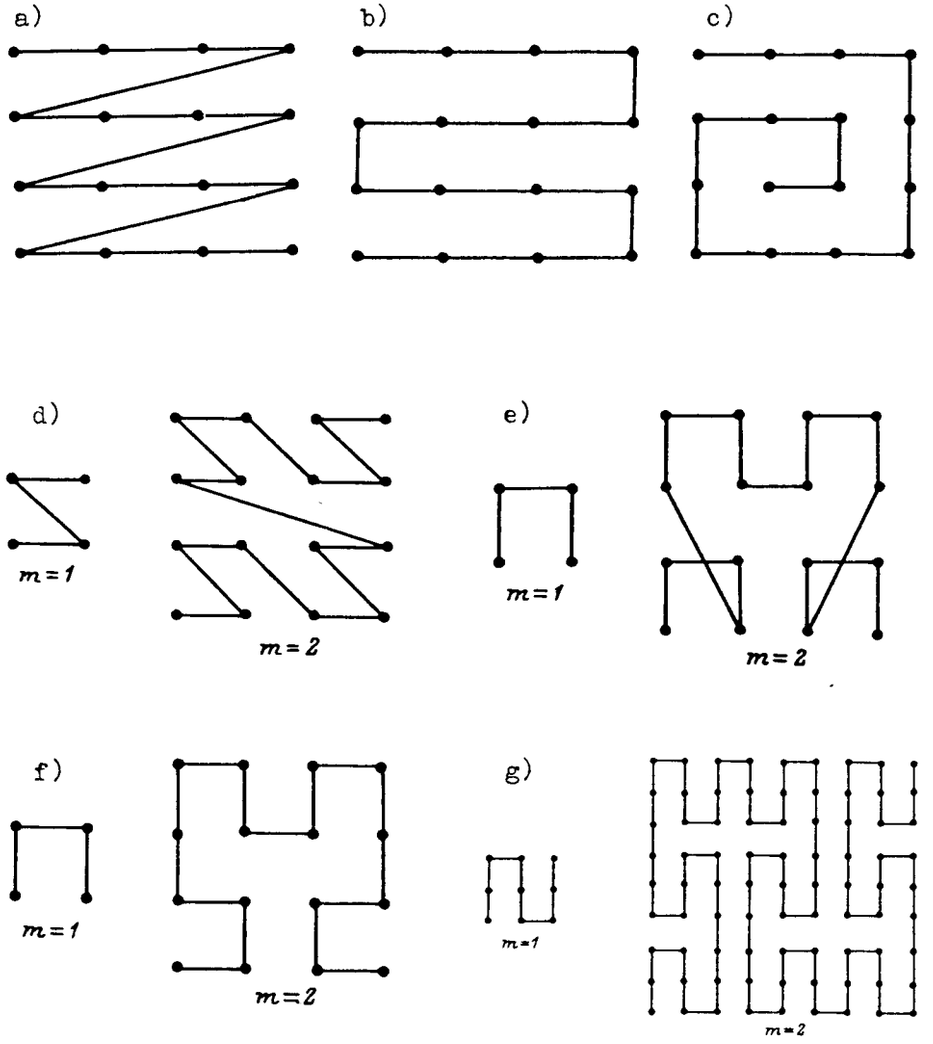


Fig. 2.6. Two-dimensional scans: TV-scan (a); continuous TV-scan (b); spiral scan (c); z-scan (d), Π-scan (e), Hilbert scan (f), Peano scan (g).

as vectors, the weighted sum of which equals the vector x of the Cartesian coordinates $(x^1,...,x^p)$ of the same cell (Fig. 2.7).

Let us take the Cartesian coordinates of the $x[t]$, read as a number in k-ary notation as a number i_t of a t-th order cell. This enumeration corresponds to a Z-scan of the $D_m{}^p$ space described in Section 2.3. For example, with $k=2$, $p=3$, we have

$$x[t]=(0,0,0) \Leftrightarrow i_t=000_2=0_8,$$
$$x[t]=(0,1,0) \Leftrightarrow i_t=010_2=2_8,$$
$$x[t]=(1,0,1) \Leftrightarrow i_t=101_2=5_8.$$

In the general case, we can write for the enumeration mode chosen:

$$i_t=(\sum_{s=1}^{p} x^q[t]k^{p-s}) mod k^p. \qquad (2.4.3)$$

There are, of course, other methods ($p!$ in total - due to the number of permutations of k-ary digits $x^1[1],...,x^p[t]$) of the interrelation between i_t and $x[t]$, which corresponds to other enumeration laws of cells in the multidimensional discrete space Dm^p. These can be considered in a similar way.

The following expression can be used as the corollary of (2.4.3):

$$x^q[t]=k\,entier(i_t/k^{q-1})-entier(i_t/k^q),\ q=1,...,p. \qquad (2.4.4)$$

This enables finding the t-th k-ary digit of the q-th component of the vector x if the t-th digit i_t of the positional coordinate is known.

For recursive structures, it is sometimes more convenient to manipulate integer numbers of nodes, but not with the coordinates of respective cells normalized to [0,1). In this case, the set of integers $J=(j_1,...,j_p)$, which are the numbers for a cell in a line, column, layer, etc. of a multidimensional array of cells, corresponds one-to-one to the Cartesian coordinates $x=(x^1,...,x^p)$ of this cell:

$$J=k^t x, \quad i.e. \quad j_q=k^q x^q,\ q=1,\ ...\ ,p. \qquad (2.4.5)$$

Combining the expressions (2.2.2), (2.3.1) and (2.4.2)-(2.4.5), we obtain the interrelation of a cell's recursive number (or the number of the recursive structure node) and the respective integer Cartesian coordinates. This interrelation can be written as follows:

$$G=i_1...i_t=Q_t(J)=Q_t((j_1,...,jp))$$
$$J=(j_1,...,j_p)=Q_t^{-1}(G)=Q_t^{-1}(i_1...i_t),$$

where $i_t=0,...,k^p-1$; $t=1,...,m$; $j_q=1,...,k^t$.

This relation enables the cell position to be found using its number, and vice versa. The mapping Q_t is nothing but a description of the enumeration law of t-th order cells or the law of multidimensional discrete space scanning.

Let us now consider operations with positional coordinates. As has been shown in Section 2.3, the positional coordinates and node numbers of the recursive structure under certain conditions provide information about the spatial relations of

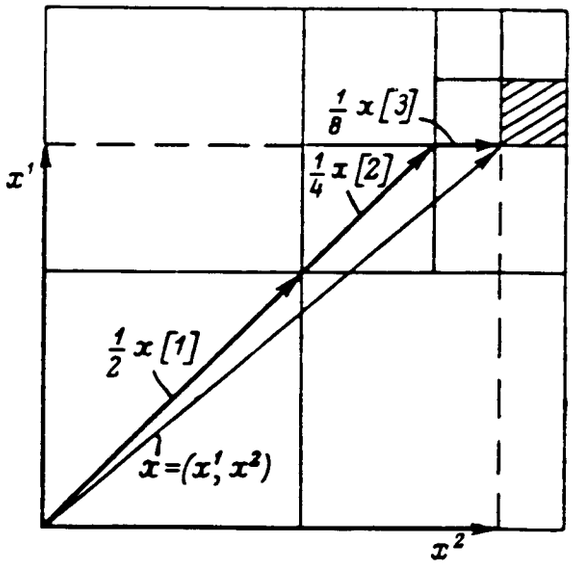


Fig. 2.7. Interrelation of Cartesian coordinates x and the positional coordinate g_x.

discrete space cells; for example, enable the distance between them to be evaluated. Let us introduce analogues of vector operations for the positional coordinates (addition, multiplication, scalar product) [6, 11, 53]. This makes it possible to achieve two main goals: first, to define operations with vector images (for example, color) by means of operations with positional coordinates, i.e. to reduce vector fields to scalar ones; second, to introduce operations with image regions as analogues of operations with numbers.

Addition of Vectors. Let us consider the vectors $x, y, z \in D^p$ and $x+y=z$. Two representations are given for each vector:

$$\begin{cases} x=(x^1,...,x^p), \\ x=.i_1...i_m; \end{cases} \quad \begin{cases} y=(y^1,...,y^p), \\ g_y=.j_1...j_m; \end{cases} \quad \begin{cases} z=(z^1,...,z^p), \\ g_z=.h_1...h_m. \end{cases}$$

It is necessary to find a relation of the form $h_1...h_m=f(i_1...i_m,j_1...j_r)$, i.e. the "addition" rule of the positional coordinates (denote this as $\oplus$) corresponding to the addition of the Cartesian coordinates of the vectors x and y. Express z using the decomposition of the vectors x and y (2.4.2):

$$z=x+y=\sum_{t=1}^{m}(x[t]+y[t])k^{-t}.$$

The sum $x[t]+y[t]$ depends only on the value of the k-ary components of the vectors $x[t]$ and $y[t]$, although it does not depend on t if the enumeration law of

the multidimensional discrete space cells remains one and the same for any decomposition number t. Thus, the addition rule of the k-ary digits i_t and j_t, which are, at the same time, the identifiers of the vectors $x[t]$ and $y[t]$, remains the same for any $t=1,...,m$. It is called the positional coordinate digit addition rule. To express this by usual manipulations, find the recursive number G_t of the vector $x[t] + y[t]$. Each component of this vector $x^q[t] + y^q[t]$ can range from 0 to $2k-1$, i.e. it contains not more than two k-ary digits. Then G_t contains two k^p-ary digits. From (2.4.3) one can obtain:

$$i_t \oplus j_t = G_t = k^p g_1 + g_2 = k^p \sum_{q=1}^{p} k^{p-q} C_k(x^q[t] + y^q[t]) + \\ + \sum_{q=1}^{p} k^{p-q}(x^q[t] + y^q[t])_{mod\ k}, \qquad (2.4.6)$$

where $C_k(\bullet)$ is the value of the highest-order k-ary digit of $(\bullet)$.

The digit g_2 is the contribution of $i_t + j_t$ to the t-th order of the positional coordinate of the vector z - digit h_t, as $(x^q[t] + y^q[t])_{mod\ k}$ influences only the component $x^q[t]$. The digit g_1 is a "carry" from summing $i_t + j_t$ into the $(t-1)$th order of the positional coordinate of the vector z - digit h_t, as $C_k(x^q[t] + y^q[t])$ influences only the component $z^q[t-1]$. So, the relation (2.4.6) describes the addition rule for orders of the positional coordinates. In particular, the following table of addition for the digit of the positional coordinates for two-dimensional space and $k=2$ can be found from (2.4.6):

$i \oplus j$		i			
		0	1	2	3
j	0	0	1	2	3
	1	1	10	3	12
	2	2	3	20	21
	3	3	12	21	30

The first digit in certain positions of the table is a "carry" into a higher order. Below, we give is an example of summing two vectors with positional coordinates $g_x = .12033$ and $g_y = .02102$ (the respective Cartesian coordinates $x = (0.59375, 0.34375)$, $y = (0.125, 0.28125)$

.12033 - g_x
$\oplus$ .02102 - gy
10131 - partial sum
$\oplus$ 2 2 - carries
30311 - sum $g_z = g_x \oplus g_y \Leftrightarrow$ $x+y = (0.71825, 0.625)$.

Multiplication of a vector by an integer. Let $y = nx$, where n is a positive integer. The rule for determining the positional coordinate $g_y = .j_1...j_m$ as a function of n and $g_x = .i_1...i_m$ is to be found (denote this as $g_y = n^* g_x$).

Let us replace multiplication by the sequential addition of n vectors x and use the addition rule obtained (2.4.6). Suppose that $0 \le n \le k$. Then $n^* i_t$ contains not more than two k^p-ary digits, and, for any $t=1,...,m$, one can obtain:

$$n^* i_t = k^p \sum_{q=1}^{p} k^{p-q} C_k(nx^q[t]) + \sum_{q=1}^{p} k^{p-q}(nx^q[t])_{mod\ k} \qquad (2.4.7)$$

In particular, $0^* i_t = 0$, $1^* i_t = i_t$, $k^* i_t = k^p i_t = i_t 0$.

For example, for k=2 and two-dimensional space we have from (2.4.7) the following table of multiplication of positional coordinate digits for a number $0 \le n \le 2$:

$n * i$		i			
		0	1	2	3
n	0	0	0	0	0
	1	0	1	2	3
	2	0	10	20	30

Using (2.4.7), it is easy to obtain the formula for the multiplication of positional coordinates by an arbitrary integer $n \ge 0$. For this, let us factorize n by orders of k:

$$n = \sum_{i=1}^{u} k^{u-1} n[l] \quad , \text{where } 0 \le n[l] \le k. \quad \text{Then,}$$

$$n^* i_t = \bigoplus_{l=1}^{u} ((k^{u-l} n[l])^* i_t) = \bigoplus_{l=1}^{u} k^{p(u-l)} (n[l])^* i_t).$$

Taking into account that $g_x = .i_1...i_m = \bigoplus_{l=1}^{u} k^{-pt} i_t$, we have

$$n^* it = n^* \bigoplus_{t=1}^{m} k^{-t} i_t = \bigoplus_{t=1}^{m} k^{-t} \bigoplus_{t=1}^{m} k^{p(u-l)} (n[l]^* i_t), \qquad (2.4.8)$$

where the operation $n[l]^* i_t$ is accomplished according to (2.4.7) or the equivalent multiplication table. The relation (2.4.8) is completely analogous the usual multiplication of two numbers, but the operations + and $\oplus$ are replaced by $\oplus$ and *, respectively. Below, we give an example of multipling a two-dimensional vector $x = (0.09375, 0.3125)$ having a positional coordinate $g_x = .02102$ by an integer

number 3 *(k=2)*:

$$\begin{array}{rl} & .02031 \quad - g_x \\ * & (11)_2 \quad - n \\ \hline & 2031 \\ \oplus & 2031 \\ \hline & 23221 \end{array}$$

- product $g_y=n*gx \Leftrightarrow nx=(0.28125, 0.9375)$.

Scalar product of vectors. Let $a=(x,y)$. We require to find a rule of calculation a using positional coordinates g_x, g_x of the vectors x and y (designate $a=g_x \otimes g_x$).

According to the definition:

$$a=(x,y)=\sum_{q=1}^{p} x^q y^q.$$

Using (2.4.2) one can write:

$$\sum_{q=1}^{p} x^q y^q = \left(\left(\sum_{q=1}^{p}\sum_{t=1}^{m} k^{-t} x^q[t]\right)\left(\sum_{r=1}^{m} k^{-r} y^q[r]\right)\right)$$

$$=\sum_{t=1}^{m}\sum_{t=1}^{m}\left(k^{-(r+t)}\sum_{t=1}^{m} x^q[t]y^q[r]\right).$$

Let us consider the expression in brackets

$$\sum_{q=1}^{p} x^q[t]y^q[r] = (x[t],y[r])=i_t \otimes j_r. \qquad (2.4.9)$$

This depends only on the component values of the vectors $x[t]$, $y[r]$, rather than on t and r. Thus, the rule of the scalar product of k^p-ary digits i_t and j_r, corresponding to the vectors $x[t]$ and $y[r]$, is the same at any t, $r = 1,...,m$. We call this rule the positional coordinate digit multiplication rule. The relation (2.4.9) shows that to obtain the product $i \otimes j$ it is enough to carry out binary logic multiplication i by j and then to calculate the number of units in the product. In particular, for the two-dimensional space, and the decomposition base $k = 2$, we obtain the following multiplication table:

$i \otimes j$		i			
		0	1	2	3
j	0	0	0	0	0
	1	0	1	0	1
	2	0	0	1	1
	3	0	1	1	10_2

As the result of this multiplication is an ordinary number, it then follows from (2.4.9):

$$a=g_x \otimes g_y=\sum_{r=1}^{m}\sum_{t=1}^{m} k^{-(r+t)}(i_t \otimes j_r),), \qquad (2.4.10)$$

i.e. the operation "⊗" of the scalar product of vectors represented by positional coordinates is the usual number multiplication, but using the digit multiplication table described by (2.4.9). Further, an example is given of the scalar multiplication of the vectors $x=(0.75, 0.625)$ and $y=(0.125, 0.625)$ with positional coordinates $g_x=.312$ and $g_y=.203$:

$$\begin{array}{rl} & .312 \quad - g_x \\ \otimes & .203 \quad - g_y \\ & 1011 \\ & 101 \end{array}$$

0.011111_2 - product $g_x \otimes g_y=(x,y)=0.484375$.

Projection of a vector into a coordinate subspace. Let y be a projection of the vector $x=(x^1,...,x^p)$ into a $(p-1)$-dimensional subspace, i.e. $y = (y^1,...,y^{p-1}) = (x^1,...,x^{n-1}, x^{n+1},...,x^p)$. The positional coordinate $g_y=j_1...j_m$ is to be found using $g_x=i_1...i_m$ and q (denote this as $g_y=g_x/n$).

It follows from (2.4.2) that:

$$y=\sum_{t=1}^{m} y[t]k^{-t},$$

where $y[t]=(x^1[t],...,x^{n-1}[t],x^{n+1}[t],...,x^p[t])$. Then,

$$j_t=\sum_{q=1}^{p-1} y^q[t]k^{p-q-1}=\sum_{q=1}^{n-1} x^q[t]k^{p-q-1}+\sum_{q=n+1}^{p} k^{p-q}x^q[t]$$

$$=k^{p-n}(i_t \div k^{p-n+1})+(i_t)_{mod\,k^{p-n}}, \qquad (2.4.11)$$

where "÷" denotes exact division[1]. The result of the operation with a digit i_t does not depend on the number t of this digit position. Thus, (2.4.11) describes the rule of projection for an arbitrary k^p-ary digit of the positional coordinate. Applying (2.4.11) for each digit of g_x, we obtain the result $g_y=g_x/n$:

$$g_y=\sum_{t=1}^{m} k^{(p-1)m}((i_t \div k^{p-n+1})k^{p-n}+(i_t)_{mod\,k^{p-n}}).$$

1) This unwieldly expression is one other than the elimination on n-th k-ary order from it

In particular, to project a three-dimensional discrete space into a two-dimensional space (with a $k=2$ decomposition base), the following table of results can be obtained (n is the number of eliminated Cartesian coordinates):

$j=i/n$		i							
		0	1	2	3	4	5	6	7
n	1	0	1	2	3	0	1	2	3
	2	0	1	0	1	2	3	2	3
	3	0	0	1	1	2	2	3	3

Below is an example of the projection of the vector $x = (0.75, 0.625, 0.5)$ with the positional coordinate $g_x=.712$ into a two-dimensional space having axes x^1 and x^3:

$$.712/2 = .310 = (0.75, 0.5).$$

If the projection of a vector into a $(p\text{-}q)$-dimensional subspace is required, then the operation "/" should be applied sequentially q times in decreasing order of the deleted Cartesian coordinate axis numbers (in order that the remaining axis numbers are not changed), for example:

$$.712/3/1 = .301/1 = .101 = (0.625).$$

The relations (2.4.6)-(2.4.11) determine the main vectorial operations with the positional coordinates of multidimensional discrete space cells. This means that all other vectorial operations (for example, distance calculations in multidimensional space, matrix operations, and so on) can be expressed also by the transformation of positional coordinates. It should be noted that to do this, it is necessary to apply the main operations to the vectors having both an integer and a fractional part and an arbitrary sign of each component.

To do this, let us introduce into a positional coordinate an integer part determined by the same rules as the fractional part and placed before the point. Then the operation * can be used for the multiplication of the positional coordinate by an arbitrary large number. Let also the positional coordinate have a number of digits in the fractional part which are not fixed beforehand (i.e. let the cells have an arbitrary decomposition number), then the operation * can be used to multiply the positional coordinate by a number α with both an integer and a fractional part according to the same rules. By analogy, the sign of a positional coordinate can be introduced.

Thus, operations with positional coordinates are similar to operations with ordinary numbers (scalars). The difference is that instead of k-ary digits, all manipulations are made with k^p-ary digits - the corresponding cell numbers of discrete space. The rules relating to the manipulation of these digits are new. All the operations with digits are "local" and can be performed in parallel; their result being independent of the digit position in the positional coordinate. The basic result is that irrespective of the space dimension, all operations are carried out with scalar values (positional coordinates); only the notation base (or the length of the binary representation of every digit) is changed.

2.5. The Representation of Images with Pyramidal-Recursive Structures

The concept of an information field is a convenient abstraction for the description of digital images of different types. It can be specified by establishing a correspondence between two bounded sets: X - the domain of the field definition, and S - the set of field characteristic values, which is called brightness or color.

We consider X as a p-dimensional discrete space ($p=1$ corresponds to a one-dimensional signal, $p=2$ - to a "flat" image, $p=3$ - to a three-dimensional image).

In accordance with the characteristics of the set S, let us introduce the following classification of information fields (the most frequent cases are represented):

1. Black-and-white graphics or binary image - there are only two brightness levels, i.e. $S=\{0,1\}$. An example of such an image is a drawing, or a solid body in a three-dimensional space. The value "0" corresponds to the background or is interpreted as "void", the value "1" means an object (pattern).

2. Color graphics. There is an unordered finite set of colors which are different identifiers of the domain elements, i.e. S is a scale of names. Examples of color graphic images are a placard, a geography map, etc.

3. Greyscale image. There are k^n ordered brightness ranges, i.e. $S=D_n{}^1$ is a discrete scale of ratios. An example of such an image is an image on the greyscale display.

4. Color halftone image. There is a discrete color space, where each coordinate axis corresponds to the intensity of a certain color, i.e. $S=D_n{}^q$. Examples of such images are a color TV-image ($q = 3$) or a multispectral image surface received from remote sensor ($q = 3 - 10$).

Let the domain X of an information field be a p-dimensional unit hypercube D^p. Construct on this a dynamic discrete space $\mathbf{D}^p$ *as a set of discrete spaces* $D^p{}_1,\ldots,D^p{}_m$. In this case, a regular k^p-ary tree is brought into correspondence to the field domain, the nodes of which have a one-to-one correspondence to the cells of the discrete spaces $D_1{}^p \ldots D_m{}^p$. Each node is marked by the number of the respective cell, then the node can be identified by this number or by the positional coordinate of the cell (2.2.3) which is uniquivocally related to the number.

An important aspect relates to filling the constructed structure with data. The brightness or color values of the information field correspond to $N=k^{pm}$ elements of the lowest structure level (tree leaves). It is necessary to find, for every structure node of the levels $t=1,\ldots,m-1$, the brightness (color) value characterizing a set of descending elements of the initial field and probably the nearest cells. We denote this brightness (color) value compared to the structure node by s with an index set, showing the cell position in the multidimensional discrete space.

Depending on the need, two different index sets for "s" are used below: one for the identification of the cell by Cartesian coordinates, the other - for indicating the positional coordinate of the same cell. So, the expression $s^t{}_{j1\ldots jp}$ means that s characterizes the brightness value in the cell of the t-th level in the j_1-th row, in the j_2-th column, in the j_3-th layer, ... of D_t^p; while $s(i_1\ldots i_t)$ means that s characterizes the cell $d(i_1\ldots i_t)\in D_t^p$ (Fig. 2.8). Let us suppose that the correspondence of these two index sets, giving the location of the brightness elements, is specified by the known dependences

$$i_1\ldots i_t = Q_t(j_1,\ldots,j_p)$$
$$j_1,\ldots,j_p = Q_t^{-1}(i_1\ldots i_t),$$

where $t=1,\ldots,m$; $i_h=0,\ldots,k^p-1$; $h=1,\ldots,t$; $j_n=1,\ldots,k$; $n=1,\ldots,p$; which enable the cell's location to be found in the multidimensional discrete space by its number, and vice versa.The mapping Q_t is nothing but a description of the enumeration law of the t-th order cells or a law of multidimensional space scanning (Section 2.3).

Suppose that, for some initial image, a recursive structure is constructed and filled by data, i.e. all brightness values compared to the tree nodes are found. Then, we obtain a set of m images, each of them refining the image of the upper level. These are converge to the initial image with increase in the level number. Such a structure for image representation is known as an image pyramid. Each image of the pyramid corresponds to one structure level. Each structure node with a number $i_1\ldots i_t$ of the t-th level with its brightness (color), we call a brightness element of the t-th level or a pixel of the t-th level from the pyramidal-recursive structure. For pixels, we use the same denotations as for structure nodes.

Let us consider how the brightness value of a t-th level pixel $s(i1\ldots it)$ can be determined. This pixel is a characteristic of some local domain of the initial image which can either coincide with the t-th order cell $d(i_1\ldots i_t)$ or exceed it in size by overlapping the neighboring cells. The general approach to the determination of the $s(i_1\ldots i_t)$ problem is as follows ([146]). Subject the initial p-dimensional image to the action of some "roughening" operator, for example, a low-pass filter. Then, to obtain the $(m-1)$-th level image from the pyramid, the "roughened" image is to be discretized into $k^{p(m-1)}$ elements, and then the brightness scale is to be quantized into a given number of levels.

To obtain pixels of the t-th level image, the same process is carried out for the $(t+1)$-th level image. Convolution of each level image with bell-like window functions [34] are sometimes used as a "roughening" operator for greyscale images (analogously to the procedure for time series fitting), for example:

$$S^t_{j_1\ldots j_p} = \sum_{l_1=-r}^{r} \sum_{l_p=-r}^{r} w(l_1,\ldots,l_p) S^{t+1}_{kj_1+l_1\ldots kj_p+l_p} \qquad (2.5.1)$$

where $w(l_1,\ldots,l_p)$ is a window function which is even all over the arguments with

$$\sum_{l_1=-r}^{r} \sum_{l_p=-r}^{r} w(l_1,\ldots l_p) = 1;$$

outside the segment $[-r,+r]$ the function equals zero, while, inside it, it is positive.

If the window size at the $(t+1)$-th level image fitting exceeds the t-th order cell, then the value in (2.5.1) becomes dependent not only on the brightness values of the directly subordinate cells of the $(t+1)$-th level, but also on the neighboring elements of these cells. Figure 2.9 illustrates this case for $k=2$, $p=1$, when each pixel depends on four pixels of the lower level. Note that the structure of the brightness calculation (dotted lines) remaining recursive is no more a tree, but a lattice. Some authors refer to this as a grid, and call this structure *an interleaved* or *overlapped pyramid* [3, 43].

However, it is more often considered that the window function is of a size which coincides with that of the cell of a previous decomposition (in this case, the data structure and the calculation structure coincide). The brightness of a pixel is determined in this case by the relations:

$$\begin{cases} s(i_1\ldots i_t) = s^m{}_{j1\ldots jp}, \\ s(i_1\ldots i_t) = F(s(i_1\ldots i_t 0),\ldots,s(i_1\ldots i_t k_{p-1})), \end{cases} \qquad (2.5.2)$$

where $(j_1,\ldots,j_p) = Q_t^{-1}(i_1\ldots i_t)$, which, in short, describe *the pyramidal-recursive representation* of an image. The way of calculating the brightness of a node-ancestor $s(i_1\ldots i_t)$ using the known brightness values $s(i_1\ldots i_t i_{t+1})$,

$i_{t+1}=0,\ldots,k^p-1$ of nodes-descendants, which are later called *subordinate* nodes, depend on the character of the problems being solved, image type, etc. Different authors determine the function F as an average, median, minimum or maximum value of the subordinate pixels [92, 123, 142].

We take the average value of greyscale image elements as an F function, as the average value minimizes the mean square error of approximation the values $s(i_1\ldots i_t i_{t+1})$, $i_{t+1}=0,\ldots,k^p-1$ by the value $s(i_1\ldots i_t)$:

$$\sum_{i_{t+1}=0}^{k^p-1} (s(i_1\ldots i_t) - s(i_1\ldots i_t i_{t+1}))^2 \to min,$$

$$if\ s(i_1\ldots i_t) = k^{-p} \sum_{i_{t+1}=0}^{k^p-1} s(i_1\ldots i_{t+1}) =$$

$$= k^{-p(m-t)} \sum_{i_{t+1}}^{k^p-1} \cdots \sum_{i_m=0}^{k^p-1} s(i_1\ldots i_m) \qquad (2.5.3)$$

This expression can be rewritten in another way using indices which give the Cartesian coordinates of a cell $d(i_1\ldots i_m)$:

$$S^t_{j_1\ldots j_p} = k^{-p} \sum_{l_1=kj_1}^{k(j_1+1)-1} \cdots \sum_{l_p=kj_p}^{k(j_p+1)-1} S^{t+1}_{i_1\ldots i_p} = k^{-p(m-t)} \sum_{l_p=k^{m-t}j_p}^{k^{m-t}(j_p+1)-1} S^m_{i_1\ldots i_p} \qquad (2.5.4)$$

The values $s(i_1\ldots i_t) = s^m{}_{j1\ldots jp}$ are pixels of the initial image at the lowest level. Thus, a pixel of the t-th level is the average of its subordinate pixels of the h-th level for any $h=t+1,\ldots,m$. Note that the choice of another approximation criterion might lead to another function F, for example, in minimizing an absolute error, F turns out to be the median, and in the case of the Chebyshev criterion (minimizing the maximum error) F is half the sum of the maximum and minimum elements of $\{s(i_1\ldots i_{t+1}),\ i_{t+1}=0,k^p-1\}$. The preference for the mean square error criterion is explained not because it excels over other criteria, but by a tradition established in signal processing, the convenience of analytical transforms, and the possibility to compare results with those of other authors.

For a binary image, the brightness of the initial field elements have one of two conditional values, 1 or 0. The pixel of the t-th level can also be determined in this case according to (2.5.3) or (2.5.4). However, it is considered more often that $s(i_1\ldots i_t)$ can have three values

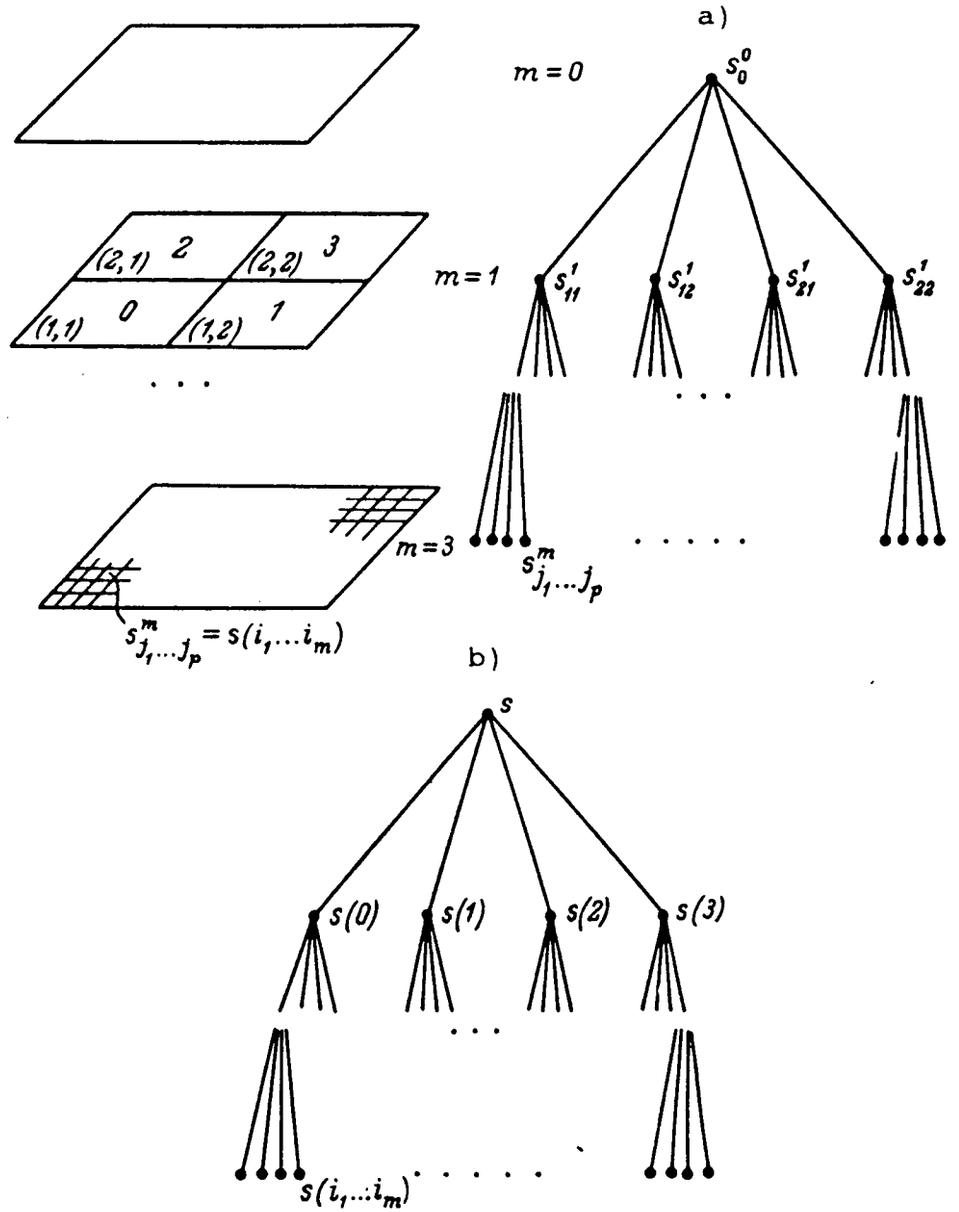


Fig. 2.8. The identification of data structure using (a) Cartesian coordinates, (b) and positional coordinate.

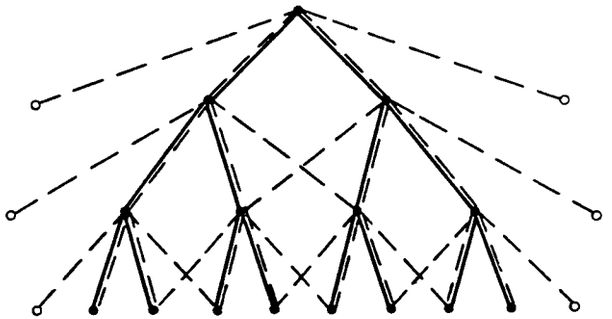


Fig. 2.9. Overlapped structure.

$$s(i_1...i_t) = \begin{cases} 0, \text{ if } s(i_1...i_t i_{t+1}) = 0 \\ \text{for all } i_{t+1} = 0,...,k^p - 1, \\ \\ 1, \text{ if } s(i_1...i_t i_{t+1}) = 1 \\ \text{for all } i_{t+1} = 0,...,k^p - 1, \\ \\ U, \text{ in other cases.} \end{cases} \qquad (2.5.5)$$

Here, U means that among the descendants of $s(i_1...i_t)$ there are elements having a different brightness. The value "0" is usually interpreted as "background" or "white", the value "1" as a "figure" or "black", the value "U" as "uncertain" or "grey". Descendants of a node with brightness "0" or "1" are usually omitted (truncated) as they evidently have the same brightness (Fig. 2.10). Such incomplete trees with a decomposition base $k=2$ have special names: quadtree or quad-tree for $p=2$ [45, 70], oct-tree or octree [73, 87] for $p=3$ and hyperoctree or 2^p-tree [50, 73] for p3. It is easy to see that all the leaves (terminal nodes) of quad-, oc- or hyperoctree have a brightness "0" or "1" and all nonterminal nodes are grey.

The same can be done for the representation of color graphic images. To get the color of the t-th level node, "voting" of colors compared to subordinate nodes of the $(t+1)$ decomposition is usually carried out. The color occupying the greater area is transmitted to the t-th level [123]; if all subordinate structure nodes are of the same color; they are omitted, i.e. the tree becomes incomplete, as is the case for a binary image.

The representation of a color halftone image can be based on a color space representation by a pyramidal-recursive structure. It is shown in Section 2.4 that superimposing the recursive structure on a multidimensional space enables space

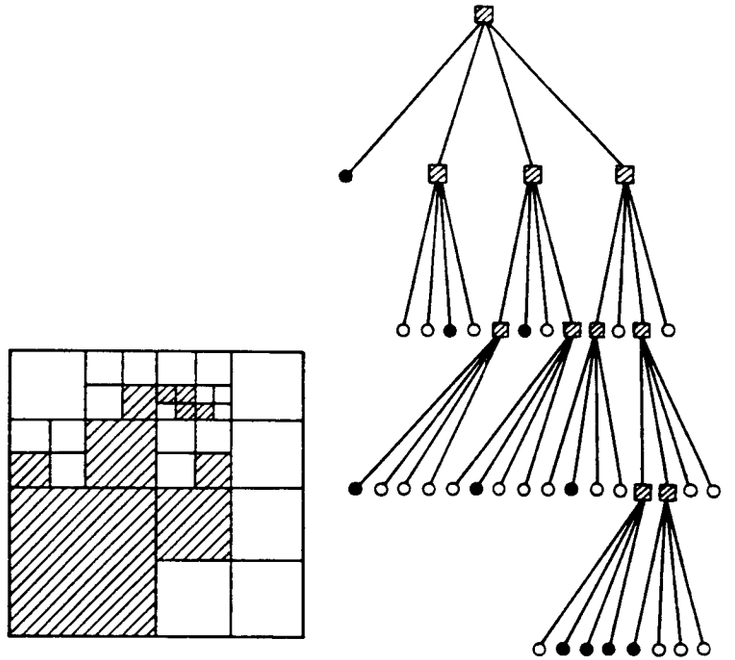


Fig. 2.10. Representation of a binary image by a quadtree (pyramidal-recursive structure for $p=2$, $k=2$, and $S=\{0,1,U\}$.

cells to be identified using the positional coordinate having features similar to a number in the k^p-ary notation. Let us represent a multidimensional color space where each coordinate axis corresponds to a certain color (or, for example, to one of the spectrazonal channels) by the dynamic discrete space $\mathbf{D}^q$. Then, each point of the color space is given the positional coordinate which defines its belonging to cells of all decompositions, from the first to the n-th (consider the number of distinguishable gradations in each color to be equal to k^n; usually $k=2$, $n=3$-8).

Now, instead of a point (vector) of a multidimensional color space, each image element will be compared to a one-dimensional positional coordinate (scalar); hence, we come to the case discussed above of a black-and-white greyscale field. This representation can be easily interpreted: taking the t-th level image from the pyramid, we obtain an approximation of the initial image, where each color has k^t gradations (but not k^n as in the initial case).

The most important result which follows from the reduction of the vector, to the scalar, field is the possibility to process color images in the same way as black-and-white images, which is possible due to the properties of the positional coordinate described in 2.2 - 2.4. For example, taking a quasi-continuous

Peano-Gilbert scan as a way of enumerating color discrete space cells, we find that close brightness values (black-and-white characteristics of the reduced image) correspond to points close in the color space, or similar color tints. To a certain extent, the opposite is true: close brightness values correspond to close color tints satisfying the conditions (2.3.2). In particular, this enables effective algorithms of color halftone image coding and compression [127, 128] to be constructed.

Operations with positional coordinates, introduced in Section 2.4, make possible the processing of brightness values obtained by the reduction of color images in a way quite similar to that used for greyscale images. The only difference is that the summation and multiplication signs of the brightness samples are changed by the signs of the respective operations with the positional coordinates for colors. The possibility of a combined representation of a color halftone image, when both the image domain (p-dimensional space) and the intensity scale (q-dimensional color space) are structured, is based on this fact.

Let us first reduce the color image to a black-and-white image, comparing each domain element not with a q-dimensional color vector, but with a one-dimensional positional coordinate. Then $s^m_{j1...jp}$ is not a vector, but a scalar (pixel) expressed by the positional coordinate of the color space cell. Second, let us represent the image domain itself by a recursive structure. While filling this structure by data according to (2.5.1), it is necessary to take into account that $s^m_{j1...jp}$ is not an ordinary number but a positional coordinate, therefore, the operations signs in this expression are to be replaced:

$$S^t_{j_1...j_p} = \bigotimes_{l_1=-r}^{r} \bigotimes_{l_p=-r}^{r} w(l_1,...,l_p) * S^{t+1}_{kj_1+i_1...kj_p+i_p}, \quad t=m-1,..,1.$$

Instead of (2.5.3), for the same reasons we have:

$$s(i_1...i_t) = k^{-p} * \Big(\bigotimes_{i_{t+1}=0}^{k_p-1} s(i_1...i_t i_{t+1}) \Big), \qquad (2.5.6)$$

Thus, all brightness values $s(i_1...i_t)$, $i_t = 0,...,k^p-1$, $t = 1,...,m-1$ (as positional coordinates, of course) at all structure levels can be determined, or a pyramidal-recursive representation of the halftone color image can be obtained.

Similarly, it can be considered that the processing of color images is also reduced to the processing of greyscale images with the replacement of the respective operations with numbers by operations with positional coordinates. Therefore, the case of a color image will not be further discussed separately.

Thus, the main types of the information field described can be presented based on a pyramidal-recursive structure, with the method of calculation of pixel values being different for different types of images.

Chapter 3

PYRAMIDAL IMAGE MODELS

This chapter considers the main properties of pyramidal-recursive image representation. Attention is paid to preserving the initial image properties at the upper levels of the structure. It makes possible the approximate solution of problems requiring the analysis of considerably less data.

In the first section of this chapter, the interrelation of the pyramidal-recursive representation and some widely used orthogonal two-dimensional transforms is considered. The property of energy concentration at upper levels of the pyramid is presented.

The second section describes theoretical and experimental research into the pyramidal model of a greyscale image. An image is simulated by a separable Markov field. This model provides estimations of the capacity and computational complexities of the representation. It is used in subsequent chapters to forecast the accuracy of results of various image processing algorithms using pyramidal-recursive structures.

The third section presents analogous results for the model of binary images. Two kinds of simple images - "spot" and "line" - as well as a model for complex images, are considered.

The results of this chapter have been published in [11, 53, 54, 57, 60].

3.1. Comparison of Pyramidal-Recursive Structures and Two-dimensional Transforms

The comparison of a new way of representing data with those already existing helps to establish its advantages and disadvantages. It is also useful to be able to use one form or another depending on the problem.

Comparison of the pyramidal-recursive representation with Hadamar transforms can be easily illustrated using a graph of fast Hadamar transforms. Figure 3.1 shows the graph of one-dimensional transforms (p=1), for the case m=3. The nodes and edges of the tree shown by the solid line are just a graph of a one-dimensional pyramidal-recursive structure. The corresponding brightness values are calculated (with a accuracy of the normalization factor) at these nodes. As one can see, these values are used to a large extent to calculate the coefficients F(2) - F(8) of Hadamar transforms.

In the general case, provided the pyramidal-recursive structure is constructed already, it takes

$$K = N \log_2 N - N\sum_{t=0}^{m-1} 2^{-t} \approx N(\log_2 N - 1)$$

operations to calculate the transform coefficients using fast algorithms.

It is of importance, in this case, that the pyramidal-recursive representation reduces the number of operations so as to determine only the high-order coefficients, i.e. those which have the greatest values. Thus, to calculate the first $2^t < 2^m = N$ coefficients it is sufficient to perform

$$K_t = N_t(\log_2 N_t - 1) = 2^t(\log_2 2^t - 1) = 2^t(t-1) \qquad (3.1.1)$$

operations. Since the multidimensional Hadamar transforms are separable, for transforms of an information field of arbitrary dimension p, instead of (3.1.1), we have:

$$K_t = 2^{tp}(t-1)^p;$$

that is, $2^{pm}(2^p/(2^p-1))$ operations less than for fast Hadamar transform for obtaining the same coefficients.

The results can be repeated for some other orthogonal transforms also. Without actually presenting this, we illustrate this point for Haar transforms, using the graph of fast Haar transforms for p=1, and m=3 (Fig. 3.2). Haar coefficients also can be obtained from the values of the pyramidal-recursive structure for the initial sequence $s_1,\ldots,s_8$. The solid lines in Fig. 3.2 indicate once more the intermediate calculations performed for the construction of the pyramidal-recursive structure. Haar transforms coefficients are calculated by performing only two operations per coefficient with the values of the structure pixels .

Thus, the pixels from the image pyramid are strictly connected to Hadamar and Haar transform coefficients. However, the transform coefficients characterize integrally the whole image, though pixels of different levels characterize image

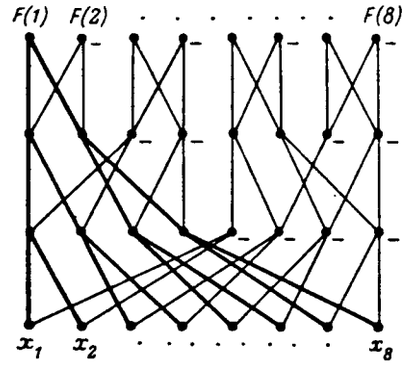


Fig. 3.1. Graphs of Hadamar transforms and a pyramidal-recursive structure (bold lines).

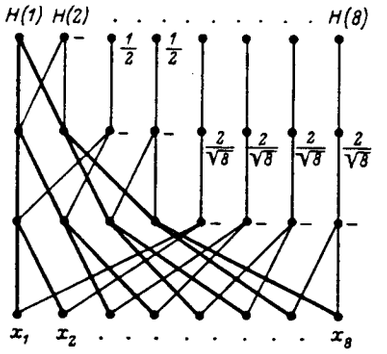


Fig. 3.2. Graphs of Haar transforms and a pyramidal-recursive structure (bold lines).

fragments of different size and can be interpreted as transforms coefficients (or their linear functions) for image fragments.

The expressions which enable to go from the values of the pixels of pyramidal-recursive structure to the values of Hadamar transform coefficients and back again are given in [57].

For the one-dimensional case, we have

$$F(0) = \sqrt{N}\, s_0^o$$

$$F(u) = \frac{\sqrt{N}}{M}\sum_{j=0}^{M-1} s_j^t(-1)^{q(jL,u)}, \qquad (3.1.2)$$

where $2^{t-1} \le u < 2^t$, $M=2^t$, $L=2^{m-t}$,

$$q(j, u) = \sum_{j=0}^{m-1} j\,[i](u\,[m-i\,]+[m-i-1\,])_{mod2} + u\,[m-1\,],$$

and where $u[i]$ and $j[i]$ are the i-th bits of the binary representation of u and j. It follows from (3.1.2) that each t-th level of the pyramidal-recursive structure enables the determination of coefficients of 2^t high-order Hadamar transforms having the values $u=0,...,2^{t-1}-1$. As for the previous $(t-1)$-th structure level, Hadamar transform coefficients with numbers $u=0,...,2^{t-1}-1$ can be determined, then the transfer from the $(t-1)$-th level to the t-th level yields the 2^{t-1} coefficients with numbers $u=2^{t-1},...,2^t-1$.

For the first 2^t high order coefficients $F(u)$ pixels s_j^t, $j=0,...,2^t-1$ are, as can be seen from (3.1.2), an initial signal. Then, according to the Parceval theorem,

$$\sum_{u=0}^{2^t-1} (F(u))^2 = \sum_{u=0}^{2^t-1} (s_j^t)^2$$

Hence, the energy falling upon the t-th level of the structure equals the energy corresponding to 2^t high order Hadamar transform coefficients, or, which is the same, the mean square error of the numerical sequence (one-dimensional image) approximation by the t-th level image from the pyramid equals the mean square error of this sequence restoration with the use of 2^t high order Hadamar transform coefficients.

The one-dimensional reverse Hadamar transforms in this case can be written as:

$$s_j^t = \frac{1}{\sqrt{N}} \sum_{n=0}^{2^t-1} F(u) \quad (-1)^{q(jL,u)}, \quad L=2^{m-t}. \qquad (3.1.3)$$

This shows that for determining the structure element s_j^t it is necessary to know only 2^t high order one-dimensional Hadamar transform coefficients.

As the multidimensional Hadamar transforms are separable and can be performed as p sequential one-dimensional transforms for each of the Cartesian coordinates, the relations (3.1.2) and (3.1.3) are easily generalized for the p-dimensional case.

For direct transforms one can obtain:

$$F(u_1,...,u_p) = \frac{\sqrt{N}}{M^p} \sum_{j_1=0}^{M-1} \cdots \sum_{j_p=0}^{M-1} s_{j_1...j_p}^t (-1)^{q(j_1L,...,j_pL,u_1,...u_p)},$$

where $2^{t-1} \le u < 2^t$.)

The reverse transforms are described as:

$$s_{j1...jp}^t = \frac{1}{\sqrt{N}} \sum_{u_1=0}^{M-1} \cdots \sum_{u_p=0}^{M-1} F(u_1,...,u_p)\,(-1)^{q(j_1L,...,j_pL,u_1,...u_p)}.$$

The mean square error of an image approximation by the t-th level image from the pyramid, as well as in the one-dimensional case, equals the field restoration error using 2^{pt} "left-lower" Hadamar transform coefficients with $0 \le u_l < 2^t$, $l=1,...,p$. In particular, for a two-dimensional field of $2^m x 2^m$ pixels, the energies corresponding to sequential levels of a pyramid structure equals the energies of Hadamar coefficients found in the shaded zones of the transform matrices, Fig. 3.3. Thus, pyramidal structure as a whole exhibits such an important property as energy concentration in upper levels.

Now, let us consider representation at each level which is determined by the properties of the respective scannings or ordering of pixels. Let us show that if a two-dimensional image of each level is scanned by a recursive quasicontinuous scan of the Peano-Hilbert type (Section 2.2), then such a pixel sequence acquires some statistical features of a two-dimensional field. Let us consider for this correlation functions and power spectra of both the initial two-dimensional and the one-dimensional signals obtained after scanning. Coincidence of their properties will indicate that integral transforms on an image (or on each level of the pyramid of images) can be performed using a one-dimensional sequence of pixels, but not the initial array.

A wide class of images can be described using a Markov model. It is considered in this case that correlation between two pixels in a row (column) depends only upon the distance d between these elements and equals r^d, where r is a correlation coefficient between neighboring elements. As is shown by experiment [102], the real images have correlation coefficients r over rows and columns which are in agreement with the Markov autocorrelation function for the range $r = 0.9 - 0.97$. Let us consider a two-dimensional separable Markov field for which the correlation coefficients of the neighboring pixels along the rows and columns are similar. The correlation matrix for a row (or column) of a Markov image is:

$$W = \begin{pmatrix} 1 & r & r^2 & \dots & r^{\sqrt{N}-1} \\ r & 1 & r & \dots & r^{\sqrt{N}-2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r^{\sqrt{N}-1} & r^{\sqrt{N}-2} & & \dots & 1 \end{pmatrix}, \qquad (3.1.4)$$

where $\sqrt{N} = 2^m$.

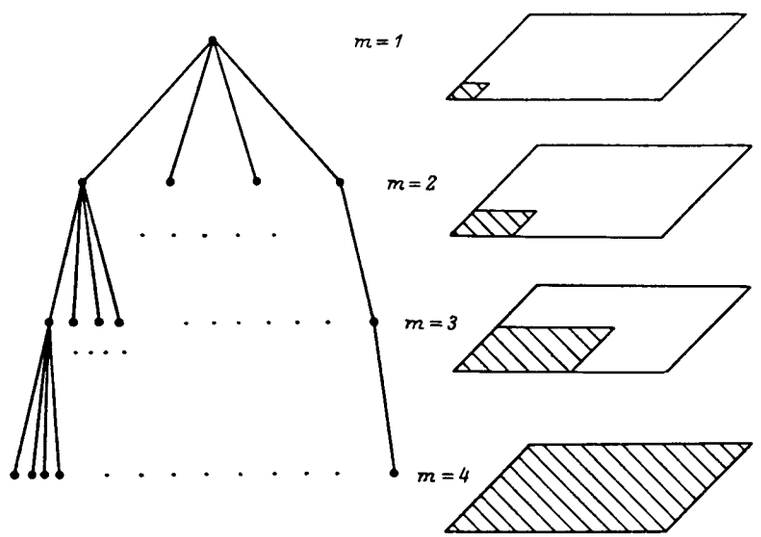


Fig. 3.3. Energy at each level of the pyramid structure equals the energy of a certain zone of the Hadamar transform coefficients matrix .

Let us scan this field into a one-dimensional sequence of brightness samples by two methods: the Hilbert scan and the TV-scan, denote these sequences as $u_H(T)$ and $u_V(T)$, $T=1,...,N$, and find their autocorrelation functions.

For the sequence $u_V(T)$ the correlation matrix (i.e. the direct product of the $u_V(T)$ sequence considered as a vector by itself) is:

$$K_V = \begin{pmatrix} W & rW & \cdots & r^{\sqrt{N}-1}W \\ rW & W & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ r^{\sqrt{N}-1}W & \cdots & \cdots & W \end{pmatrix},$$

where W is determined from (3.1.4) [60].

To estimate the τ-th sample of the autocorrelation function $R_V(\tau)$, $\tau=0,...,N-1$ it is sufficient to average all elements of K_V situated at a distance τ from the main diagonal:

$$R_V(\tau) = r^h((\sqrt{N}-h-1)(r^c(\sqrt{N}-c)+r^{\sqrt{N}-c+1}c)+r^{c(\sqrt{N}-c)})/(N-\tau),$$

where $\tau=0,...,N-1$, $h = entier(\tau/\sqrt{N})$, $C = \tau - h\sqrt{N}$.

An estimation of $R_H(\tau)$ can be obtained by calculating $\Omega(\tau)$ - the average Hamming distance between elements of a multidimensional array with the distance τ in a one-dimensional sequence $u_H(T)$:

$$R_H(\tau) = r^{\Omega(\tau)}.$$

The upper boundary for $\Omega(\tau)$ is given by the expression (2.2.2), but following the way used to derive this expression (see [9], p.48), the average value of the distance $\Omega(\tau) \approx 1.1\sqrt{\tau}$ can be found with a large enough value. Then one obtains

$$R_H(\tau) \approx r^{1.1\sqrt{\tau}}$$

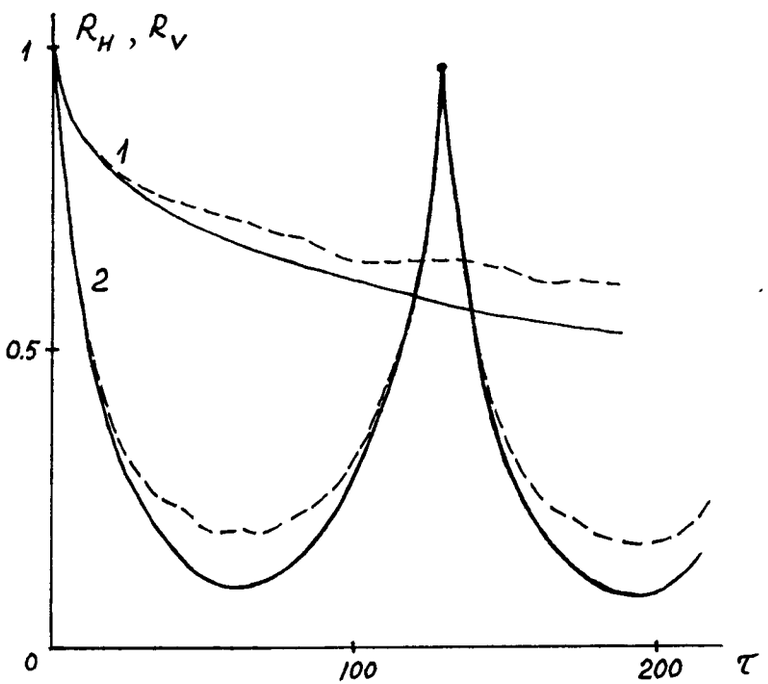


Fig. 3.4. Theoretical (solid lines) and experimental (dotted lines) autocorrelation functions of an image scanned by Hilbert scan (1) and TV-scan (2).

Figure 3.4 shows graphs of the functions $R_H(\tau)$ and $R_V(\tau)$ for the Markov field of 128x128 elements with r = 0.95. The same graph shows similar autocorrelation functions for the real image of 128x128 elements with 256 grey levels and correlation coefficients in rows and columns equal to 0.954 and 0.945, respectively. The first conclusion to be drawn based on this graph is that the Hilbert scan is much more efficient in so far the preservation of correlations of pixels is concerned.

The exponential decrease of the autocorrelation function $R_H(\tau)$ with the exponent index $\sqrt{\tau}$ (but not τ) shows that the one-dimensional sequence $u_H(\tau)$ has actually the statistical features of a two-dimensional array. Really, if the vicinity contains τ elements in a two-dimensional Markov field, then the correlation coefficient between the central element and that on the boundary approximately equals $r^{\sqrt{\tau}}$; just the same takes place for a one-dimensional Hilbert scanned sequence. The second conclusion is that the theoretical Markov model agrees well with real experimental data.

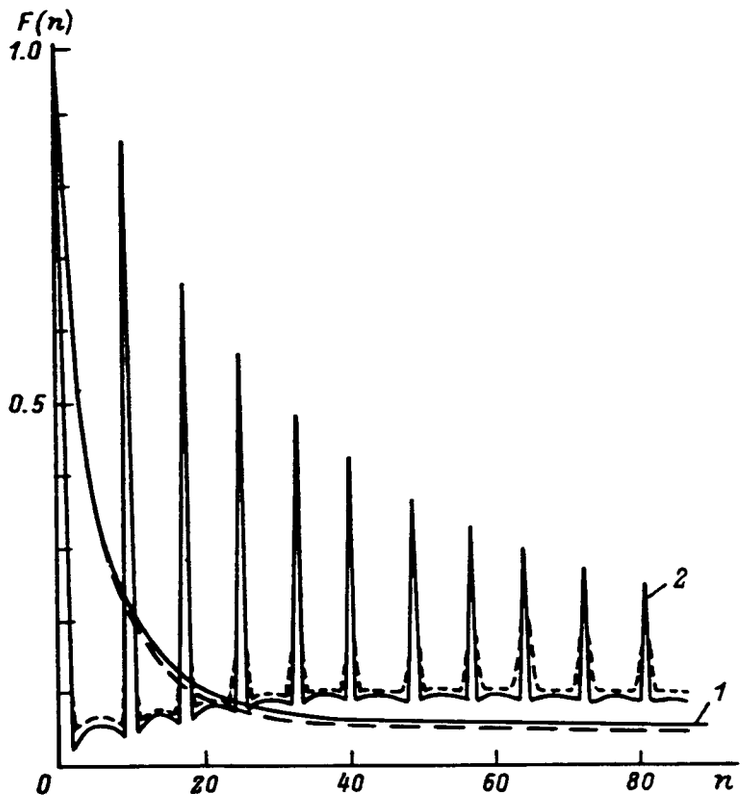


Fig. 3.5. Theoretical (solid lines) and experimental (dotted lines) power spectra of an image scanned by Hilbert scan (1) and TV-scan (2).

Figure 3.5 shows graphs of energy spectra obtained by Fourier transform of autocorrelation functions shown in Figure 3.4. It is evident that the energy concentration rate at low frequencies is higher for the sequence $u_H(T)$. This concentration rate is rather close to those resulting from a calculation of the

two-dimensional energy spectrum of the initial image (compare the curves shown in Fig. 3.6).

These curves show a dependence of the residual dispersion (initial image restoration error) from a number of the Fourier transform coefficients used in the reconstruction. In the first case, n high order coefficients of *one-dimensional transforms* of the sequence $u_H(T)$ have been taken; in the second case, n coefficients have also been taken, but representing a nxn zone of the highest energy coefficients of *a two-dimensional transform*.

Thus, it can be concluded that the pyramidal-recursive structure, having level elements being ordered by a quasicontinuous recursive scan, yields integral (energy) features of images as good as the well-known two-dimensional transforms used for image processing.

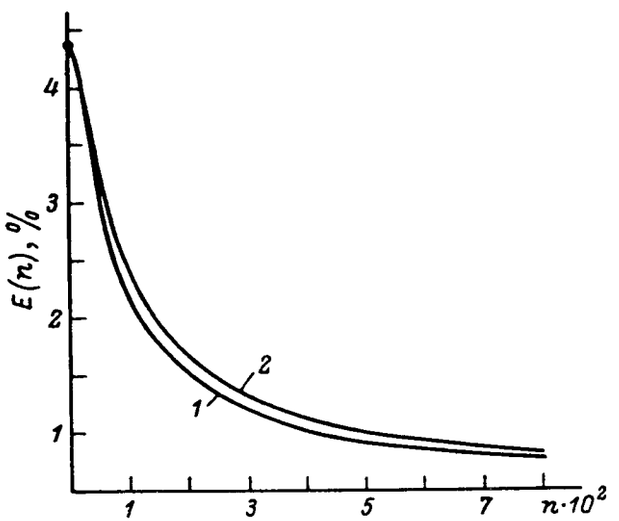


Fig. 3.6. Error of image restoration depending on the number of coefficients of: two-dimensional Fourier transforms (1); and one-dimensional Fourier transforms along the Hilbèrt scan (2)

3.2. The Model of Greyscale Image

A Markov model of an information field is often used in image processing [102] making possible the qualitative and sometimes quantitative theoretical analysis of methods and algorithms being developed. This section deals with the pyramidal-recursive representation of a separable p-dimensional Markov field. As is shown below, the results obtained are in agreement with experimental data; they are of importance for image coding and image transmission algorithms (Chapter 4), and fast template matching algorithms (Chapter 5).

Let a p-dimensional greyscale image (Markov field) be represented by a pyramidal-recursive structure as described in Section 2.5, with the t-th level pixel being the average value of its direct descendent pixels of the $(t+1)$-th level, i.e. the condition (2.5.4) is satisfied. Considering the brightness of each pixel of an initial image to have a random value, suppose, that dispersions of the pixel's brightness are known, which are equal to each other and equal to some value D. The correlation coefficient between the brightnesses of neighboring pixels is also supposed known, and to be independent of their position and equal to r. Let us find the dispersion and the correlation coefficient of the t-th level pixels.

Let us analyze first an elementary cell of a structure at the m-th level and then summarize the results obtained. Denote subordinated pixels as $s_1,\dots,s_{k^p}$, and their average value as s_0 (Fig. 3.7). Find the dispersion of s_0 (denote it as D_p). Taking into account the known theorem relating to the dispersion of a sum of random values and the fact that $D(s_i)=D$ for $i=1,\dots,k^p$, one obtains:

$$D_p = k^{-2p}\Big(\sum_{i=1}^{k^p} \mathbf{D}(s_i) + 2D\sum_{i<j} \frac{\mathbf{K}(s_i,s_j)}{\sqrt{\mathbf{D}(s_i)\mathbf{D}(s_j)}}\Big) = \frac{D}{k^{2p}}\sum_{i=1}^{k^p}\sum_{j=1}^{k^p} r(s_i,s_j) \qquad (3.2.1)$$

where $\mathbf{K}(s_i, s_j)$ is the correlation moment of values s_i and s_j, and $r(s_i, s_j)$ is the respective correlation coefficient. It is known (see [102], for example) that for the Markov field $r(s_i, s_j)$ depends only on the correlation coefficient r and the distance between the elements s_i, s_j. Taking this into account, one can find D_p.

For a one-dimensional Markov field $r(s_i, s_j)=r^{|i-j|}$ the calculation of the sum in (3.2.1) is equivalent to summation of the elements of the matrix:

$$\begin{pmatrix} 1 & r & r^2 & \dots & r^{k-1} \\ r & 1 & r & \dots & r^{k-2} \\ r^2 & r & 1 & \dots & r^{k-3} \\ \dots & \dots & \dots & \dots & \dots \\ r^{k-1} & r^{k-2} & & \dots & 1 \end{pmatrix}$$

and the sum can be rewritten in the following way:

$$\sum_{i=1}^{k}\sum_{j=1}^{k} r^{|i-j|} = 2\sum_{i=1}^{k}(k-i)r^i + k = k\frac{1-r}{1+r} - 2r\frac{1-r^k}{1-r^2} \qquad (3.2.2)$$

Substituting (3.2.2) into (3.2.1), one finds for $p=1$:

$$D_1 = \frac{D(k(1-r^2)-2r(1-r^k))}{k^2(1-r)^2} \qquad (3.2.3)$$

Note 1. Due to (2.5.3), the t-th level pixel is an average value both for subordinated pixels of the $(t+1)$-th level and for the subordinated pixels of the initial image. Therefore, the expression (3.2.3) allows to calculate also the dispersion of the pixel of level t, if we put the value k^{m-t} into (3.2.3) instead of k.

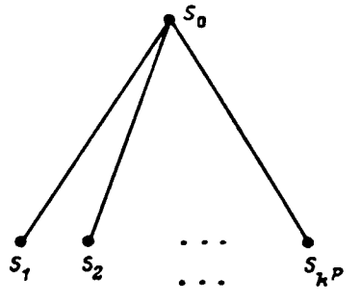


Fig. 3.7. Denotation for the inference of (3.2.1)

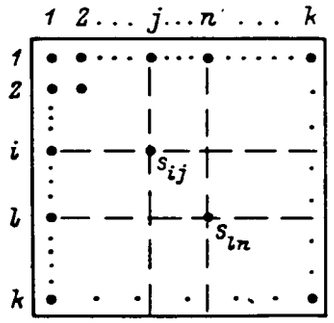


Fig. 3.8. Denotation for the inference of (3.2.4).

In the case of a two-dimensional field (p=2), it is convenient to rewrite (3.2.1) as:

$$D_2 = Dk^{-4}\sum_{}^{k}\sum_{}^{k}\sum_{}^{k}\sum_{}^{k} r(s_{ij}s_{ln}),$$

where s_{ij} and s_{ln} are the brightnesses of pixels of size kxk of a two-dimensional fragment (Fig. 3.8). For the separable Markov field $r(s_{ij}, s_{ln}) = r^{|i-l|+|j-n|}$ using (3.2.2), one can write:

$$D_2 = Dk^{-4}\sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{l=1}^{k}\sum_{n=1}^{k} r^{|i-1|+|i-n|} = \frac{D}{k^4}\left(k\frac{1+r}{1-r} - \frac{2r(1-r^k)}{(1-r)^2}\right)^2 \qquad (3.2.4)$$

Generalizing this result for a p-dimensional field, we have

$$D_p = Dk^{-2p}\left(k\frac{1+r}{1-r} - \frac{2r(1-r^k)}{(1-r)^2}\right)^p \qquad (3.2.5)$$

As Note 1 is valid in this case too, the expression (3.2.5) is applicable to the calculation of the dispersion of a pixel's brightness at any level t, $t=1,...,m-1$. For this, it is enough to put k^{m-t} into (3.2.5) instead of k. Then, the general expression for the dispersion depending on the image dimension p, the level number t, and the decomposition base k can be written as:

$$D(p, k, t) = D\left(\frac{\alpha(1-r^2) - 2r(1-r^\alpha)}{\alpha^2(1-r)^2}\right)^p, \qquad (3.2.6)$$

where $\alpha = k^{m-t}$.

Figure 3.9 shows $D(p, k, t)$ as a function of t for different correlation coefficients r with (a) $p=1$, (b) $p=2$, and (c) $p=3$. The decomposition base k equals 2 (solid lines) or 3 (dotted lines). The graphs show that images of lower levels from the pyramid have nearly the same brightness dispersion as the initial one, especially with a high value of r.

Let us determine the correlation coefficient between pixels of the t-th level. The same as for determining the dispersion, first we analyze this problem for two elementary structure cells. Let us denote the brightnesses of subordinate pixels in these cells as $s_1,...,s_{kp}$ and $s_1',...,s_{kp}'$, and their average values as s_0 and s_0' (Fig. 3.10). Then, considering that $D(s_i) = D(s_i') = D$ and $D(s_0) = D(s_0') = D_p$, one can obtain:

$$r_p(s_0, s_0') = \frac{M(s_0 s_0') - M(s_0)M(s_0')}{D_p} = \frac{1}{k^{2p}D_p}\sum_{t=1}^{k^p}\sum_{j=1}^{k^p}\left(M(s_j s_j') - \right.$$

$$\left. - M(S_j)M(S_j')\right) = \frac{D}{D^p k^{2p}}\sum_{i=1}^{k^p}\sum_{j=1}^{k^p} r(S_j, S_j') \qquad (3.2.7)$$

For a one-dimensional Markov field $r(s_i, s_j') = r^{|k-i-j|}$, putting this value into (3.2.7) and using (3.2.3), one can obtain:

$$r_1(s_0, s_0') = \frac{r(1-r^k)^2}{k(1-r^2) - 2r(1-r^k)}. \qquad (3.2.8)$$

In the case of a two-dimensional field (p=2):

$$r_2(s_0, s_0') = \frac{D}{D^2k^4}\sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{l=1}^{k}\sum_{n=1}^{k} r(s_{il}, s_{jn}'),$$

where s_{il}, s_{jn}' are brightnesses of pixels in two neighboring two-dimensional arrays each of size k x k (Fig. 3.11). For a separable Markov field $r(s_{il}, s_{jn}') = r^{k-i+j+|l-n|}$, using (3.2.2) and (3.2.4) we have:

$$r_2(s_0, s_0') = \frac{r^k\sum_{i=1}^{k} r^{-i}\sum_{j=1}^{k} r^j}{\sum_{i=1}^{k}\sum_{j=1}^{k} r^{|i-j|}} = \frac{r(1-r^k)^2}{k(1-r^2) - 2r(1-r^k)} = r_1(s_0, s_0').$$

The same result can be found for the p-dimensional field (under the condition that two p-dimension arrays of size kx...xk have a common "side" of the order $p-1$), i.e. (3.2.8), is actually independent on the dimension of an image. In particular, with $k=2$:

$$r_p = r(1+r)/2.$$

Taking into account Note 1, expression (3.2.8) can be rewritten as an expression for the correlation of neighboring pixels of the t-th level for any p and $t=1,...,m-1$:

$$r(k, t) = \frac{r(1-r^\alpha)^2}{\alpha(1-r^2) - 2r(1-r^\alpha)}, \qquad (3.2.9)$$

where $\alpha=k^{m-t}$. We shall call the $r(k, t)$ value *the inside-level correlation*.

Figure 3.12 shows graphs of $r(k, t)$ for different t and r values, and $k=2$ (solid lines) and $k=3$ (dotted lines). The graphs show that low level images from the pyramid have great statistical redundancy, nearly the same as the initial image. This proves that images of lower levels faithfully copy the statistical structure of the initial image making possible their use to obtain an approximate result while processing the pyramid structure top-down level-by-level.

Now, let us determine the *interlevel correlation* - the correlation coefficient between the brightnesses of a pixel of the $(t-1)$-th level $s(i_1...i_{t-1})$ and its direct descendent at the t-th level $s(i_1...i_t)$ (denote this correlation coefficient $r^*(p, k, t)$). Let us consider an elementary cell of the structure (Fig. 3.7):

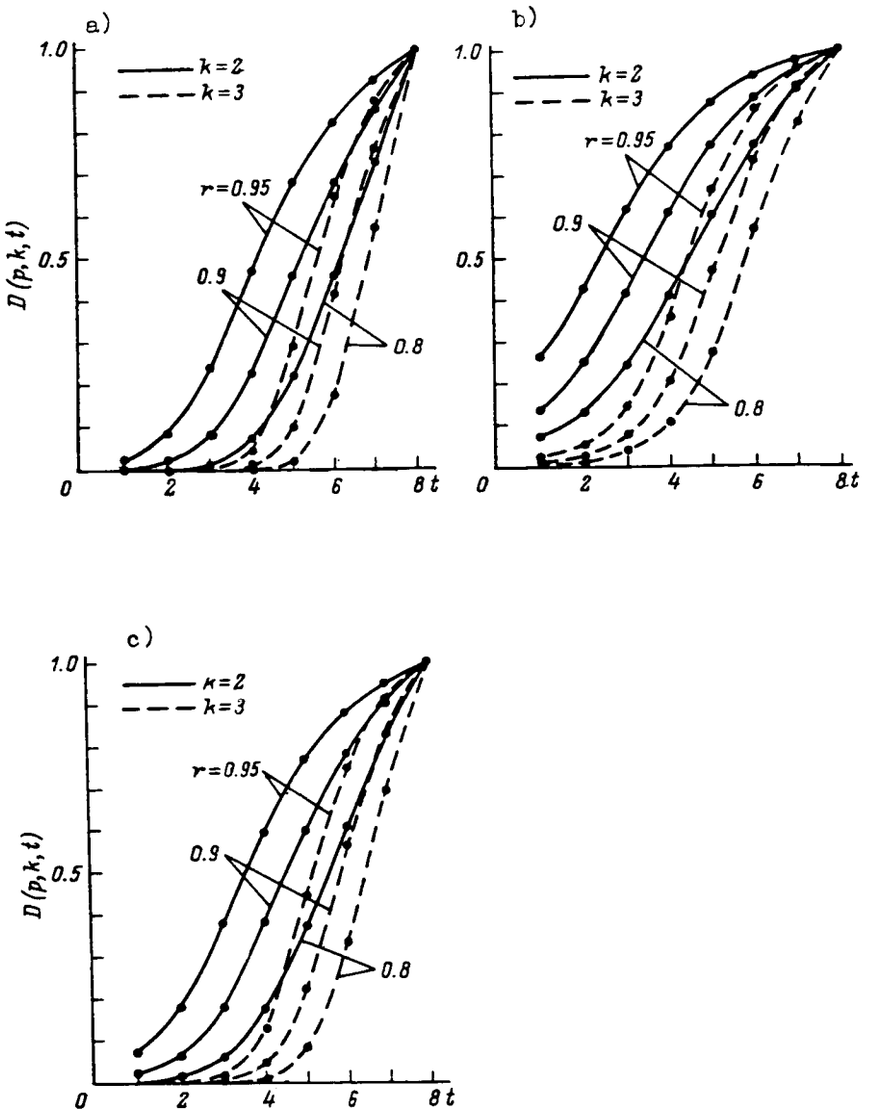


Fig. 3.9. Brightness dispersion at various pyramid levels: (a) p=1, (b) p=2, (c) p=3.

Fig. 3.10. Denotations for the inference of (3.2.8).

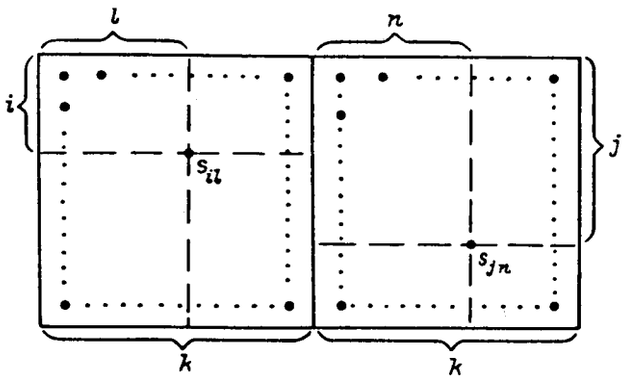


Fig. 3.11. Denotations for the inference of (3.2.9).

$$r(s_0, s_i) = \frac{\mathbf{M}(s_0,s_i)-\mathbf{M}(s_0)\mathbf{M}(s_i)}{\sqrt{\mathbf{D}(s_0)\mathbf{D}(s_i)}} = \frac{k^{-p}\sum_{j=1}^{k^p} \mathbf{M}(s_i s_j)-\mathbf{M}(s_1)\sum_{j=1}^{k^p} \mathbf{M}(s_j))}{\sqrt{\mathbf{D}(s_0)\mathbf{D}(s_1)}}$$

$$= k^{-p}\frac{\sqrt{\mathbf{D}(s_i)}}{\sqrt{\mathbf{D}(s_0)}} \sum_{j=1}^{k^p} r(s_i,s_j)$$

By averaging over all elementary cells of the same level and considering that s_i is the t-th level pixel, s_0 is the $(t-1)$-th level pixel, and the index i is of k^p different values, we have:

$$r^*(p, k, t)=\mathbf{M}(r(s_0, s_i))=$$

$$k^{-2p}\left(\frac{D(p,k,t)}{D(p,k,t-1)}\right)^{p/2} \left[\sum_{i=1}^{k^p} \sum_{j=1}^{k^p} r(s_i,s_j)\right].$$

As follows from (3.2.1), the factor in square brackets is the ratio of the dispersion of the $(t-1)$-th and t-th levels. Then:

$$r^*(p,k,t)=\left(\frac{D(p,k,t-1)}{D(p,k,t)}\right)^{p/2} \qquad (3.2.10)$$

Substituting $D(p, k, t)$ from (3.2.6), one obtains:

$$r^*(p,k,t)=\left(\frac{k\alpha(1-r^2)-2r(1-r^{k\alpha})}{k^2(\alpha(1-r^2)-2r(1-r^{\alpha}))}\right)^{p/2}, \qquad (3.2.11)$$

where $\alpha=k^{m-t}$.

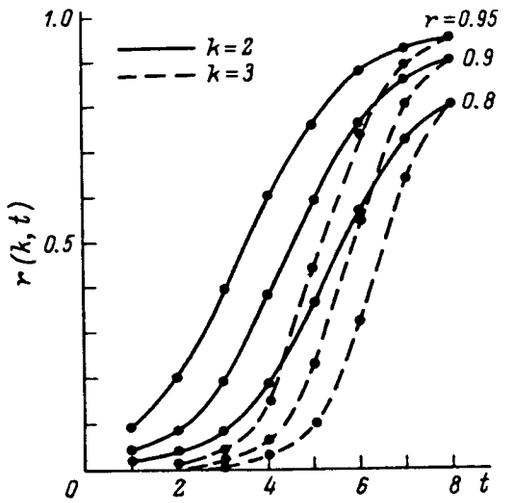


Fig. 3.12. Inside-level correlation against the level number.

Figure 3.13 shows graphs of $r^*(p, k, t)$ for different p, k and r values. As the graphs show, the interlevel correlation is practically always higher than that of the inside-level (compare Fig. 3.13 with Fig. 3.12). In particular, for $k=2$ (the most useful case), (3.2.11) can be rewritten as:

$$r^*(p,2,t)=\left(\frac{1}{2}\left(1+\frac{r(1-r^{\alpha})^2}{\alpha(1-r^2)-2r(1-r^2)}\right)\right)^{p/2}=\left(\frac{1+r(2,t)}{2}\right)^{p/2} \qquad (3.2.12)$$

where $r(2, t)$ is the inside-level correlation determined from (3.2.9). The analysis of (3.2.12) shows that for $p=1, 2, 3, 4$ the interlevel correlation is *always more* than the inside-level one; for instance, with $p=2$ and $t=m$ $r^*(2,2,m)=(1+r)/2$. With $p>4$, $r(k,t)$ and $r^*(p,k,t)$ relation depends upon r. Hence, for most useful image dimensions and decomposition bases, the interlevel correlation dominates, and it is this that should be used in algorithms of image data compression.

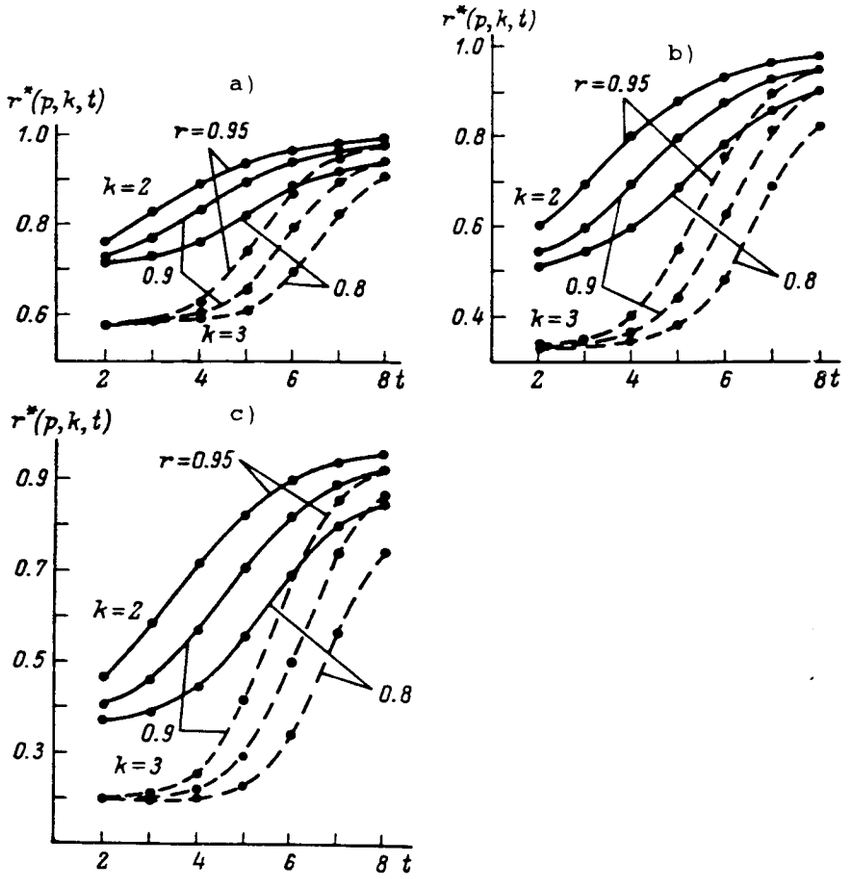


Fig. 3.13. Interlevel correlation against the level number for (a) one-dimensional, (b) two-dimensional, and (c) three-dimensional images.

Figure 3.14 shows graphs of $D(2,2,t)$, $r(2,t)$ and $r^*(2,2,t)$ plotted against the level number $t = 3,...,7$. The solid lines shows the theoretical dependencies determined from (3.2.6), (3.2.9), (3.2.11) with $r=0.95$, the dotted lines indicate similar dependencies for real images of 128x128 elements with practically the same correlation coefficients $r=0.94$-0.96. As the figure shows, the pyramidal model of the greyscale image based on a Markov field abstraction provide a good enough description of the important statistical characteristics of real images represented by pyramidal structures, the best coincidence being achieved at the most informative lower levels.

Another characteristic important for further image processing algorithms is the difference in the brightnesses of pixels from adjacent levels, which belong to the

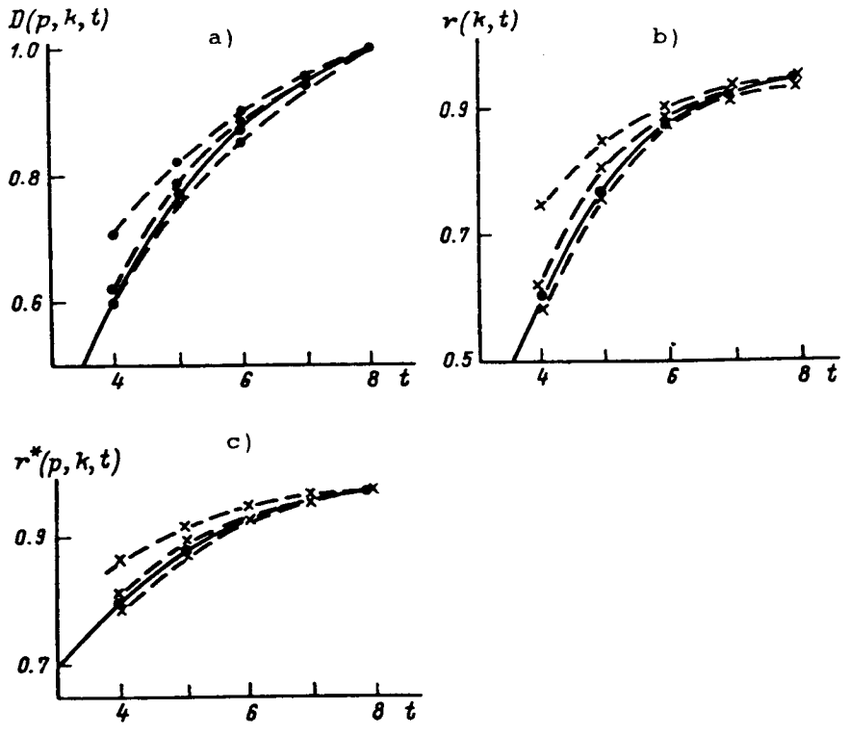


Fig. 3.14. Comparison of predicted (solid lines) and experimental (dotted lines) dispersions (a), inside-level correlations (b), and interlevel correlations (c) against the level number for two-dimensional images.

same elementary cell. Let us find the dispersion of this difference (Z_t), provided pixels are at the t-th and $(t-1)$-th pixel.

For the m-th level in Fig. 3.7, one can find:

$$Z_m = \underset{i}{\mathrm{M}}(\mathrm{D}(s_i - s_0)) = \underset{i}{\mathrm{M}}(\mathrm{D}(s_i)) + \underset{i}{\mathrm{M}}(\mathrm{D}(s_0)) - 2\underset{i}{\mathrm{M}}(\mathrm{M}(s_0 s_i) - \mathrm{M}(s_0)\mathrm{M}(s_i)) =$$

$$= D + \mathrm{D}(s_0) - 2D\underset{i}{\mathrm{M}}\left((k^{-p} \sum_{j=1}^{k^p} r(s_i, s_j)\right) \qquad (3.2.13)$$

As i have k^p different values, then according to (3.2.1):

$$\underset{i}{\mathrm{M}}\left(k^{-p} \sum_{j=1}^{k} r(s_i, s_j)\right) = k^{-2p} \sum_{i=1}^{k^p} \sum_{j=1}^{k^p} r(s_j, s_j) = \mathrm{D}(s_0)/D.$$

Then (3.2.13) can be rewritten in the following way:

$$Z_m = D - \mathrm{D}(s_0) = D - D(p, k, m-1).$$

By similar reasoning, one can get for the t-th level pixels (in this case $s_t = s(i_t \ldots i_t)$, $s_0 = s(i_1 \ldots i_{t-1})$):

$$Z_t = D(p, k, t) - D(p, k, t-1),$$

where $D(p, k, t)$ is determined according to (3.2.6). Using this expression, we get finally for a p-dimensional Markov field:

$$Z_t = D\left(\left(\frac{\alpha(1-r)^2 - 2r(1-r^{\alpha})}{\alpha^2(1-r)^2}\right)^p - \left(\frac{k\alpha(1-r^2) - 2r(1-r^{k\alpha})}{k^2\alpha^2(1-r)^2}\right)^p\right)$$

where $\alpha = k^{m-t}$. It follows, in particular, from (3.2.14) that

$$\sum_{t=1}^{m} Z_t = D.$$

Fig. 3.15 shows graphs of Z_t (in fractions of the original image dispersion D) as a function of level number t for $k=2$, different correlation coefficients r, and image dimension p. The graphs show that the dispersion of the pixel difference is much less than that for the initial image. For r = 0.9 the dispersion Z_t decrease very fast with growth of level number, which is an indication of great information redundancy of lower pyramid levels.. This result is of great significance for image data compression with pyramidal structures.

3.3. The Model of the Binary Image

The variety of binary images is very large (meteo maps, texts, graphs, solids in space, etc.), therefore, a universal theoretical model for all types of graphics can hardly be found. The present section considers the simplest models of a binary image and describes the experimental results for some image types. Complexity and space efficiency of truncated pyramidal-recursive structures representing p-dimensional binary images (quadtrees for $p=2$, octrees for $p=3$ or k^p-trees for arbitrary k and p) are investigated. Theoretical models that describe quantities of grey, black and white nodes at different levels of the structure for several type of images are presented.

Quad-, oct- and k^p-trees have been used for image coding, transmission, feature extraction, pattern recognition, etc. (see, for instance, [87, 113, 149]). Available algorithms are mainly based on sequential or parallel traversing of tree

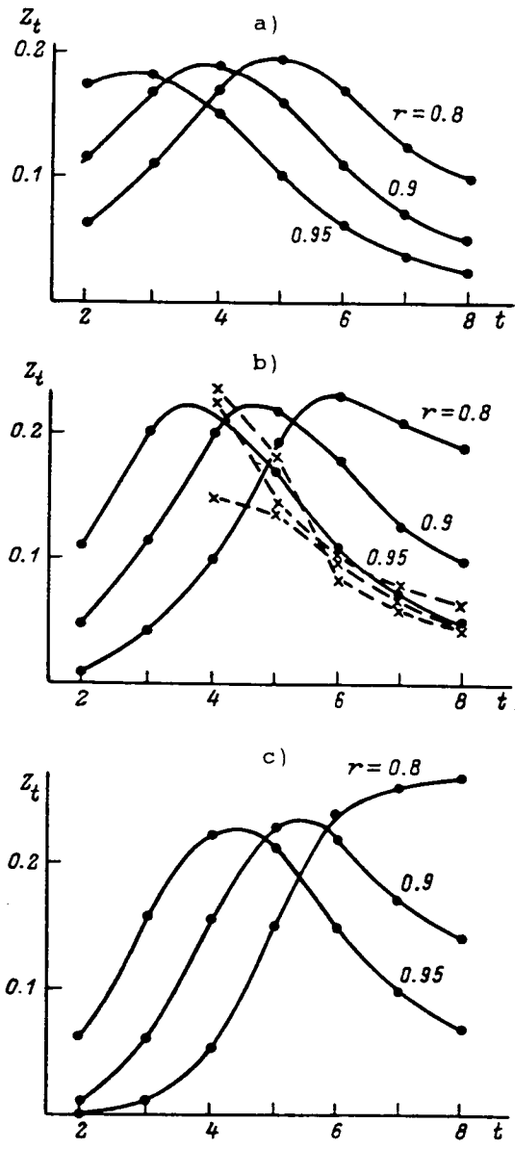


Fig. 3.15. Dispersion of brightness difference for pixels of adjacent pyramid levels: (a) one-dimensional, (b) two-dimensional, (c) and three-dimensional images; solid lines denote the prediction according (3.2.14), dotted lines - experimental data for images with r=0.94 - 0.96.

nodes, and analyze data connected with these nodes [49, 76, 113]. The complexity of such algorithms directly depends on either the total number of nodes in a tree or on the number of "black" nodes corresponding to the black areas of an image. If the number of nodes of each type is known, then the complexity of processing algorithms and their computational efficiency could be predicted.

Another important task is the image coding and transmission of compressed image data. Pyramidal-recursive structures provide a high compression ratio in picture coding and a gradual refinement of the transmitted image during its reconstruction at the receiving side [44, 76, 123]. If the number of nodes at various structure levels were known for different types of images, then the space efficiency of k^p-trees and reconstruction errors during the refinement process could be predicted.

Thus, the development of k^p-tree models describing quantities of black, white and grey nodes at different levels seems to be important. The fact that such models depend on some parameters appears to be related to image properties such as the square and perimeter of black or white areas, the dimension of the boundaries of black areas, etc., which are often used as image features. Having a k^p-tree representation for a particular image enables the parameters of the model to be identified, and also the corresponding image features.

Let us consider a pyramidal-recursive structure (k^p-tree) representing a p-dimensional binary image of size $k^m \times .. \times k^m$, in which every terminal node is black or white and nonterminal nodes are grey (Fig. 2.10). Let the number of grey nodes be equal to G. Every grey node has exactly k^p descendants in a k^p-tree, so there are in total $k^p G$ descendants in a tree. Every node is a descendant of some other node except the root, which means that the total number of nodes in a k^p-tree is equal to

$$M = k^p G + 1 \qquad (3.3.1)$$

To estimate a quantity of grey nodes let us investigate two models of simple binary images (Fig. 3.16). First, consider a two-dimensional case for k=2.

The image at Fig. 3.16a will be called a "region" and the image at Fig. 3.16b, a "line". A line is a discrete analog of a straight line segment (pixels crossed by the segment are black). A region is a discrete analog of the part of a square restricted by a straight line (pixels crossed by the boundary are black if their centers are in the black area).

Suppose a quadtree for a region is constructed and consider a t-th level pixel (cell) which is crossed by the straight line P (such a cell corresponds to a grey node in a quadtree). This cell consists of four cells of $(t+1)$-th order (Fig. 3.17a).

Let the projection of a t-th order cell onto the normal to P have a length l, then

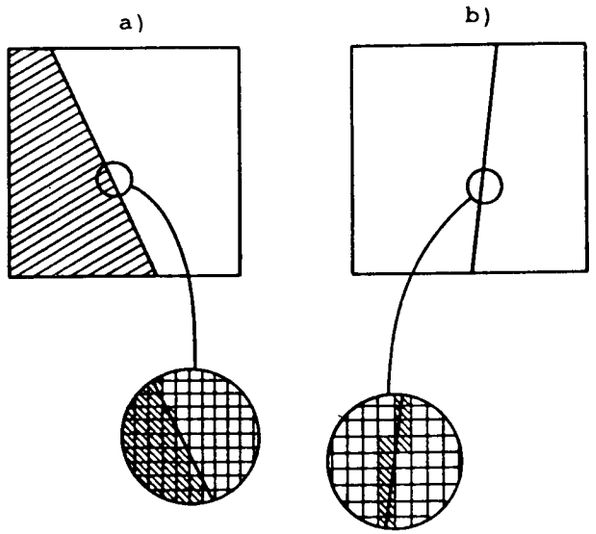


Fig. 3.16. Simple images: a region (a) and a line (b).

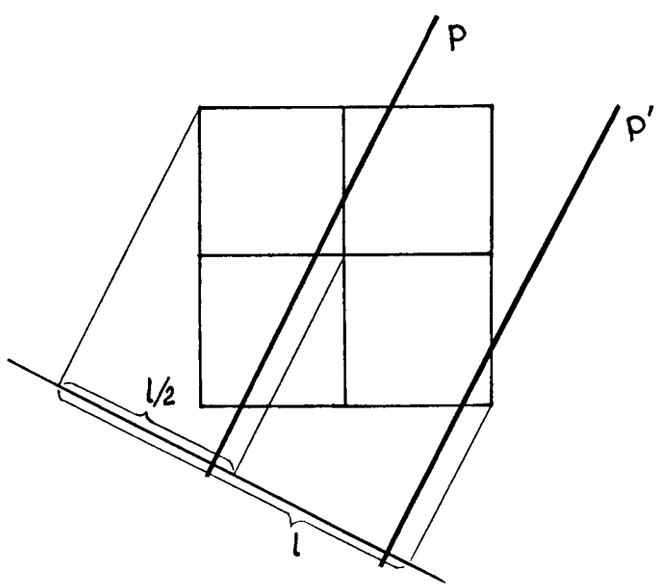


Fig. 3.17. To the calculation of the average number of (t+1)-th order cells crossed by straight line.

a the projection of any *(t+1)*-th order cell onto this normal has a length *l/2*. Let us

build an additional straight line *P'* at the distance *l/2* from *P* and parallel to it. It is obvious that every *(t+1)*-th order cell is crossed either by *P* or *P'* (a cell can have one common point with *P* and one with *P'* but relative quantity of such cells equals to zero when *P* is arbitrarily positioned). Thus, two lines cross all cells. Then every line *P* or *P'* crosses on average half the *(t+1)*-th order cells (i.e. crosses two of them) taking into account possible shifts of *P*.

From symmetry considerations, it follows that half of the remaining cells are on the left side of the boundary *P* and the other half are on the right-hand side. Thus, a grey node (corresponding to a crossed t-th order cell) has on average 2 grey descendants, 1 black descendant and 1 white descendant (corresponding to the *(t+1)*-th order cells which are not crossed by a boundary). Then for the model of a region one obtains (average values):

$$g(t)=2^t;\ \ b(t)=w(t)=2^{t-1};\ \ t=1,2,\ldots,m-1; \qquad (3.3.2)$$

where $g(t)$, $b(t)$, $w(t)$ are the quantities of grey, black and white nodes at the t-th level.

At the lowest (m-th) level of the quadtree there are no grey nodes, therefore $b(m)=w(m)=4g(m-1)/2=2^m$. The whole tree has:

$$G = \sum_{t=0}^{m-1} 2^t = 2^m-1$$

grey nodes and the total number of nodes can be obtained using (3.3.1):

$$M=4G+1=2^{m+2}-3; \qquad (3.3.3)$$

Now it is easy to find the average quantities of black and white nodes in a quadtree for a simple region:

$$W=B=(M-G)/2=3*2^{m-1}-1; \qquad (3.3.4)$$

For the model of the line the above inference is also valid except for the ratio of black and white nodes. Quadtree of a line has no black nodes at the t-th level, but at the m-th level every pixel crossed by a line produces a black node. Then instead of (3.3.4) we have

$$B=2g(m-1)=2^m;\ \ W=M-G-B=2^{m+1}-2. \qquad (3.3.5)$$

Thus, for simple images the number of nodes in a quadtree increases with the level number t such as 2^t instead of 2^{2t} in ordinary 4-ary tree; the total number of nodes in a quadtree approximately equals 2^{m+2} instead of the 2^{2m} pixels in an original image.

The above consideration for inferring of (3.3.1) - (3.3.5) can be repeated for an image of arbitrary dimension p and decomposition base k. (For this, k parallel hyperplanes of dimension $p-1$ crossing the p-dimensional hypercube - a t-th level cell - should be considered). In this case, one can obtain:

$$g(t) = k^{t(p-1)},\ t = 1,\ldots,m-1;$$
$$G \approx k^{m(d-1)}/(k^{p-1} - 1);$$
$$M \approx k^{m(p-1)+p} / (k^{p-1} - 1); \qquad (3.3.6)$$

For the model of a p-dimensional region:

$$B = W \approx k^{m(p-1)}(k^p-1)/(2(k^{p-1}-1)), \qquad (3.3.7)$$

and for the model of a line (in the p-dimensional case - a model of a $(p-1)$-dimensional hyperplane):

$$B \approx k^{m(p-1)};\quad W \approx k^{(m+1)(p-1)}(k^p-1)/(k^{p-1} - 1) \qquad (3.3.8)$$

Real images contain straight and curved lines, regions, spots, etc. Models of a simple region and line are acceptable for these images only beginning from some level number t_0, which can be different for different parts of an image. Up to the level t_0, the majority of nodes in a tree are grey, but, after that, cells of lower levels can be considered as simple regions and lines. In this case, the following *two-layer model* of a complex image can be proposed: up to, and including, the level t_0 all nodes in a tree are grey, i.e. the number of nodes on the t-th level $(t \le t_0+1)$ is equal to k^{pt}, but starting from the level t_0 the models of simple images are valid

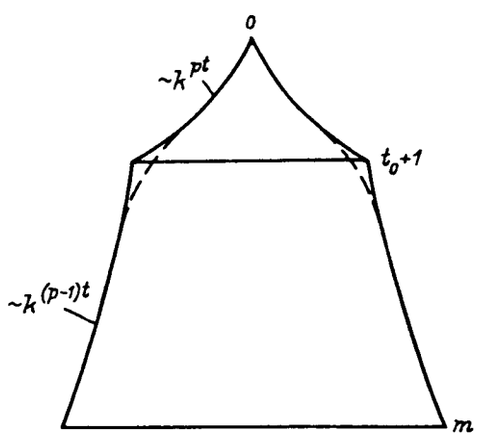


Fig. 3.18. The total number of nodes in a k^p-tree for the two-layer model.

(Fig. 3.18).

Certainly, in real images the level t_0 can be "fuzzy" because of the presence of different size objects (see the dotted curve on Fig. 3.18) but formally the value t_0 can be determined as a point where the border lines of two tree layers are intersected, the upper layer growing as k^{pt} with the level number increasing, and the lower layer growing as $k^{(p-1)t}$. In such a case, the t_0 value may be fractional.

It can be said that t_0 reflects the intrinsic detailness of an image: up to this level objects (spots, curves) retain their individuality and integrity and on lower levels they appear to be divided into blocks which have no "sense". One can suppose the t_0 value to be an invariant characteristic for a particular class of images and an indicator or measure of the complexity for this class. Using this measure the space efficiency of a k^p-tree representation and the complexity of image processing algorithms can be determined.

It is interesting to estimate the t_0 value for different classes of images. Some experimental results are given in the literature. In particular, the quantities of black and white nodes in quadtrees for different images of a size $2^{10}\times 2^{10}$ have been calculated in [76]. The t_0 value can be estimated from these data: (1) $t_0 = 3.1$ for a simple radiochart; (2) $t_0 = 3.5$ for drawing with thick lines and black regions; (3) $t_0 = 5.0$ for a map, etc.

Another example can be found in [149] where quantities of black, white and grey nodes in an octree representing a three-dimensional image of the human brain are evaluated. For this image, $t_0 = 1.2$ and the ratio B/G equals 0.97. Thus, there is a high similarity to the region model. The results of our experiments with a binary version of the well-known girl image (Fig. 3.19) are shown in the Table 3.1; for this image $t_0 = 4.1$.

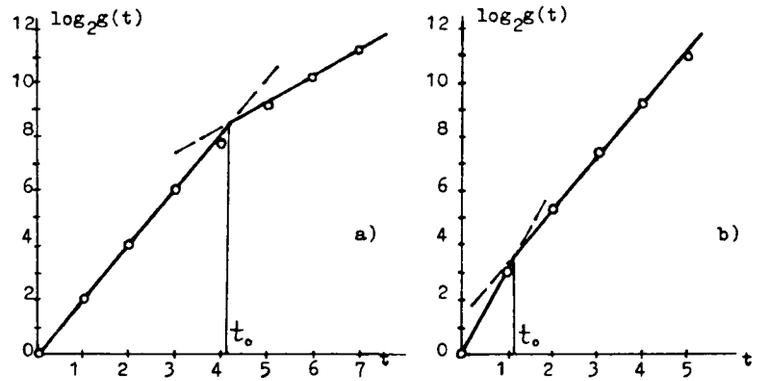
Table 3.1. Quantities of different color nodes in the quadtree for the "girl image".

Color of nodes	Level number								
	1	2	3	4	5	6	7	8	total
black	0	0	1	7	54	272	1081	4518	5933
white	0	0	0	45	207	631	1614	4750	7247
grey	4	16	63	200	539	1253	2317	0	4392
total	4	16	64	252	800	2156	5012	9268	17572

Theoretical estimations for a two-layer image model with $t_0 = 4.1$ and

experimental results from the Table 3.1 are compared on in Fig. 3.19. As can be seen, the correspondence between theory and experiment is high.

Fig. 3.19. The amount of grey nodes in the quadtree for the "girl image" (a) and in



the octree for the brain image [149] (b); solid lines - prediction according two-layer model, points - experiment.

In a two-layer image model the total number of nodes M can be estimated as $k^{pt_o}M_0$, where M_0 is the number of nodes in a subtree having its root at the level t_0, and k^{pt_o} is the quantity of such subtrees (or the number of nodes at the t_0-th level). To find M_0 it is sufficient to substitute m-t_0 (the height of a subtree) instead of m in (3.3.6):

$$M_0 = k^{(m-t_o)(p-1)+p}/(k^{p-1} - 1).$$

We then obtain

$$M = k^{m(p-1)+t_o+d} / (k^{p-1}-1) \qquad (3.3.9)$$

M can be called the complexity of the k^p-tree. It determines the complexity of the algorithms of manipulations with k^p-trees. Estimations for G, B, W for a two-layer k^p-tree model can be obtained from (3.3.6)-(3.3.8) in a similar way.

Let us consider an image perimeter estimation with a two-layer model. It is known [115], that the average number of square lattice cells crossed by some curve is equal to $n = 4L/(\pi a)$, where L is the curve length and a is the length of a lattice cell side. It follows from this theorem that, for a large enough level number t, the number of grey nodes at this quadtree level can be employed for the evaluation of the length L of the boundary between black and white regions. This is just the

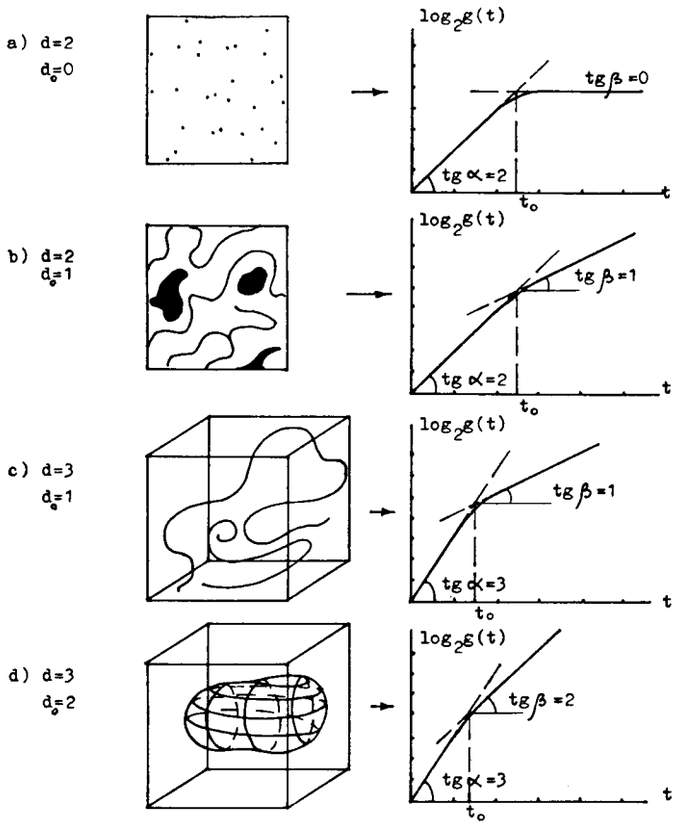


Fig. 3.20. The logarithm of quantity of grey nodes at the t-th structure level grows proportionally to the image dimension p for $t<t_0$, and proportionally to the intrinsic dimension p_0 of the boundary of the black regions for $t>t_0$.

perimeter of the object, figure, etc.

$$L = \frac{\pi 2^{-t} g(t)}{4}.$$

If the two-layer k^p-tree model is valid and the parameter t_0 is estimated using a linear approximation of experimental as like in Fig. 3.19, the perimeter can be estimated, taking into account that for the two-layer model $g(t) = 2^{t_o+t}$:

$$L = \pi \frac{2^{t_o}}{4}. \qquad (3.3.10)$$

Thus the t_0 value is related with perimeter of black regions and may be used

for its prediction (note, that L is the true perimeter of black regions, measured before image discretization). The more detailed is the image, the more complex is their configuration, and the larger is the value L and, hence, t_0.

It can be also be shown that a k^p-tree representation allows to evaluate the intrinsic dimensionality of an image (more precisely, the dimension of the boundary of black regions). If the boundary of black regions has the dimension p_0, the quantity of grey nodes (which determines the space efficiency and the complexity of the tree) increases for $t>t_0$ in the proportion $k^{p_0 t}$ independently of the original image dimension p. Then intrinsic dimension p_0 can be estimated from the quantities of grey nodes at levels $t>t_0$: p_0 equals the tangents of the angle between the t-axis and the straight line approximating $\log_2 g(t)$ experimental points (see Fig. 3.20).

For instance, a set of uniformly distributed points ($p_0=0$) in a p-dimensional space produces a k^p-tree, which does not grow in width after some level number; a typical two-dimensional "girl image" having a one-dimensional boundary ($p_0=1$, $p=2$) produces a quadtree in increased the proportion 2^t for $t>t_0$; the brain image has a dimension $p=3$ and the boundary dimension $p_0=2$, so the corresponding octree grows in the proportion 2^{2t}.

Thus, some important image features such as the perimeter or square of the black areas, the intrinsic dimension of the boundary of the black areas can be estimated using pyramidal-recursive structures of k^p-tree type. One can suppose that for particular classes of images (e.g. for a set of dark particles in a light area) more complex and accurate models can be developed which allows to estimate other features related with connectivity, concavity, average size of connected components, etc.

Chapter 4

IMAGE CODING AND PROGRESSIVE TRANSMISSION WITH GRADUAL REFINEMENT

Image representation by means of the pyramidal-recursive structures described in the previous chapters is a means of uniformization of data organization to provide their storage and processing in a computer. One of the important tasks here is to reduce the information flow entering the processing system, or compression of the images to be stored. This reduction can result in a decrease in the complexity of image processing algorithms and the simplification of some image analysis problems. The coding and compression algorithms should provide, as far as possible, a final form of data representation which might be useful for further processing without decoding or other conversions necessary to restore the initial data volume.

The existing coding and compression techniques are, by and large, not closely connected with the processing methods, and the computational complexity of the latter is determined, as a rule, not by the reduced, but by the initial data volume [68, 102, 103, 148, 152]. The pyramidal-recursive representation enables image data compression and processing to be carried out in a manner which preserves, for the compressed data, the same structure as for the initial field (i.e., pyramidal-recursive structure) and a description of processing algorithms based not on the original image, but on the condensed structure.

The present chapter is devoted to problems of image data compression on the basis of the elimination of their statistical (or "information" in terms of information theory) and, to some extent, semantic, redundancy. The first is connected with the high mutual correlation of pixel values in the image and the presence of

homogeneous monotonic areas; the second is determined by the problem being considered and by the means of processing, i.e. it is connected with the problem orientation of the system which perceives the image and makes a decision.

The first two sections describe methods of image data coding, which are based on the elimination of statistically redundant elements in a pyramidal-recursive structure for greyscale and binary images. The algorithms described in these sections are of interest as they provide rather high compression ratios and preserve some properties which are important for further image processing or transmission.

Problems of image data transmission based on the level-by-level processing of an image pyramid are discussed in the third section. This provides a gradual image "refinement" during its decoding or transmission and enables image analysis to be initiated, or a decision to be made at the receiving side, before the moment when the whole data volume is reconstructed or received. This can be interpreted as be account of the different semantic value of different data elements. Algorithms of greyscale image coding and their progressive transmission are given in [15, 55, 58], while methods and algorithms for binary images are described in [11, 54].

4.1. Coding of Greyscale Images

Different algorithms of image coding, which use pyramidal or recursive structures for image representation have been developed recently. Two main approaches to their construction are the decorrelation of pixel values to eliminate statistical redundancy and the removal of redundant elements corresponding to image areas uniform in brightness. The first approach was developed in [1, 28, 34, 35, 92, 98]; however, exhaustive quantitative models of such algorithms have not yet been described . The second approach is developed in the papers [17, 70, 80, 87, 141, 142].

The algorithms of image coding described in the present section combine both the approaches mentioned, they are based on the decorrelation of pyramidal-recursive structure pixels, and the elimination of redundant elements. The coding ability of the algorithms is estimated.

At present, simple, but rather efficient, data compression algorithms which are based on the calculation and coding of the difference between highly correlated image or signal elements (delta-modulation, differential pulse coding modulation methods, etc.) are widely employed [23, 103, 152]. As is shown in Section 3.2, the most significant correlations in the pyramidal-recursive structure for a greyscale image are interlevel correlations which, as follows from (3.2.12), in most cases

exceed the inside-level ones. Therefore, it may be expected that the use of interlevel correlations enables more effective algorithms to be constructed than the existing algorithms based on the analysis of neighboring pixels.

Let us use a differential schema for accounting for interlevel correlations. Let us apply the following procedure to each elementary cell of a pyramidal-recursive structure beginning with the top level: each node-descendant with the number $i_1...i_t$ is assigned a new brightness value $\delta(i_1...i_t)$ defined by the relation

$$\delta(i_1...i_t) = s(i_1...i_t) - s(i_1...i_{t-1}), i_t = 0,...,k^p-1,\ t = 1,...,m.$$

Then the brightness of each pixel of the original image $s(i_1...i_t)$ can be expressed via the brightnesses of the transformed structure, which, in what follows, is called the *difference structure*, in the following way:

$$s(i_1 \dots i_m) = s_0+\delta(i_1)+...+\delta(i_1...i_m), \qquad (4.1.1)$$

where s_0 is the average image brightness.

It can be expected that the dispersion of brightnesses $\delta(i_1...i_t)$, $t=1,...,m$ is significantly less than dispersions of corresponding brightnesses $s(i_1...i_t)$. Therefore, a smaller number of digits is required to store the values of difference structure pixels in the memory as a result of which compression can be realised.

The dispersion Z_t of the difference signal $\sigma(i_1...i_t)$ can be calculated using the expression (3.2.14). It follows from this expression and Fig. 3.15 that this dispersion for lower levels of the structure is about ten times less than that of the initial signal. It enables significantly fewer quantization levels to be used for brightness coding. The latter is equivalent to a volume reduction of the information stored, so far as the fewer number of bits is required to describe the brightness value of each pixel of the difference structure as compared to the brightness values of the initial structure.

If we would like to decode an image by restoring the pixels of the initial structure $s(i_1...i_t)$ according to (4.1.1), an error may appear, which is determined by the quantization technique of each of the $\delta(i_1...i_t)$ values, $t=1,...,m$ and the number of quantization levels. Let the sign "'" near the symbol designate the quantized value. Then the mean-square deviation U of the reconstructed brightness values from that of the original image (assuming the independence of quantization errors at various levels) is equal:

$$U = \mathbf{M}(s(i_1...i_m)-s'(i_1...i_m))^2 =$$

$$= \mathbf{M}((s-s')+(d(i1)-d'(i1))+...+(d(i1...im)-d'(i1...im)))2 =$$

$$= \mathbf{M}(s-s')2+...+M(d(i1...im)-d'(i1...im))2 = \sum_{t=0}^{m} Z_t E(t), \qquad (4.1.2)$$

where Z_t is the dispersion of the brightness of t-th level pixels, and $E(t)$ is a normalized quantization error at the t-th level.

This error is determined by the type of distribution law of value $\delta(i_1...i_t)$. This is in inverse proportion to the square of the values of the quantization levels. The optimal quantization techniques are, in the general case, rather complex, and are described, for example, in [102, 148]. However, there exist tables of optimal quantization thresh holds and levels minimizing the quantization error for some distribution laws (Gaussian, uniform, Laplace, etc.) and for the given number of quantization levels [85]. Both the level values themselves and the quantization errors $E(t)$ can be found using these tables and the respective formulae. Vice versa, given the maximum error $E(t)$, the required number of levels n can be found (the-so called Max algorithm).

Empirical probability density functions for the values $\delta(i_1...i_8)$, $\delta(i_1...i_7)$, and $\delta(i_1...i_6)$ for the two-dimensional images considered are shown in Fig. 4.1. These functions can be approximated by a Gaussian distribution density having the respective dispersion. Therefore, the error arising from the quantization of pixel values of the difference structure can be estimated according to (4.1.2) using Max's algorithm for the Gaussian distribution and the relation (3.2.14) for the dispersion of the t-th level pixels.

The question arises as to how the minimum volume V of the data stored can be obtained with the given mean square reconstruction error U. This volume (in bits) is determined by the formula:

$$V = \sum_{t=0}^{m} k^{tp} n_t$$

where n_t is the number of bits needed to store the $\delta(i_1...i_t)$ value.

An exact solution of the problem $V \to min$, $U = const.$ is difficult due to the large number of variables and the complexity of the function being optimized. However, a reasonable practical result can be obtained applying the following reasoning. First, almost all the volume of data stored falls on the two or three lowest levels of the structure. Thus, minimization should be initiated by an examination of the influence of n_m. Second, the quantization error $E(t)$ exponentially decreases with the linear growth of n_t; therefore, taking $n_t = 5 - 8$ for $t=1,...,m-l$ $(l = 1 - 3)$, it is

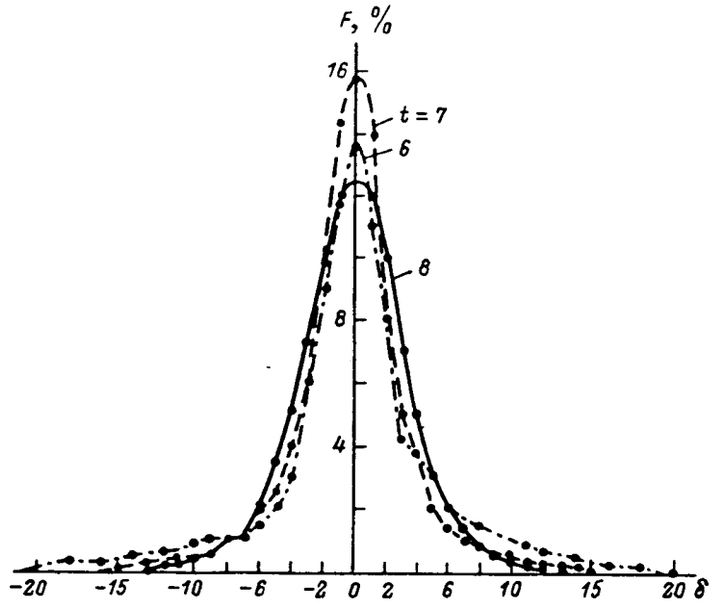


Fig. 4.1. Empirical probability density functions of the pixel values for the difference structure.

possible to reduce the contribution of the upper m-l levels into the error to an insignificant value without any practically increase of the data volume. In this case, the following simple algorithm can be proposed for the selection of the number of bits coding the difference signal $\delta(i_1...i_t)$.

Algorithm A41 $(U, l, n0)$. **(This algorithm finds the number of the binary digits n_t necessary to save the difference signal $\sigma(i_1...i_t)$ of the t-th level $(t=m,...,l)$ for a given error level U.)**

1. Set $U_0 = U\left(\frac{k^p}{1+k^p}\right)$, set $t=m$.

2. For the current U_0 and t, find, using the Max algorithm, (or with tables) the minimum number of bits n_t and the quantization error $E(t)$ for which $Z_t E(t) \le U_0$. Set $U_1 = Z_t E(t)$. (Z_t is determined according to (3.2.14).)

3. If $n_t > n_0$, then go to 6, else, go to 4.

4. Set $U_0 = U - U_1$, $t = t-1$.

5. If $t=1$, then go to 6, else go to 2.

6. Set $n_t = n_0$ for $t-1,...,l$; stop.

It is supposed that Z_t is determined from (3.2.14) and n_0 is selected such that $E<<U$. The reconstruction error provided approximately equals U, and the volume of the data stored in bits is determined by the relation

$$V = k^{pm} n_m + \ldots + k^{p(m-l)} n_{m-l} + (k^{p(m-l+1)} + \ldots + k^p + 1) n_0.$$

The drawback of the Algorithm A41 results from the accumulation of quantization errors $E(t)$ of different levels. This negative property can be easily avoided by the reconstruction of the encoded brightness at each level (a similar scheme is usually used in differential pulse coding modulation (DPCM) algorithms). Such a modification of the algorithm is discussed in the next section.

Figure 4.2. shows an example of a reconstructed image after its decoding. The image size is 256×256 pixels, and the brightness scale has 256 gradations.

The second compression mechanism is based on the modification of image areas with slow changes in brightness for blocks with constant brightness. Note that, with an image representation by a pyramidal-recursive structure, this mechanism may be used independently of the elimination of interlevel correlations, as the brightness of each image block of $k \times \ldots \times k$ pixels is changed for a constant value when the difference structure is used. Thus, monotonic image areas remain monotonic in the difference structure too.

Hence, it may be expected that a definite number of pixels of the t-th $(t<m)$ level may be found, the descendant pixels of which at the lowest m-th level have a very small brightness scattering. All such pixels (corresponding to a p-dimensional hypercubic fragment of the original image) can be replaced by an average brightness, the divergence of the transformed image from the original image increasing insignificantly. In this case, it is sufficient to store only information about a single pixel of the t-th level, which has an average brightness, instead of information about all pixels belonging to a fragment.

Because of this, the volume of the data stored can be reduced for $n(k^{pt} + k^{p(t-1)} + \ldots + k^p)$ bits, where n is the number of binary digits necessary save a brightness value. At the same time, the reconstruction error is introduced for the original image, and is equal to

$$u = \sum_{i_{t+1},\ldots,i_m=0}^{k^p-1} (s(i_1 \ldots i_t) - s(i_1 \ldots i_m))^2$$

If this error does not exceed the given maximum value then the node with the number $G=i_1 \ldots i_t$ is called the terminal. All descendants of terminal nodes may be eliminated, reducing the volume of the data stored. We call this operation the truncation of the structure, and we call the resulting structure the truncated structure.

a.) b.)

Fig. 4.2. The original image (a) and the reconstructed image with the equivalent volume of information 4 bits per pixel (b). Coding of the difference interlevel signal is used. A noise suppression effect can be seen on some smooth image areas.

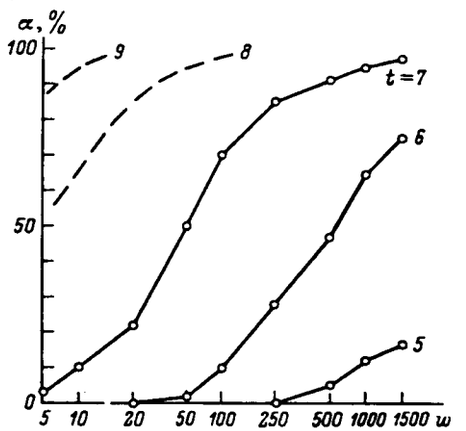


Fig. 4.3. The number of terminal nodes at different structure levels depending on the error threshold.

An interesting problem is to predict the number of nodes in the truncated structure depending on the given reconstruction error threshold. Unfortunately, it is not yet possible to find a sufficiently simple theoretical model for this case, because it is necessary to know the mutual distribution of the brightnesses of pixel groups in the initial or difference structures. Experimental data obtained for the two-dimensional image shown in Fig. 4.2(a) and for the structure with a decomposition base $k=2$ are given below. Figure 4.3. shows the number of terminal nodes at the three levels of the truncated difference structure for various error thresholds. The threshold value is expressed in absolute units — the number of brightness gradations. For example, with $w=500$, the mean square deviation of truncated pixels of the 7-th order from their reconstructed values is found to be

$(500/4)^{1/2}=11$ gradations. If an image is of a higher resolution (512x512, 1024x1024, etc.), then corresponding curves for levels *8, 9, ...* are still closer to the ordinate axis in the diagram (see the dotted lines in Fig. 4.3.).

This means that the volume of data stored (when an image is represented by a truncated structure) is determined by the internal complexity of the original image on which the number of terminal nodes depends, but not by the resolution of the image or by the input hardware.

Algorithm A42 (w, V). (This algorithm marks the terminal nodes of a pyramidal-recursive structure with the given threshold of the reconstruction error w, and finds the volume of data stored V of a truncated structure).

1. Set $V=\dfrac{k^p}{k^p-1}k^{pm}$.

2. For all pixels of the $(m-1)$-th level find

$$y=\sum_{i_m=0}^{k^p-1}(s(i_1\ldots i_m)-s(i_1\ldots i_{m-1}))^2.$$

If $y<w$, then (mark the node $i_1\ldots i_{m-1}$ as the terminal one; set $V=V-k^pn+1$). (One bit is used to save the information concerning the terminality of nodes).

3. Check for terminality (similar to item 2) all nodes of the t-th level ($t=m-2$, $m-3$, ...) for which all descendants have been marked as terminal. If there are no such elements at the t-th level, then stop.

4.2. Comparison of Some Image Coding Algorithms

Several image coding algorithms have been developed for comparative experiments with pyramidal-recursive structures. An image representation by difference and truncated difference structures, and ordering of the pixels according to the Hilbert scan have been used in these algorithms. In addition, an algorithm of the DPCM technique has been programmed.

Algorithm A43. (DPCM coding based on a TV raster).

1. For every single row of the N×N image, the first three pixels are saved and every next pixel in the row is predicted sequentially as follows:

$$s'_i = a_1s'_{i-1} + a_2s'_{i-2} + a_3s'_{i-3},\ i=4,\ldots,N$$

where s_i is the predicted brightness of the i-th row pixel and $s_1=s'_1$, $s_2=s'_2$, $s_3=s'_3$.

2. The difference $\delta_i=s_i-s'_i$ is quantized with 4, 8 or 16 levels according to the Max algorithm (assuming a Gaussian distribution for δ) and saved.

3. Reconstruction of the pixels is performed according to the relation

$$w_i = entier(a_1w_{i-1} + a_2w_{i-2} + a_3w_{i-3} + \delta'),\ i=4,\ldots,N,$$

where entier(·) denotes the nearest integer value of the argument and $w_1=s_1$, $w_2=s_2$, $w_3=s_3$.

Algorithm A44. (DPCM coding with variable rate sampling.)

The main difference between this algorithm and the previous one is that difference signals of low magnitude can be omitted according to the criterion described in [59]. Reconstruction of the difference signal is performed using the numbers of the quantization levels. Reconstruction of the original pixels is performed as in the previous algorithm.

Algorithm A45. (DPCM coding with variable rate sampling based on the Hilbert scan.)

This algorithm differs from Algorithm A44 in that no pixels along the image row are estimated but the corresponding chain (sequence) of pixels along the Hilbert scan of the image is analyzed. Reconstruction is performed as in Algorithm A44.

Algorithm A46. (This algorithm performs truncation of the difference pyramid structure.)

In this case a 4-ary structure of m levels is constructed, where the lowest m-th level contains $2^m \times 2^m$ pixels of the original image. For every node $s(i_1\ldots i_t)$ of such a structure starting from the root, the difference between its brightness and the brightness of its ancestor on the previous level is evaluated:

$$\delta(i_1\ldots i_t) = s(i_1\ldots i_t) - s'(i_1\ldots i_{t-1}), i_t=0,\ldots,k^p-1, t=1,\ldots,m,$$

where $s'(i_0)=s(i_0)=s_0$ is the average image brightness. The difference is encoded according to the Max algorithm (assuming a Gaussian distribution for the difference signal) with 16 quantization levels. Subsequently, the encoded signal $\delta'(i_1\ldots i_t)$; $i_t=0, 1, 2, 3$; $t=1,\ldots,m$, can be transmitted. To avoid error accumulation due to the quantization errors, the values of the nodes $s(i_1\ldots i_t)$ are reconstructed:

$$s'(i_1\ldots i_t)=s'(i_1\ldots i_{t-1})+\delta'(i_1\ldots i_t).$$

Once the reconstructed values are obtained, truncation of the tree can be carried out . For every node the mean square error is evaluated on the proper

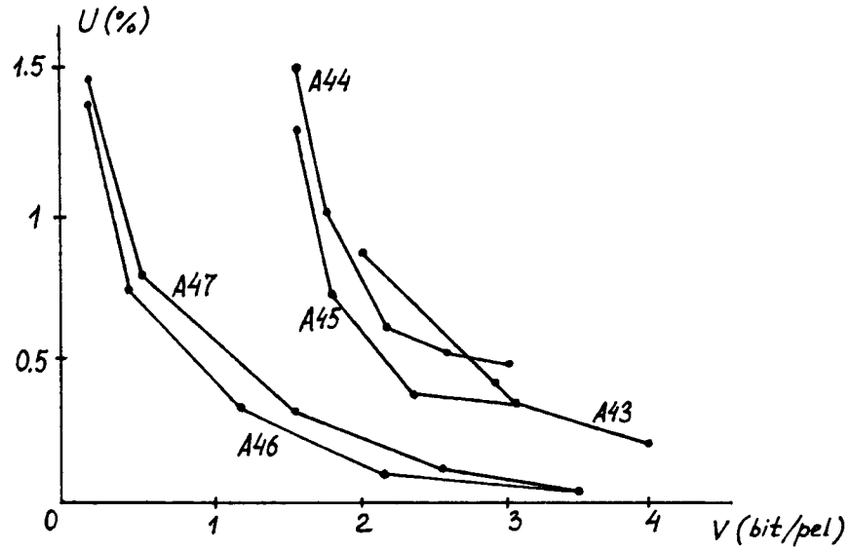


Fig. 4.4. Reconstruction error as a function of the compression ratio for Algorithms A43-A47.

segment of the image as if this node were a terminal one. In the case that this error is less than the given threshold, the node is marked as a terminal one and will be considered further. This encoding procedure is repeated recursively from the top of the pyramid to its lowest level. Thus, every subsequent step gradually refines the fidelity of the image and does not change the image segments already obtained to satisfactory precision. The complexity of the algorithm is $O(N^2)$, where $N=2^m$ is the number of image rows or columns.

Algorithm A47. (Truncation of the structure with a slant plane approximation of the reconstructed image.)

A tree similar to that in Algorithm A46 is constructed. Then, for every four nodes of the t-th level $(t=2, 3,...,m)$, a slant plane is constructed to minimize the mean square error E_0 of the brightnesses of these nodes. In the case $E_0 \leq H$, where H is a given parameter, this plane is applied to approximate the corresponding square fragment of the original image. If the mean square error E_1 of the approximation of this fragment by the slant plane does not exceed the given threshold H_1, the four nodes of the t-th level are considered as terminal ones and their descendants on levels $t+1$, $t+2,...,m$ are eliminated. The differences between the obtained terminal node brightnesses and those from the slant plane are evaluated and encoded as in Algorithm A46.

For all the above algorithms, corresponding programs were developed. The

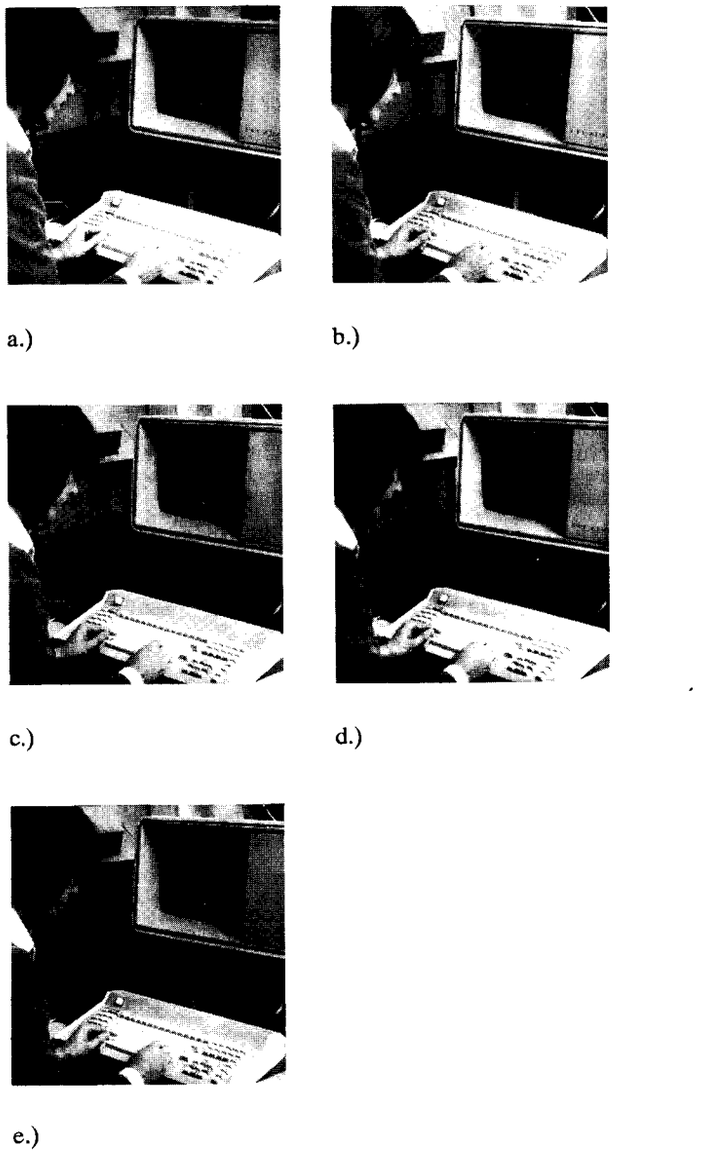


Fig. 4.5. Examples of reconstructed images in the coding of a difference signal and the truncation of redundant structure nodes: (a) original, V=8 bit/pel, normalized mean square error (NMSE)=0; (b) using Algorithm A46, V=2.0 bit/pel, NMSE=0.08%; (c) using Algorithm A46, V=1.5 bit/pel, NMSE = 0.15%; (d) using Algorithm A47, V=2.5 bit/pel, NMSE = 0.07%; (e) using Algorithm A47, V=1.6 bit/pel., NMSE = 0.2%.

image was stored in two different forms:

(a) TV-scanned for Algorithms A43 and A44;

(b) Hilbert-scanned and TV-scanned for Algorithms A45 - A47.

It should be noted that the use of a pyramidal-recursive structure provides an opportunity to obtain a rough approximation of the whole image, whereas traditional linear representation provides smaller fine fragments only, the same volume of information having been decoded.

Examples of the decoded images are presented in Fig. 4.4. Figure 4.5 shows the NMSE of the reconstructed image and the corresponding coding rate achieved, defined as the amount of bits per pixel. As can be seen from the figure, Hilbert scan utilization has advantages over TV scan, and algorithms based on the use of a pyramidal structure work better than all DPCM algorithms.

An important feature of these algorithms should be stressed. Tree structure data representation enables image data both to be decoded and processed progressively from the top to the base of the pyramid, gradually refining the results. This could give rise to an additional improvement of the actual compression ratio in the case when the results already obtained are satisfactory [94, 123].

Recursive pixel enumeration in the recursive pyramid representation preserve some topological properties of the image. We have not obtained significant improvement of the compression ratio in the image coding based on this. Nevertheless, this property might be useful for other applications in image processing, where the topology of an image is important. The compressed form of the image representation, i.e. the truncated difference pyramid structure, might be rather efficient in respect of a decreasing volume of computations when different image processing tasks are to be performed.

4.3. Transmission of Greyscale Images with Gradual Refinement

The semantic redundancy of image data makes it possible for a human being to perceive distorted or noisy image, to make decisions based on the observation of separate parts or as a result of receiving a very limited data volume, for example, a contour image. This redundancy is closely connected with a particular problem being solving at the moment. It may be different for one and the same image depending on the purpose of processing, and, therefore, its formal measurement is quite difficult. At the same time, for each individual problem the semantic image redundancy can be indirectly estimated via those decisions which are being made

based on the analysis of the given image. In particular, two images can be considered as semantically equivalent if they lead to identical decisions (after their analysis by a human being or a computer).

Based on this thesis, it may be said that omitting part of the image data does not disturb the semantics of the image perception, provided the decision regarding this image and found within the frames of the problem under consideration remains the same. Then, one possible approach to the elimination of semantic redundancy during image data transmission consists in determining the succession of the data to be transmitted which provides the quickest decision making at the receiving side. After the decision is made, data transmission (or data selection from the memory) may be ended; which is equivalent to a decrease in the transmission volume or semantic data compression.

Hence, it is the most semantically saturated data part which should be selected, transmitted, and analyzed first. A question arises as to how to extract this part? If there is no a priori information about the image to be transmitted, then, evidently, the only useful criterion for data selection relates to the possibility to reconstruct approximately the image transmitted, supposing that this approximation is adequate for decision making. An example of the most valuable data extraction is shown in Section 2.2: "the right" recursive definition of a binary number allows to identify it (to make a decision regarding its belonging to a certain scale gradation) reading bits from the high-order to low-order ones. Extraction ceases when a certain numerical accuracy, adequate for its identification, is reached.

Let us process the image data in a similar way, analogously to the scheme (2.1.4) and Fig. 2.1. Let us extract and transmit data in a stepwise manner in order that each succeeding data portion refines the preceding one, thus gradually "developing" the image at the receiving side. Concerning regular hierarchical structures, this idea of image transmission with a step-by-step reconstruction at the receiving side was proposed in [123], and subsequently a number of algorithms was derived on this basis [74, 92, 94, 125].

The present section describes a theoretical model of image transmission with gradual refinement using pyramidal-recursive structures as proposed in [11], and presents original data extraction algorithms. The main assumption is that a certain part of the semantic information is preserved at the upper levels of the image pyramid, which are an approximate description of the image under consideration. This data should be extracted and analyzed first. It is only a small part of the total data volume that corresponds to these levels, as a result of which, in the decision making process, a high equivalent compression ratio may be reached or the

transmission time may be significantly reduced.

Let us take the time needed to obtain an approximate image description with a given accuracy as a formal criterion of the efficiency of the extraction (or transmission) technique. The notion "accuracy" in each particular case should use the *a priori* information on the class of images treated. In this section, due to the lack of such information, we limit ourselves to an image reconstruction error at the receiving side of some abstract communication channel (without interference).

We set the problem in the following way: it is necessary to transmit a p-dimensional greyscale image of $N=k^{pm}$ elements each having a brightness in the range 0 to 2^n-1. Thus, the total number of bits describing an image equals nN. It is necessary to determine the order of data element transmission[1)] for the optimum (for the given criterion) image reconstruction at the receiving side with the limited transmission time.

As the criterion of the received image quality, we analyze the traditional mean square deviation of the received image from the original one (see Section 2.5 concerning criterion selection):

$$u(T)=\frac{\sum_{i=1}^{N}(z_i(T)-s_i)^2}{\sum_{i=1}^{N}s_i^2} \qquad (4.3.1)$$

where s_i, $i=1,...,N$ is the brightness of the i-th sample of the original image being transmitted; and where $z_i(T)$ is the corresponding brightness of the received image at the time T. Let also δ be the elementary time for the transmission of one bit.

Note that, according to (4.3.1), the optimum transmission technique consists of the ordering and subsequent transmission of all image bits in accordance with their contributions in the criterion (or reconstruction error) value; however, in this case the implicit order of both samples and the enumeraton of their bits will be lost. Additional time will be required to transmit information of relating to the "coordinates" of the bits and will considerably decrease the criterion value. Therefore, the transmission technique should combine both implicit enumeration and the selection of the most significant (for the criterion) bits and samples.

Let us consider some techniques. Suppose the brightness of all pixels of an image at the receiving side to equal zero before the transmission begins, i.e. $z_i(0)=0$, $i=1,...,N$. Then the criterion value at the time $T=0$ equals $u(0)=1$.

1. The traditional technique consists of linear image scanning row-by-row and the transmission of successive samples for each row. In this case, the $u(T)$ value

1) Data element means here both a pixel and an individual bit.

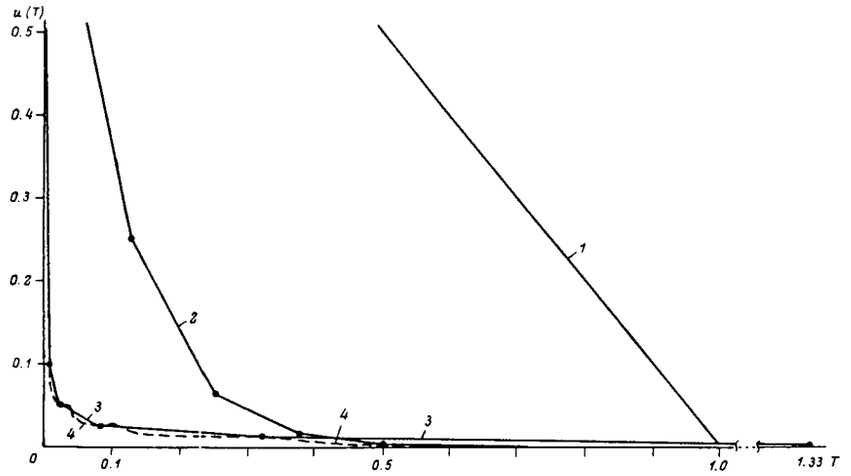


Fig. 4.6 Dependence of the image reconstruction error at the receiving side on the transmission time: (1) bit-serial transmission according to TV-scan; (2) bit-slice transmission; (3) level-by-level transmission of the pyramidal structure; (4) combination of 2 and 3.

decreases after the transmission of the next sample s_j in s_j/N. As the order of the samples is not connected with their absolute values, the $u(T)$ dependency is, on average, linear (Fig. 4.6, curve 1). Should the transmission time be limited by the value $T_0=Kn\delta$, (transmission of K samples), then the reconstruction error is, on average, equal to:

$$u(T_0)=1-T_0/n\delta N=1-K/N, \qquad (4.3.2)$$

a part of the image (the first few rows) being reconstructed exactly, and another part not being reconstructed at all.

2. Let us make use of the different significance of bits in each brightness value and pass to an image transmission "by bit-slices", i.e., by transmitting the sequence of binary matrices, each being a set of bits of one number from all samples. This is equivalent to applying a recursive structure which represents the domain of the intensity function (see Section 2.5). Let us evaluate the criterion value after transmission of the first binary matrix. Suppose that the brightness distribution is uniform all over the range. One outcome, in particular, is that the frequency of appearance of zeros and ones in all binary matrices is equal (for real images this is true beginning with the 2nd-3rd bit-slice). Then

$$u(N\delta) = \frac{\sum_{i:s_1<2^{n-1}} s_i^2 + \sum_{k:s_k>2^{n-1}} (s_k - 2^{n-1})^2}{\sum_{i=1}^{N} s_i^2} \qquad (4.3.3)$$

with the number of summands in both sums of (4.3.3) being equal to $N/2$ due to the supposition made.

After the transformation of (4.3.3), we get:

$$U(N\delta) = \frac{\sum_{j=0}^{2^{n-1}-1} \frac{jN}{2^n} - 2^n \sum_{j=2^{n-1}}^{2^n-1} \frac{jN}{2^n} + 2^{2n-3}N}{\sum_{j=0}^{2^{n-1}} \frac{jN}{2^n}} = \frac{2^n-2}{2^{n+2}-2} \qquad (4.3.4)$$

Comparing the value from (4.3.4) with $u(0)$, one can see that the criterion value decreases with the ratio of $(4\cdot 2^n-2)/(2^n-2)$, i.e. a little greater than 4 times. After the transmission of the first binary matrix, one again comes to the initial task, but now for an image having a twice smaller brightness gradation number equal to 2^{n-1}. The second matrix transmission also decreases the error with the ratio of $(4\cdot 2^{n-1}-2)/(2^{n-1}-2)$, and so on. Thus, with a limited transmission time $T_0=lN\delta$, (after transmission of l bit slices) one can obtain:

$$u(T_0) \approx 4^{\frac{-T_0}{N\delta}}$$

In all time periods from $T_0=KN\delta$ to $T=(K+1)N\delta$ the function is linear on average (Fig. 4.6, curve 2). It should be stressed that the information concerning the whole image field will be received starting with $T_0=N\delta$. This is a principal difference between the bit-slice transmission technique and the previous method.

3. Now let us consider a transmission technique based on pyramidal-recursive image representation. Let us begin the transmission with the top level (the root of the k^p-ary tree). Evaluate the reconstruction error after transmission of the t-th level pixels from a pyramid. At this moment, the t-th level image is obtained at the receiving side. The fragment of this image corresponding to the node with the number $i_1...i_t$ has a constant brightness equal to $s(i_1...i_t)$. The contribution of this fragment into the mean square error equals:

$$E(i_1...i_t) = k^{p(t-m)} \sum_{i_{t+1}} \dots \sum_{i_m} (s(i_1...i_m) - s(i_1...i_t))^2 =$$

$$= k^{p(t-m)} \Big(\sum \dots \sum (s(i_1...i_m) - s(i_1...i_t))^2 - k^{p(m-t)} (s(i_1...i_t))^2 \Big).$$

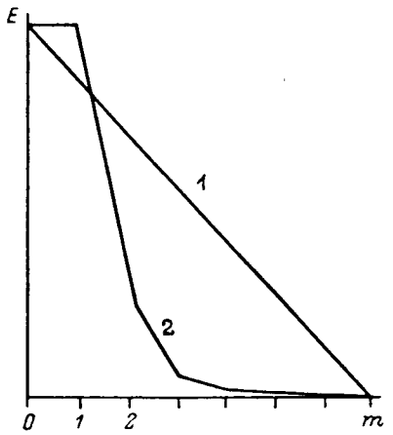


Fig. 4.7. Change of each linear segment of curve 3 (Fig. 4.6) in using the bit-slice transmission technique of every difference image instead of successive bit-serial transmission of the corresponding pyramid level.

By averaging this contribution over the entire image, one can get:

$$E = k^{-pt} \sum_{i_1} \dots \sum_{i_t} E(i_1...i_t)$$

$$= k^{-pm} \Big(\sum_{i_1} \dots \sum_{i_t} (s(i_1...i_m))^2 - k^{p(m-t)} \sum_{i_1} \dots \sum_{i_t} s(i_1...i_t))^2 = D_m - D_t,$$

where D_m and D_t are statistical dispersions of the m-th and t-th level images. Then the criterion value at the moment

$T_0=n\delta\sum_{i=0}^{t} k^{\pi}$ (i.e. after transmission of t levels) equals:

$$u(T_0) = (D_m - D_t)/(D_m + s_0^2) = (1 - D_t/D_m)/(1 + s_0^2/D_m), \qquad (4.3.5)$$

where s_0 is the average brightness of the image transmitted.

In particular, for a p-dimensional separable Markov field using (3.2.6) one can obtain from (4.3.5) the value of the expected reconstruction error:

$$U(T_0) = (1 - ((a(1-r^2) - 2r(1-r^a))/a^2(1-r)^2)^p)/(1 + s_0^2/D), \qquad (4.3.6)$$

where $a=k^{m-t}$.

Figure 4.6 shows (4.3.6) dependency for the case $k=2$, $p=2$ (curve 3) with $1+s_0^2/D$ taken to equal 4, which is valid, for example, for a uniform brightness distribution of the original image. Comparing curves 1, 2, and 3, we see that image

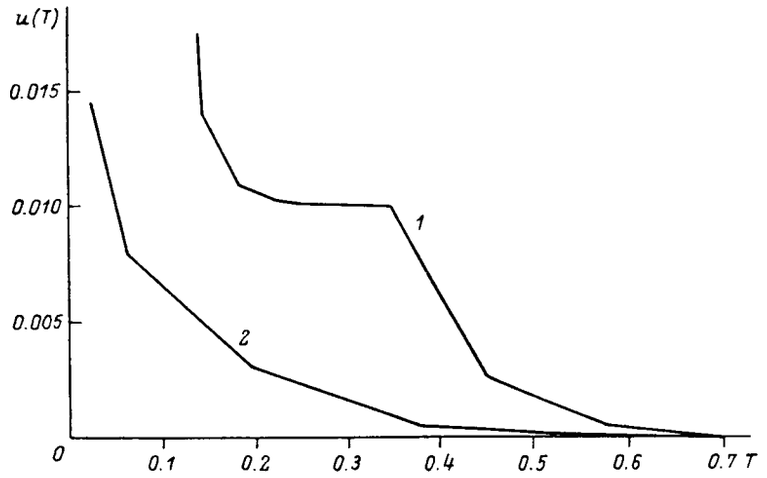


Fig. 4.8. Reduction in the reconstruction error while transmitting a complete tree (1), and the truncated difference structure (2) (experiment).

transmission based on the use of pyramidal representation provides a very fast criterion decrease during a short time, though later the bit-slice transmission becomes more efficient. Let us combine the advantages of these two techniques.

4. One can use the bit-slice transmission scheme in sending each following image from the pyramid. Note, that the direct application of this technique to the arrays of absolute pixel values corresponding to pyramid structure levels is not acceptable, as the high-order bits of each image from the pyramid duplicate the corresponding bits of the previous level image, i.e. the data that have been already transmitted. Thus, it is more reasonable to use the bit-slice scheme to transmit the difference structure, but not the initial one, when interlevel correlations have been already eliminated.

Experiments presented in Section 4.1 show that the brightness distribution at lower levels of the difference structure is close to a Gaussian distribution. Therefore relation (4.3.4) describing the bit-slice image transmission with a uniform brightness distribution can be used with some limitations only. As the difference signal has a zero mean value, its first bit of each brightness value is a sign, and transmission of the first bit-slice does not affect the criterion value. After transmission of the first slice, the remaining bits of difference samples have a distribution density function which to a first approximation may be considered already as uniform. After the second bit slice transmission the distribution becomes uniform to a high degree of accuracy.

Thus, model (4.3.4) may be used for each image from the difference pyramid

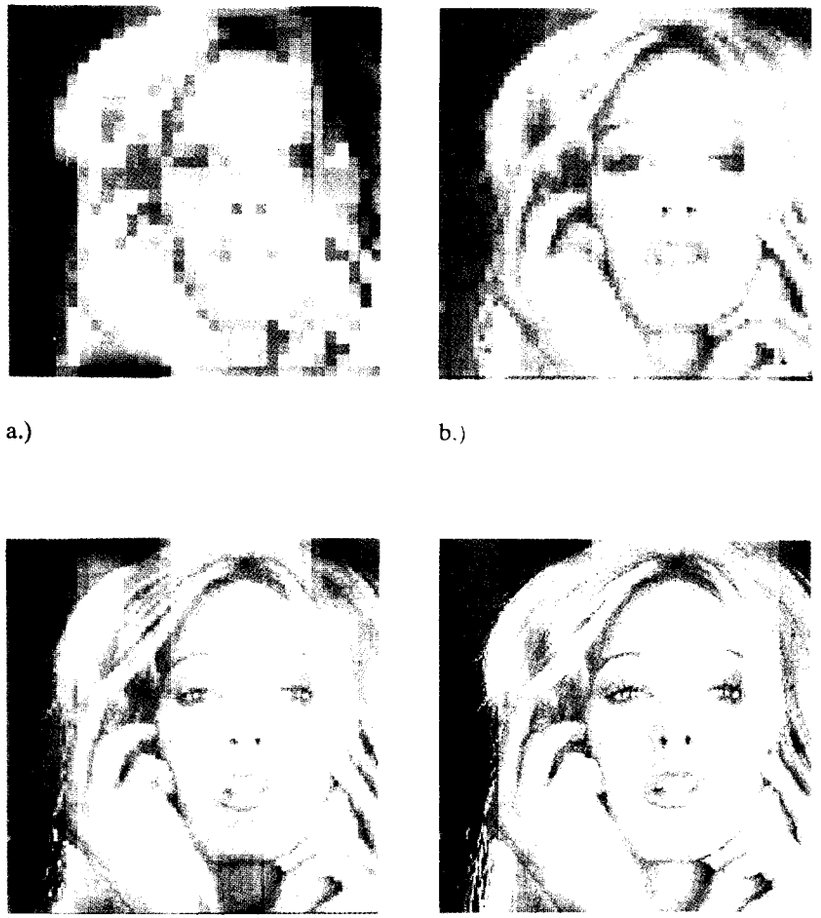


Fig. 4.9 Gradual image refinement at the receiving side during the combined transmission of the truncated difference structure:
(a) T=0.005; (b) T=0.02; (c) T=0.08; (d) T=0.3.

beginning with its second slice transmission. Hence, by applying the bit-slice transmission technique of every difference image instead of the successive bit-serial transmission of the corresponding pyramid level, each linear segment of curve 3 at Fig. 4.6 may be changed for a curve of type 2, as is shown in Fig. 4.7.

As a result, we obtain curve 4 (Fig. 4.6) corresponding to the successive transmission of difference images from the pyramid into a channel with bit-slice transmission of each image. One can see from the diagram, that this means of transmission is the most preferable of those analyzed. A further development is the

transmission of a nonlinearly quantized difference signal, which is not considered here.

The transmission techniques described do not use image data compression, i.e. they are also applicable to facsimile transmission. Should a compressed image represented by a truncated difference structure be transmitted, then the efficiency can be increased still more. The general transmission scheme described in item 4 remains the same, the difference signals corresponding to descendants of terminal nodes being omitted. In this case, it is necessary to transmit additionally information relating to the structure node terminality for unequivocal reconstruction of each brightness sample "coordinates" at the receiving side. This is achieved by adding one bit indicating the terminality of each sample transmitted.

Figure 4.8 shows an example of the criterion $u(T)$ graph change when using the combined technique to transmit the truncated difference structure for a real image. Figure 4.9 shows examples of reconstructed images at the receiving side at different times for the case of the combined transmission of truncated difference structure.

4.4. Compression and Transmission of Binary Images

The main idea used for the encoding and transmission of binary images represented by tree and pyramidal structures is the same as for greyscale images. It consists of the elimination of redundant structure nodes which describe image areas of the same color and scanning the structure from top to the bottom during transmission.

One of the first algorithms of such a type was described in [95], and, later on, some modifications were proposed [76, 114, 125, etc.]. However, a theoretical estimation of the encoding ability of pyramidal representation for binary pictures and transmission models has not yet been given. The results described below are based on a two-layer model of a binary image (Section 3.3). These results and the encoding-decoding algorithms for binary image data transmission with a gradual reconstruction are described in [11, 54].

Let us consider the compression ability of the encoding algorithm for binary images based on the two-layer image model proposed in Section 3.3. The volume of data stored with this representation for a p-dimensional image of $N=K^{pm}$ elements is equal to the number of nodes M in the k^p-tree representing the image multiplied by V_0, the volume of information on a particular node. As the tree contains three

types of nodes ("black", "white", and "grey"), with the probability of each given by the relations (3.3.6) - (3.3.8), it is possible to find the minimum value of V_0. For example, with $k=2$, $p=2$:

$$V_0 = -\frac{3}{8}\log_2\frac{3}{8} - \frac{3}{8}\log_2\frac{3}{8} - \frac{1}{4}\log_2\frac{1}{4} = 1.55$$

If the position of particular structure node is set up implicitly by giving the tree traversal algorithm, then no other information about the node except of V_0 is required. In this case, the compression coefficient equals the ratio of the initial data volume to MV_0:

$$\alpha = \frac{N}{mV_0} = \frac{k^{pm}}{MV_0}. \qquad (4.4.1)$$

It is possible to propose various coding techniques in which V_0 approaches its minimum value. For example, coding algorithms based on linear grammars and a quadtree traverse in a direct order for which $V_0=1.63$ and 2 bits are given in [76]. Another algorithm is described below. This is equivalent to the algorithms already mentioned regarding obtaining a value for V_0, but is suitable for arbitrary image dimension and is based on the tree traverse level-by-level, from top to bottom. This enables images to be decoded and transmitted with gradual refinement (similar to [94, 123], see also item 4.1).

Let us estimate the number of nodes in the tree M. Based on the proposed two-layer model of a binary image, M can be approximately estimated as $K^{pt}M_0$, where M_0 is the number of nodes in a subtree with the root at the level t_0. M_0 can be obtained from (3.3.6) by substituting $m-t_0$ instead of m; t_0 can be estimated for each class of images individually. Taking these values for (4.4.1), we get:

$$\alpha(p,k,m-t_0) = \frac{k^{pm}(k^{p-1}-1)}{V_0 k^{pt_0} k^{(m-t_0)p-(m-t_0)+p}} = \frac{k^{m-t_0-p}(k^{p-1}-1)}{V_0}. \qquad (4.4.2)$$

In particular, for a two-dimensional image with $k=2$ one can obtain:

$$\alpha(2,2,m-t_0) = \frac{2^{m-t_0-2}}{V_0}$$

The simplest coding scheme is based on a constant length code for a node color: a "gray" node is coded as 00, "white" as 10, "black" as 11, with V_0 being equal to 2 bits.

Algorithm A48 (T, A). (The algorithm scans the k^p-ary tree T representing a binary image and forms the binary code sequence in the binary array A.)

1. Set t=0; i=1.

2. For each node of the t-th level do

(if the node color is "white", then set (A(i)=1; A(i+1)=0; i=i+2);

if the node color is "black", then set (A(i)=1; A(i+1)=1; i=i+2);

if the node color is "gray", then set (A(i)=0; A(i+1)=0; i=i+2)).

3. If t=m, then stop, else (set t=t+1; go to 2).

On completion of the encoding algorithm, the array A contains a binary sequence which represents encoded information. In this case V_0=2 bits.

In decoding, implicit information on the tree traversal method is used and the fact that every "grey" node of the *(t-1)*-th level contains just k^p direct descendants at the *t*-th level. This provides unequivocal reconstruction of node relations in the tree on completion of the decoding algorithm.

Algorithm A49 (A, T). (The algorithm reconstructs the colors of nodes of the truncated k^p-ary tree T representing a binary image according to a binary code sequence from the array A.)

1. If A(1)=1, then (if A(2)=0, then (the root of T is "white"; stop)), else (the root of T is "black"; stop)).

2. Set (t=1; i=3; g(t-1)=1; g(t)=0).

(g(t) is the number of "grey" nodes at the level t.)

3. Repeat k^p times: (if A(i)=0, then (the following node of the t-th level is "grey"; set g(t)=g(t)+1), else (if A(i+1)=0, then the following node is "white", else it is "black"); set i=i+2).

4. Set g(t-1)=g(t-1)-1, if g(t-1) not equal 0, then go to 3.

5. If g(t)0, then (set (g(t-1)=g(t), g(t)=0, t=t+1); go to 3), else stop.

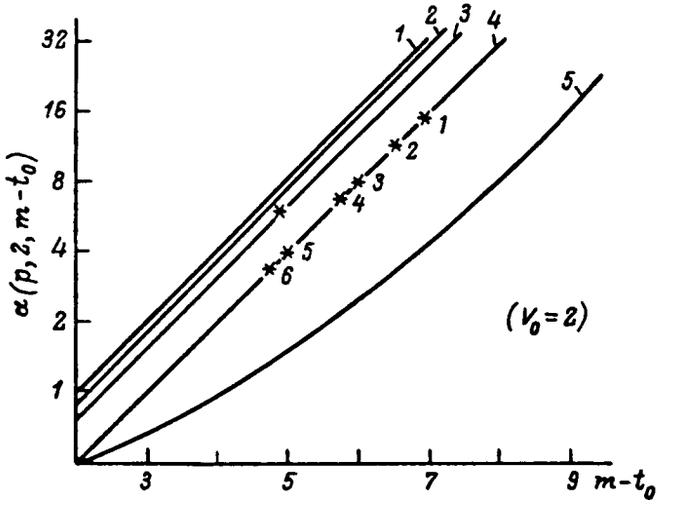
It is not difficult to see that the second bit in the description of a "grey" node is redundant. Algorithms A48 and A49 are easily modified for the case of a "grey" node encoding by one zero bit. In this case, the average data volume V_0 for a node coding (based on the region model with k=2) equals

$$V_0 = 1\times3/8 + 2\times3/8 + 2\times1/4 = 1.625 \text{ bits.}$$

Fig. 4.10 shows the compression ratios $\alpha(p, k, m\text{-}t_0)$ depending on the m-t0 value for different p and V_0=2, which corresponds to the Algorithm A48. In the same figure the results of coding images (1)-(6) from [76] and a three-dimensional image from [149] are indicated. The compression ratio α fluctuates for images of varied complexity from 4 to 15, which is better than the results of a block coding

and run length coding (see also the comparative estimations in [149]).

Fig. 4.10. The compression ratio of a binary image described by a two-layered



model depending on the $m\text{-}t_0$ value for a different image dimension p.

Let us study the transmission of an encoded binary image. In this case, a normalized mean square error is not already a satisfactory criterion of image reconstruction at the receiving side. It is more convenient to calculate the absolute deviation of the image received from the original, which is equivalent to determining the number of noncoinciding pixels. Let us consider an image reconstructed at the receiving side from successively taken elements of array $A(i)$, $i=1,\ldots,M$ according to the Algorithm A49.

As there are no "grey" pixels in the original image, it is clear that the reception of a "grey" node does not decrease the uncertainty in the reconstructed image, i.e. "grey" does not coincide with either "white" or "black". Then the deviation (reconstruction error) of the reconstructed image from the original at a given time is the total square of pixels in the received image considered at that time as grey.

The number $g(t)$ of "grey" nodes in a tree depending on the level number for models of a simple region and a line is given by the relations (3.3.6). The area of each "grey" pixel at the *t*-th level equals k^{-pt} (taking the area of the whole image to equal 1), then, limiting the transmission period by the time of transmission of t levels, one can obtain the criterion value:

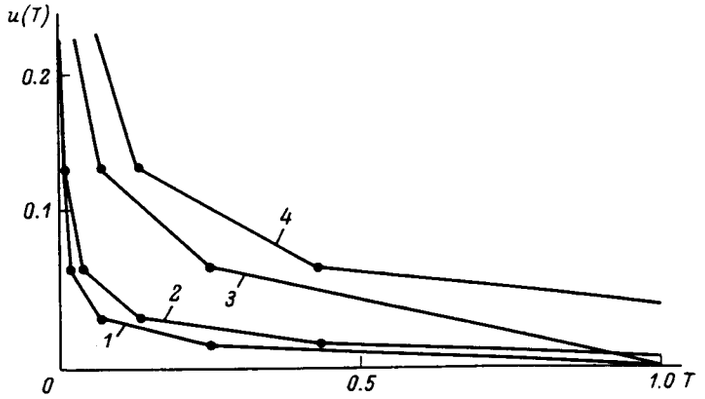


Fig. 4.11. The reconstruction error at the receiving side during the progressive transmission of a binary image for $t_0=0$ (curve *1* - $p=2$, curve *2* - $p=3$) and $t_0=2$ (curve *3* - $p=2$, curve *4* - $p=3$).

$$u(T_0) = k^{-pt} g(t) = k^{-t}. \qquad (4.4.3)$$

It can be seen from (4.4.2) that for the models of a region and a line, the criterion decreases exponentially with a linear increase of the level number t. To find how this value depends on the absolute time T, it should be taken into consideration that the t-th level of a tree contains $n(t)$ nodes, i.e. requires the transmission time $\delta k^{t(p-1)+1}$. Fig. 4.11 shows this dependence at time

$$T_0 = \delta \sum_{i=1}^{t} n(i),$$

with $t=1,...,6$ for $p=2$, *3* and $k=2$. Considering the uniform distribution of "grey" nodes along each level, the decrease in the criterion during the transmission of every level must be linear (curves 1 and 2 at Fig. 4.11).

For a two-layer model of a real image, this inference is valid only beginning with the t_0 level, as prior to this moment all the nodes are considered "grey" and the criterion value does not decrease (curves 3 and 4 at Fig. 4.11).

To summarize, it should be noted that the representation of image data by pyramidal-recursive structures provide high compression ratios; besides, the possibility arises of stepwise transmission and the utilization of compressed encoded information together with gradual refinement at the receiving side or in the processing device. This enables information analysis to begin before the time when the total data volume is received. This feature may be rather important for data transmission from sources which are under the risk of being destroyed and in situations requiring operative (in real time) decision making.

CHAPTER 5

IMAGE PROCESSING WITH PYRAMIDAL-RECURSIVE STRUCTURES

Chapter *5* is devoted to the processing of images represented by pyramidal-recursive structures. The primary purpose of this chapter is to show that various operations with images may be performed not based on their initial description, but by considering the hierarchical data structure. This enables showing that many algorithms of image processing, based on a pyramidal-recursive representation, have a complexity that depends on the number of structure nodes M rather than on the number of pixels of the initial image N. As follows from the previous chapter, M is 2-8 times less than N for greyscale images, and 5-40 times less for binary images, depending on the class and complexity of the image processed. This results in a considerable reduction of computational expenditure as compared with traditional algorithms.

A second purpose is to concentrate attention on such processing algorithms which utilize significantly the representation hierarchy and which allow to specify the result gradually when accomplishing a successive analysis of an image pyramid from upper levels to lower ones. This may reduce still more the time of task resolving, if a decision can be made before all the data is processed.

The third purpose is to show, with the help of some examples, the advantages of positional coordinates, which enables the algorithms of processing the images of different types and different dimensions to be described.

The material of this chapter has been published in [11, 16, 21, 56, 58].

Section one deals with the simplest operations with images which usually refer to problems of preliminary processing; some algorithms relating to the extraction of image features are also discussed. The main task in this case is to demonstrate the

possibilities of specifying the gradual feature value when the estimation is carried out in the course of a level-by-level tree processing.

In the second section one of the key problems of image processing, that is, the problem of object identification, is discussed. Approaches to one of its modifications — the task of hierarchical template matching — are considered. With this, more than a two-order decrease of computation as compared with to the classical correlation matching scheme is achieved.

The third section considers an identification algorithm for an arbitrarily oriented plane object which is based on image and object representations by regular hierarchical structures.

5.1. Simple Operations with Images Represented by Truncated Trees

Let us consider performing some operations on images represented by pyramidal-recursive structures. In this section the simplest image transformations are described, so to speak: basic actions which can be used for the construction of more complex algorithms . It should be noted that, at present, a large number of various algorithms of image processing based on pyramidal and regular tree structures are described in the literature (see, for instance, [22, 49, 63, 92, 99, 136, 147]). However, these descriptions vary a great deal; many of them deal with particular cases only and lead to no specific conclusions. They employ different terminology and different mappings of the data structure into the computer memory are used. The latter is significant, as the complexity of one and the same algorithm depends upon the data structure organization technique.

For example, a popular structure for a two-dimensional image representation is a quadtree [51, 70, 113]. With this it is often supposed that brightness and references to ancestors and descendants are stored in the memory for each tree node. In this case, it is easy to make forward and backward traverses in the tree, but it is not convenient to work with nodes at one and the same level. Given a linear representation of trees, successive access and processing of terminal nodes is most convenient [49]. Other data organization is possible ([46, 111], Section 4.4 of the present book) for which some access techniques are worthwhile and others are not.

The algorithms described in this section differ in their orientation for the gradual refinement of results, and their complexity, depending on the number of nodes in the truncated tree structure. The material given below is based on the results presented in [11, 55].

To be able to estimate the complexity of the algorithms described, without being limited by the framework of any particular representation in the memory, we suppose that we have at our disposal a truncated k^p-ary tree T, containing M nodes and representing a pyramid of images. Each node of the t-th level is brought into correspondence with the number $G=i_1...i_t$ of the respective cell of the t-th decomposition $d(i_1...i_t) \in D_t^p$, and a pixel $s(i_1...i_t)$ from the image pyramid.

The number G corresponds one-to-one to the positional coordinate $g=.i_1...i_t$ by which it is also possible to identify the node. The following operations of unit complexity in the tree T are considered :

(a) transitions to the previous and the next nodes at the t-th level (consider the nodes as ordered in each level by their numbers or positional coordinates). Within an elementary cell of the structure this operation is described by the relation $G_2=G_1 \pm 1$, outside the cell this may not be correct, as a some of the nodes can be missing due to a truncation of the tree;

(b) going up one level from a given node with coordinate $g_1=.i_1...i_t$ to a node with the coordinate $g_2=.i_1...i_{t-1}$;

(c) going down one level from a given node with coordinate $g_1=.i_1...i_t$ to a node with coordinate $g_2=.i_1...i_t i_{t+1}$.

This set of operations covers most of the requirements for actions on the structure enabling access to its various elements and the realization of different processing algorithms. It provides for both the transition between nodes at the same level and interlevel transitions. The complexity of the algorithms described below changes with the selection of a particular physical data structure in a serial or special purpose parallel computer, but it can be recalculated taking into account particular realizations of the basic operations indicated.

Access to a pixel. In many image processing tasks the need arises to find the pixel at a given distance in a given direction from the pixel being considered. If direct access to the data is provided, then it is not difficult. However, many special systems of image processing (matrix processors, for example) pay for the possibility of parallel processing by having access bound to time consumption which is proportional to the distance b to the element searched. In a pyramidal structure, this access does not require a time $O(b)$, but (on average) $O(\log b)$.

Consider a node (cell) having the coordinate $g_1=.i_1...i_t$ and suppose the coordinate of a node being searched is g_0 (respective to g_1). The brightness is to be found corresponding to the node being searched. Then the absolute positional coordinate of the node searched is $g_2 = .j_1...j_t = g_1 \oplus g_0$. However, this node in the tree may be missing (because of truncation); this means that the required brightness may only be determined from the brightness of the nearest ancestor

node for g_2. Therefore, after finding g_2, it is necessary to move from the tree root down along the branch (which is determined by the digits $j_1...j_t$) until either the t-th level node or the terminal node at some h-th level ($h<t$) *is reached. The brightness of the node reached is the required brightness.*

It should be noted that if the arithmetical difference $|G_1 - G_2|$ is small, then the node $j_1...j_t$ is located close to the node $i_1...i_t$ in the ordered set of nodes of the t-th level. In this case, a route along the level may be shorter than that through the tree root. However, the problem is that the existence of the $j_1...j_t$ node is not guaranteed if it is not acknowledged that $i_1=j_1$, ..., $i_{t-1}=j_{t-1}$. Therefore, an inside-level search is hardly reasonable. Thus, the algorithm described "visits" t nodes of a tree on average, or has a complexity $O(t)=O(log(N))$ irrespective of the distance g_0 between the nodes g_1 and g_2.

It is more advantageous to go up from g_1 to the nearest common ancestor of g_1 and g_2, rather than to descend to g_2. The level number l of the common ancestor of g_1 and g_2 is, on average, inversely proportional to the logarithm of the distance between cells having coordinates g_1 and g_2 in discrete space. Hence, the number of steps of the access algorithm (ascent and descent along the tree) proportional to the value t-l depends on the distance b as $O(log\ b)$. As an example, we shall consider an algorithm of access to a neighboring pixel.

Algorithm A51 *(g_1, g_2, s, l, t, h).* **(This algorithm finds the brightness** s**, the level number** h **and positional coordinate of a pixel** g_2**, which is a neighbor, in the direction** l**, of the pixel** $s(i_1...i_t)$ **having the coordinate** $g_1=.i_1...i_t$**. Direction** l **coincides or is in the opposite direction to the** l**-th Cartesian coordinate axis depending on the sign of** l**.)**

1. Find $g_2=.j_1...j_t$ according to $g_2=g_1 \oplus .0...0\ k^{l-1}$

2. Find the greatest number n for which $i_1=j_1$, ... , $i_n=j_n$. (This number can be determined from step 1, taking into account the range of the last transfer while performing the addition).

3. Ascent from the node g_1 to t-n levels; set $h=n$.

4. Set $h=h+1$. Descend down the branch determined by the digit j_h at one level.

5. If the node reached is terminal or $h=t$, then (set $s=s(j_1,...,j_h)$; $g_2=.j_1...j_h$; stop) else go to 4.

The algorithm A51 is of interest. It should be noted that according to Property 7 (see Section 2.3) just one half of the neighbors of cell $d(i_1...i_t)$ are included in the cell $d(i_1...i_{t-1})$. To reach such a neighbor, the algorithm requires two steps (an ascent by one level and a descent by one level). Just one half of the remaining neighboring cells form part of $d(i_1...i_{t-2})$; to reach these, the algorithm makes four

step (two ascents, two descents), etc. Then to reach the neighboring pixel, the following number of steps will be carried out on average:

$$\Phi = 2\times1/2 + 4\times1/4 + 6\times1/8 + ... + 2t \times 2^{-t} < 4,$$

i.e. the complexity of Algorithm A51 equals $O(1)$.

Image masking. The simplest operations which can be made with images are operations involving unification, superimposition, intersection, projection, etc. These can be described under the general name masking, as in their execution some pixels (or an image as a whole) become a mask through which other pixels or an image are examined. For instance, the elimination of "invisible" lines in a drawing or the axonometric projection of a detail, can be achieved by masking by visible surfaces; or the superimposing of an inscription on an image field involves masking some of the pixels. The particular feature of these operations is that the masking of an element (or several elements) by another does not require the analysis of those pixels which neighbor the masked one, but only requires a knowledge of their mutual location. Therefore, a typical approach in solving these problems is a parallel traverse of the mask and the image masked (or the determination of the masked elements in the process of a traverse).

By representing an image by a truncated tree, the possibility arises to perform masking by large blocks corresponding to terminal tree nodes. This results in a masking algorithm of complexity $O(M)$, and not $O(N)$, where M is the number of nodes in the tree, and N is the number of pixels of the initial image. It is also important that large blocks are considered first, i.e. it is advisable to scan a tree from the root to leaves, which yields at each intermediate step an approximate result useful for decision making.

Two algorithms for black-and-white image processing are analyzed below. The first is intended for the intersection of two images, the second for the projection of a multidimensional image into a space of lower dimension. Despite the apparent difference in these problems, algorithms are very similar to each other. Their essence is the parallel traversing of representing trees "in width", i.e. the nodes of each following level are traversed sequentially. At each step, the node which masks the others from the set of analyzed nodes is found. If this is a terminal node, then the solution for the given pixel is found, otherwise, all the descendants of the node need to be taken into account in order to make a decision at the next level, etc. This is done by joining together several subtree roots (preserving the node numbers). This is called "handing up" in the text of the algorithm below.

Algorithm A52 (*T1, T2, T3*). (For k^p-ary trees, *T1* and *T2*, representing binary images I_1 and I_2 of size $k^m \times \ldots \times k^m = k^{pm}$, the algorithm forms the k^p-ary tree *T3* representing the image $I_3 = I_1 \cap I_2$).

1. If the root of *T1* or the root *T2* are of "white", then perform (the root of *T3* is "white"; stop).

2. Set $t=1$. Hand up *T2* to the root of *T1*; call the result *T3*. Further work with T3.

3. If $t>m$ or there is no node on the t-th level of *T3*, then stop, else go to 4.

4. For each set *W* of nodes of the t-th level, which have a common ancestor at the $(t-1)$-th level, perform 5.

5. For each $i_t=0,\ldots,k^p-1$ perform 6.

6. For all nodes with the same numbers i_t from *W*, find the node having the most "light" color, other nodes of the same number i_t are to be eliminated. (White color masks grey and black; grey color masks black.) If the color of the node found is "grey", then hand up to it all subtrees the roots of which were the eliminated "grey" nodes.

7. Set $t=t+1$; go to 3.

Algorithm A52 scans in parallel two trees *T1* and *T2* with each tree nodes being passed not more than once, i.e. the complexity of the algorithm is determined by the sum of the number of nodes *T1* and *T2* and equals $O(M_1+M_2)$.

The algorithm of multidimensional binary image projection into a cartesian coordinate subspace operates in a similar way. The difference is that a set of nodes, from which a single masking node is chosen at each step, is formed by the projection of nodes having the same ancestor into the subspace.

Algorithm A53 (*T1, T2, r*). (From a k^p-ary tree *T1* representing a black-and-white image I_1 of size $k^m \times \cdots \times k^m = k^{pm}$, the algorithm forms the (k^p-1)-ary tree *T2* representing the projection of I_1 into a subspace orthogonal to a coordinate axis with number *r*.)

1. If the root of *T1* is not "grey", them perform (the color of the root *T2* set equal to the color of the root T1; stop).

2. Set $t=1$; copy *T1* and mark the copy by *T2*. Further work with *T2*.

3. If $t>m$ or if there is no node on the t-th level of *T2*, then stop, else go to 4.

4. Change digit i_t in all numbers of nodes of the t-th level for digit $j_t=i/r$. (Projection operation is made according to (2.4.11).)

5. For each set *W* of nodes of the t-th level having a common ancestor at the $(t-1)$-th level, perform 6.

6. For each $j_t=0,\ldots,k^{p-1}-1$ perform 7.

7. For all nodes with the same numbers j_t from *W*, find the node with the most "dark" color, other nodes with number j_t are to be eliminated. (If some nodes or all nodes are of the same color, then any of them may be taken). If the color of the node found is "grey", then hand up to it all subtrees, roots of which were the eliminated "grey" nodes.

8. Set $t=t+1$; go to 3.

This algorithm also scans each node *T1* not more than once, i.e. has complexity $O(M)$. In a similar way, other algorithms based on a masking operation can be constructed: image projecting in variously oriented subspaces, the unification of two or several images, the removal of invisible lines in three-dimensional images, etc.

Feature extraction. Reduction of semantic redundancy in image processing is realised by describing an image or its separate fragments by sets of features significant for solving the given problem. For different problems, different features are required and it is impossible to determine an adequate set of these beforehand. That is why a set of various algorithms is required which enables images to be described in a different way depending on the problem orientation of the system. Let us consider two types of algorithms dealing with feature extraction - some of these are based on taking into account pixel brightess or coordinates in an image (or its fragment), while others are based on the analysis of pixels' neighborhood.

Algorithms for local estimations of intensity, dispersion (energy), correlation, object area in highlight areas, image moments, etc. can be considered to be of the first type. The second type includes detectors of spots, lines, and contours, as well as algorithms based on quantitative characteristics of the estimation of neighborhood properties, for example, object perimeter, brightness gradient at a given point, etc.

Different algorithms of feature extraction based on the use of pyramidal and tree-like structures are described in the works [27, 33, 43, 120, 147]. Some algorithms distinguished by the possibility of obtaining a gradual result, with the operation time proportional to the number of structure nodes, are given below.

Let us begin by considering the algorithms for those features, the extraction of which supposes an analysis of the brightness value of each pixel taken separately (without neighboring pixels) and, possibly, by also taking into account the coordinates of these pixels in the image field.

These algorithms are based on traversing the truncated tree representing an image. In order to obtain, at each given moment, an approximate value of a feature, it is convenient to perform the traverse level-by-level starting with the higher ones, and to scan nodes at each level in turn, from left to right.

Suppose it is necessary to estimate a feature value for a given image fragment. There can be several different cases at each step of tree scanning. If a cell represented by the following node of the tree completely belongs to the predetermined image fragment (where the feature value should be estimated) and the node is terminal, then its contribution is taken into account for estimating the feature value. It is just the same as if all values of pixels corresponding to this node were processed.

If the node is nonterminal, or if the cell does not belong to the fragment, then this node is ignored.

Finally, if the node is terminal, and the respective cell does not completely belong to the fragment but partially intersects it, then it is necessary to find the area of cell intersection with the fragment and to take into consideration the contribution of this area assuming that its brightness is even. The intersection of a square or rectangular fragment with the cell is easy to find having restored Cartesian coordinates of the cell by its number (positional coordinate) and comparing these with Cartesian coordinates of the fragment.

As an example of algorithms of this type, let us consider an algorithm for image volume calculation (that is, for estimating the number of "black" pixels in a given fragment of a p-dimensional binary image), and an algorithm for determining the brightness gradient of a fragment of a greyscale image. The complexity of these algorithms equals $O(M)$ due to the fact that they are based on a truncated tree traverse, where M is the number of nodes of a subtree covering the fragment.

Algorithm A54 (T, F, V). (This algorithm finds the volume V, normalized to [0, 1], of the figure formed by "black" pixels belonging to a hypercubic fragment F of side $k^{m-1} < a < k^m$ pixels. The p-dimensional image of the side of k^m pixels is represented by the truncated k^p-ary tree T, the nodes of which are marked by "+", "0", or "-" depending on whether the respective cell belongs to the fragment, intersects it, or doesn't belong to it.)

1. Set ($V=0$, $t=0$, $N_g=0$). (N_g is the number of "grey" nodes at the current level).

2. For each "not white" node of the t-th level perform 3.

3. If the color of a node is "grey", then set $N_g=N_g+1$, else (if the mark of the node is "+", then set $V=V+k^{-pt}$; if the mark of the node is "0", then (find the volume V_0 of the intersection area of fragment F and the cell of the t-th decomposition corresponding to the current node; set $V=V+V_0$)).

4. Set $t=t+1$; if $N_g=0$ or tm, then stop, else go to 2.

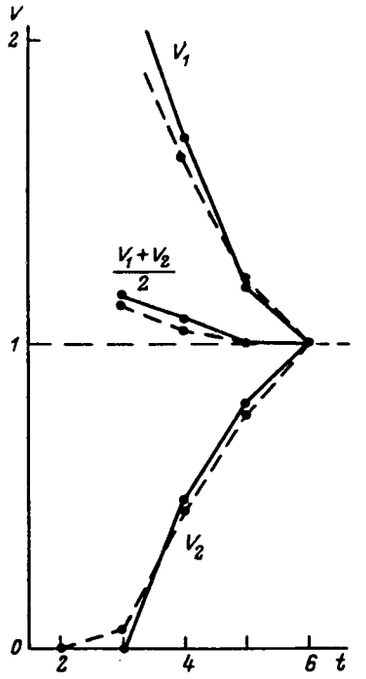
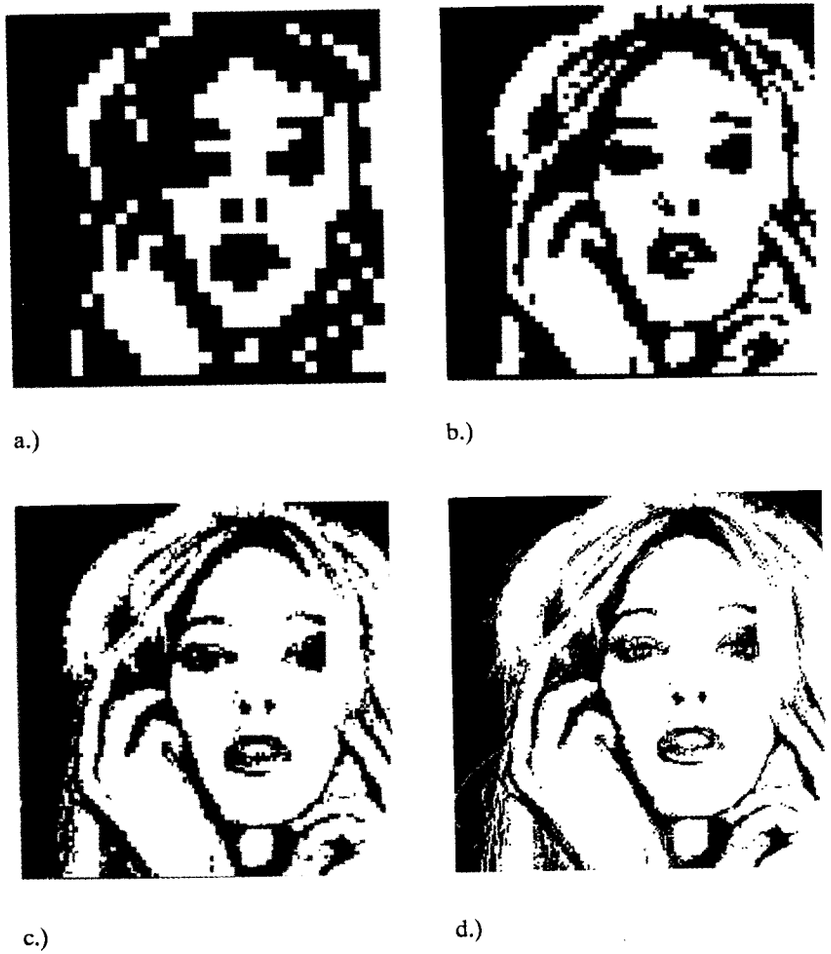


Fig. 5.1. Convergence of an object area estimations with the increase of level number.

An approximate estimation of the volume with deficiency is found in V after the scanning of t tree levels. A similar estimation with redundancy may be obtained by adding the value $N_g k^{-pt}$ to V, i.e. the volume of "grey" cells of the t-th level. If a two-layer model (see Section 4.4) is valid for the fragment F, then a nondisplaced estimation of V will be the average value of estimations with redundancy and with deficiency, since, as it follows from the model, half of the "grey" cells on average will turn out to be "black" at lower levels.

The results of area estimations of the projections of different planes after the processing of t ($t=1,...,6$) tree levels are given in [147] for a separate image and averaged for 120 images. Figure 5.1 shows respective graphs indicating that averaged estimates readily converge to a true value. Figure 5.2 shows similar estimations of the "black" pixels area of an image.

Now, consider an algorithm which enables the brightness gradient of a greyscale image to be found. For a fragment of a digital image, this is usually determined as the vector



a.) b.)

c.) d.)

Fig. 5.2. Gradual estimation of the square of white regions:
(a) image of the 4-d level; deviation of area estimation from the real value =12.3%;
(b) the 5-th level, =1.7%; (c) the 6-th level, =0.1%; (d) the 7-th level, =0.0%.

$$z = \sum_i s(x_i)x_i \qquad (5.1.1)$$

where $s(x_i)$ is the brightness of the pixel, the cartesian coordinates of the center of which are x_i. The calculation goes along the whole fragment area.

Let us suppose that the gradient for a fragment coinciding with a cell of the h-th decomposition and with positional coordinate $g=.i_1...i_h$ is to be found.

Then, to find z, it is sufficient to sum the vectors corresponding to the terminal nodes of the representing tree. The weight of each such vector is determined by the

number of the node level and the brightness of the cell. Should we take it in mind to get the next refinement of the approximate value of the gradient at each step, then it is necessary to traverse all the nodes of the tree, beginning with the root along the levels from top to bottom. The following algorithm provides an approximate gradient value after the processing of every level of the truncated difference tree where each node is assigned the brightness value.

$$\delta(i_1...i_t) = s(i_1...i_{t-1}) - s(i_1...i_t) \qquad (5.1.2)$$

In (5.1.2), $s(i_1...i_t)$ is the absolute value of the t-th level element with $\delta(i_1...i_t)$ being the respective value of the difference structure element. The gradient $grad_m$ as a positional coordinate is obtained from (5.1.1) by replacing the vector operations with those of positional coordinates:

$$grad_m = \sum_{i_1,...,i_m=0}^{k^p-1} s(i_1...i_m) \; * (.i_1...i_m * g_m),$$

where g_m describes the transition to the center of the m-th level cell. Considering the linearity of (5.1.1) and taking into account the relations (5.1.2) and (2.5.3), the t-th estimation of the gradient can be expressed through the (t-1)-th one:

$$grad_t = grad_{t-1} + k^{-pt} * \Big(\bigoplus_{i_1,...,i_t=0}^{k^p-1} (i_1...i_t) * (.i_1...i_t \oplus g_t) \Big) \qquad (5.1.3)$$

The expression (5.1.3) allows to estimate the gradient in a gradual manner, analyzing the structure from upper levels to the lower ones, with the possibility of gradient estimation refinement not only over all t-th decomposition cells, but also locally for each separate cell. If a cell corresponds to a terminal node, then further calculations for this branch of the tree may be omitted. The algorithm based on the use of (5.1.3) is given below.

Algorithm A55 (*T, g, grad*). (The algorithm determines the brightness gradient *(grad)* of a hypercubic fragment of a p-dimensional greyscale image, the fragment coinciding with the cell $d(i_1...i_h)$ of positional coordinate $g=.i_1...i_h$. The image is represented by a truncated k^p-ary tree T.)

1. Set ($t=h$; $grad=.0$). (The gradient is calculated as a positional coordinate with respect to the center of the cell with the number $i_1...i_h$.) If the node $i_1...i_h$ is terminal, then stop, else go to 2.

2. Set $t=t+1$; for each node of the t-th level with a coordinate $g_0=.j_1...j_t$, where $i_1=j_1$, .., $i_h=j_h$, perform:

$$grad = grad \oplus (k^{-pt}(j_1 \ldots j_t) * (g_0 \oplus g_t))$$

(g_t describes the transition to the center of the cell $j_1 \ldots j_t$)

3. If no nodes are found at the t-th level or $t=m$, then stop, else go to 2.

Let us now consider algorithms for the extraction of features based on the analysis of the neighborhood of each pixel, or by taking into account adjacent pixels. The computation of an object's perimeter, the detection of different objects, such as the crossing of lines, angles, and spots in an image, can be considered as algorithms of this type. Provided the image is represented by a truncated tree, it is convenient to base these algorithms on a tree traverse where each step is appraised of those cells which neighbor the cell being considered (or, possibly, some neighborhood of the cell being considered - this depends on the type of feature).

The algorithm given below is based on an "in width" tree traverse,i.e. from top to bottom with a scanning at every level of those nodes which enable a converging feature value approximation to be obtained at each step. Let us take as an example the algorithm of a binary image perimeter computation (to be more exact - the algorithm of a $(p-1)$-dimensional hypersurface area computation, where p is the image dimension). Note, that if the "black" cell of the t-th decomposition has a "white" neighbor, then their common side should be considered in the perimeter computation. If the "black" cell of the t-th decomposition has a "gray" neighbor, then their common side should also be taken into account in the perimeter computation, but only in that part where the boundary between black and white happens to be found at higher decompositions.

Algorithm A56 (T, L). (This algorithm determines the perimeter L of "black" regions of the p-dimensional binary image of k^{pm} pixels represented by a k^p-ary truncated tree T. The external boundaries of the image are taken into consideration).

1. If the root T is not "grey", then (if the root is "black", then ($L=2p$; stop), else ($L=0$; stop)), else (set ($L=0$; $t=1$); go to 2).

2. For each pixel s_1 corresponding to the "black" node of the t-th level perform 3.

3. For each pixel s_2 neighboring to s_1, perform (if s_2 is not "black", then $L=L+k^{-t(p-1)}$; if s_2 is black and $h<t$, then $L=L-k^{-t(p-1)}$) (h is the level number of the pixel s_2).

4. If at the t-th level there are no "grey" nodes or $t=m$, then stop, else (set $t=t+1$; go to 2).

After scanning each tree level, we obtain the perimeter estimation L_t making

more precise the preceding value L_{t-1}. Depending on the "convexity" or "concavity" of the figure forming an image, the estimation may be one with redundancy or with deficiency.

Another estimation of the perimeter L_t' can be obtained when considering "grey" nodes as "black" ones in items 2 and 3 of the Algorithm A56. This estimation is shifted to the opposite side as compared to L_t. The most realistic is the average value of the aforementioned estimations, which can be calculated within the frames of one and the same Algorithm A56.

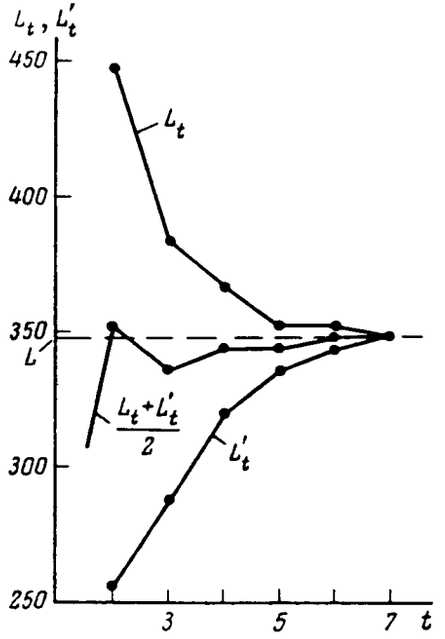


Fig. 5.3 Convergence of an object perimeter estimation with an increase in the decomposition number.

Figure 5.3 shows an examples of L_t, L_t' and $(L_t+L_t')/2$ perimeter estimations for two binary images. Note, that contrary to area estimations, perimeter estimations are displaced relative to the true black/white border length of "black" regions, if we consider an image before its discretization.

5.2. Fast Template Matching

The task of image matching (registration) is a special case of the general problem of object search and identification. This task appears at different stages of image processing, for example, while estimation of an object location changes using photos of one and the same area made at different moments; for finding the coordinates of a given object in a photo, map, or drawing; in search of an image containing a given pattern in the image data base, etc.

One widely used approach to solving problems of object search and identification in an image is the method of comparing the image with a given prototype [41, 68, 106]. The prototype would be a concrete pictorial presentation of object being sought. For the matching task, it can be considered as an image fragment which is to be found in the working image (possibly in a noisy, distorted state with respect to the prototype image), which is also called the search area. Several prototypes of one and the same object can exist; they may differ in detail, shape, brightness, size, etc.

Let us consider first the simplest case where a square prototype image (template) to be found is given and there exists its copy in the search area. They are both of the same scale and are not rotated relative to each other. The task is to find the place where the template coincides with the relevant part of the search area.

The classical approach to solve such a matching problem is to determine the similarity (difference) measure E of the template and the respective image area, over which the template is being searched. A window W filled, by the template, is sequentially shifted through all possible positions (i, j), $1 \le j(k^M - k^m + 1)$, $1 \le i(k^M - k^m + 1)$ in the search area S. k^M is the size of the search area, and k^m is the size of the window. The $E(i, j)$ value is determined in each window position (i, j) in S, and its maximum value indicates the most probable template position in the search area. Mutual correlation is often used as a similarity measure; a method known in the literature as correlation matching. The complexity of a correlation matching algorithm is estimated as $O((k^M - k^m + 1)^2 k^{2m})$ operations.

The existing methods of computation volume reduction in solving image matching problems are based on the following basic ideas: a reduction of operation in computation of the similarity measure $E(i, j)$, and object search planning (moving of window in the search area). For the first group of matching techniques, fast transforms to calculate cross-correlation [41, 148] are used as well as methods of "sequential similarity evaluation" [107] in which a reduction of the calculation volume is achieved due to a decrease in the number of pixel pairs of the

"window-search area". The drawback of this group of methods is the need to calculate $E(i, j)$ in all (i, j) positions of W. The time for solving the problem weakly depends in this case on the character of the images to be registered.

At the same time, the size and form of monotone areas, and the degree of detail of various regions of the search area must be taken into consideration in the matching process. That is why search planning methods find, at first, those areas of most probable object location, and then calculations are made which enable the object to be localized accurately [106, 132, 146].

The promising method using search planning is a sequential hierarchical matching based on image and window representation by pyramidal structures [16, 56, 107, 135, 146]. Unlike other planning techniques, the object search is carried out gradually, with progressive refinement, at every step, of the results obtained at the previous steps. This approach allows to take into account the character of the images to be registered and, at the same time, to use the existing methods of correlation computation reduction. Let us consider it.

Let the search area S contain $k^M \times k^M$ pixels and the window W contain $k^m \times k^m$ pixels with m. Let us construct pyramidal-recursive structures for the search area and the window and denote the image of the t-th level of the search area pyramid as $S^t = \{s_{ij}^t,\ i,j = 0,\ldots,k^t - 1\}$ and the image of the l-th level of the window pyramid as $W^l = \{w_{ij}^l,\ i,j = 0,\ldots,k^l - 1\}$. The search area $S = S^M$ and the window $W = W^m$ are on the base of the respective pyramids. The pixel of the t-th level s_{ij}^t is determined recurrently via the pixels of the $(t+1)$-th level according to (2.5.4).

The matching process is developed as follows. At some initial level t_0 of the image (search area) pyramid, all possible locations of the t_0-th window pyramid level are tested. If the similarity measure $E^t{}_0(i, j)$ between the window and the search area for some window location (i, j) exceed a given threshold, the location is declared to be "promising", i.e. subject to further consideration at the next level.

Then at each following level $t = t_0 + 1, t_0 + 2, \ldots$ the similarity measures $E^t(i, j)$ between the window and the image for those (i, j) locations of the window $W^{t-(M-m)}$ which are the projections of the promising locations of the $(t-1)$-th level on the t-th level and also for locations adjacent to (i, j) are evaluated. If $E^t(i, j) \ge H^t$, where H^t is a predetermined threshold for the t-th level, then the location (i, j) is declared to be promising, i.e. subject to be specified at lower levels of the structure. The process is terminated at the M-th level, where W^m locations are determined relative to S^M with extremum values $E^M(i, j)$.

Algorithm A57. (This algorithm determines at each level $t>t_0$ of the search area pyramid the matrix of promising locations P^t)

$$P^t \text{ is determined as } P^t(i,j)=\begin{cases} 1, \text{ if the location } (i,j) \text{ of } W^{t-M+m} \text{ is promising} \\ 0, \text{ otherwise} \end{cases}$$

At the initial level t_0 $P^t{}_0(i, j)=1$ for all $i,j = 0,...,k^t{}_0-1$. H^t, $t=t_0,...,M$ are thresholds for determining promising window locations at each level.

1. Set $t=t_0$.

2. For each pair (i, j) with $P^t(i, j)=1$ determine $E^t(u, v)$, where $u=i+a$, $V=j+b$; a, $b=\{0, +1, -1\}$.

3. Determine $EXTR^t= \max\limits_{u,v}\{E^t(u, v)\}$

4. If $t=M$, then stop, else determine the matrix P^{t+1}:

$$P^{t+1}(i,j) = \begin{cases} 1, \text{ if } E^t(u,v)H^t \text{ or } E^t= EXTR^t \\ 0, \text{ otherwise} \end{cases}$$

where $i=ku+[k/2]$, $j=kv+[k/2]$, and $[.]$ is an integer part of a number.

5. Set $t=t+1$. Go to 2.

By performing this algorithm, a sequential (by structure levels) refinement of the window location in the search area is carried out.

A proper window localization may also be reached before considering the lower levels on which the main bulk of the calculations take place. This allows significant reduction of the matching time and in the case of limited resources for implementing the algorithm to get at least approximate coordinates of the correct window location.

Let us consider a model of the matching process based on a pyramidal-recursive model of a p-dimensional Markov field (Section 3.2), The main purpose of the following calculations is to obtain values of the thresholds H and the criterion for algorithm termination. Let us begin by considering a one-dimensional image (signal) and then generalize the results to the two-dimensional case.

Let $S=...,s_0,s_1,...$ be a stationary Markov random process modelling a one-dimensional signal (image of the lowest level). Suppose, that the signal to be registered $W=...,w_0,w_1,...$ is just the same process, but shifted relative to S in d terms: $w_i=s_{i-d}$. Then in order to define the unknown shift parameter d, it is sufficient to find the location of the maximum value of the cross-correlation

function of the S and W processes. Meanwhile, we consider the processes S, W as unlimited and corresponding to the lowest m-th level of the structure.

Let $S_t=...,s_0^t,s_1^t,...$ and $W^t=...,w_0^t,w_1^t,...$ be the processes corresponding to the t-th level of the structure for S and W. Then

$$s_n^t = k^{t-m} \sum_{i=0}^{k^{m-t}-1} s_{i+nk}^m; \quad w_1^t = k^{t-m} \sum_{i=0}^{k^{m-t}-1} w_i^m + lk; \qquad (5.2.1)$$

are also random values and their variance, as follows from (3.2.6), is given by:

$$D(s_n^t) = D(w_l^t) = D(a(1-r^2)-2r(1-r^\alpha))/a^2(1-r)^2 \qquad (5.2.2)$$

where $a=k^{m-t}$.

Now, let us find the correlation coefficient $r(s_n^t, w_l^t)$. This value is the basis for determining the cross-correlation function of the processes S^t and W^t and is used to estimate the promising locations at the t-th level. According to the definition:

$$r(s_n^t, w_l^t) = (\mathbf{M}(s_n^t w_l^t)-\mathbf{M}(s_n^t)\mathbf{M}(w_l^t))/(\mathbf{D}(s_n^t)\mathbf{D}(w_l^t))^{1/2}.$$

Then, taking into account (5.2.1) and bearing in mind that $w_j^m=s_{j+d}^m$, we get:

$$r(s_n^t, w_l^t) = (D/a^2\mathbf{D}(s_n^t)) \sum_{i=1}^{\alpha}\sum_{j=1}^{\alpha} r^{|a(n-l)+(i-j)-d|} \qquad (5.2.3)$$

As one might expect, the correlation coefficient of the random values s_n^t, w_l^t depends only upon the difference $b=n-l$ of their numbers and the value of the shift searched d. Taking values $b=0, 1, 2,...$, we can obtain from (5.2.3) a set of samples $R(a, b, d)$ of the cross-correlation function of the S^t and W^t processes with the given shift d of the original processes $S=S^m$ and $W=W^m$. Performing summation in (5.2.3) and using (5.2.2), we get the final expressions for $R(a, b, d)$:

$$R(a, b, d) = ((a-|ab-d|)(1-r^2)-r^{|ab-d+1|}(2-r^a-r^{a-2|ab-d|})) /$$

$$/ (a(1-r^2)-2r(1-r^a)) \qquad (5.2.4a)$$

for the case of $|ab-d|<a$, and

$$R(a, b, d) = r^{|ab-d|+1}(1-r^a)^2 / (a(1-r^2)-2r(1-r^a)) \qquad (5.2.4b)$$

for the case of $|ab-d| \geq a$.

The graph of $R(a, b, d)$ for a particular d can be plotted by setting $b=0$ and changing d. The curve obtained with $r=0.95$; $k=2$, $b=0$ and $a=8$ is given in Fig. 5.4. In order to find the function $R(a, b, d)$ for a particular $d=d_0$, it should be shifted by d_0 to the right (dotted line in Fig. 5.4) and then the points corresponding to 0, a, $2a$,... arguments are marked; these will just be the curve searched.

If it were possible in practice to find the exact cross-correlation function, then only one point (or at least two points for $d=ab+a/2$) for which $R(a, b, d)$ has a maximum value should be chosen as a promising location. However, in a real situation, the number of image (signal) pixels is finite and, therefore, for each b we obtain only an estimation of the cross-correlation function value which is itself some random value. This means that the calculated values $R^*(a, b, d)$ may be both larger or smaller than expected. Then, those locations should be chosen as being promising for which $R^*(a, b, d)$ values are larger than some threshold H, depending on the number of pixels in the one-dimensional window L to be registered, and a predetermined probability q, so as not to omit the real promising location.

Note, that with an a priori unknown shift value d, the minimum value of the exact function $R(a, b, d)$, for which the location $b=b_0$ is considered promising, equals $R(a, 0, a/2)$ (Fig. 5.4). If we find now the interval into which the estimation $R^*(a, 0, a/2)$ falls with the predetermined probability q, then its lower boundary H will be just the threshold searched.

Let the window (process W^m) contain k^m pixels. Then at the t-th level, the estimation $R^*(a, 0, a/2)=r^*(s_n^t, w_l^t)$ is obtained by using $L_t=k^t$ pixels. It is known that if the values s_n^t, w_l^t are distributed normally, then the distribution of the statistics $f=0.5ln(1+r^*)/(1-r^*)$ is normal with a mean $m_f=0.5(ln(1+\mu)/(1-\mu)+\mu/(L_t-1)$ and dispersion $D_f=1/(L_t-3)$, where $\mu=R(a,o,a/2)$.

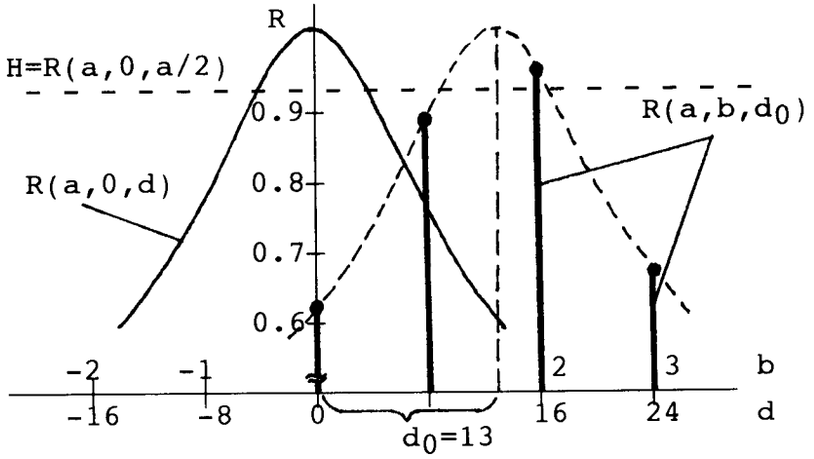


Fig. 5.4. Samples of cross correlation function of S and W processes at the 4-th level (r=0.95)

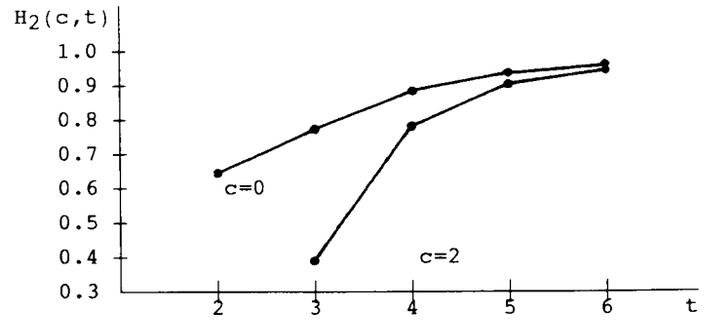


Fig. 5.5. Thresholds for determining promising locations of the window at different pyramid levels.

In real situations, the s_n^t distribution (as well as w_l^t) can be considered to be normal in the general case only for sufficiently small t. This is also valid for small t (large L_t) in the case of the Markov model, due to the limit theorem for the stationary Markov processes, as s_n^t is a normalized sum of a rather large number of summands. In this case, the threshold H is determined from the condition $f \geq m_f - c\sqrt{D_f}$, where c is the number of standard deviations from the mathematical expectation. For example, we have $q=\Phi(2)-\Phi(\alpha)=0.977$ for $c=2$, where () is the value of the normal distribution function.

Then for a given c (and, therefore, for a given probability to find the real promising location) we have the equation:

$$0.5ln(1+H)/(1-H) = 0.5(ln(1+\mu)/(1-\mu)+\mu/2(L_t-1))-c/(L_t-3)^{1/2}$$

the solution of which enables to determination of the threshold $H(c, t)$:

$$H(c,t)=(exp(A)-1)/(exp(A)+1), \qquad (5.2.5)$$

where $A = ln(1+\mu)/(1-\mu)+\mu/(L_t-1)-c/(L_t-3)^{1/2}$; $\mu = R(a, b, d)$ and $R(a, b, d)$ is determined from (5.2.4).

In the case of additive noise in the process S, the expression for the cross-correlation function changes in the following way. Let $X=S+Z$, where Z is the random noise with limited dispersion, not correlated with the process S (and, hence, with the process W). Let us find the correlations $r(w_n^t, x_l^t)$, where $X^t=S^t+Z^t$ is a process corresponding to the t-th level of the pyramidal-recursive structure for $X^m=S^m+Z^m$:

$$r(w_n^t, x_1^t) = \frac{\mathbf{M}(w_n^t(s_1^t+z_1^t)) - \mathbf{M}(w_n^t)M(s_1^t+z_1^t))}{(\mathbf{D}(w_n^t)\mathbf{D}(s_1^t+z_1^t))^{1/2}}$$

The assumption that the process Z is not correlated with the processes S and W leads to the conclusion that Z^t is not correlated with the processes S^t and W^t. Then:

$$r(w_n^t, x_1^t) = \frac{\mathbf{M}(w_n^t s_1^t) - \mathbf{M}(w_n^t)M(s_1^t)}{(\mathbf{D}(w_n^t)\mathbf{D}(s_1^t) + \mathbf{D}(s_1^t))^{1/2}} = \frac{r(w_n^t, s_1^t)}{\sqrt{(1+\mathbf{D}(z_1^t)/\mathbf{D}(s_1^t))}}$$

Thus, in the case of additive noise, the cross-correlation function at every pyramid level (expression (5.2.4)) is normalized according to (5.2.6) depending on the signal and noise dispersion ratio at this level. The expression for threshold calculation (5.2.5) in this case remains unchanged, though the thresholds are different due to a decrease in μ.

Let us estimate the number of promising locations at each level with threshold values being calculated in the case of absence of noise. The simplest way to do this is to plot the threshold values for the predetermined probability q and the correlation coefficient r in the graphs of type cross-correlation functions for different levels, and by verifying how many samples of this function are higher than the threshold. It turns out that for $t=4, 5,...$ the number of promising locations does not exceed two in the worst case and practically always equals one even for $q=0.99$. This means that no more than finding of the next k-ary digit in the value d takes place during the matching process at lower levels of the structure.

However, it is precisely the number of promising locations at the lower levels which determines the volume of calculations performed, as it is only at these levels that the number of pixels L_t in a window is large enough. Considering that the test at each of the lower levels is carried out for three window locations (shifts -1, 0, +1 from the promising location), we find that the number of "window-signal" pairs equals to:

$$N_1=3(k^m+k^{m-1}+...)+k^{M-m+1}k^2,$$

where the last item corresponds to the initial step of the algorithm (testing of all locations of the first window level relative to the $(M-m+1)$-th level of the original signal of k^M pixels). For an average algorithm of correlation template matching $N_2(k^M-k^m)k^m$. Then the number of operations of a window matching with the use of the proposed algorithm is reduced by a factor of

$$Q=N_2/N_1 \approx (k^M-k^m)(k-1)/(3k+k^{M-2m+3(k-1)}).$$

For example, with $M=10$, $k=2$, $m=7$ we get $Q=130$.

The above relations may be easily generalized in the case of a separable Markov field of arbitrary dimension p. In practice, the most interesting is the case $p=2$, which is considered below. For the separable field cross-correlation function of two-dimensional processes of the t-th level S^t, W^t equals the product of one-dimensional functions: $R_2(a, b_1, b_2, d_1, d_2)=R(a_1, b_1, d_1)R(a_2, b_2, d_2)$, where d_1, d_2 are shifts of W along absciss and ordinate axes in the original images S, and $R(a, b, d)$ is determined from (5.2.4).

The minimum value of the true cross-correlation function, with which b_1, b_2 location is considered promising, is obtained with $d_1=d_2=a/2$ and equals $\mu = (R(a,0,a/2))^2$. Let the matching image fragment contain $k^m \times k^m$ pixels. Then at the t-th level the estimation of is calculated from $L=k^{2t}$ pixels of the S^t and W^t processes and we obtain a threshold to define promising locations

$$H_2(c, t) = (exp(A)+1)/(exp(A)-1),$$

where A is determined as before from (5.2.5), but with

$$L_t=k^{2t} \text{ and } \mu=(R(a, o, a/2))^2.$$

Fig. 5.5 shows the dependence of thresholds $H_2(c, t)$ on the level number t for $c=0;\ 2$ and $r=0.95$. As well as in the one-dimensional case, the threshold value quickly becomes such that, at worst, only four locations are promising at each level. In this case, all the promising locations are in the vicinity of the maximum, and this means that "multiplication" of the promising locations does not occur while realizing the process at lower levels.

The computational expenditure of the algorithm depends on the average number of promising locations at each level which for the two-dimensional case are not higher than 2 for lower levels [16, 58]. Thus, the benefit in computational costs as compared to the traditional algorithm of correlation matching is approximately equal to:

$$Q = (k^{2M}-k^{2m})k^{2m}/(18(k^{2m}+k^{2(m-1)}+...)+k^{2(M-m+1)}k^4) \approx$$

$$\approx (k^{2M}-k^{2m})(k^2-1)/(18k^2+k^{2(M-2m+3)}(k^2-1))$$

For instance, with $k=2$ we get $Q \approx (4^M-4^m)/(12+4^{M-2m+3})$, then for $M=9$ (an image of 512x512 pixels), and $m=6$ (a window of 64x64 pixels) $Q \approx 2500$.

Two different images were used for experiments: a portrait and an aerial photograph. Each was an image of 256x256 pixels with the number of gray scale gradations equal to 256. From the image a window was randomly selected (a

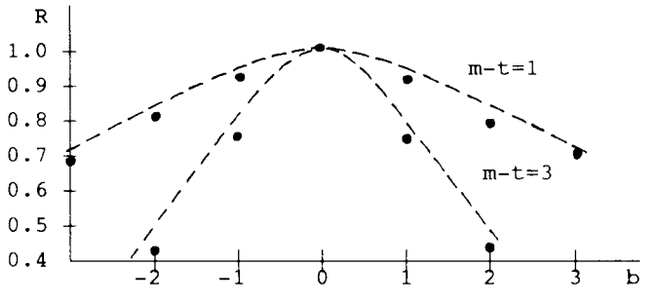


Fig. 5.6. Samples of a cross-correlation function (section along one coordinate) in the vicinity of a promising location. Points are experimental results; dotted curves are theoretical dependencies.

template to be registered) of n×n pixels (n64). Pyramidal structures were constructed for a decomposition base $k=2$. Thresholds to define promising locations were calculated according to (5.2.5) with $c = 0.5$.

Figure 5.6 shows cross correlation function values in the vicinity of the promising location window in the search area (section along one coordinate) for different levels of the structure. A shift for one sample in the graph relative to the promising location corresponds to a shift for k^{m-t} samples (pixels) at the t-th structure level. The dotted line shows the corresponding theoretical values of the function obtained from (5.2.4.).

The average number of promising locations at all levels of the search area pyramid except the two starting levels was not higher than 2, and for the majority of tested locations at the lower levels of the structure, the number of promising locations was equal to one.

Figure 5.7 shows the dependency of the time expenditure distribution (computer processing time in seconds) for different levels in the matching process (line 1), and the time expenditure dependence on the number of structure levels taken into account in the matching process (line 2). It can be concluded from Fig. 5.7 that obtaining the approximate coordinates of the true window location in the search area requires insignificant computational expenditures at the upper structure levels, while the main resources are spent for a precise window positioning at the lowest level. Thus, search planning at the upper pyramid levels allows to reduce significantly the total matching time. For a window size of 64x64 pixels, the time expenditure reduction factor as compared with the direct method of correlation matching (taking into account the time required to build pyramids) is 2000-3000, which is better than the results described in [135, 146]. For 30 tested

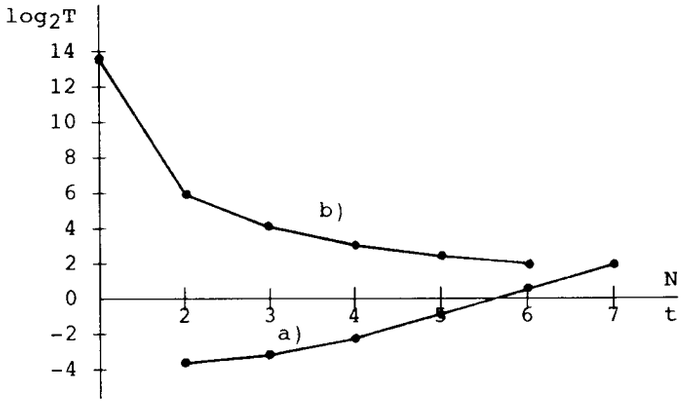


Fig. 5.7. Distribution of time expenditures at different levels of the structure (a); total search time depending on the number of pyramid levels used in the matching

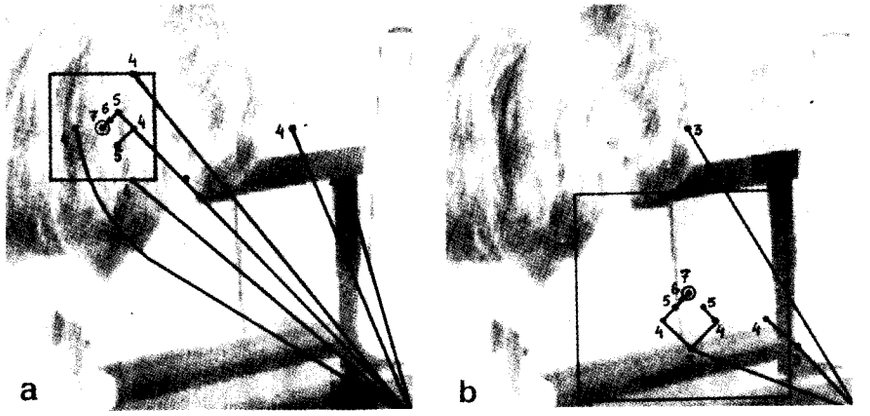


Fig. 5.8. Gradual refinement of the window (black frame) location in the search area. Points in the figure indicate promising locations of the window centre at the different pyramid levels (level numbers are near the points): (a) window of 32x32 pixels; (b) window of 64x64 pixels.

window positions in the search area, the result of performing the program was correct and window localization was true. The matching result is shown in Fig. 5.8.

Independent, additive, and normally distributed noise is overlaid on the search area in the other series of experiments. In this case, the thresholds to determine promising locations were calculated according to (5.2.5) taking into

account (5.2.6). It is interesting to note that in the case of threshold values calculated without taking into account noise effects, correct window localization is possible even with a signal/noise ratio of 1. This "resistance" to the effect of noise can be explained by the fact that noise energy is strongly reduced at the upper levels of the structure as a result of low-frequency filtration in the process of pyramid assembling, and even relatively strong interference does not result in incorrect approximate window localization at the upper levels of the structure. In the case of a significant noise/signal ratio ($D(Z)/D(S)$ about 10) thresholds decrease, especially at lower levels and this results in a 1.5 - 2 time increase of the average number of promising locations and the working time. of the algorithm.

5.3. Hierarchical Matching of Arbitrary-Oriented Template

Let us consider how the advantages of a hierarchical approach can be used for searching for a given template having a complex form, the location and orientation of which, in the search area are not known. Among multiple practical applications of this problem are indicating the predetermined object in the image, searching in pictorial databases using a prototype, etc. The preliminary selection of promising image areas, i.e. places in the search area similar to the given template, can be evidently used to speed up the process. Let us describe an algorithm for solving this problem based on the hierarchical correlation matching technique [21].

For the base, we shall use the above described matching algorithm for searching for a rectangular window-template in an image (Section 5.2). It may be formally considered as an algorithm assigned to find the maximum of the cross-correlation function in the two-dimensional search area. Generalizing the formalization for the case of an arbitrarily-oriented template, we come to the search problem in the three-dimensional discrete space, with the orientation (rotation angle) of the window with respect to the search area being used as the third coordinate.

However, an algorithm of type A57 for matching a rectangular window may not be directly used to search for a template having an arbitrary configuration. This is connected with the fact that a rectangular window which scans the search area looking for an object of complex form, covers not only the image of the template searched, but also part of the background, which acts, in this case, as noise. In order that the background found in the window together with the template does not affect too much the finding of promising locations, the summands in the

cross-correlation sum (corresponding to the cells of the t-th pyramid level) can be normalized in a way that their effect upon the final value of the correlation factor is proportional to the ratio of the area of the cell occupied by the template and the total area of this cell.

Thus, the main modifications which must be introduced into Algorithm A57 consist in normalizing the summands in the correlation sum and the introduction of the third (angular) coordinate into the search area. Let us consider each of these modifications in more detail.

Let us consider a template of arbitrary form — a prototype object — on some neutral background. The first problem to be solved is to select an adequate size for the window in which the given template should be placed in order to use this window in the matching algorithm. Too small a window will result in loss information and in a significant increase in the operation time of the algorithm due to an increase in the total number of possible window locations in the search area. Too large a window will result in the same effect due to the small ratio of the area occupied by the template to the window area, which will lead to an increase in the number of promising locations.

The optimal size of the window is calculated by coinciding the template geometrical center of mass with the window center and by finding the minimum of the template and the window difference function with variation of window size. The difference function is preset as the sum of the number of pixels covered by the window and belonging to the background and the number of template elements outside the window. It follows that, with an accurate coincidence of the window and the template (a case of rectangular image fragment search) the value of the difference function is equal to zero.

To avoid any background effect in the matching process, we introduce weights for all pixels in the window pyramid. The weights of the lowest level elements equals 1 or 0 depending on whether they belong to the template or to background, respectively. The weights of t-th level pixels are determined recurrently through the weights of the $t+1$-th level:

$$G^t_{ij} = k^{-2} \sum_{p=k_i}^{k(i+1)-1} \sum_{q=k_j}^{k(j+1)-1} \qquad (5.3.1)$$

where $i,j = 0,1, \ldots, k^t-1$

It is easy to see that the determination of the weight of the t-th level is the same as the determination of its brightness (5.2.2). Thus, the element weight which is an independent characteristic may be calculated and treated quite similarly as its brightness which is convenient for completing the pyramidal data structure.

Taking into consideration the pixel weight, the formula to calculate correlation coefficients between the search area and the template at the respective level of the pyramid may be written as:

$$r(s_{ij}^{M-t}, w_{ij}^{m-t}) = \Big(\sum_{i=0}^{a-1} \sum_{j=0}^{a-1} G_{ij}^{m-t} (s_{ij}^{M-t} - a^{M-t})(w_{ij}^{m-t} - \beta^{m-t}) \Big) /$$

$$/ \Big(\Big(\sum_{i=0}^{a-1} \sum_{j=0}^{a-1} G_{ij}^{m-t} (s_{ij}^{M-t} - a^{M-t})^2 \Big(\sum_{i=0}^{a-1} \sum_{j=0}^{a-1} G_{ij}^{m-t} (w_{ij}^{m-t} - \beta^{m-t})^2 \Big) \Big)^{1/2}$$

where s_{ij}^{M-t}, w_{ij}^{m-t} are the values of the pixels at the $(M-t)$-th and $(m-t)$-th levels of the search area pyramids and the template, respectively; a^{M-t} and β^{m-t} are their average values; G^{m-t}_{ij} is the respective weight coefficient of the window pixel at the $(m-t)$-th level, and $a=k^{m-t}$.

The algorithm correction described allows to approximate the template even more accurately while descending to the lower levels of the pyramid, thus avoiding the background effect.

The next task is taking into account in the algorithm the possible difference in the template orientation in the window and in the search area. To do this, it is necessary to measure the cross correlation when the template is rotated through a number of fixed angles respective to the search area pyramid. As was mentioned above, this is equivalent to the introduction of an additional angular coordinate in the search space. The simplest way of realising this idea is constructing a pyramid which is three-dimensional at each level. This is constructed in a quite similar manner to the two-dimensional one; however, this procedure requires additional computer memory, which may be a significant obstacle in its practical realization.

As an alternative way of accounting for angular coordinates, direct rotation of the current level of the window pyramid for a preset angle may be used in the matching process. In this case, the time of algorithm implementation does not increase significantly, as the upper levels on which the majority of the promising locations are found are rotated sufficiently quickly. It is expedient to use the conversion of the discrete coordinate system for rotation of the current pyramid level. In this case, for the matrix NxN we have:

$$I = i\cos\Theta - j\sin\Theta - (\cos\Theta - \sin\Theta - 1)(N+1)/2 + .5,$$
$$J = j\sin\Theta - j\cos\Theta - (\sin\Theta - \cos\Theta - 1)(N+1)/2 + .5,$$

where I,J are the pixel coordinates of the rotated matrix; i,j are coordinates of the pixels of the original matrix; Θ is the rotation angle.

This conversion is characterized by the following rather useful property: four pixels belonging to the 4pixel cross-like neighborhood of a given pixel will remain in its 8-pixel square neighborhood after rotation.

It is evident that with $k=2$ at the first level of the template pyramid of size 2×2 the minimal possible rotation angle is $\pi/2$. In descending to subsequent levels, the minimal rotation angle decreases twice at each level. Thus, the number of possible angles increases with the same rate as the template size, i.e. at the t-th level the measurement of correlation is potentially realizable with 2^{t-1} angles of template orientation.

Note, that a formal scheme of Algorithm A57 may be used in the case of an arbitrarily-oriented template search with the introduction of a supplementary angular coordinate in the search space. However, the corresponding threshold changes should be carried out for correct selection of promising locations. Evidently, it is still possible to use formula (5.2.1) in the calculation of new threshold values, but the value of the correlation coefficient of the search area and the template should be adequately reconsidered.

As the changes in the template form and orientation independently influence the change of the correlation measure, then their influence may be taken into account by multiplying the expression (5.2.4) by the respective correcting factor.

Let us consider first the correction connected with the template form. As in the previous Section, we shall first consider the one-dimensional Markov field. Let the window contain k^m pixels and consist of l sample of the recognized template and k^m-l samples of the background. Suppose that the background does not correlate either with the template or with the signal in the search area. In this case, only those window pixels which belong to the template and overlap the signal pixels in the search area will provide a nonzero contribution to the correlation sum (when the testing of some window location in the search area takes place).

If the template fills not less than half of the window length (and this is usually the case for the absolute majority of cases with the described technique of window size determination), then the number of pixels to be correlated as compared with the case of searching the template, the length of which coincides with the length of the window, decreases not more than by a factor of $\delta=(1-2(1-l/k^m))$. Hence, the correlation between the window and the search area decreases not more than for this value at the upper level. The indicated coefficient tends to be equal to one while descending to lower levels due to taking into account the weight coefficients. As with each descent to the next level, the number of pixels increases k times, the correlation coefficient at the t-th level can be calculated by the formula:

$$R^{*}=(1-2k^{-t}(1-l/k^{m}))\,R\,,$$

where R is the correlation at the t-th level calculated according to (5.2.4) when the background is absent.

All the above considerations may be easily repeated for the two-dimensional case. The respective formula for the correlation coefficient is the following:

$$R^{*}_{2}=(1-2k^{-t}(1-\sqrt{G^{0}_{00}}\ ))^{2}\,R_{2}\,, \qquad (5.3.2)$$

where R_2 is the correlation coefficient obtained for a rectangular template and G^{0}_{00} is calculated in accordance with (5.3.1).

To estimate the influence of template rotation through some angle on the correlation coefficient, let us consider how the correlation changes while rotating an arbitrary window filled by a two-dimensional Markov field around its center. For convenience, let us use a polar coordinate system. In this case, the location of any point is described by its distance L from the center of the coordinates and the angle with respect to the ordinate axis. For a nonseparable Markov field, the correlation coefficient of a point displaced by a distance d from its original location will change according to the formula $r(d)=r_0^{d}$, where r_0 is the autocorrelation coefficient of the Markov field. During rotation, the distance of any window point from the center of rotation remains unchanged. Therefore, according to the cosines theorem, we obtain $d=L(2(1-\cos))$. In order to obtain the correlation coefficient between the original template inscribed into the window $L\times L$ and the template rotated respective to its mass center through an angle , it is necessary to average the mentioned correlation over the entire template area. As the template size is close to that of the window, the template may be approximated by a circle of radius $L/2$ to find the correlation coefficient.

$$r(\alpha)=\frac{4}{\pi L^{2}}\int_{S}^{l\sin(\alpha/2)} r_{o}^{2l\sin(\alpha/2)}\,dS=\frac{4}{\pi L^{2}}\int_{0}^{2\pi}\int_{0}^{L/2} r_{o}^{2l\sin(\alpha/2)}\,l\;dl\;dS=$$

$$=r_0^{2l\sin(\alpha/2)}(2l\sin(\alpha/2)-1)/L^{2}\sin(\alpha/2)^{2}\ln(r_0)^{2}\Big|_{0}^{L/2}=$$

$$=(r_0^{L\sin(\alpha/2)})(L\sin(\alpha/2)\ln(r_0)-1)+1)/(L\sin(\alpha/2)\ln(r_0))^{2}.$$

In our case $L=k^{m}$. Moreover, considering that the minimal rotation angle equals $\pi/2^{t}$ at the t-th level and introducing a new variable $\tau=k^{m}\sin(\pi/2^{t+1})$, we conclude that to take into account window rotation relatively the search area, their correlation coefficient should be multiplied by the factor k_R:

$$k_R=2(r^{\tau}(\tau\ln(r_0)-1)+1)/(\tau\ln(r_0))^{2}. \qquad (5.3.3)$$

In the case of a circular template, this approximation can be used in the threshold estimation according to (5.2.5).

If the template configuration is more complex, then a more careful threshold estimation is required. The correction factor k_R can be calculated more accurately in this case. Integration in (5.3.3) is replaced by summarizing, and the correction factor dependent upon the particular template form is calculated by the formula:

$$k^{r}=\frac{\sum_{i=0}^{ddk^{m-1}}\sum_{j=0}^{k^{m}-1} G^{m}_{ij}\,r_o^{\,l_i^{2}+l_j^{2}-2\,l_i\,l_j\cos(\pi 2^{-t}))}}{\sum_{i=0}^{k^{m}-1}\sum_{j=0}^{k^{m}-1} G^{m}_{ij}}$$

where: $l_i=i-(k^{m}+1)/2$, $l_j=j-(k^{m}+1)/2$.

Thus, taking into consideration (5.3.2), we are able to finally write the expression to estimate the correlation coefficient of a template of complex form arbitrarily oriented with respect the search area fragment:

$$\sigma^{*}=R_2(1-2(1-\sqrt{G^{0}_{00}})k^{-t})k_R \qquad (5.3.4)$$

Remember that to obtain the threshold values, it is necessary to substitute this correlation coefficient σ^{*} into formula (5.2.5) instead of the σ value.

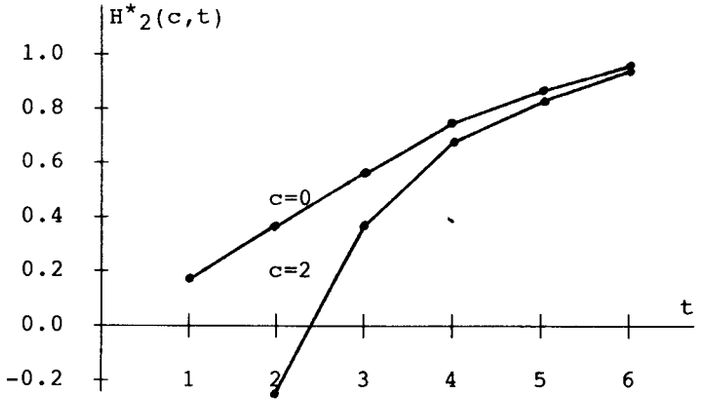


Fig. 5.9. The thresholds defining promising locations of an arbitrarily-oriented template with the scanning window and template area ratio equal to $\pi/4$.

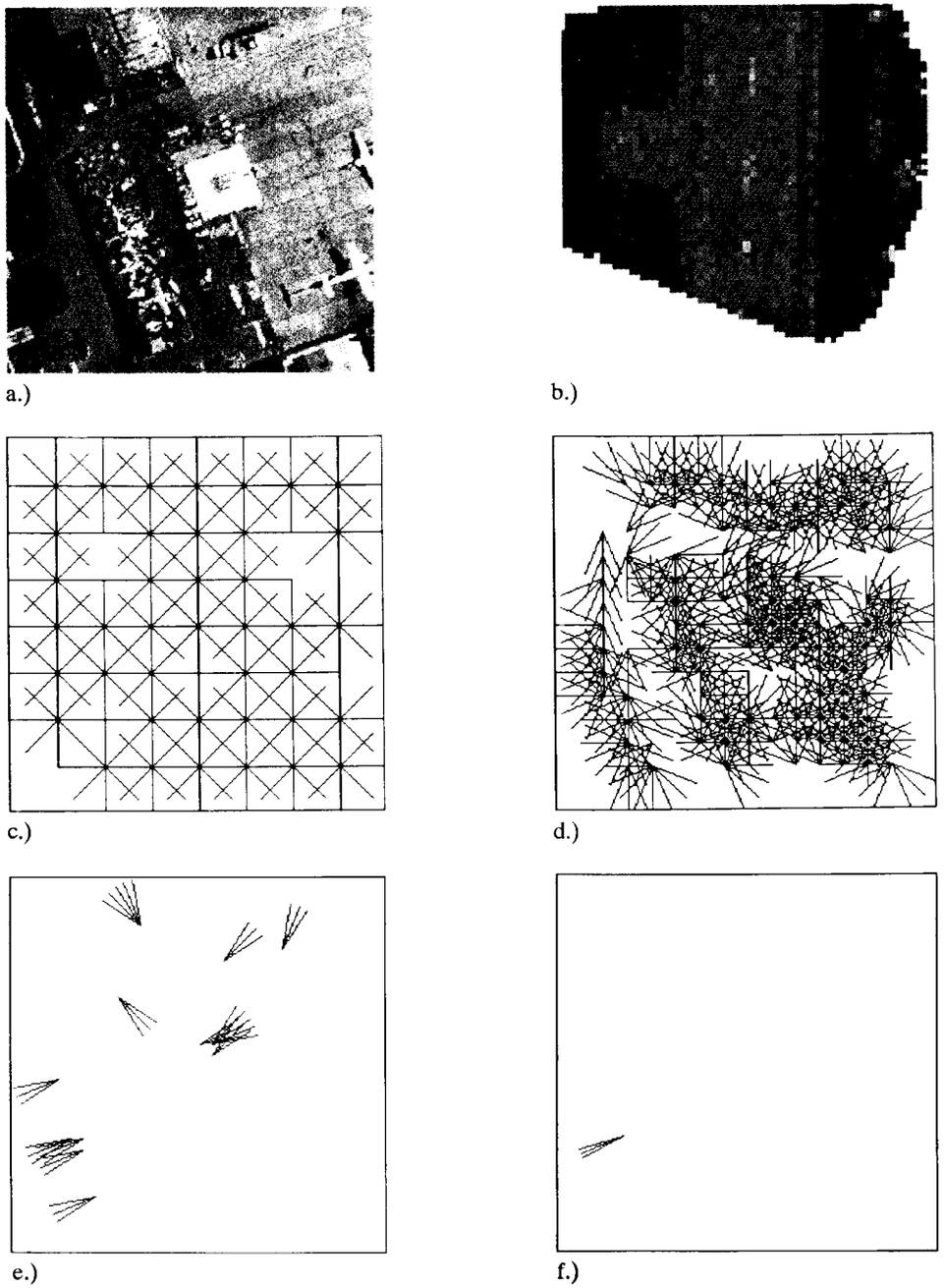


Fig. 5.10. Promising locations and orientations of the template (a) in the search area (b) at pyramid levels 2,...,5 *(c-f)*.

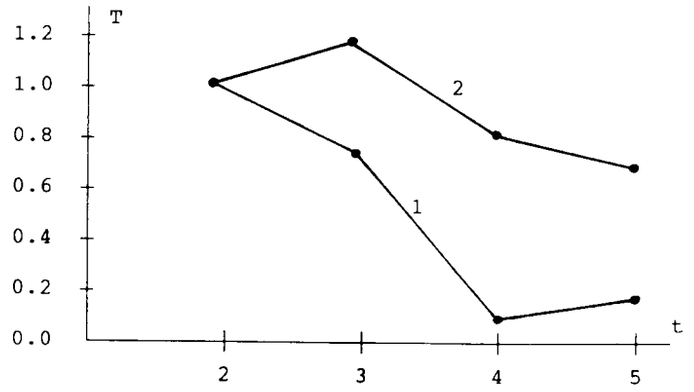


Fig. 5.11. The time distribution of an arbitrarily-oriented template matching process over pyramid levels for the cases:
(1) an unique asymmetrical template;
(2) a typical symmetric template.

As an example, thresholds to define promising locations, obtained according to (5.3.4) and (5.2.5), with the ratio of window and template areas equal to $\pi/4$ (the case of a circle inscribed into a square window) are given in Fig. 5.9. As before, these thresholds increase much more slowly than in the case of the standard oriented rectangular template search. Thus, for example, even with $c=0$ more than half the locations scanned at the first window level appear to be promising and should be considered at the second level. Hence, in the case of arbitrarily-oriented template matching, in practice it is reasonable to begin the search with the second (4x4) level.

Experiments with the algorithm were performed for three greyscale images: an air photograph, a sea bottom photo, and a portrait image. The search area size was 256x256 pixels, the window size was 64x64. Different templates occupied 70-80% of the window area.

The search began at the second level of the window pyramid. The correct localization and rotation angle of the template with $c=1$ and $c=2$ was found in 80% and 95% of the cases, respectively. Both the total search time and its distribution over the pyramid levels happens to be strongly dependent upon the template symmetry and the search area structure. In the search of the unique (for a given search area) templates with few axes of symmetry (for example, a plane in the photo) the matching process was sufficiently quick, and most time resources were spent in processing the second and the third levels (Fig. 5.10).

In the case of a search involving a highly symmetrical template having also a large number of similar prototypes in the search area image (a stone in the sea bottom photo), the time needed for the program increased significantly with the bulk of the processor time used for the lower levels. Figure 5.11 shows a typical distribution of the processor time over the pyramid levels (with $c=1$) for both the cases indicated.

In conclusion, it should be noted that the technique described allows, when necessary, to consider also other continuous template transformations (for instance, a change in scale), there by approaching the fundamental problem of identifying real objects by their images.

Chapter 6

APPLICATIONS OF PYRAMIDAL-RECURSIVE STRUCTURES AND ALGORITHMS

This chapter deals with some applications of image processing algorithms and image models based on pyramidal-recursive structures. The first section describes the features of an image processing system which uses a pyramidal-recursive representation as an internal data structure. The flow-chart of the computing process in such a system is considered.

The second section describes an optical character recognition program, which is based on an algorithm of hierarchical correlation template matching.

The third section presents an application of a pyramidal-recursive model of a binary image for the processing of microscopic images of a mineral ore to forecast the ore enrichment process during its milling at a mining concentration plant.

Section four describes alternatives of a specialized pyramidal-recursive computer architecture for image processing. Some of these are pyramids of processor elements realizing pyramidal-pipeline image processing; the other processor PRESS has a ring architecture to enable it to process image fields of arbitrary dimension.

Section five deals with the problem of expert systems for image analysis development. Some parallels are presented between human visual perception and image analysis with pyramidal structures.

The material of this chapter is drawn from [8, 10, 11, 14, 18, 58].

6.1. Data Flow Organization in Image Processing Systems

The models, methods and algorithms of image processing as described above can be used as the basis for developing special systems for image data storage, transmission and processing, with pyramidal-recursive structures being the internal image representation technique. Some principles of their construction and the particularities of their concomant data processing and decision making follow from the properties of pyramidal-recursive image representation described in Chapters 2-5. Let us summarize these: .

(1) Image data are represented in a computer memory by hierarchical structures which are described recursively. The interrelations of structure elements are regular and do not depend upon the contents of an input image. On the contrary, the filling of structure elements with concrete information is determined by input image data. The ways of structure filling for different image types are given by the relations (2.5.4)-(2.5.6) which are formulated relative to the elementary structure cell. Recursive application of these relations to the whole structure enables the pyramidal-recursive representation of different type images to be obtained.

(2) The manner of image data structuring predetermines the natural way of breaking down the image processing stages: the problem being solved is divided into subproblems, the interrelations between which are determined by the interrelations between the pyramidal-recursive structure elements. In such a decomposition, a structural unit relative to which a processing algorithm is described is an elementary cell of a pyramidal-recursive structure or its separate level. Solving the problem as a whole is realized by the application of an elementary processing algorithm to the entire structure.

In level-by-level structuring, the energy of the initial image is distributed over different structure levels as described by relations (3.2.14), (3.3.6)-(3.3.8). Thus, it is possible to say that an elementary algorithm at each level uses a certain part of the initial energy or input information. The volume of this part for images satisfying the pyramidal-recursive models proposed in Sections 3.2 and 3.3. may be predicted beforehand.

(3) The algorithms process the structure in accordance with a top-down strategy which corresponds to the scheme of the recursive data processing [6, 13] (Fig. 6.1a). They analyze data stored at the current structure level so as to refine at each level the approximate results obtained after the processing of preceding levels.

The sequence of gradually refined results is obtained while an algorithm

accomplishing at each level. The accuracy of current results depends upon the dispersion of brightness at corresponding structure level. The inherent accuracy of level-by-level processing for images satisfying the pyramidal-recursive model is determined according to relations (4.3.6), (4.4.2) and the algorithms of Sections 5.1-5.3.

(4) The elements of a pyramidal-recursive structure does not describe individual pixels of the input image but, rather, groups of pixels having given properties. Therefore, the data volume in the case of pyramidal-recursive representation depends not on the volume of input data (the number of the initial pixels N), but on the real complexity of the analyzed image expressed through the number M of nonredundant structure nodes, with M being much less than N. The results presented in Chapter 4 enable an estimation of the number of nonredundant nodes in a truncated pyramidal-recursive structure and the volume of data associated with each node sufficient to describe the input image with a given accuracy. This facilitates the estimation of the data volume subject to be processed at each structure level.

Summarizing the principles presented above, one can say that the results of Chapters 2-5 enable to determine in what way a pyramidal-recursive structure can be constructed and filled by data, what data volume is to be processed at each stage of the level-by-level processing, what fraction of the input information (energy) will be used, and at which point can results of a given accuracy can be achieved.

The utilization of pyramidal-recursive structures opens various possibilities relating to computational process control in digital image processing. Thus, if an image is processed not by a single, but by n sequential algorithms, then it is possible to organize an algorithmic pipeline through which the data to be processed are passed level by level (Fig. 6.1b).

In this case, the result of processing the first level by the first algorithm $RES_1(1)=A_1(I(1))$, is used, first, for refining by the same algorithm $A1$:

$$RES_1(2)=A_1(RES_1(1), I(2)),$$

and, second, for the processing by algorithm $A2$ to obtain the first approximation of the image processing result by two algorithms $A1$, and $A2$:

$$RES_2(1)=A_2(RES_1(1))=A_2(I(1)).$$

In the general case, after the $(t+n)$-th step, the n-th algorithm provides the t-th approximation of the image processing result by all n algorithms to a depth of t levels:

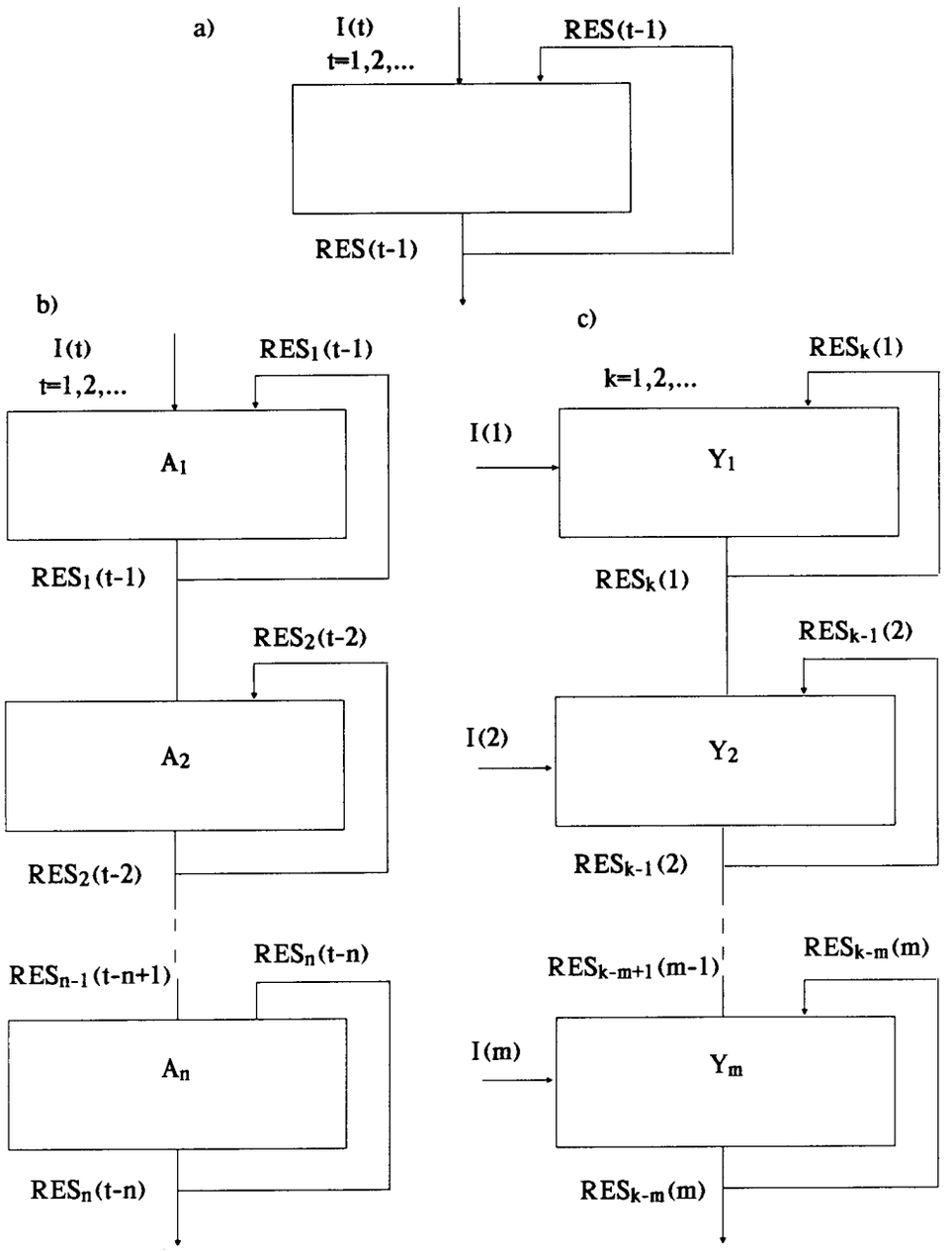


Fig. 6.1. Flow-chart of progressive data processing with a gradual refinement of results: (a) by a single algorithm;
(b) by a set of algorithms; (c) by a set of processors.

$$RES_n(t)=A_n(RES_n(t-1), RES_{n-1}(t)).$$

If the result obtained after the processing of t levels by all n algorithms is acceptable, then no further work is needed. In the case of the parallel performance of all n algorithms (for example, by n parallel processors), $n+t$ steps will be required to achieve this result, while a satisfactory realization of the goal using a serial one-processor computer requires nt steps.

In both cases, a significant reduction in the time required for decision making ($n(m-1)+t$ steps) can be achieved compared with the alternative procedure when every subsequent algorithm has to accomplish a complete processing of the structure, and only then is control transferred to the next algorithm. It should be taken into consideration that those steps which correspond to the processing of lower levels of the structure need the greatest time, because it is during this stage that the greatest data volume is processed. The pipeline of algorithms allows to execute these steps (if the decision at this moment has not yet made) at the end of the procedure.

If several parallel processors are available, it is possible to organize another pipeline for the processing of an image represented by a pyramidal-recursive structure. Each processes a certain structure level by a sequence of different algorithms (Fig. 6.1b). In this case, the t-th approximation of the image processing result is received after t+n steps using a chain of n algorithms:

$$RES_n(t)=A_n(RES_{n-1}(t), RES_n(t-1)).$$

The difference between this and the previous alternative is that the processor power should be proportional to the average data volume treated at the corresponding level. The second alternative might be more preferable than the first in problem-oriented image processing systems, as it enables pipeline processing of both an individual image and a sequence of images even by a single algorithm (the basis of parallel processing is the level-by-level data structuring, and not the presence of several sequential processing algorithms).

There exist various possibilities of flexible control of computations when the processing of a sequence, or a set, of images is required due to various alternatives in the sequencing of a level-by-level processing of corresponding pyramids by one or several algorithms. While processing a set of images, the t-th level of each image can be processed just after the t-th level of the previously processed image. Thus, the first approximation of the result can be obtained for the whole set of images in a given time, then the second approximation for all the images can be interpreted, and so on. When a certain accuracy in the result for each image is realized, for

instance, it is determined that certain images do not correspond to the pattern being searched for, all levels of their structures which have not yet been analyzed can be excluded from further consideration and the pipeline then continues to refine results for a smaller number of (remaining) images. Thus, it is possible to organize the change in the accuracy of the result for a single image (while processing images one after another and making a decision for each of them) to the rate at which the approximate result for a set of images is received (in "parallel" processing of all images level after level and making decisions at each step for all images under consideration).

The choice of the processing strategy can be of great importance, for instance, in the analysis of multispectral images or stereo pairs, and motion analysis and control over sequences of images. It should be noted that in many cases it is necessary to process not only brightness data, but also dispersion, texture characteristics, etc. In such cases the image analyzed is represented by several pyramidal-recursive structures, each of which corresponds to its own characteristics. The time required for the analysis of such images may be significantly reduced by the choice of an appropriate strategy of decision making in different combinations of scanning a set of structures "in depth" (by levels) and "in width" (by a set of structures).

Thus, image representation with pyramidal-recursive structures provides some new possibilities in image processing systems such as enlarging the set of tasks which can be solved, a gradual refinement of intermediate results and decisions, and the realization of various decision making strategies in accomplishing one or several algorithms.

6.2. Optical Character Recognition

The task of automated typewritten text reading using optical character recognition is, at present, one of the popular problems both in research [24, 25, 93] and commercial [75, 126] applications. There are two main approaches to solving this problem: the development of "taught" and "intellectual" programs.

The operation principle of algorithms of the former type is the successive comparison of each character scanned by the program with all templates (character prototypes) stored in the computer memory. When the template identical (or similar) to the character concerned is found, the character is identified as being recognized and the respective code is recorded in the memory. The "teaching" of a recognition program consists of introducing a new set of character prototypes (new fonts).

When intellectual programs are operating, it is not the specific character which is compared with a set of templates, but some distinctive features of their abstract generalizations which are obtained as a result of character image transformations.

Both approaches to character identification have their advantages and drawbacks. "Taught" programs can be adjusted in practice by the user to any font of any alphabet and are not very sensitive to document quality, although they not can usually recognize more than one font at a given time and need to be taught any new font. On the contrary, "intellectual" programs need not be taught any new type of text to be read and can process documents using variable fonts. However, difficulties in identification can appear with documents of average or poor quality. Besides, it is usually not possible to teach these programs new alphabets. In practice, both recognition techniques are often used together.

This section presents some ideas of image processing with pyramidal structures implemented into the algorithm of typewritten character recognition. The operation is based on the successive comparison of every character to be recognized with prototype images of the alphabet characters.

Let s_{ij} and w_{ij} be pixels of the character recognized and a template from a set of prototypes, respectively. Comparison of these images is made in the rectangular window $l \times n$, so that $i=0,1,\ldots,l-1$; $j=0,1,\ldots,n-1$. The normalized deviation of the character image and the template in the simplest case is calculated using the formula:

$$d(s,w) = \left(\sum_{i=0}^{l-1} \sum_{j=0}^{n-1} | (s_{ij} - w_{ij}) | \right) / nl \qquad (6.2.1)$$

Let $d(s,w(i))$ be the deviation of the character identified from the i-th character (or from the i-th template) of the alphabet; then the condition that the character identified is the a-th character of the alphabet can be written as:

$$d(s,w(a)) = \min_i \{d(s,w(i))\} \text{ and } d(s,w(a)) \leq d^*,$$

where d^* is the maximum deviation (threshold) with which the decision concerning character recognition can be made.

A primary parameter effecting recognition quality and speed is the size of the rectangular window in which templates are stored (this dimension is related to the scanner resolution with which images are entered). Characters become faintly distinguishable with too rough a resolution, and the algorithm operation time significantly increases with high resolution. A way out of this situation which provides acceptable recognition quality at a sufficient rate is the use of hierarchical template matching. Comparison of characters in this case is done progressively

using their pyramids in a level-by-level manner from top to bottom (see Section 5.3).

Some alphabet templates which are non-promising for subsequent comparison can be refused at each pyramid level during a top-down comparison process. Only a few templates having the greatest similarity to the character identified "reach" the lowest level. At the same time, the calculation time of the deviation decreases by a factor k^2 if the deviation is estimated at the *(t-1)*-th pyramid level instead of the *t*-th level. As a result, the algorithm operation time decreases several times without any practical decrease of recognition quality.

An important problem impressing the algorithm operation efficiency is the selection of the threshold d^*_t optimum value at each *t*-th level of the pyramid. The theory described in Section 5.2 fails to help in this case, as it is not possible for images of character type to be adequately described by a Markov field with a sufficiently high autocorrelation coefficient. Let us consider another approach to this problem.

The image of the character identified, *s*, can be conditionally represented as the sum of the ideal template *w* and some random additional value *s*. This value is attributable to the difference between any real character image and the same character prototype image.

In turn, the value *s* can be decomposed into two independent summands: that connected with defects and specific attributes of the typing device - β; and that connected with distortions arising in the image of a character entered into the computer memory (noise of the scanner, discreteness of the image received, etc.) - α: $s=\beta+\alpha$ An example of the first summand could, for instance, result from a change of a character image solidness or some character "striking". A typical example of the second summand would arise from a specific "fringe" along the character boundaries.

It should be noted that the summands β and α at upper levels of the pyramid have a principally different behavior. As the value β is connected with statistically stable correlations of a large number of pixels, it is practically constant at upper levels. The value α, on the contrary, is connected with random independent variations of pixels of the original character image. Hence, contributions of these variations to are mutually compensated at upper pyramid levels and, as a result, decreases rather quickly with pyramid level growth. Thus, it turns out that in most cases we can assume $s^t \approx \beta^t \approx \beta^m$ (*t* is the level number; and *m* is the quantity of levels in the character and template pyramids) for text recognition with a typical scanner resolution (150 dots/inch or more). From this, an important conclusion can be drawn: the threshold d^*_t can be taken to be a constant for all pyramid levels: $d^*_t = d^*$.

To understand how to choose the particular value of d^* and to represent more clearly the essence of the problems which arise, let us consider the simplest case: identification of an alphabet consisting of only two characters. Call these *A* and *B*. According to the assumption made, an image of the character identified can be decomposed into three components:

$$s_{ij}(A) = w_{ij}(A) + \beta_{ij} + \alpha_{ij};$$
$$s_{ij}(B) = w_{ij}(B) + \beta_{ij} + \alpha_{ij}. \qquad (6.2.2)$$

Taking into account (6.2.2), and condition $\alpha_{ij} < \beta_{ij}$ which is true for many real cases, (6.2.1) can be rewritten as:

$$d(s(A), w(A)) \approx |f_d(A)|\ ; \qquad (6.2.3a)$$
$$d(s(B), w(A)) \approx |f_w(A,B) + f_d(B)|\ , \qquad (6.2.3b)$$

where $f_d(A)$ and $f_d(B)$ is the noise of typing device for characters *A* and *B*, and $f_w(A,B)$ is the difference of characters *A* and *B*.

It should be noted that in (6.2.3) the value $f_w(A,B)$ is not dependent on the particular character recognized, and hence is a constant for a given pair of characters.

Let us introduce two more assumptions. Let random values $f_d(Q)$ be uniformly distributed in the interval $[-f^*_d(Q), f^*_d(Q)]$ and, in addition, $f^*_d(A) = f^*_d(B) = f^*_d$. Neither of these suppositions are of a principal character, however, they enable simple analytical expressions for d^* to be obtained. There are two principally different alternatives as illustrated in Figs. 6.2 and 6.3.

If the condition $f_w(A,B) \geq 2f^*_d$ (Condition 1) is satisfied, then densities corresponding to possible deviations of the characters *A* and *B* are completely separated (Fig. 6.2). Conformably, with any threshold satisfying the condition

$$f^*_d \leq d^* \leq |f_w(A,B)| - f^*_d \qquad (6.2.4)$$

the probability of a correct identification is $P_{tr}=1$, the probability of false identification is $P_{false}=0$, and the probability of identification reject is $P_{br}=0$.

Another situation takes place if $f_w(A,B) < 2f^*_d$ (Condition 2). Densities corresponding to the characters A and B partially overlap in this case and correct identification of these characters is principally impossible (Fig. 6.3). It is reasonable to take the threshold d^* within the ranges:

$$f^*_d \leq d^* \leq |f_w(A,B)| - f^*_d$$

A case of error-free identification with the maximum amount of rejects ("distrustful" recognizer) corresponds to $d^*_d = |f_w(A,B)| - f^*_d$. The respective probabilities are:

$$P_{br} = ((2f^*_d - f_w(A,B))/f^*_d)^2;\ P_{tr} = 1 - P_{br};\ P_{false} = 0;$$

Obligatory character identification with the maximum number of errors ("trustful" recognizer) corresponds to $d^*_r = f^*_d$. In this case:

$$P_{false} = (2f^*_d - f_w(A,B))/f^*_d)^2/2;\ P_{tr} = 1 - P_{false};\ P_{br} = 0;$$

When the threshold d^* fluctuates within the ranges given above, the respective probabilities P_{br}, P_{false} and P_{tr} change continually within the respective interval. Having defined a certain P_{false} value (the remaining probabilities being determined ambiguously) depending on the particular recognition task, the appropriate threshold d^* can be found.

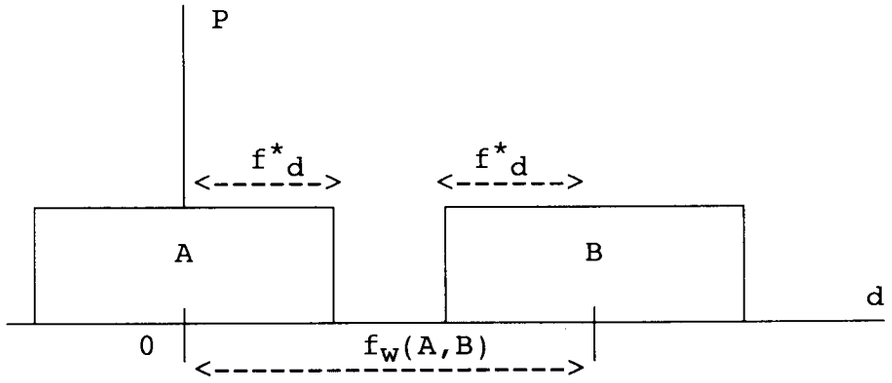


Fig. 6.2. Density of distributions of $f_d(A)$ and $f_w(A,B) + f^*_d(B)$ values for the case of $f_w(A,B)$ $2f^*_d$

Generalization of the inference for the case of an arbitrary volume alphabet and for a more realistic distribution of the $f_d(Q)$ value does not qualitatively change the conclusions drawn. The density of the $f_d(Q)$ distribution is close to a Gaussian, than to a uniform, distribution. Parameters of distributions for various characters are significantly different; however, as before, there exists a varying interval of d^*, within the limits of which a reasonable compromise between the number of characters erroneously recognized and those rejected can be reached.

A program for typewritten text recognition has been developed based on the ideas described. Its specific features are a sufficient tolerance to the quality of the

documents to be entered into the computer and the ability to work with comparatively low scanner resolution (150 dots/inch). The latter enables working in a "conveyor" mode even with IBM PC AT and IBM PC XT class computers by recording 10-20 files with images of text pages at a time and subsequently their automatic recognition.

A wide range of thresholds d^* was experimentally investigated during the study of the program operation. In practice, no identification error ($P_{false} \le 0.005$) was observed for a mean quality text with $d^* = 0.08$ and a significant number of rejects ($P_{br} = 0.3$). Moreover, with an increase in d^* the number of errors gradually increased while rejections decreased. With $d^* = 0.2$, P_{false} reached 0.09 while P_{br} dropped to 0.001. Further increase of the threshold had practically no effect upon the change in probabilities mentioned above. These probabilities can be observed for a wide class of character pairs for which Condition 2 is satisfied.

As the investigation of a set of template characters shows, the number of character pairs with a dissimilarity less than a certain given f^*_w limit set increases from 1 at $f^*_w = 0.08$ to 184 at $f^*_w = 0.2$. Figure 6.4 shows a distribution graph for the mentioned number of characters M depending on f^*_w.

An average template dissimilarity from an image of a real character equals 0.17. As can be seen from Fig. 6.4, there are about 100 pairs of characters having a dissimilarity less than this value. Condition 2 is fulfilled for these. Thus, the predictions of the theory are proved.

While program operation in the mode of character recognition entered with a resolution of 150 dots/inch, about 10% of the processor time was used for reading and unpacking each following line of the text, about 30% for the construction of a pyramidal structure and about 60% for the progressive character recognition by the pyramid levels.

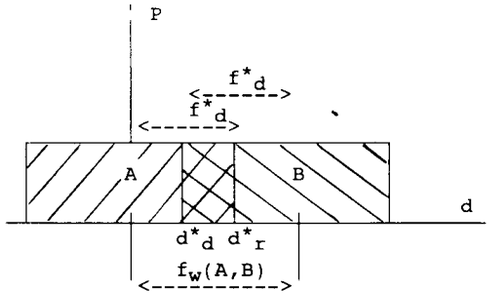


Fig. 6.3. Density of distributions of fd(A) and $f_w(A,B) + f_d(B)$ values for the case of $f_w(A,B)$ $2f^*_d$

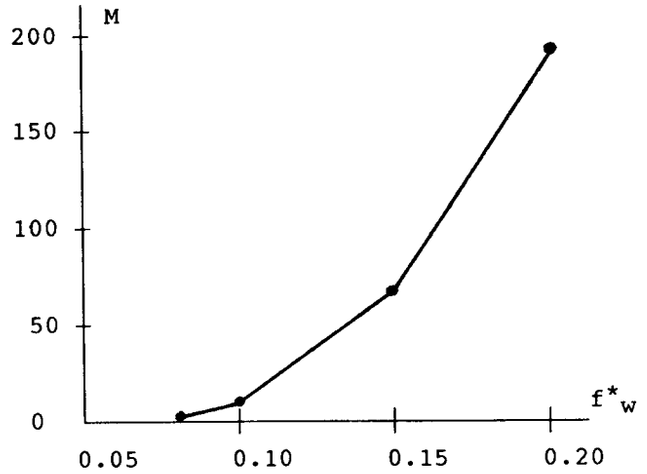


Fig. 6.4. Amount of character pairs M having a dissimilarity less than a threshold f^*_w for different thresholds.

Figure 6.5 shows the typical distribution of the number of characters tested at each pyramid level in the identification process of Cyrillic alphabet characters. As can be easily seen, in average, only 3 characters out of 90 (the standard set on a Russian typewriter) could be considered at the lowest level of the pyramid. Document quality varied from poor to good, the speed of recognition with an IBM

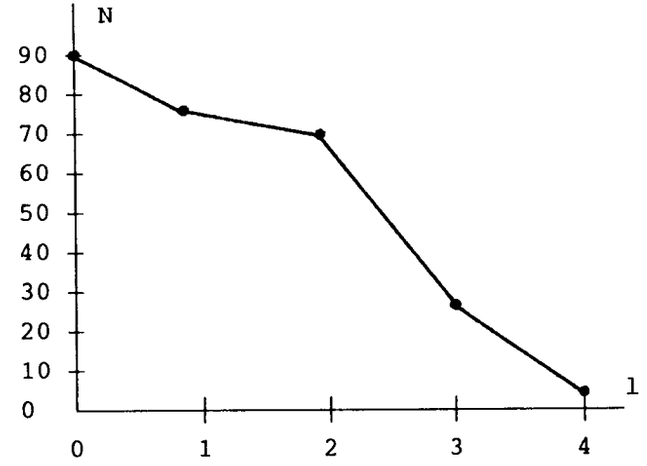


Fig. 6.5. Typical distribution of the number of characters tested (N) at each pyramid level *t*.

PC AT (12 Mhz) computer changing from 5 to 10 characters per second and the percentage of correctly identified characters improved from 90 to 99, respectively.

6.3. Modelling of Ore Milling

Digital image processing is widely used now in mineralogy; in particular, for the analysis of the micro-structural composition of ores at mining concentration plants for forecasting the enrichment ability of the raw materials. Microscopic images of flat cuts of ore specimens with grains (regions) of a useful product included in the mineral mass are used for this purpose. Various parameters of grain textures and structures are measured on these images. Using these data, models of the separation of ore phases (i.e. separation of the useful product from dead rock) in the process of mineral mass crushing are developed.

An important role in ore quality forecasting is played by the so-called ore milling-opening diagram which is a three-dimensional graph showing the percentage of ore particles containing the preset proportions of the useful product in particles of a fixed size. It can be briefly described as follows.

First, ore is crushed into large pieces of size significantly larger than that of the useful product grains (magnetite for iron-ores). In this case, the curve of the density distribution of the volume fraction of magnetite in ore pieces is approximately Gaussian. At the subsequent milling stages, the diameter of ore particles decrease, and the distribution of the magnetite volume fraction changes (its dispersion increases, particles consisting of pure rock quartz or pure magnetite appear). At least, the distribution changes to a discrete one with the magnetite fraction content in a single particle equalling 0 or 100%. The quicker the valuable mineral can be separated from the ore (i.e. the less is the milling rate required) the higher is the quality of the ore.

Enrichment strategies depend on the ore quality. If the milling-opening diagram is known, then some technological parameters of milling plants and separation strategies can be chosen to obtain a high quality of concentrate with optimal energy expenditures for mineral mass milling. Milling-opening diagrams are often developed experimentally during the physical milling of ore samples and by taking particle samples of different grades from the dispersion mass. However, it is also essential to be able to construct computer-based models describing milling-opening diagrams, which enable various technologies of milling and separation to be considered beforehand, by analyzing digital images of ore cuts.

The application of pyramidal-recursive models of binary images developed in

Sections 3.3 and 4.4 makes possible the simulation of some peculiarities of the milling process and the construction of a milling-opening diagram based on the analysis of microscopic images of ore specimens.

Actually, progressive image decomposition into cells of different order, when the construction of a pyramidal-recursive structure takes place, resembles the stepwise ore crushing process. In this case, cells of the first decomposition order correspond to particles of the first grade class, cells of the second partition — to particles of the second class, etc. (usually average particle sizes in different grade classes are in a constant ratio, for example 1:2:4:8:16...).

Each grade class is subdivided into a number of mineral value classes depending on the volume fraction of the useful product. In the structure representing the binary image, the characteristic of the mineral value class is the node color: black nodes correspond to the useful product, white nodes to dead rock, and grey nodes to aggregations of ore and magnetite which have not yet been separated. If the node color does not have three gradations, but its brightness is determined more accurately (for instance, as for greyscale images), then it is possible to imitate separation into a larger number of mineral value classes. However, investigation of four classes is sufficient in the first approximation (the color of each mode is characterized by two bits in this case).

There are some results in integral geometry and stereology [115, 118] which enable linear and surface estimations to be transformed into volume estimations. In particular, the volume fraction of the useful product in each grade class and mineral value class is determined by the corresponding surface fraction estimation for a cell of the respective level in the pyramidal-recursive structure.

To estimate the milling-opening diagram for four mineral value classes, it is sufficient to find at each level of the pyramidal-recursive structure the number of cells of four different types: black (B); grey, not less than 50% of which is black on the image of lowest level (G_1); grey, less than 50% of which is black (G_2); and white (W). These areas can be found directly for each image analyzed or in the process of the structure construction by the summation of the black areas of cells of lower levels.

Let us use the two-layer model of a binary image to simulate the ore milling-opening process. Magnetite is often included in a quartz mass as grains, for which reason microscopic images of ore cuts often have a "spot" structure.

Suppose that sections of magnetite grains have a circular shape. Then, the parameter t_0 of the two-layer model (see Section 3.3) is related to the sizes and the amount of grains in an image. The sum of the perimeters of grains is approximately equal

$$L = \pi \sum_{i=1}^{N} D_i = ND_0,$$

where N is the number of grains, and D_0 their average diameter. The total area of the grains can be estimated as

$$S = \sum_{i=1}^{N} S_i = \frac{\pi NK}{4},$$

where K is a square of the average grain diameter. Then by estimating the parameter t_0 for the image, it is possible *to* estimate the average diameter of the magnetite grains, and its dispersion. These are important characteristics of ore quality.

The same parameter allow to estimate the diagram of ore milling-opening for four mineral value classes of particles. Let us assign particles consisting of pure magnetite to the first mineral value class (denote B). Their volume fraction in the t-th grade class is determined by the square of a "black" cell of the t-th decomposition. This can be estimated using (3.3.6):

$$B(t) = \begin{cases} 0, & t<t_0 \\ S(1-k^{-(t-t_0)}), & t \geq t_0. \end{cases} \qquad (6.3.1)$$

where S is the area of black regions of the image. Aggregations of magnetite and quartz are assigned to the second and the third classes (G_1 and G_2, respectively). Their summary volume fraction in the t-th grade class is determined by the number of "grey" cells of the t-th level:

$$G(t) = G_1(t)+G_2(t) = \begin{cases} 1, & t<t_0 \\ k^{-2t}g(t) = k^{-(t-t_0)}, & t \geq t_0. \end{cases} \qquad (6.3.2)$$

The fourth mineral value class (W) into which dead ore falls is determined, taking into account that $B(t) + G_1(t) + G_2(t) + W(t) = 1$, as follows:

$$W(t) = \begin{cases} 1, & t<t_0 \\ (1-S)(1-k^{-(t-t_0)}), & t \geq t_0. \end{cases} \qquad (6.3.3)$$

It follows from relations (6.3.1) and (6.3.3) that at each decomposition level the volume fraction of magnetite in a class $G(t) = G_1(t) + G_2(t)$ remains constant and equals S. The ratio of volume fractions of classes $G_1(t)$ and $G_2(t)$ equals $S/(1-S)$, therefore:

$$G_1(t)=\begin{cases} S, & t<t_0 \\ Sk^{-(t-t_0)}, & t\geq t_0. \end{cases} \qquad (6.3.4)$$

$$G_2(t)=\begin{cases} 1-S, & t<t_0 \\ (1-S)k^{-(t-t_0)}, & t\geq t_0.. \end{cases} \qquad (6.3.5)$$

The relations (6.3.1) - (6.3.5) enable ore opening to be forecasted at different stages of milling (Fig. 6.6). They also allow to predict more accurately the best moment for halting the milling process, which, in turn, would result in a decrease in energy consumption for ore crushing.

Some existing technologies of fraction separation assume that different mineral value classes are extracted from a disperse mass upon completion of the crushing process. At the same time, there should be possibilities to choose some ore phases, particles of which need not be crushed further, at each of the milling stages during the separation process. The above model helps to describe such a technique. For example, if the first mineral value class corresponding to pure magnetite is extracted at each stage of milling from a mineral mass, and residuals are directed into dead rock, then volume fractions of different mineral value classes are described by the following relations:

$$b(t)=\begin{cases} 0, & t<t_0 \\ S(k-1)k^{-(t-t_0)}, & t>t_0. \end{cases}$$

$$g_1(t)=G_1(t);\ g_2(t)=G_2(t); \qquad (6.3.6)$$

$$G_2(t)=\begin{cases} 0, & t<t_0 \\ (1-S)(k-1)k^{-(t-t_0)}, & t>t_0. \end{cases}$$

The volume fraction of ore being milled compared to the original mass permanently decreases in this case and equals k^{t_0-t+1} $(t>t_0)$, which provides a sharp decrease in energy expenditure for mass crushing to obtain particles of grade class $t>t_0$.

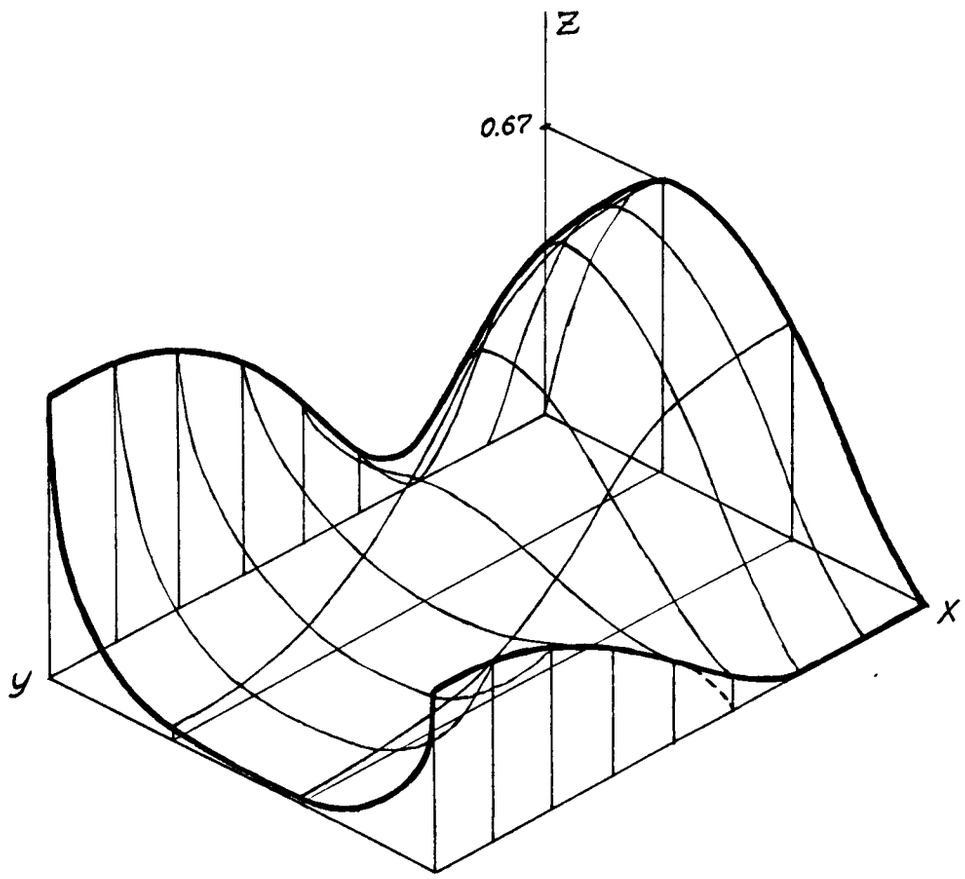


Fig. 6.6. Milling-opening diagram of a two-phase ore, based on the two-layer model of a binary image (t_0=3, S=0.67)

x - mineral value class; y - particle grade class; z - percentage of particles (fraction volume).

6.4. Special Devices for Image Processing

Recent progress in computer hardware development makes it possible to create special processors for operating with practically any data structure, or realizing any algorithmic structure. Pyramidal-recursive structures are very suitable for implementation in high-power special devices with parallel data processing. Work in this direction began in the early a 70's [134, 138, 139] and were developed

simultaneously with the theoretical study of pyramidal data structures and their processing algorithms [3, 37, 61, 88, 91].

Two features of pyramidal structures - the possibility of simultaneous operations with many pixels, and regular hierarchical interrelations of elements - predetermine a natural way for the construction of special hardware. This reflects the pyramid data structure in the computer architecture. In this case, each structure node is supposed to be brought into correspondence with the so-called processor element (not a very complex device which accomplishes elementary arithmetic or logic operations with data). As is shown in [140], this approach provides the best time characteristics for performing a large range of various operations with images as compared with some other approaches to the construction of special computers for the processing of two-dimensional images.

There exist already many different proposals and realizations of this type of device [30, 37, 42, 43, 116, 133, 136, 144]. The main idea being the construction of these devices is the parallel processing of pixels at each individual pyramid level and successive level processing. The data flow from bottom to top, or from top to bottom, through the pyramid of processor elements. An additional opportunity for computational process paralleling is in operating with several images at the same time due to the organization of a data processing pipeline (see Section 6.1), i.e. activation of all processing elements at each moment and operating with different images at different levels. This is a distinctive feature of pyramidal computational structures which distinguish it from matrix processors or homogeneous computational media, where processing of only one image element normally takes place [20, 47, 61, 133].

This section considers some features of the construction of pyramidal processors and a variant of specialized computer architecture for the processing of pyramidal-recursive data structures. The presentation is based on [11, 14].

Important differences in pyramidal computers, as suggested by different authors, are the quantity of processor elements (PE) and the "topology" of their interconnections (it is usually supposed that all PE's of the same level are identical). The main types of interconnections are determined by the existence or absence of lines of communication between processors of the same level, and the existence or absence of two or several PE's lines "overlays" at the lower level (see Fig. 6.7).

Pyramidal structures with overlays are aimed at facilitating the work with the neighborhoods of pixels. "Horizontal" interconnections of PE's serve the same purpose, therefore a combination of overlay and horizontal links of PE's of the same level (Fig. 6.7g) is not usually used. Interconnections of the type shown in

Fig. 6.7a provide access to neighboring pixels only through the upper levels.

One other important difference of the proposed pyramidal processors is the direction of data flow. If the data flow in the pyramid goes in one direction only, from bottom to top or from top to bottom, then the organization of pipeline processing of an image set is possible with the pipeline capacity being proportional to the number of pyramid levels and inversely proportional to the operation time at each level. Horizontal data flow among processor elements of the same level do not prevent pipeline operation if the access time to data at this level is the same as that in the "vertical" direction. If, on the contrary, data flow is allowed to be both ascending and descending, then a continuous pipeline not can be organized without additional hardware requirements.

It should be noted that an analogous problem arises in the processing of an image by a pyramid having a descending data flow, if the filling of the PE's structure by data (i.e. the process of the initial image representation by a pyramidal structure) is organized as an ascending data flow. Usually in this case, pipelining is used only in performing individual operations (algorithms) of image processing [147].

The advantages of pyramid-like computer systems for image processing are the simplicity of the architecture of individual processor elements, the regularity of the interconnections of PE's, and the pipeline processing of a set of images. At the same time, such computers have some disadvantages.

First, a computer having a pyramidal architecture cannot take advantage of those algorithms which work with truncated trees: the complexity of such algorithms depends, as a rule, not upon the number of nodes in a full tree, but only upon the number of terminal nodes, which are usually one order less. The complexity of algorithms in pyramidal processors is usually determined by the number of levels with PEs, i.e. it depends on the number of nodes in a full tree. This means that a great number of PEs corresponding to redundant nodes execute unnecessary operations or simply wait. The rigidity of interconnections in the structure does not allow unattached PEs to be used for any other operations, and the levels of the structure are forced to work synchronously in the mode "one command - multiple data".

Second, only pyramidal machines for two-dimensional image processing are being developed. In the three- and multi- dimensional case the number of both PEs and their interconnections sharply increases, which complicates the engineering and technological realization of the pyramidal processor concept. Certainly, one PE is able to process not just a single tree node but several nodes at a time, leading to a tree of a less height [3]; however, in this case, no proposals are known which are

directed to developing machines for the processing of multidimensional fields, or to processing fields of different dimensions.

This motivate us to sketch here some considerations on special computer architecture which is able to process images (information fields) of an arbitrary dimension represented by truncated pyramidal-recursive data structures. We call this PRESS [11, 14]. An important difference of PRESS from processors having a pyramidal structure is its possibility to operate with a varying number of PEs irrespective of their malfunctions, the absence of PE's downtimes, and the processing of multidimensional images. This is attained due dynamic organization of the necessary PE's connections based on the use of positional coordinates of the structure nodes instead of using of fixed PEs connections.

PRESS is based on the representation of the image processed as a truncated pyramidal-recursive structure. For each operation with an image, this structure can be considered to consist of a number of elementary structure cells processed in serial by levels and in parallel inside each level; all operations with the elementary structure cells being of the same type. Therefore, each PE processes at any moment a single elementary cell of the structure. A principle distinction is that elementary cells are processed in the required order, i.e. an important feature is the opportunity to reorder the input data flow. For this, the array of elements of the pyramidal-recursive structure is recorded in a special manner into a high-speed ring dataway, where data then circulates permanently (Fig. 6.8).

This ring, dataway is the memory of PRESS. It can be considered to be a set of memory cells which quickly rotate relative to unmovable PEs. The data array to be processed is opened by some specially marked memory cell. Data ordering in the dataway is determined by the positional coordinates of the pyramidal structure nodes. The contents of memory cells are brightness values and optionally some other characteristics of tree nodes. The total amount of memory cells reserved for these data is equal (or proportional) to the number of nodes in a full (not truncated) structure; however, the memory is filled in each particular case with

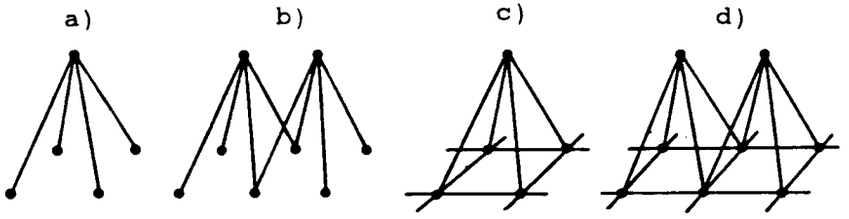


Fig. 6.7. Different types of PE interconnections in pyramidal processors.

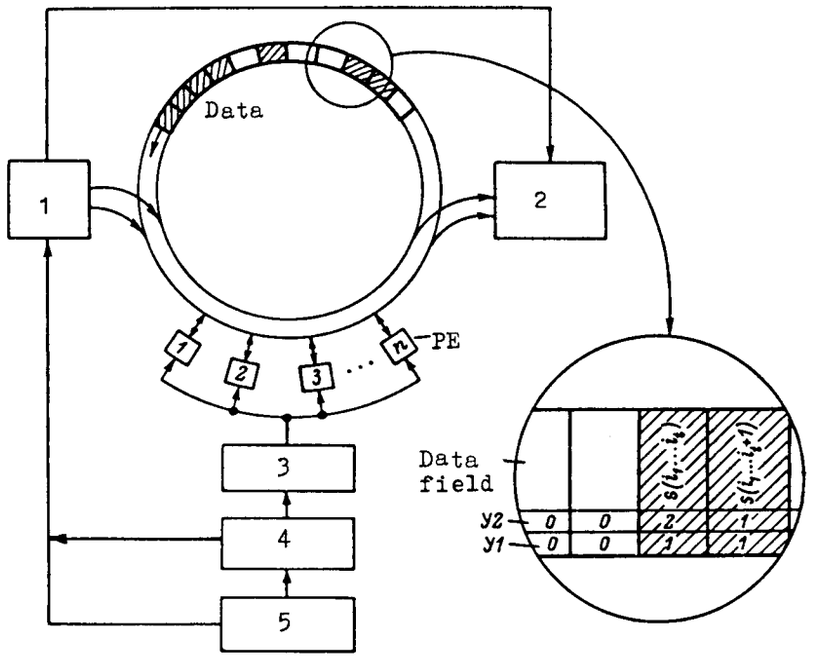


Fig. 6.8. Flow-chart of the PRESS processor: 1 - input preprocessor, 2 - output postprocessor, 3 - command memory, 4 - interpreter, 5 - user interface, Y1, Y2 - flags.

only those data which correspond to the nodes of the truncated tree. Thus, the number of each cell in the circulating array is in a one-to-one correspondence with the positional coordinate of the respective structure node, and access to the data in the memory of PRESS is realized through the calculation of positional coordinates of tree nodes.

A separate PE at any moment is considered to be "connected" with one of the nodes of the structure which it processes. A PE can read-out the brightness value from the circulating array and record into it the results of calculations at those moments when the respective memory cells are passing it during the following data circulation. The PE functions are:

- operations with positional coordinates to determine the addresses of brightness values for subsequent processing and to access them in the dataway according to the calculated positional coordinates;
- operations of reading and writing data from (to) the dataway;
- operations with data from the dataway; for example, calculation of the new

brightness value of the node to be processed.

PRESS operates in the following way. The input preprocessor represents the input image by a truncated pyramidal-recursive structure and fill with data the array circulating in the memory. At the same time, the controller translates the user task of performing the next operation with an image into one, or several, commands describing the necessary manipulations with an elementary structure cell. The command memory accessible for all PEs.

During the first cycle of the data array in the dataway, PEs (suppose there are n of them) "connect" themselves with the first n non-empty array cells. (Let us consider that the number of PEs is less than the quantity of nodes in the tree — this is a typical situation.) After that, the cells connected with PEs are marked as occupied to prevent the processing of one node of the structure by two PEs simultaneously. PEs readout the first command from the command memory. Depending on the command, each PE can need additional data (for instance, the values of pixels which neighbor that being processed) which it must readout from the dataway having determined previously their numbers. The PEs which then have all the necessary data (this is equivalent to permission for work start-up) begin to work.

The initial starting moment is initiated by the PE corresponding to the tree root. Having performed the required operation, it records a new value into the array cell and marks it as already processed (the array circulates in the dataway during this operation - its rotation period is not of great importance, though it is desirable that it does not exceed the time of operation). Then, the PE which has processed the root finds the first non-empty cell not yet being processed, changes its state indicator and begins to operate with it.

As soon as the first cell (i.e. the zero level of the tree) is processed, it becomes possible for PEs which correspond to the nodes of the first tree level to begin processing their cells (the change of state of the first cell indicator is a signal for them that all necessary data are available). They work in parallel, and having finished their operations they go to processing the following non-processed and non-empty cells, similarly to the PE which processed the root. Further level-by-level processing of the tree is developed in a similar way: each PE begins to operate depending on data availability; when a given operation is completed, the PEs turn their attention to tree nodes not yet processed. (These features make PRESS related to a dynamic architecture computer [101]).

During the work of the PEs, the output postprocessor permanently reads out the cells having been processed and displays the current state of the circulating array as an image on the screen. Thus, a gradual refinement of the resulting image

takes place at the output.

The distribution strategy of a non-allocated PE over the data can be rather flexible. For example, a PE can go not to the following non-processed array cell, but to the next command from the command memory. If a sufficiently large number of PEs is available, then a part of the unallocated PE can be directed to the processing of another image (or images) in the pipeline regime. This can be done by recording new data set into free cells of dataway.

Note, that there is a major possibility to use PRESS also as a matrix processor or in the mode of parallel processing of individual image fragments. This depends on the method of data ordering in the dataway. For this, the input preprocessor must perform also some functions of the controller and to be able to record data structures of different types into the dataway, for instance, two- or multi-dimensional arrays ordered in different ways. It can be said that the problem orientation of the whole processor is achieved by the selection of the data structure in the dataway in accordance with the problem solved.

Thus, the main complexity in PRESS is transferred from the organization of the physical interconnections of PEs to the organization of the data stream in the dataway. In practice, the data structure in the dataway performs the role of a control device for all PEs which defines the sequence of data elements, processor element connections, and the paralleling of operations. For the rest, PEs work independently of each other and asynchronously in time.

A PE in PRESS must have some specific features to carry out the required data access and data manipulations successfully. First of all, this requires an ability to operate with positional coordinates. Such actions are required not only to calculate the addresses of data elements in the dataway, but also for operation with positional coordinates describing the characteristics of vector information fields, i.e. color or multispectral images. Hence, each PE must consist of two elementary processors: one for the calculation of access addresses and the other for operations with field characteristics. Both processors must "be able" to work with positional coordinates of a discrete space cell of arbitrary dimension p. In particular, with $p=1$, or while processing a scalar field (greyscale or binary image), they must process ordinary scalar values.

Note, that the usual arithmetical processor can be adapted for work with positional coordinates by limited modifications. For instance, performing the addition operation $i \oplus j$ is described by the relation (2.4.6). With $k=2$ (or utilizing the binary notation) it follows that the addition of 2^p-ary digits of positional coordinates is carried out according to standard rules of binary addition with only the following difference: the binary carry after addition of the following binary

digits is added not to the neighboring left bit, but to the p-th bit on the left hand (in particular, with $p=1$ we get an ordinary scalar addition). Hence, to convert a standard summator into a "positional" one for dimension p it is sufficient to introduce an extra carry chain or to use the existing chain, but to divide the addition into p cycles. In the latter case, the time of operation will increase.

Another modification is required to realize multiplication of positional coordinates. As follows from (2.4.7), this operation can be realized by logic multiplication and the calculation of the number of units in each p bits of the result. This calculation is nonstandard for ordinary processors, and requires extra resources for its realization. The remaining operations with positional coordinates can be realized using the standard set of microcommands of arithmetic-logic devices operating in binary notation. As in the particular case $p=1$, digits of positional coordinates coincide with ordinary bits, and special processors for ordinary numeric operations are not required.

6.5. On Expert Systems Simulating Human Visual Perception

The recent development of expert systems for image analysis (ESIA) [18, 77, 108] may be considered to be a natural outcome of research in the area of digital image processing and analysis. The class of the problems dealt with in this direction has increased for more than three decades: starting from digital image coding up to problems of image analysis, interpretation and image databases. Despite the fact that a lot of positive results have been obtained in these areas, such tasks as invariant object recognition, image understanding, retrieving image data by a visual pattern query, still cause serious difficulties. It seems that the utilization of achievements in the artificial intelligence area, as well as recent results in human perception research will make the development of ESIA more grounded. The present section considers some principles of human visual perception in relation to pyramidal-recursive structures and their implementation in expert systems for analyzing visual data. It is based on the works [8, 10, 18].

The first distinctive feature of an expert system for image analysis lies in two subsystems, each of which works with its own type of information. Visual data are treated in a way which is different from that used in the processing of descriptive data and knowledge. This feature stems from the incorporation of two types of expertise: one from the problem area and the other from image processing. While in traditional expert systems (ES), direct input data usually consists of features of

objects or situations which have been formalized, in ESIA we deal with raw image data, where it is necessary first to extract features, combine them into sensible groups and measure different parameters, in order to obtain initial material for semantic analysis in the frame of the particular problem area. Thus, in ESIA knowledge of both problem area specialists and image processing specialists are inherent.

Another distinctive feature of ESIA, which is also due to the two types of information presented in the system, is based on the following aspect. Along with the traditional logical inference providing generation and verification of hypotheses by means of symbolic data manipulations, it is necessary to involve in ESIA another mechanism for visual data. In contrast to logical inference which is presumably mainly sequential, these needs to be a parallel mechanism which is used for image processing, pattern matching and associative search.

The peculiarities of the two main parts of ESIA are summarized in Fig. 6.9. Looking at the functions which the two parts perform one can notice an analogy with some peculiarities of human visual perception. In dealing with two aspects of human mental activity: the left (logical verbal) cerebral hemisphere and the right (spatial pattern) cerebral hemisphere, one should take into account that they are both activated when the brain is processing, but that they realize different functions.

Logical verbal reasoning is responsible for the analysis of cause-sequence relations, providing the construction of internally consistent world models and extracting from the total amount of possible relations between an object and phenomena only several spatial ones. Pattern reasoning gives us the gestalt perception of the world picture, which is necessary to make complete contact with reality. Thus the difference between the hemisphere functions and the types of reasoning consists in what the dominating technique of information processing is. As can be seen, the situation is rather similar to the two parts of ESIA.

1. Two types of information

FORMALIZED	NONFORMALIZED
numerical data, symbolic data (rules, predicates,...)	picture, pictograms, visual patterns

2. Two types of information processing

SEQUENTIAL	PARALLEL
symbolic processing, logic inference,...	image processing, pattern matching, associative search,...

3. Two types of expertise

LOGIC-VERBAL	SPATIAL-PATTERN
problem-area expertise (knowledge formalization, implementation,...)	image processing expertise (preprocessing, segmentation, feature extraction,...)

Fig. 6.9. Symbolic and iconic mechanisms in expert system for image analysis.

In the traditional ES, where a process of reasoning by a specialist in some problem area is simulated, an inference engine which corresponds to the logical verbal type of human mental activity plays the most important role. The second type of mental process is usually absent. Such a situation satisfies, in many cases, the conventional ES requirements. This happens because input data to search ES are already formalized (numbers, predicates, rules, feature parameters), and point inherently to the corresponding analysis mechanism — logic inference. In contrast to the traditional ES, in ESIA input data are given as spatial patterns, from which formalized data are to be extracted.

Thus, in ESIA, where it is often necessary to analyze situations similar to those that occurred before (for instance, estimation of the changes that took place in some region, object identification based on associativity and a search process based on visual pattern query), lack of a perception mechanism based on spatial-pattern reasoning significantly reduces the possibilities of the system. This points to the necessity for developing techniques for the modeling of simultaneous visual pattern processing, where basic elements are represented by blocks

(fragments, symbols) of a picture rather than a pixel.

The important object arises of constructing an expert system for image analysis which would support the two types of expertise (problem area and image processing) and simulate the left and right cerebral hemisphere mechanisms of perception. It is worth noting that a problem area expertise, presumably based on formalized knowledge, is more adequate for the logical formal mechanism. Concept hierarchy - the description of hierarchical relations between abstract objects - could be chosen as a model in this case.

As for image processing expertise, this is based to a large extent on the right hemisphere mechanism which is oriented to work with spatial patterns. That is, we deal with the visual hierarchy of instances. If so, then one of the main tasks of the ESIA is to find a correspondence between elements of the two hierarchies [97, 108]. The implementation of this process is, in essence the situation of human perception with the left and right hemisphere mechanisms working together.

Biomedical and physiological investigations indicate that before the image passes from the retina to the brain it undergoes an important change. Low spatial frequencies (or, equivalently, low resolution image) go to the brain first, then medium frequencies, and so on. Thus, several images of varying resolution (direct analog of an image pyramid) follow into the brain, this process being repeated every time we shift our glance, i.e. when a new retinal image appears which differs from the preceding one.

Two basic hypotheses exist which account for this phenomenon [40, 52, 100]. According to one of these, the receptive field of the retina enlarges quickly after one shifts one's glance, and this is followed by a step-by-step decrease, which is equivalent to the progressive increase in the clarity of the image transmitting into the brain. By another hypothesis, there exists several physically different photosensitive rasters in the retina, some of which are rough, and the others more fine. Initially, the image is perceived by the coarse raster (a set of perceptive elements), then by intermediate ones and so on, to the finest raster. As a result, a stage-by-stage refinement of the observed pattern in the visual cortex occurs.

If the duration of one's glance is too short, the brain can perceive only the roughest copies of the scene observed. The volume of these data is significantly less than that received for the whole image. To process these data takes much less time than that which is necessary to analyze the image in detail. It is this data that carries the most important information on a general situation. Fine points, and details of an observed picture are important later after the first rough copy has already been formed. Then each detail at which one desires to look, and to which our glance is shifted, is also analyzed in the same stage-by-stage manner.

Thus the information is analyzed according to the coarse-fine principle, where the approximate result at the initial stages of the process is gradually refined subsequently. In this case, the channel of data transference between the retina and the memory is not only used efficiently, but smoothly over the defects of the retinal image, thus increasing the system's noise immunity.

Other research shows that rather deterministic visual perception violations take place when certain areas of the brain are affected. For instance, when visual areas of the left cerebral hemisphere are damaged, the ability to estimate visual pattern features, to extract valuable features and to find pattern classification principles are affected to a great extent. Damage in the corresponding areas of the right cerebral hemisphere lead to violations of the perception of specific features (especially parallel feature perception), memorizing of individual features and shapes with unknown names and spatial relation estimation processes [90, 130].

This provides a clue which is expedient to the implementation of the functions of both hemispheres. The right hemisphere mechanism should perform the location of specific individual object features, compare them with already known ones (the prototypes), analyze spatial relations of the features and objects, and remove noise in the image. Left hemisphere functions include feature significance classification, construction of conceptual hierarchy, formulation of rules to recognize an object and remembering of visual symbolic descriptions (Fig. 6.10).

Patients suffering damages to the visual areas of the right and left hemispheres have quite different strategies of object recognition when recognition time is rather short. For example, patients with damage in the right occipital lobe have a recognition strategy consisting in logical inference deductions which are based on the recognition (that often happens to be incorrect due to short observation period and the right hemisphere damage) of some image features. The conclusions are generic and lack concreteness. They are gradually refined using the results of new deductions (Fig. 6.11a).

Patients with damage in the left occipital lobe perceive mainly by way of accumulating different image features without taking into account their significance. Recognition takes place only after most of the features have been extracted, correctly recognized and summarized (Fig. 6.11b).

It should be stressed that for healthy persons with two cerebral hemispheres working in parallel, the recognition process runs almost 50 times faster (average recognition time is several miliseconds) than for patients with impairment of either hemisphere.

It is interesting to note the different principles for making decisions, when recognizing an object, by persons with left and right orientation [90, 143]. For the

former, the hierarchy of features (their relative importance) plays the main role in

LEFT HEMISPHERE	RIGHT HEMISPHERE
	Destruction of image into elements
	Noise elimination, pattern extraction
Recognition of general features	Recognition of specific features
Pattern similarity analysis	Pattern specify analysis
Estimation of feature hierarchy	
Analysis and fixing of temporal and causal relations	Analysis and fixing of spatial relations
Learning - obtaining of generalized features and notions	Remembering of integral patterns
Memory for generalized features, short-term visual memory	Long-term memory for patterns and their elements with assotiative access

Fig. 6.10. Basic functions of the human visual cortex.

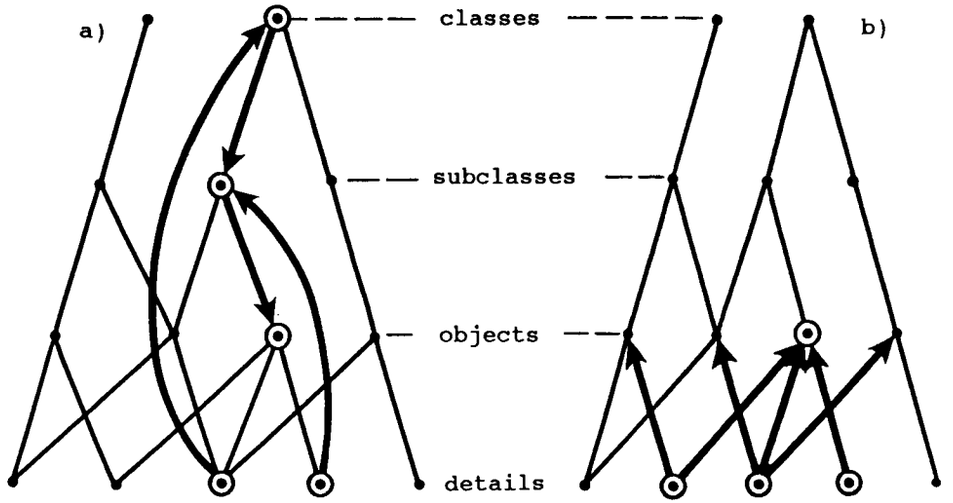


Fig. 6.11. Recognition strategies for left (a) and right (b) oriented persons.

the recognition process. A feature (or detail) - say, "a wheel" - for such patients, leads to the conclusion "this is a kind of transport". This means that the relation between the feature "a wheel" and the class "transport" has maximum weight in spite of the possibility of relating the feature with objects from other classes (for instance, the object could be a mill). Recognition of the next feature - "an arc" - (within the context of "transport") allows the choice of a subclass with the highest weight and so on, down to the recognition of a specific object.

In contrast to this, among right oriented persons "a wheel" feature causes associations with many concrete patterns where similar wheels have been observed (a car, a watch, a motorcycle, a ship, etc.). The next feature (randomly taken and regardless of importance and relation with the previous one) diminishes the set of possible solutions and so on up to the situation when the final set of features defines the unique object.

Thus, in contrast to the way the left hemisphere makes decisions based on the *highest weight criterion*, we discover in the right hemisphere a radically different criterion of decision making that may be called *an absence-of-alternatives principle*: once the accumulated amount of information narrows the choice to a single alternative, and the decision is accepted.

Based on the data given above and presented in the literature [10, 84, 105, 150, 151] a functional schema of a system simulating the human perception process could be suggested (Fig. 6.12). The left formal-logical part of the system is to be filled with domain knowledge constituting conceptual hierarchy. In the right spatial

pattern part, accumulation of visual data and feature identification takes place, providing a hierarchy of patterns. Goal definition and context should play an important role in the system. It is difficult to guide both image analysis and synthesis when there is no distinct task to be done. (For example, in dreaming, when only the right hemisphere is working, contradictory and logically inconsistent pictures are retrieved from the visual memory in the form of dreams, though they are accepted as rather natural when sleeping.) To exclude such self-generation in the right part of the system, the left part should specify a set of goals or tasks to work with visual data, which will define the logic of "system attention" switching.

Recognition strategies in ESIA should be based, on the one hand, on the accumulation of specific features and, on the other hand, on forecasting the type of object to be recognized using its dominant features. To prevent a "combinatorial explosion", both strategies should be utilized first of all in parallel and, second, hierarchically, using hierarchical structuring of input image data. For this purpose, the right and left subsystems may continually exchange decisions made thus permanently refining the scope of the alternatives under consideration. In reality, the implementation of such a strategy due to its complexity may be effective only in developed systems working with a large volume of knowledge and a significant amount of patterns.

The inherent characteristic of the hierarchy of perception processes predetermines the natural choice of regular pyramidal structures as the base for ESIA development. A significant reduction in computer resources can be achieved when performing operations not on input images, but on corresponding pyramidal-recursive structures in a top-down strategy. It is worth noting that these structures operate with an image block rather than a pixel, thus approaching the right hemisphere type of information processing.

An important aspect of pyramidal-recursive structures is the possibility of specifying a correspondence between a visual hierarchy, which is proved by the structure construction mechanism itself, and a conceptual hierarchy, which is the bases for selecting strategies in ESIA. For instance, once a pyramid is constructed on the basis of an aerial photo image of a city, we could locate "large" objects such as roads and buildings on the upper pyramid levels where images with low resolution are situated. These correspond to classes of objects (Fig. 6.11b). Therefore the location of a crosswalk could be achieved on the lower levels (where fine details of objects appear) only in those places previously defined as road objects. This prevents the need for a large number of segmentation reducing solutions to be checked by a system for the whole picture. Noise influence is reduced as well.

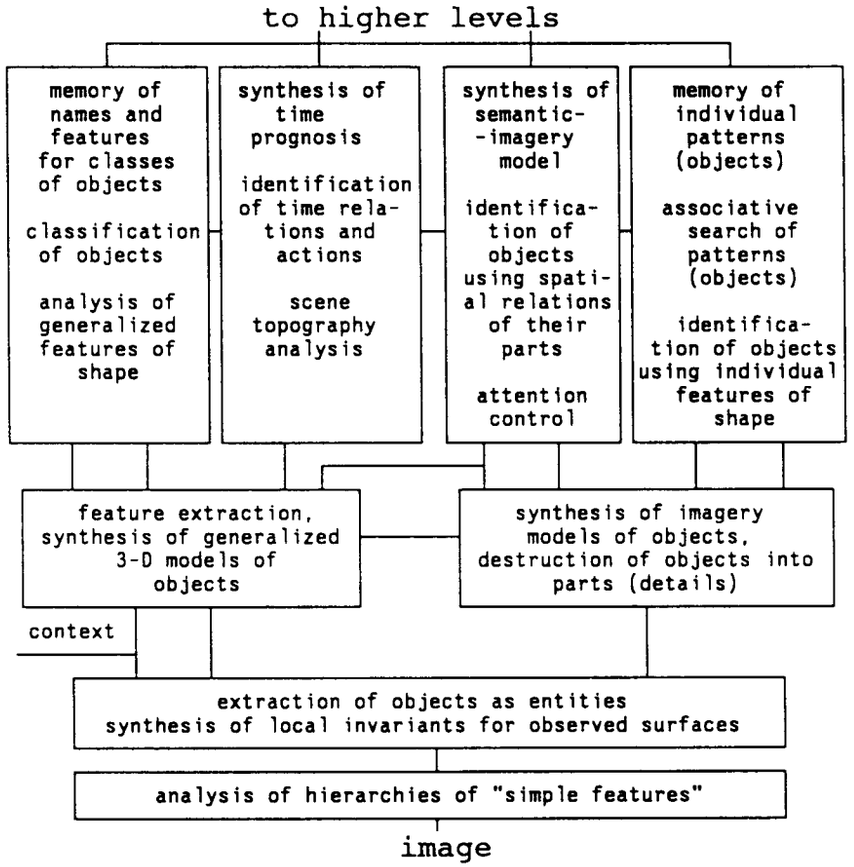


Fig. 6.12. Structure of the system modeling some functions of human visual

On the other hand, the search for a particular object is performed in a particular context, thus indirectly utilizing specialist knowledge of a problem area. To summarize, image processing in ESIA based on pyramidal-recursive structures rather than a picture as an array of pixels, allows considering the analysis process as a process of establishing a correspondence between two hierarchies.

The hypothetical system shown in Fig. 6.13 includes a pyramidal-recursive processor for image operations and a knowledge-based processor working with a conceptual hierarchy. An image is loaded in parallel into the pyramid of processing elements (PE) and the description of the current goal activates the corresponding

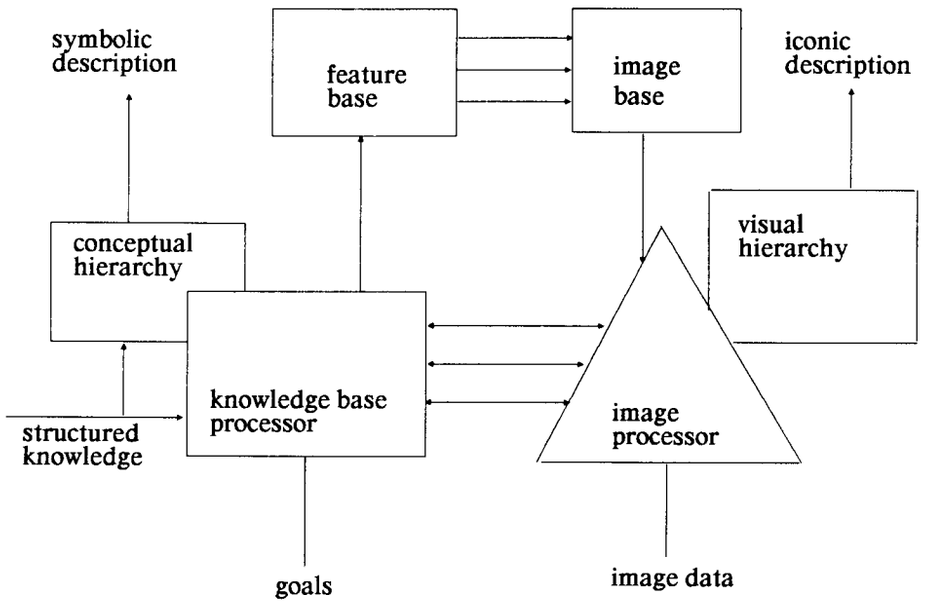


Fig. 6.13. Function schema of the ESIA prototype.

erarchy in the knowledge-based processor.

In the first stage, image data flow moves in the upward direction through the pyramid. Processing elements extract the individual features of the image on different hierarchy levels. At the same time, the feature identification process is initiated (starting from upper levels) taking into account the activated formalized knowledge represented in a conceptual hierarchy. On the basis of a set of identified features, as well as individual ones, hypotheses about the presence of an object in the image are generated. yAssociative retrieval of visual patterns (templates) from the image database constitutes this process. After that a downward pyramid process starts. At this stage, a fast technique for template location is used to verify whether the predicted objects (features) are really present in the picture. This concludes the primitive identification process. The described procedure, considered as a model of the perception process in the ESIA prototype, should be repeated at different generalization levels.

BIBLIOGRAPHY

[1] Adelson, E.H., Burt, P.J., Image Data Compression with the Laplassian Pyramid, *Proc. IEEE Comp. Soc. Conf. Pattern Recogn. Image Proc., Dallas, 1981,* pp. 218-223.

[2] Ahuja, N., Approaches to Recursive Image Decomposition, *Proc. IEEE Comp. Soc. Conf. Pattern Recogn. Image Proc., Dallas, 1981*, pp. 75-80.

[3] Ahuja, N. and Swamy S., Interleaved Pyramid Architectures for Bottom Up Image Analysis, *Proc. 6-th Int. Conf. Pattern Recogn., Munich, 1982,* pp. 398-390.

[4]* Alexandrov, V.V., Selfsimilar Recursive Structures as Techniqué of Knowledge Representation, *Information Problems of Research Automation*, Nauka, Moscow, 1983, pp. 65-75.

[5]* Alexandrov, V.V. and Arsentjeva A.V., *Information and Developing Structures*, Leningrad Research Computer Center, 1984.

[6]* Alexandrov, V.V. and Gorsky, N.D., A Recursive Approach to Data Processing, *Applied Informatics*, **2(7)**, 1984, pp. 59-68.

[7]* Alexandrov, V.V. and Gorsky, N.D., A Recursive Approach to Associative Storage and Search of Information in Data Bases, *Proc. Finnish-Soviet Symp. on Data Base Management Systems, Turku, Finland, 1980*, pp. 270-284.

[8] Alexandrov, V.V. and Gorsky, N.D., Expert Systems Simulating Human Visual Perception, *Int. Journ. of Pattern Recogn. and Artif. Intel.*, **3**, 1989, pp. 19-28.

[9]* Alexandrov, V.V. and Gorsky, N.D., *Algorithms and Programs of the Structural Data Processing Method*, Science, Leningrad, 1983.

[10] Alexandrov, V.V. and Gorsky, N.D., *From Humans to Computers: Cognition*

* in Russian

through Visual Perception, World Scientific, Singapore, 1991.

[11]* Alexandrov, V.V. and Gorsky, N.D., *Image Representation and Processing - A Recursive Approach*, Science, Leningrad, 1985.

[12] Alexandrov, V.V., Alexeev, A.I. and Gorsky, N.D., A Recursive Algorithm for Pattern Recognition, *Proc. 6-th Int. Conf. on Pattern Recogn., Munich, Germany, August 1982*, pp. 431-433.

[13]* Alexandrov, V.V., Arsentjeva A.V. and Gorsky, N.D., Recursive Approach to Associative Data Storage and Search, *Computer Systems and Methods in Research Automation*, Science, Moscow, 1982, pp. 68-74.

[14]* Alexandrov, V.V., Butuzov, V.V., Gorsky, N.D. Kosatchev, L.V., Mysko, S.N. and Sadovnikova A.I., Pyramidal-Recursive Processor for Image Processing and Image Database Management, *Proc. 2-nd Nat. Conf. on Image Processing and Remote Sensing, Novosibirsk, USSR, November 1987*, pp. 23-24.

[15] Alexandrov, V.V., Gorsky, N.D. and Mysko, S.N., Recursive Pyramids and Their Use for Image Coding, *Pattern Recogn. Lett.*, 2, 1984, pp. 301-310.

[16] Alexandrov, V.V., Gorsky, N.D. and Mysko, S.N., A Fast Technique for Recursive Scene Matching Using Pyramids, *Pattern Recogn. Lett.*, 3, 1985, pp. 413-419.

[17]* Alexandrov, V.V., Gorsky, N.D. and Polyakov, A.O., Recursive Algorithms of Data Representation and Processing, *Algorithms and Systems of Computer-Aided Research and Design*, Science, Moscow, 1980, pp. 40-78.

[18]* Alexandrov, V.V., Gorsky, N.D., Iskanderov, P.M. and Mysko S.N., *Expert Systems for Image Analysis*, Leningrad Institute for Informatics and Automation, Preprint No. 25, 1987.

[19]* Alexandrov, V.V., Latchinov, V.M. and Polyakov, A.O., On Recursive Algorithmization of a Curve that Fills a Multidimensional Interval, *Engineering Cybernetics*, No. 1, 1978, pp. 192-198.

[20]* Andreev, V.P., et al., *Experiments in Computer Vision*, Moscow, Science, 1987.

[21] Anisimov, V.A. and Gorsky, N.D., Fast Hierarchical Matching of an Arbitrarily Oriented Template, *Pattern Recogn. Left*, 15, 1993, pp. 95-101.

[22] Antonisse, H.J., Image Segmentation in Pyramids, *Comp. Graph. Image Proc.*, 1982, v. 19, pp. 367-383.

[23] Ballard, D.H. and Brown, H.C., *Computer Vision*, Prentice Hall, 1982.

[24] Baird, H.S., Graft, H.P., Jackel, L.D., and Hubbard W.E., A VLSI Architecture for Binary Image Classification, *From Pixels to Features*, North-Holland, Amsterdam, 1989, pp. 275-286.

[25] Baird, H.S., Kahan, S., and Pavlidis, T., Components of an OmniFont Page Reader, *Proc. 8-th Int. Conf. on Pattern Recognition*, Paris, vol. 1, 1986, pp. 344-348.

[26] Barron, D.W., *Recursive Techniques in Programming*, McDonald Londres, New York, 1969.

[27] Baugher, E.S. and Rosenfeld, A., Boundary Localization in an Image Pyramid, *Pattern Recogn.*, 1986, v. 19, No. 5, pp. 373-395.

[28] Benelli, G. et al., Image Data Compression by Using Adaptive Huffman Encoding and the Laplacian Pyramid, *Proc. 2-d Int. Conf. on Image Process. Appl.*, London, IEE, 1986, pp. 21-24.

[29] Bially, T. Space Filling Curves: Their Generation and Their Application to Bandwidth Reduction, *IEEE Trans.*, vol. IT-15, No. 6, 1969.

[30] Blanford, R.P. and Tanimoto, S.L., A Pyramid Machine Simulator for the Symbolics 3600, *Proc. IEEE Comp. Soc. Conf. on Comp. Vision and Pattern Recogn.*, IEEE Comp. Soc., 1986, pp. 427-429.

[31] Browing, J.B. and Tanimoto, S.L., Segmentation of Pictures into Regions with a Tile-by-Tile Method, *Pattern Recogn.*, vol. 15, No. 1, 1982, pp. 1-10.

[32] Bulter, M.C., Top-Down Recursive Partition Algorithm for Nonparametric Classification, *Proc. IEEE Comp. Soc. Conf. on Comp. Vision and Pattern Recogn.*, IEEE Comp. Soc., 1985, pp. 412-415.

[33] Burt, P.J., Algorithms for Estimating Local Image Properties, *Proc. IEEE Comp. Soc. Conf. on Pattern Recogn. and Image Proc.*, Las-Vegas, 1982, pp. 669-671.

[34] Burt, P.J., Fast Filter Transforms for Image Processing, *Comp. Graph. Image Proc.*, 1981, v.16, No. 1 pp. 20-51.

[35] Burt, P.J., The Pyramid as a Structure for Efficient Computation, *Multiresolution Image Processing and Analysis*, Springer, 1984, pp. 6-20.

[36] Butz, A.R., Space Filling Curves and Mathematical Programming, *Inform. Contr.*, 1968, v.12, pp. 314-330.

[37] Cantoni, V., Levialdi, S. (eds.), *Pyramidal Systems for Computer Vision*, Springer, Berlin, 1986.

[38] Chang, S.K. and Kunii, T.L. Pictorial Data-base Systems, *Computer*, 1981, v. 14, No. 11, pp. 13-21.

[39] Crettez, J.P., The Relationship Between Two Types of Hierarchies in Hexagonally Digitized Images, *Proc. 8-th Int. Conf. on Pattern Recogn.*, Paris, 1986, pp. 1286-1289.

[40] Crettez, J.P. and Simon, J.-C., A Model for Cell Receptive Fields the Visual Striate Cortex, *Comp. Graph. Image Proc.*, 1982, v. 20, No. 2, pp. 299-318.

[41] Duda, R.O. and Hart, P.E., *Pattern Classification and Scene Analysis*, John Wiley, 1973.

[42] Dyer, C.R., A VLSI Pyramid Machine for Hierarchical Parallel Image Processing, *Proc. IEEE Comp. Soc. Conf.Pattern Recogn. Image Proc.*, Dallas, 1981, pp. 381-386.

[43] Dyer, C.R., Pyramid Algorithms and Machines, *Multicomputers and Image Processing Algorithms and Programs*, Academic Press, 1982, pp. 409-429.

[44] Dyer, C.R., The Space Efficiency of Quadtrees, *Comp. Graph. Image Proc.*, 1982, v. 19, pp. 335-348.

[45] Dyer, C.R., Rosenfeld, A. and Samet, H., Region Representation: Boundary Codes From Quadtrees, *Comm. ACM*, 1980, v.23, No. 3, pp. 171-179

[46] Finkel, R.A. and Bentley, J.I., Quad Trees: A Data Structure for Retrieval on Composite Keys, *Acta Inform.*, 1974, v. 4, pp. 1-9.

[47] Fountain, T., *Processor Arrays: Architecture and Application*, Academic Press, 1987.

[48] Fu, K.S., *Syntactic Pattern Recognition and Applications*, Prentice Hall, 1982.

[49] Gargantini, I., Linear Octtrees for Fast Processing of Three-Dimensional Objects, *Comp. Graph. Image Proc.*, 1982, v. 20, pp. 365-374.

[50] Gibson, L. and Lucas, D., Spatial Data Processing Using Generalized Balanced Ternary, *Proc. IEEE Comp. Soc. Conf. Pattern Recogn. Image Proc.*, Las Vegas, 1982, pp. 566-571.

[51] Gillespie, R. and Davis, W.A., Tree Data Structure for Graphics and Image Processing, *Proc. 7-th Canadian Man-Comp. Comm. Conf.*, Waterloo, 1981, pp.155-162.

[52]* Glezer, V.D., *Vision and Thinking*, Nauka, Leningrad, 1985.

[53]* Gorsky, N.D., A Recursive Model for Image Representation, *Automated Processing of Complex Graphics*, Gorky State University, 1984, pp. 159-178.

[54] Gorsky, N.D., On the Complexity of the Quadtree and 2D-tree Representations for Binary Pictures, *From Pixels to Features, Proc. of a Workshop held at Bonas, France,* North-Holland, Amsterdam, 1988, pp. 393-402.

[55]* Gorsky, N.D., On the Efficiency of Data Processing Algorithms, *Proc. Nat. Conf. INFORMATICS'87, Erevan, October 1987*, pp. 238-239.

[56]* Gorsky, N.D. and Mysko S.N., Fast Template Matching Using Pyramidal Structures, *Methods and Systems of Automation in Science and Technology*, Science, Moscow, 1986, pp. 165-175.

[57]* Gorsky, N.D. and Mysko S.N., Image Representation by Pyramid Structure and its Relation with Hadamar Transforms, *Automation Systems in Research and Technology*, Science, Moscow, 1984, pp. 65-74.

[58]* Gorsky, N.D., Iskanderov, P.M. and Mysko S.N., Pyramidal-Recursive Structures for Image Processing and Analysis, *Information Support of Integrated Manufacturing Systems*, Leningrad Institute for Informatics and Automation, Leningrad, 1987, pp. 52-66.

[59]* Gorsky, N.D., Mysko S.N. and Sukharichev V.P., *Comparative Research of Two-dimensional Scans Characteristics*, Leningrad Research Computer Center, Preprint No. 44, 1982.

[60]* Gorsky, N.D., Mysko S.N. and Sukharichev V.P., Estimations of Auto-correlation Functions and Power Spectra of Processes while Image Scanning, *Automation Research and Industrial Systems*, Science, Moscow, 1985, 156-160 pp.

[61] Granlund, G.H. and Arvidsson, J.B., Computer Architectures for Image Processing, *Proc. 4-th Scandinavian Conf. on Image Analysis*, Trondheim, 1985, pp. 20-30.

[62] Gregory, R.L., Gombrich, E.H. (eds.), *Illusion in Nature and Art*, Duckworth, 1980.

[63] Grosky W.I. and Jain R. A Pyramid-based Approach to Segmentation Applied to Region Matching, *IEEE Trans Pattern Anal. Machine Intel.*, 1986, v. PAMI-8, No. 5, pp. 639-650.

[64] Hanson, A.R. and Riseman, E.M., Segmentation of Natural Scenes, *Computer Vision Systems*, Academic Press, 1978.

[65] Hilbert, D., Uber die stetige Abbildung einer linie auf ein Flachenstuck, *Math. Annalen*, 38, 1891, pp. 459-460.

[66] Hilbert, D. and Bernice, P., *Foundations of Mathematics*, Science, Moscow, 1981.

[67] Hofstadter, D.R., *Gödel, Esher, Bach: An Ethernally Golden Braid*, Harvester Press, New York, 1979.

[68] Horn, B.K.P., *Robot Vision*, MIT Press and McGraw-Hill, 1986.

[69] Horowitz, S.L. and Pavlidis T., Picture Segmentation by a Tree Traversal Algorithm, *ACM Journ.*, 1976, v. 23, No. 4, pp. 368-388.

[70] Hunter, G.M., and Steiglitz, K., Operation on Images Using Quad Trees, *IEEE Trans.*, 1979, v. PAMI-1, No. 2, pp. 145-153.

[71] Icnikawa, T., A Pyramidal Representation of Images and Its Feature Extraction Facility, *IEEE Trans.*, 1981, v. PAMI-3, No. 3, pp. 257-264.

[72] Jackins, C. and Tanimoto, S.L., Oct-trees and Their Use in Representation Three-Dimensional Objects, *Comp. Graph. Image Proc.*, 1980, v. 14, No. 2,

pp. 249-270.

[73] Jackins, C. and Tanimoto, S.L., Quad-Trees, Oct-Trees, and k^d-Trees: A Generalized Approach to Recursive Decomposition of Euclidean Space, *IEEE Trans.*, 1983, v. PAMI-5, No. 5, pp. 533-539.

[74] Johnsen O., and Netravali A., Progressive Transmission of Two-Tone Images, *IEEE Trans.*, 1980, v. c-29, No. 12, pp. 1934-1941.

[75] Jones, G., Recognition to the Resque, *Personal Computer Magazine*, September 1989.

[76] Kawaguchi, E. and Endo, T., On a Method of Binary Picture Representation and Its Application to Data Compression, *IEEE Trans.*,1979, v. PAMI-2, No. 1, pp. 27-35.

[77] Kim, J.H., Payton, D.W. and Olin, K.E., An Expert System for Object Recognition in Natural Scenes, *Proc. 1-st Conf. on Artificial Intelligence Applications, Sheraton, December 1984,* pp.170-174.

[78] Klinger, A. and Dyer, C.R., Experiments on Picture Representation Using Regular Decomposition, *Comput. Graphics Image Process.*, 1976, v. 5, No. 2, pp. 305-321.

[79] Koenderink, J.J. and Van Doorn, A.J., New Type of Raster Scan Preserves the Topology of Image, *Proc. IEEE,* 1979, v. 67, No. 10, pp. 1465-1466.

[80] Kunii, T.L., Fujihro, I. and Mao X, G-quadtree: A Hierarchical Representation of Grey-scale Digital Images, *The Visual Computer*, 1986, v. 2, No. 4, pp. 219-226.

[81] Labonte, A.E., Micro-Adaptive Picture Sequesing (MARS) in a Display Environment, *Proc. Soc. Photo-Optical Instr. Eng.*, 1980, v. 249. pp. 61-69.

[82] Li, M., Grosky, W.I. and Jain R. Normalized Quadtrees with Respect to Translations, *Comp. Graph. Image Proc.*, 1982, v. 20, No. 1, pp. 72-81.

[83] Mandelbrot, B., *Fractals: Form, Chance, and Dimension*, Freeman, San Francisco, 1975.

[84] Marr, D., *Vision*, Freeman, San Francisco, 1982.

[85] Max, J., Quantizing for Minimum Distortion, *IRE Trans.*, 1960, v.IT-6, No. 1, pp. 7-12.

[86] McBurney, D.H. and Collings, V.B., *Introduction to Sensation/Perception*, Prentice-Hall, 1984.

[87] Meagher, D., Geometric Modeling Using Octree Encoding, *Comp. Graph. Image Proc.*, 1982, v. 19, pp. 129-147.

[88] Merigot, A., Zavidovique, B. and Devos, F., SPHINX - A Pyramidal Approach to Parallel Image Processing, *Proc. 1985 IEEE Computer Soc. Workshop on Computer Architecture for Pattern Analysis and Database Management*, 1985, pp. 107-111.

[89] Meyer, B. and Baudoin, K., *Methodes de Programmation*, Paris, 1978.

[90]* Meyerson, Ya.A., *The High Level Visual Functions*, Nauka, Leningrad, 1986.

[91] Miller, R. and Stout, Q., The Pyramid Computer for Image Processing, *Proc. 7-th Int. Conf. on Pattern Recognition*, 1984, pp. 240-242.

[92] *Multiresolution Image Processing and Analysis* (A. Rosenfeld, ed.), Springer, 1984.

[93] Murray, J.J., A Fast Binary Template Matching Algorithm for Document Image Data Compression, *Pattern Recognition*, (J. Kittler, ed.), Proc. 4-th Int. Conf., U.K., March 1988, pp. 230-239.

[94]* Mysko, S.N. and Polyakov, A.O. On a Technique of Image Representation while Their Receiving, *Computer Systems and Methods Research Automation*, Science, Moscow, 1982, pp. 91-98.

[95] Newman, W. M. and Sproull, R.F., *Principles of Interactive Computer Graphics*, McGraw-Hill, 1979.

[96] Nguyen, P.T. and Quinqueton, J., Space-Filling Curves and Texture Analysis, *Proc. 6-th Int Conf. Pattern., Munich*, 1982, pp. 282-285.

[97] Nishihara, H.K., Recognition of Shape Visible Surfaces, *Physical and Biological Processing of Images*, Springer, Berlin, 1983, pp. 335-348.

[98] Noar, J. and Peleg, S., Hierarchical Image Representation for Compression, Filtering and Normalization, *Pattern Recogn. Letters*, 1983, v. 2, No. 1. pp. 43-46.

[99] Pietikainen, M. and Rosenfeld, A., Image Segmentation By Texture Using Pyramid Node Linking, *IEEE Trans.*, 1981, v. SMC, No. 12, pp. 822-825.

[100]* Podvigin, N.F., Makarov, F.N. and Shelepin, Yu.E., *Elements of Structural-Functional Organization of Vision and Eye-Movement System*, Nauka, Leningrad, 1986.

[101]* Ponomarev, V.M., Plyasnin, V.U. and Torgashev, V.A., *Distributed Computations and Dynamic Architecture Computers*, Leningrad Research Computer Center, Preprint No. 54, 1982.

[102] Pratt, W.K., *Digital Image Processing*, John Wiley, 1978.

[103] Quinqueton, J. and Berthod, M., A Locally Adaptive Peano Scanning Algorithm, *IEEE Trans.*, 1981, v. PAMI-3, No. 4, pp. 403-412.

[104] Renade, S., Rosenfeld, A. and Samet H. Shape Approximation Using Quadtrees, *Pattern Recogn.*, 1982, v. 15, No. 1, pp. 31-40.

[105]* Rock, I., *Introduction to Visual Perception*, Pedagogika, Moscow, 1980.

[106] Rosenfeld, A. and Kak, A.C. *Digital Picture Processing*, Academic Press,

1983.

[107] Rosenfeld, A. and Vanderburg, G.J., Coarse-Fine Template Matching, *IEEE Trans.*, 1977, v. SMC-7, No. 1, pp. 104-107.

[108] Rosenthal, D.A. and Bajcsy, R., Visual and Conceptual Hierarchy: A Paradigm for Studies of Automated Generation of Recognition Strategies, *IEEE Trans. PAMI*, 6, 1984, pp. 319-325.

[109] Samet, H., An Algorithm for Converting Rasters to Quadtrees, *IEEE Trans.*, 1981, v. PAMI-3, No. 1, pp. 93-95.

[110] Samet, H., et al., A Geographic Information System Using Quadtrees, *Pattern Recogn.*, 1984, v. 17, No. 6, pp. 647-656.

[111] Samet, H., Neighbor Finding Techniques in Images Represented by Quadtrees, *Comp. Graph. Image Proc.*, 1981, v. 18, No. 1, pp. 35-57.

[112] Samet, H., Region Representation: Quadtrees from Binary Arrays, *Comp. Graph. Image Proc.*, 1980, v. 13, No. 1, pp. 88-93.

[113] Samet, H., Quadtrees and Related Hierarchical Data Structures, *ACM Comp. Surv.*, 1984, v. 16, No. 2, pp. 187-260.

[114] Samet, H. and Webber, R., Storing a Collection of Polygons Using Quadtrees, *ACM Trans. Graphics*, 1985, v. 4, No. 3, pp. 182-222.

[115] Santalo, T., *Integral Geometry and Geometric Probabilities*, Addison-Wesley, New York, 1976.

[116] Schaefer, D.H., et al., The MPP Pyramid Computer, *Proc. Int. Cont. on Cybernetics and Society*, 1985, pp. 617-675.

[117]* Sergeev, V.V., Image Processing with Peano Scan, *Autometrics*, No. 2, 1984, pp. 30-36.

[118] Serra, J., *Image Analysis and Mathematical Morphology*, Academic Press, 1982.

[119] Shapiro, L.G., Data Structures for Picture Processing: A Survey, *Comp. Graph. Image Proc.*, 1979, v. 11, No. 2, pp. 162-184.

[120] Shneier, M., Calculations of Geometric Properties Using Quadtrees, *Comp. Graph. Image Proc.*, 1981, v. 16, No. 2, pp. 296-308.

[121] Simon, J.-C., *Patterns and Operators*, McGraw Hill Co., N.Y., 1986.

[122] Simon, J.-C., *La reconnaissance des formes par algoritmes*, Masson, Paris, 1984.

[123] Sloan, K.R. and Tanimoto, S.L., Progressive Refinement of Raster Image, *IEEE Trans.*, 1979, v. c-28, No. 11, pp. 871-875.

[124] Spann, M. and Witson, R., A Quad-Tree Approach to Image Segmentation, *Pattern Recognition*, 1985, v. 18, No. 3-4, pp. 257-269.

[125] Srihari, S.N., Hierarchical Data Structures and Progressive Refinement of

3-D Image, *Proc. Recogn. Image Proc.*, Las Vegas, 1982, pp. 485-490.

[126] Stanton, T., OmniPage: OCR made easy, affordable, *Personal Computing*, March, 1989.

[127] Stevens, R.J. and Lehar, A.F., High - Speed Manipulation of the Color Chromaticity of Digital Image, *Comput. Graph. and Appl.*, 1984, v. 4, No. 2, pp. 34-39.

[128] Stevens, R.J., Lehar, A.F. and Preston, F.H., Manipulation and Presentation of Multidimensional Image Data Using the Peano Scan, *IEEE Trans.*, 1983, v. PAMI-5, No. 5, pp. 520-526.

[129]* Strongin, R.G., *Numerical Methods in Multiextremal Tasks*, Science, Moscow, 1978.

[130]* Suvorova, V.V., Matova, M.A. and Turovskaya, Z.G., *The Asymmetry of Visual Perception*, Pedagogika, Moscow, 1988.

[131] Tamura, H. and Yokoya, N., Image Database Systems: A Survey, *Pattern Recogn.*, 1984, v. 17, No. 1, pp. 29-43.

[132] Taniguchi, R., et al., Picture Understanding and Retrieving System of Weather Chart, *Proc. 6-th Int. Conf.Pattern Recogn.*, Munich, 1982, pp. 802-805.

[133] Tanimoto, S.L., Programming Techniques for Hierarchical Parallel Image Processors, *Multicomputers and Image Processing Algorithms and Programs*, Academic Press, 1982, pp. 421-429.

[134] Tanimoto, S.L., Regular Hierarchical Image and Processing Structures in Machine Vision, *Computer Vision Systems*, Academic Press, 1978.

[135] Tanimoto, S.L., Template Matching in Pyramids, *Comp. Graph. Image Proc.*, 1981, v. 16, No. 2, pp. 356-369.

[136] Tanimoto, S.L. and Pavlidis, T., A Hierarchical Data Structure for Picture Processing, *Comp. Graph. Image Proc.*, 1975, v.4, No. 1. pp. 104-119.

[137]* Tarian, R., On the complexity of Combinatorial Algorithms, *Cybernetic Collection*, vol. 17, 1980, pp. 60-109.

[138] Uhr, L., Recognition Cones, and Some Test Results, *Computer Vision Systems*, Academic Press, 1978.

[139] Uhr, L., Layered Recognition Cone Networks that Preprocess, Classify, and Describe, *IEEE Trans. Comput.*, 1972, v. C-21, No. 7, pp. 758-768.

[140] Uhr, L., Schmitt, L. and Hanrahan, P., Cone/Pyramid Perception Program for Arrays and Networks, *Multicomputers and Image Processing Algorithms and Programs*, Academic Press, 1982, pp. 179-191.

[141]* Vasin, Yu.G. and Pleskov, A.V., Edge Detection in a Greyscale Image with Preliminary Data Compression, *Automated Processing of Complex Graphics*,

Gorky State University, 1985, pp. 17-31.

[142] * Vasin, Yu.G., *Well Appropriate Basis and Problems of Experimental Data Processing*, Gorky State University, 1979.

[143] * Velichkovsky, B.M. and Zinchenko, V.P., Methodological Problems of Modern Cognitive Psychology, *Voprosy Philosophii*, No.7, 1979, pp. 67-79.

[144] Veronis, A.M., General Review of Pyramid Computer for Image Processing, *Proc. IEEE Conf. on Workstations Technology and Systems*, 1986, pp. 14-17.

[145] Weng, J. and Ahuja, W., Octree Representation of Objects in Arbitrary Motion, *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 1985, pp. 524-529.

[146] Wong, R.Y., and Hall, E.L., Sequential Hierarchical Scene Matching, *IEEE Trans. Comput.*, 1978, v. C-27, No. 4, pp. 359-366.

[147] Wu, A.Y., and Rosenfeld, A., Top-down Cellular Pyramids, *Pattern Recogn. Letters*, 1983, v. 2, No. 1, pp. 47-52.

[148] * Yaroslavsky, L.P., *Introduction to Digital Image Processing,* Soviet Radio, Moscow, 1979.

[149] Yau, M. and Srihari, S., A Hierarchical Data Structure for Multidimensional Digital Images, *Comm. ACM*, 1983, v. 26, No. 7, pp. 504-515.

[150] * Zaporogetz, A.V., Perception Development and Activity, *Proc. 30-th Symp. XVIII Int. Psychological Congr.*, Moscow, 1966, pp. 35-44.

[151] * Zenkin, G.M. and Petrov, A.P., Functional Organization of the Vision Process and the Gestalt Principle, *Intellectual Processes and Their Simulation*, Nauka, Moscow, 1987, pp. 265-293.

[152] * Zuckerman, I.I., (Ed.), *Digital Coding of TV-images*, Moscow, Radio and Communications, 1981.

INDEX

Other *Mathematics and Its Applications* titles of interest:

P.H. Sellers: *Combinatorial Complexes. A Mathematical Theory of Algorithms.* 1979, 200 pp. ISBN 90-277-1000-7

P.M. Cohn: *Universal Algebra.* 1981, 432 pp.
ISBN 90-277-1213- 1 (hb), ISBN 90-277-1254-9 (pb,

J. Mockor: *Groups of Divisibility.* 1983, 192 pp. ISBN 90-277-1539-4

A. Wwarynczyk: *Group Representations and Special Functions.* 1986, 704 pp.
ISBN 90-277-2294-3 (pb), ISBN 90-277-1269-7 (hb)

I. Bucur: *Selected Topics in Algebra and its Interrelations with Logic, Number Theory and Algebraic Geometry.* 1984, 416 pp. ISBN 90-277-1671-4

H. Walther: *Ten Applications of Graph Theory.* 1985, 264 pp.
ISBN 90-277-1599-8

L. Beran: *Orthomodular Lattices. Algebraic Approach.* 1985, 416 pp.
ISBN 90-277-1715-X

A. Pazman: *Foundations of Optimum Experimental Design.* 1986, 248 pp.
ISBN 90-277-1865-2

K. Wagner and G. Wechsung: *Computational Complexity.* 1986, 552 pp.
ISBN 90-277-2146-7

A.N. Philippou, G.E. Bergum and A.F. Horodam (eds.): *Fibonacci Numbers and Their Applications.* 1986, 328 pp. ISBN 90-277-2234-X

C. Nastasescu and F. van Oystaeyen: *Dimensions of Ring Theory.* 1987, 372 pp.
ISBN 90-277-2461-X

Shang-Ching Chou: *Mechanical Geometry Theorem Proving.* 1987, 376 pp.
ISBN 90-277-2650-7

D. Przeworska-Rolewicz: *Algebraic Analysis.* 1988, 640 pp. ISBN 90-277-2443-1

C.T.J. Dodson: *Categories, Bundles and Spacetime Topology.* 1988, 264 pp.
ISBN 90-277-2771-6

V.D. Goppa: *Geometry and Codes.* 1988, 168 pp. ISBN 90-277-2776-7

A.A. Markov and N.M. Nagorny: *The Theory of Algorithms.* 1988, 396 pp.
ISBN 90-277-2773-2

E. Kratzel: *Lattice Points.* 1989, 322 pp. ISBN 90-277-2733-3

A.M.W. Glass and W.Ch. Holland (eds.): *Lattice-Ordered Groups. Advances and Techniques.* 1989, 400 pp. ISBN 0-7923-0116-1

N.E. Hurt: *Phase Retrieval and Zero Crossings: Mathematical Methods in Image Reconstruction.* 1989, 320 pp. ISBN 0-7923-0210-9

Du Dingzhu and Hu Guoding (eds.): *Combinatorics, Computing and Complexity.* 1989, 248 pp. ISBN 0-7923-0308-3

Other *Mathematics and Its Applications* titles of interest:

A.Ya. Helemskii: *The Homology of Banach and Topological Algebras.* 1989, 356 pp. ISBN 0-7923-0217-6

J. Martinez (ed.): *Ordered Algebraic Structures.* 1989, 304 pp. ISBN 0-7923-0489-6

V.I. Varshavsky: *Self-Timed Control of Concurrent Processes. The Design of Aperiodic Logical Circuits in Computers and Discrete Systems.* 1989, 428 pp. ISBN 0-7923-0525-6

E. Goles and S. Martinez: *Neural and Automata Networks. Dynamical Behavior and Applications.* 1990, 264 pp. ISBN 0-7923-0632-5

A. Crumeyrolle: *Orthogonal and Symplectic Clifford Algebras. Spinor Structures.* 1990, 364 pp. ISBN 0-7923-0541-8

S. Albeverio, Ph. Blanchard and D. Testard (eds.): *Stochastics, Algebra and Analysis in Classical and Quantum Dynamics.* 1990, 264 pp. ISBN 0-7923-0637-6

G. Karpilovsky: *Symmetric and G-Algebras. With Applications to Group Representations.* 1990, 384 pp. ISBN 0-7923-0761-5

J. Bosak: *Decomposition of Graphs.* 1990, 268 pp. ISBN 0-7923-0747-X

J. Adamek and V. Trnkova: *Automata and Algebras in Categories.* 1990, 488 pp. ISBN 0-7923-0010-6

A.B. Venkov: *Spectral Theory of Automorphic Functions and Its Applications.* 1991, 280 pp. ISBN 0-7923-0487-X

M.A. Tsfasman and S.G. Vladuts: *Algebraic Geometric Codes.* 1991, 668 pp. ISBN 0-7923-0727-5

H.J. Voss: *Cycles and Bridges in Graphs.* 1991, 288 pp. ISBN 0-7923-0899-9

V.K. Kharchenko: *Automorphisms and Derivations of Associative Rings.* 1991, 386 pp. ISBN 0-7923-1382-8

A.Yu. Olshanskii: *Geometry of Defining Relations in Groups.* 1991, 513 pp. ISBN 0-7923-1394-1

F. Brackx and D. Constales: *Computer Algebra with LISP and REDUCE. An Introduction to Computer-Aided Pure Mathematics.* 1992, 286 pp. ISBN 0-7923-1441-7

N.M. Korobov: *Exponential Sums and their Applications.* 1992, 210 pp. ISBN 0-7923-1647-9

D.G. Skordev: *Computability in Combinatory Spaces. An Algebraic Generalization of Abstract First Order Computability.* 1992, 320 pp. ISBN 0-7923-1576-6

E. Goles and S. Martinez: *Statistical Physics, Automata Networks and Dynamical Systems.* 1992, 208 pp. ISBN 0-7923-1595-2

Other *Mathematics and Its Applications* titles of interest:

M.A. Frumkin: *Systolic Computations.* 1992, 320 pp. ISBN 0-7923-1708-4

J. Alajbegovic and J. Mockor: *Approximation Theorems in Commutative Algebra.* 1992, 330 pp. ISBN 0-7923-1948-6

I.A. Faradzev, A.A. Ivanov, M.M. Klin and A.J. Woldar: *Investigations in Algebraic Theory of Combinatorial Objects.* 1993, 516 pp. ISBN 0-7923-1927-3

I.E. Shparlinski: *Computational and Algorithmic Problems in Finite Fields.* 1992, 266 pp. ISBN 0-7923-2057-3

P. Feinsilver and R. Schott: *Algebraic Structures and Operator Calculus. Vol. 1. Representations and Probability Theory.* 1993, 224 pp. ISBN 0-7923-2116-2

A.G. Pinus: *Boolean Constructions in Universal Algebras.* 1993, 350 pp. ISBN 0-7923-2117-0

V.V. Alexandrov and N.D. Gorsky: *Image Representation and Processing. A Recursive Approach.* 1993, 200 pp. ISBN 0-7923-2136-7

L.A. Bokut´ and G.P. Kukin: *Algorithmic and Combinatorial Algebra.* 1993, 469 pp. ISBN 0-7923-2313-0