SEGMENTATION–BASED SHAPE–ADAPTIVE IMAGE CODING

ZHAO YING

SEGMENTATION–BASED SHAPE–ADAPTIVE IMAGE CODING

RESEARCH THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL ENGINEERING

ZHAO YING

SUBMITTED TO THE SENATE OF THE TECHNION — ISRAEL INSTITUTE OF TECHNOLOGY
IYAR, 5759 HAIFA MAY, 1999

THIS RESEARCH THESIS WAS SUPERVISED BY PROFESSOR D. MALAH
UNDER THE AUSPICES OF THE ELECTRICAL ENGINEERING
DEPARTMENT

ACKNOWLEDGMENT

I am grateful to Professor David Malah, for his dedicated supervision, wise advice and generous help.

I wish to express gratitude to the staff of the Signal Processing Laboratory and all my colleagues. Thanks for the love and friendship, which accompanied me during the years in Israel.

I also would like to thank to my family, for their encouragement and confidence in me.

THE GENEROUS FINANCIAL HELP OF LADY DAVIS AND SIGNAL AND IMAGE PROCESSING LABORATORY IS GRATEFULLY ACKNOWLEDGED

Dedicated to my parents

Contents

List of Figures

List of Tables

Abstract

This research addresses the issue of segmentation-based shape-adaptive image coding. The purpose of the research is to achieve bit-rate reduction by coding image-dependent arbitrary shaped segments, instead of regular blocks. Shape-adaptive image coding techniques belong to a more general field, known as second generation image coding. These image coding techniques gain growing attention in recent years.

The main motivation for applying segmentation-based techniques is that at low bit rates, *blocking effects* are caused in the reconstructed image by block-based 2D-DCT coding techniques, such as JPEG. To overcome this drawback, segmentation-based shape-adaptive coding methods aim to exploit the characteristics of the human visual system (HVS). The HVS tends to be sensitive to distortion along the region boundary, while larger segment content distortion is more tolerable. Hence, the main idea is to build an image segmentation system which partitions the original image into regions as the human viewer does. The content information within the segment is coded by texture coding techniques, so that it can be concisely represented. Even if we want to apply JPEG type coding, after the segment information is known, a meaningful bit-rate reduction is still achievable. This is done by coding blocks inside a segment with rougher quantization while blocks which include part of the segment boundary are treated in a special way by block-based shape-adaptive coding techniques, discussed in

the thesis. The shape (contour) information is coded independently and is transmitted to the receiver, since the image partition does not follow the rigid rectangular grid.

Our research focuses on the first two topics: Image segmentation and texture (segment content) coding. Shape (contour) coding performed in this research is by a known modified chain code technique.

Concerning image segmentation, we first implemented two known morphological image segmentation algorithms. Simulation results show that several innate drawbacks exist in the known algorithms. Therefore, we propose an *edge detection, local-activity classification* segmentation algorithm. This algorithm can sufficiently simplify the input image, and flat zones are generated within the segments. Conventional *labelling techniques* can then be applied to identify them. Following *marker extraction*, *region growing* is used as the *decision* tool to delimit the region boundaries. The simulation results show that if steep edges exist between adjacent segments, separate segments are well identified. On the contrary, if adjacent segments have similar characteristics, segments are clustered into one segment. Therefore, it is possible to find a compact representation for the texture content of the segment. Moreover, the partition is relatively simpler than the partition generated by the other algorithms, so that contour coding cost is reduced. With some further study of *region refinement*, we believe that the proposed algorithm is a promising candidate for a generic image segmentation system.

Concerning shape-adaptive image coding, two main categories are considered, namely, block-based and region-based approaches.

The block-based shape-adaptive approach partitions a given segment into 8×8 blocks. As mentioned before, rougher coefficients quantization is possible for coding blocks inside the segment. Blocks which include part of the segment boundary are

coded by other techniques, such as, shape-adaptive DCT (SADCT) or low-pass extrapolation (LPE) padding. The SADCT technique transforms the block data within the segment to the DCT domain by performing vertical and horizontal DCTs of variable size, according to row and column sizes of inner pixels. While extrapolation fills in the block data outside the segment based on the data inside the segment. Even though these two algorithms are quite simple, they are not efficient enough for bit reduction. Therefore, we further propose a novel extrapolation method, which aims to minimize the l_1 norm of the corresponding transform coefficients. Because it derives from a basis pursuit (BP) approach, we name it BP extrapolation. According to simulation results, the BP extrapolation method shows a compression advantage over the other examined block extrapolation methods. Its disadvantage is its rather high complexity, as it requires the use of linear programming.

Contrary to the block-based approach, the region-based approach aims to find the best approximation function of a given order (in least squared-error sense) of the original segment gray-level function, and use a concise representation of the approximation function. From a theoretical point of view, the region-based approach exploits the global characteristics of the segment and has potential for higher compression than the block-based approach. Current research works are mainly concentrated on *orthogonal basis function generation*, because of the simplicity and stability in calculating the representative coefficients. However, due to numerical problems in generating sufficiently high order shape-adaptive orthogonal functions needed to describe typical image segments, we have preferred the block-based approach and include in the thesis only the theoretical formulation of the region-based approach.

Chapter 1

Introduction

1.1 Motivation

The conventional block DCT based image coding system (such as JPEG) generally results in reconstructed images with sufficiently good quality when the code bit rate is higher than 0.5 bits/pixel [1]. The advantages of this kind of image coding technique include:

- No need to transmit block-shape information, which could be quite costly from the coding point of view, due to fixed rectangular block partition pattern.
- Low computation complexity, appealing for hardware realization.

However, at low bit rates (lower than 0.5 bits/pixel), the rigid rectangular partition of the blocks in the picutre may result in discontinuities in the reconstructed gray-level values between adjacent blocks. Thus, an annoying *blocking effect* is caused. That is, a rectangular grid is embedded in the reconstructed picture. The goal of this work is to encode images at low bit rates while still obtaining reconstructed images with

acceptable perceptual quality.

Research works in the field are motivated by the idea of exploiting the special properties of the human visual system (HVS) , which means to generate a system that can partition the original image into regions as a human viewer does. Psychometric experiments indicate that within the partition segments, lower bit rate coding is possible, because of the fact that human viewers tend to ignore the larger distortion of the inner content within the region boundary. Various methods have been exploited to segment the image into regions with uniform texture according to some suitable homogeneity criterion. The resulting internal gray-level signal within the segment, which can be considered as *texture* of the segment, and the image partition, correspondingly segment *contour* are then coded and transmitted. The obtained texture and contour coding is expected to be more natural, because the coding performed coincides with psychophysical concepts of vision. If powerful description tools for contours as well as texture are at hand, so that the local image characteristics is fully exploited. Then, one can also expect the attainable compression ratios to be significantly higher than in regular block-oriented coding schemes.

As stated earlier, the necessity of suitable texture and contour coding techniques inspires researchers to investigate the possibility of new coding techniques. The new generation (*second generation*) coding techniques [2] are focusing on fully exploiting the shape information and the homogeneity of the inner content of the partitioned regions, and people did obtain quite impressive achievements (e.g., [3, 4, 5, 6]). Based on our investigation, there are two approaches to shape-adaptive image coding, i.e., block-based and region-based approach. The region-based approach aims to find the best approximation function of the original image segment, so that it is possible to represent the approximation function more concisely. The block-based approach still

follows the context of conventional techniques, such as JPEG. But, bit-rate reduction can be achieved by coding the inner blocks with rougher quantization and boundary blocks with special techniques. Compared with the region-based approach, the block-based approach is simpler and is compatible with existing systems, such as JPEG and MPEG-4 [7]. We will discuss the various existing techniques that we studied in the following chapters.

Concerning image segmentation, there is an abundant literature published in recent years [8, 9, 10, 11, 12]. This means that it is hard to exploit all of the available possibilities. Therefore, we decided to concentrate our effort on one of the categories: image segmentation using *mathematical morphology*. We will first provide some background material concerning mathematical morphology before we go into the details of the different segmentation algorithms implemented.

1.2 Main Contributions & Thesis Organization

The main original contributions of this work can be divided in two parts:

1. Concerning image segmentation, an *edge detection, local-activity classification* segmentation algorithm is proposed. Simulation results indicate that the proposed algorithm is better than the other two mathematical morphology-based segmentation algorithms for the coding task under consideration, while at the same time, greatly reducing the computation time.

2. Concerning block-based shape-adaptive coding, a novel extrapolation technique is proposed for block extrapolation coding. It draws on a method used for basis pursuit (BP). BP extrapolation minimizes the l_1 norm of the transform

coefficients and is particular useful for coding, as indicated by simulation results which are compared to conventional JPEG coding.

The thesis is organized as follows:

Chapter 2 provides an introduction to gray-scale morphology and contour coding. Since mathematical morphology tools are used in the examined segmentation algorithms, the purpose of this chapter is to introduce mathematical morphology basics to readers who are not familiar with it. Contour (shape) coding is not in the focus of our research, therefore, only known essentials concerning this topic are briefly introduced in the second part of this chapter.

Chapter 3 centers on image segmentation. In the first two sections, two known morphological segmentation algorithms are described. Following a discussion of the drawbacks of these algorithms which we actually implemented, we propose our segmentation algorithm in the third section. The advantages of the proposed algorithm are further discussed.

Chapter 4 and the appendix are related to the issue of block-based shape-adaptive image coding. Available coding techniques: Shape-adaptive DCT (SADCT) and Low-pass extrapolation (LPE), are discussed in section 4.1. and section 4.2, respectively. Motivated by the block data extrapolation idea, a novel extrapolation algorithm is proposed (BP). Simulation results indicate the advantage of the proposed algorithm in image compression. In the appendix, a general overview of the JPEG baseline system is described, so that the reader can refer to the system implementation.

Chapter 5 discusses region-based shape-adaptive image coding. Several algorithms are addressed, but our conclusion is that because of practical numerical problems, the available algorithms are not yet well suited to the task.

Finally, chapter 6 provides conclusions and outlines issues for further research.

Chapter 2

Background Overview

The purpose of this chapter is to provide some essential background information which would help readers to better understand segmentation techniques discussed in the thesis. Topics to be covered include: basic mathematical morphology tools, watershed algorithm, and contour (shape) coding techniques.

2.1 Mathematical Morphology

Mathematical Morphology (MM) is a theory which provides useful tools for image processing and in particular for image segmentation algorithms. MM-based segmentation algorithms are quite popular [13, 14, 15, 12, 10]. However, some readers may not be familiar enough with this topic, so we introduce here the definitions and terminology related to the segmentation algorithm proposed in this work. Interested readers may refer to [16] [17] for detailed information on MM.

2.1.1 Morphological Operators and Filters

A large number of morphological tools rely on two basic sets of transformations known as *erosion* and *dilation*. Generally speaking, two sets of erosion and dilation are widely used. The first one deals with erosion and dilation with a *flat structuring element*. If $f(x)$ denotes an input signal, and M_n, a flat structuring element of size n (by "flat"it is meant – constant gray-level within the element), the *erosion* and *dilation* by M_n are defined by:

$$\text{Erosion:} \quad \varepsilon_n(f)(x) = Min\{f(x+y), y \in M_n\} \tag{2.1}$$

$$\text{Dilation:} \quad \delta_n(f)(x) = Max\{f(x-y), y \in M_n\} \tag{2.2}$$

A gray-scale image with simple geometric shapes is used to demonstrate the dilation and erosion operations. Shown in Fig. 2.1, are the original image which is a gray-scale image of size 100×100, and the images which result from dilation and erosion operations that were performed with a flat structuring element of size 7×7. As seen in Fig. 2.1, light elements expand in the "dilation image", and the dark elements shrink; the reverse procedure happens to the "erosion image".

Another set of erosion and dilation operations, defined with respect to a reference function r, are also called *geodesic dilation(erosion)*. The *geodesic dilation of size one* (smallest size structuring element, usually a 3×3 element) is defined as the minimum between the dilation of size one of the original function f and the reference function r. The *geodesic erosion* is defined by duality.

$$\text{Geodesic dilation of size 1:} \quad \delta^1(f,r) = Min\{\delta_1(f), r\} \tag{2.3}$$

$$\text{Geodesic erosion of size 1:} \quad \varepsilon^1(f,r) = -\delta^1(-f,-r) \tag{2.4}$$

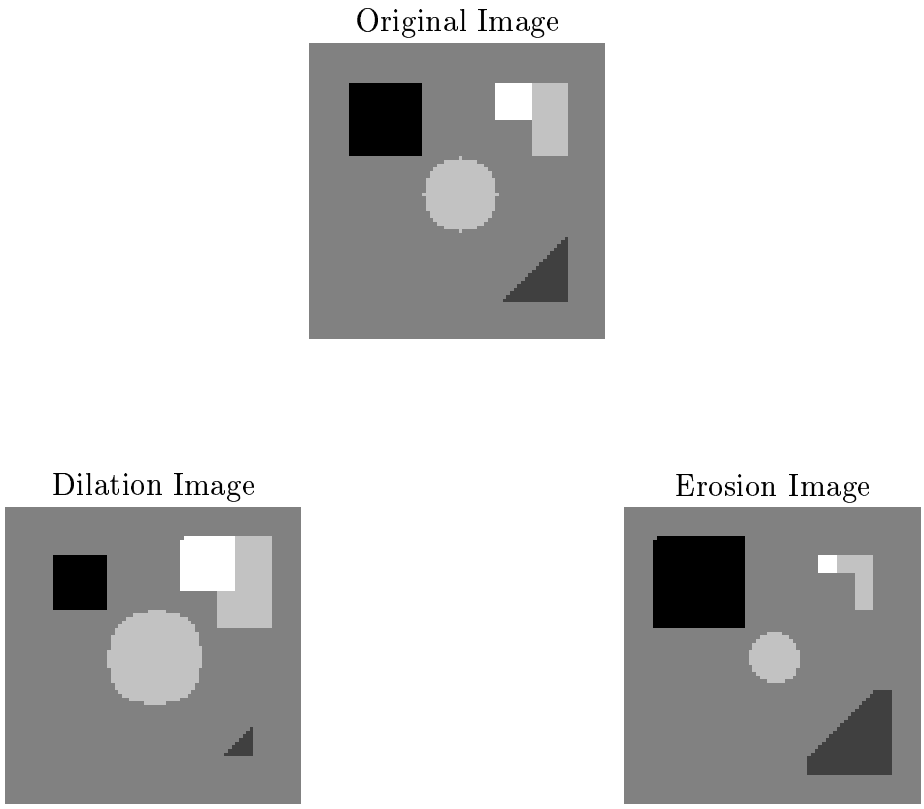


Figure 2.1: Morphological Dilation & Erosion
איור 2.1: הרחבה וכרסום מורפולוגיים

Geodesic dilation (erosion) of infinite size, which have more practical applications, are also called *reconstruction by dilation* (*by erosion*), and the definitions are given by:

$$\text{Reconstrucion by dilation:} \quad \gamma^{(rec)}(f,r) = \delta^{\infty}(f,r) = (\ldots\delta^{1}(\ldots\delta^{1}(f,r)\ldots,r)) \tag{2.5}$$

$$\text{Reconstrucion by erosion:} \quad \phi^{(rec)}(f,r) = \varepsilon^{\infty}(f,r) = (\ldots\varepsilon^{1}(\ldots\varepsilon^{1}(f,r)\ldots,r)) \tag{2.6}$$

Correspondingly, the operation of reconstruction by dilation (erosion) is illustrated in Fig. 2.2. The light element in the reconstruction-by-dilation image can not expand outside of the constraint of the reference image. The same situation happens to the dark element in the reconstruction-by-erosion image.

Elementary erosion and dilation allow the definition of morphological filters such as the morphological *opening* and *closing*:

$$\text{Morphological opening:} \quad \gamma_n(f) = \delta_n(\varepsilon_n(f)) \tag{2.7}$$

$$\text{Morphological closing:} \quad \phi_n(f) = \varepsilon_n(\delta_n(f)) \tag{2.8}$$

A morphological opening (closing) simplifies the original signal by removing the bright (dark) components that do not fit within the structuring elements (Fig. 2.3). If the simplification has to deal with both bright and dark elements, an *open-close* $\gamma_n(\phi_n(f))$ or a *close-open* $\phi_n(\gamma_n(f))$ has to be used. These filters are not self-dual, but in practice they approximately remove the same kind of information. After the original image is passed through these filters, a set of *flat zones* (areas with constant gray-level) are produced, but the contours of the regions may be seriously corrupted. Instead, filters

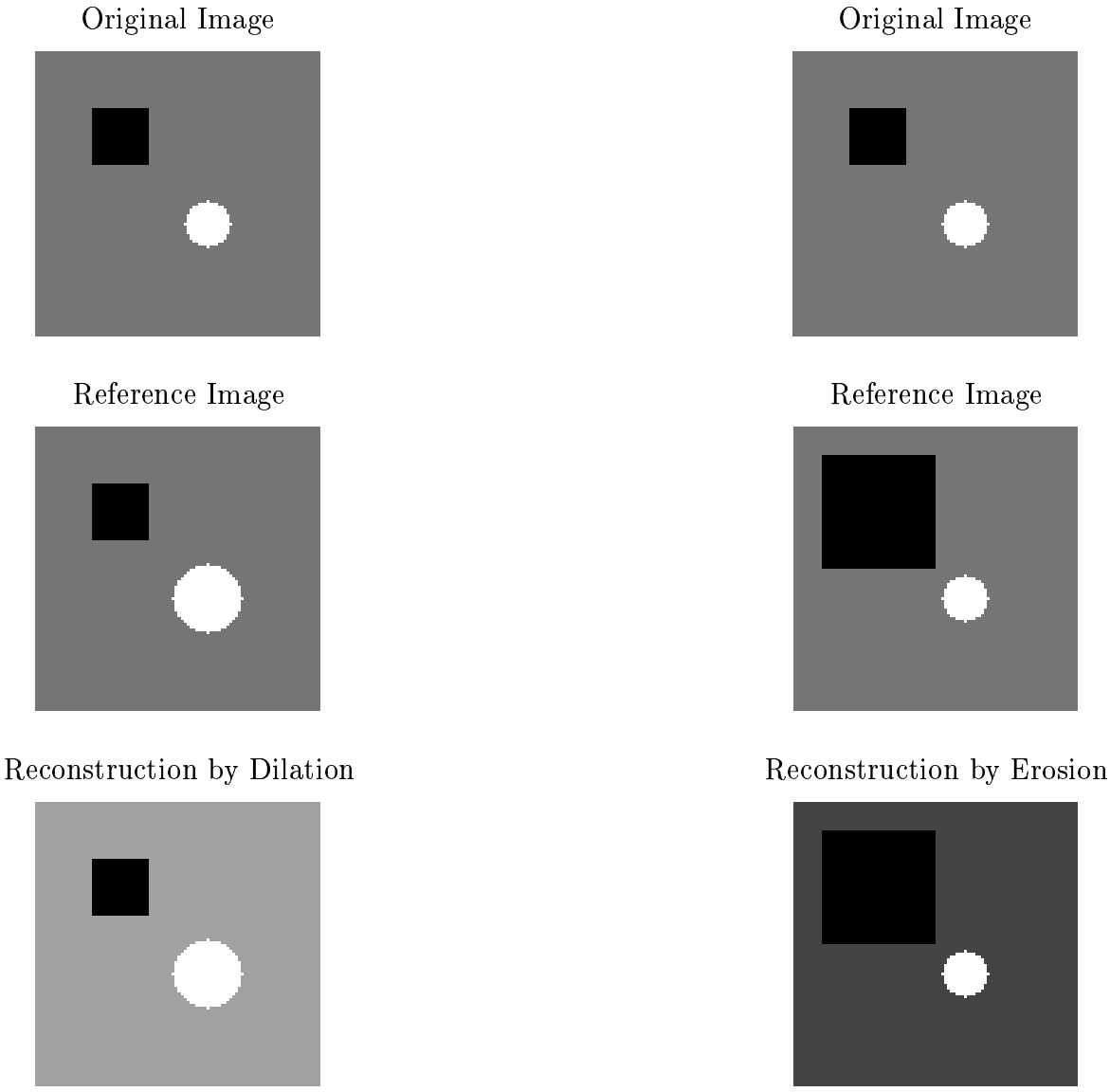


Figure 2.2: Morphological Reconstruction by Dilation & Erosion
איור 2.2: שחזורים מורפולוגיים באמצעות הרחבה וכרסום

by reconstruction are very useful in morphological image segmentation, because they have the property of keeping or removing the original contours of the regions without causing severe distortion. Referring to Fig. 2.3, the contour preserving property of these filters is obvious.

The most popular filters by reconstruction are the *opening by reconstruction of erosion* $\gamma^{(rec)}(\varepsilon_n(f), f)$. And, by duality, a *closing by reconstruction of dilation* can be defined: $\phi^{(rec)}(\delta_n(f), f)$. These two filters have almost the same filtering capacity.

Finally, the *h_maxima* and *h_minima* operators are defined in terms of reconstruction. If h is a constant,

$$\text{H\_maxima:} \quad h\_max(f) = \gamma^{(rec)}(f - h, f) \tag{2.9}$$

$$\text{H\_minima:} \quad h\_min(f) = \phi^{(rec)}(f + h, f) \tag{2.10}$$

The attraction of these operators are their ability to remove poorly contrasted regions and produce a set of flat zones with high contrast without causing severe distortion to the contours of the original regions as illustrated in Fig. 2.3. In this figure, the size of the structuring element is 25×25, and the value of h (in eq. (2.9) and (2.10)) is 20.

2.1.2 Morphological Gradients

Three Morphological gradients are generally used:

$$\text{Morphological gradient:} \quad g(f) = \delta_1(f) - \varepsilon_1(f) \tag{2.11}$$

$$\text{Gradient by erosion:} \quad g^{-}(f) = f - \varepsilon_1(f) \tag{2.12}$$

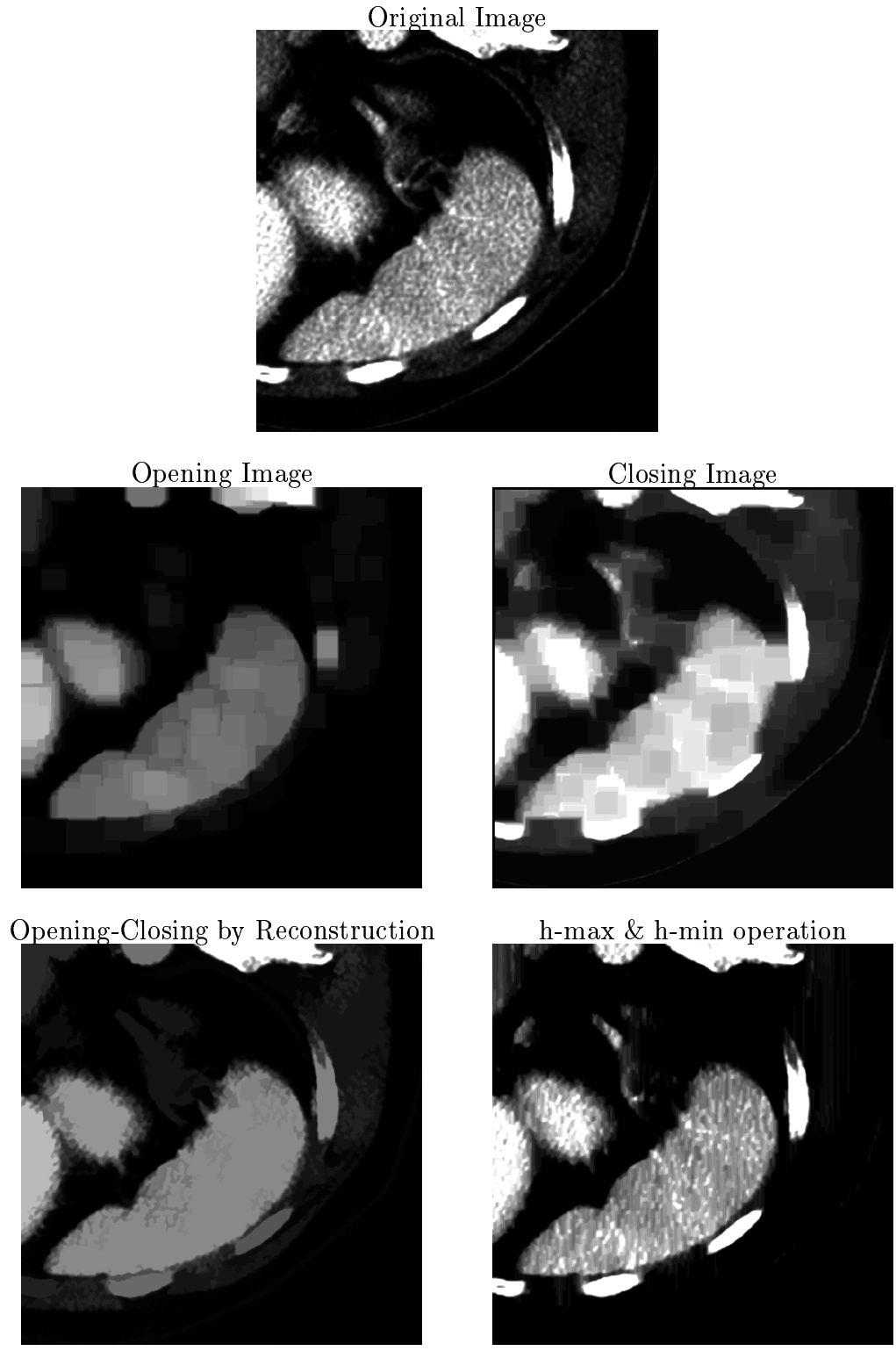


Figure 2.3: Image Simplification by Morphological Filters (Operators)
איור 2.3: פישוט התמונה באמצעים מסננים ואופרטרים

$$\text{Gradient by dilation:} \quad g^{+}(f) = \delta_1(f) - f \tag{2.13}$$

Since morphological gradients are only mentioned later but are not directly computed in our implementation of the segmentation algorithms, we do not discuss them in details.

2.2 Morphological Watershed Algorithm

The watershed algorithm is a classical morphological tool used for image segmentation. Referring to the drawing in Fig. 2.4, we can consider the gray-level values of a picture as the altitude of an imaginary relief. *Watershed lines* partition the space by associating a region called a *catchment basin* to each local minimum. *Immersion* simulation consists of flooding the surface from its local minimum. Starting from the minimum of lowest altitude, the water progressively fills up the catchment basins. When the water level reaches the altitude of other minima, these minima start to be active, and the flooding process also originates from these minima. Now, when the water coming from two different minima would merge, an imaginary dam is built to prevent any mixing of water. The procedure is ended when the water level is higher than the absolute maximum. In this case, each minima is surrounded by water, defining its *catchment basin*, and a dam delimiting its border, its *watershed line*. The catchment basins constitute a partition of the space. To get the contours of objects, the conventional watershed algorithm is applied to the *morphological gradient* of the signal. Indeed, a contour in the image corresponds to a bright line in the morphological gradient. However, the direct segmentation of the gradient image by the watershed algorithm results in extreme *oversegmentation*, because the algorithm assigns a different region for each individual local minimum of the gradient. To solve this problem,

the gradient can be simplified to segment only the objects of interest. It was found in recent research that the conventional morphological watershed algorithm [18] has the disadvantage of causing false contours, and improvements have been made [19]. In the "Hierarchical Morphological Synthesis by Analysis Segmentation Algorithm" given in [13], the improved watershed algorithms is used. The description of the improved watershed will be given later on, when the complete algorithm is discussed.

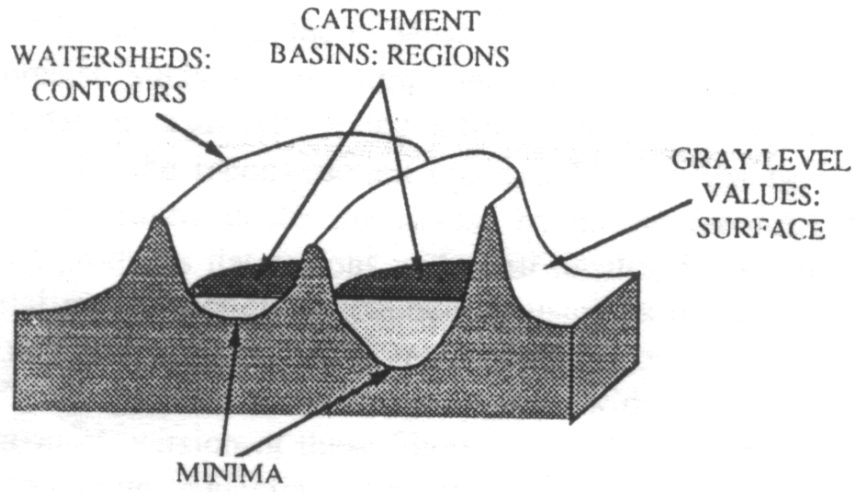


Figure 2.4: Morphological Watershed
איור 2.4: אלגוריתם מורפולוגי watershed

In general, the attraction in using morphological operations is their object-oriented characteristics which helps in simplifying the source image. Mathematical morphology deals with geometric features, such as size, shape, contrast or connectivity, so that the important information, big flat areas or high contrast areas with smaller size, is well kept. On the other hand, less important areas are eliminated in the processed image. It is also claimed in [10] that the morphological watershed, widely used,

does not integrate geometric information, such as curvature and continuity, for low-contrast images, so that the final contours are not well localized. Therefore, the authors suggest using an active contour algorithm [10] instead of the watershed tool, to finely localize the boundary of the regions. The discussion of this topic is out of the scope of this work, and interested readers may refer to [10].

2.3 Contour Coding

Contour coding, although not the main subject of our work, is definitely essential in building up the whole segmentation-based image-coding system. Through the years, people proposed many methods for contour coding. An overview of which can be found in [20]. Recent progress is reported in [21, 7]. The difficulty of the task lies in the trade off between the cost (in bit rate) of contour coding and the precise description of contour information. According to [14], the average coding cost of the algorithm proposed in [22] is 1.34 bits per *contour pixel*. In the sequel, the popular contour coding techniques, the chain code and the modified chain code, are addressed.

For the convenience of the description later on, we first provide the definitions of a partition (label) image and a contour image. The first one refers to the original pixel domain, where each pixel is given a certain number (label) indicating the region it belongs to. Hence, a contour is not directly represented by a single element but by two neighboring pixels with different labels. The definition of a contour image is illustrated in Fig. 2.5, where the circles denote pixels in the original partition image and the rest (lines and their intersection points, the squares) indicate the contour image. A line is set to zero (broken line) if its two closest circles have the same label which indicates the same region. Otherwise, it is set to one (solid line). The squares

are set to one if one of its associated line element is one. In this way, a partition image will correspond to a unique binary contour image and vice versa.

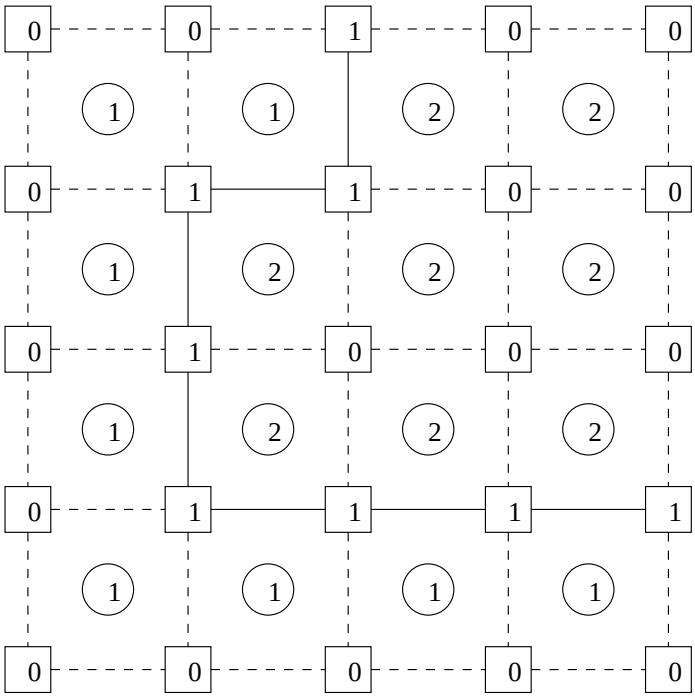


Figure 2.5: Label Image vs. Contour Image
איור 2.5: תמונת פסיפס לעומת תמונת שפות

As stated earlier, proper contour coding is essential for properly addressing of the boundary information after image segmentation. There are mainly two kinds of contour coding techniques – lossless and lossy. To properly choose a contour coding method involves a trade off between the distortion endurable and coding efficiency. Lossy coding techniques include Fourier Descriptor, Contour Approximation by Splines or Straight Lines, etc. Since the contours produced by our segmentation algorithm, presented in Chapter 3, are already simplified, further lossy coding will cause too much distortion. Therefore, we chose the lossless modified chain code

method for contour coding.

2.3.1 Chain Code

Chain code is a classic lossless contour coding method [20]. In the coding procedure, the directional vectors between successive boundary pixels are encoded. For example, a commonly used chain code (Fig. 2.6) employs four directional vectors which could be coded by 2-bit code words. The chain codes contain the starting pixel address A followed by a string of code words. In the case of Fig. 2.6, the chain code is:

$$A00330003232221221101 \tag{2.14}$$

Another version of chain code makes use of derivative directional vectors. In this case, only three symbols are needed to code the four-connected boundary. These symbols are "straight ahead", "turn left" and "turn right". Although a restriction has to be made which prevents to "turn back", it works in most cases. However, from the starting point, the first movement is still in one of four-directions, so that the original chain code is applied to code the first step. If we use 0, 1 and 2 to represent these three derivative symbols, the boundary in Fig 2.6 could be coded as

$$A00201002212002102021 \tag{2.15}$$

Entropy coding can be applied to reduce the bit rate per contour pixel.

2.3.2 Modified Chain Code

The traditional chain code only provides a way to describe the boundary. In fact, there is another important information to code, as far as the coding of a partition image is concerned – the starting points. A starting point defines where the chain

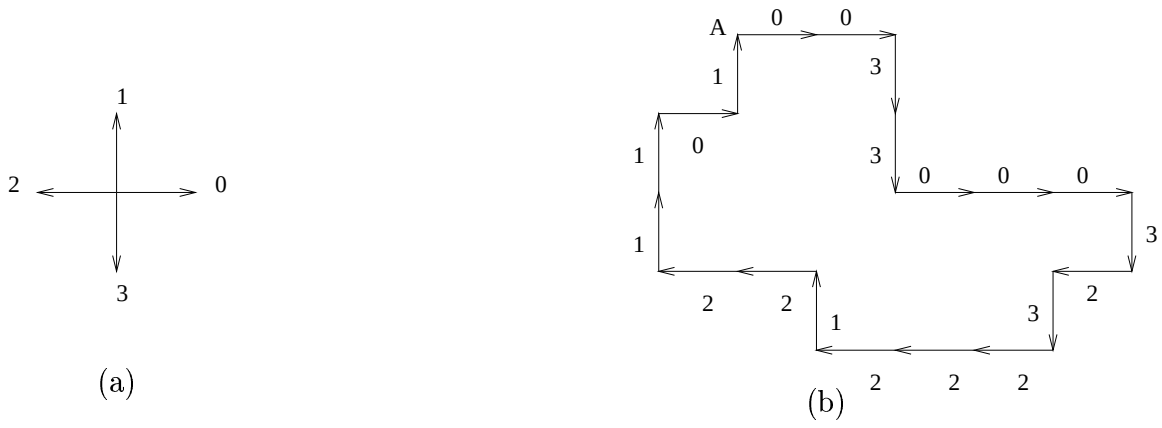


Figure 2.6: Chain Code
איור 2.6: קוד שרשרת

codes begins. This factor can not be neglected because there could be many regions in the partition image and each region needs a starting point. The way to code the starting points was ignored during a long period since the invention of chain codes. In [22], a *modified chain code* which takes in account the coding of starting points is proposed.

Assuming a square image of size $N \times N$, the maximal number of bits needed to code the starting points for M regions are $2 \times M \times \log_2 N$, before entropy coding. An efficient way to reduce this number is to utilize the special positions of these points in the contours. These positions could be *triple points*. A triple point is the intersection of two contour segments and it can serve as a mark of the beginning of two new regions. Since two or more segments share the same starting point (the triple point), the number of bits needed to code the absolute address of the starting points are reduced. Therefore, in the modified chain code [22], a special symbol is inserted in the normal chain code to indicate the triple point. Not all starting points can be treated this way. The starting points of isolated regions have to coded directly

because there is no other contour segment attached to its boundary. Fortunately, the number of this kind of isolated regions is very small.

An example which illustrates the modified chain code algorithm is presented in Fig. 2.7. The original partition image of size 8×8 with 4 regions is displayed in different gray-levels. The absolute starting points (solid dots) and triple point (solid triangle) with associated chain code in the contour image are shown. The contours are generated by using the procedure described which converts the partition image into a contour image. The starting points are the intersections of the line elements in the contour image. A new symbol, e.g. 3, is assigned to represent a triple point. Coding the contours in the contour image, the absolute starting points are $(2, 2)$ and $(10, 0)$. The associated derivative chain codes are 00202020 and 30030020000100, respectively. It should be noted that the first step of each chain code is still a direct chain code instead of a derivative one.

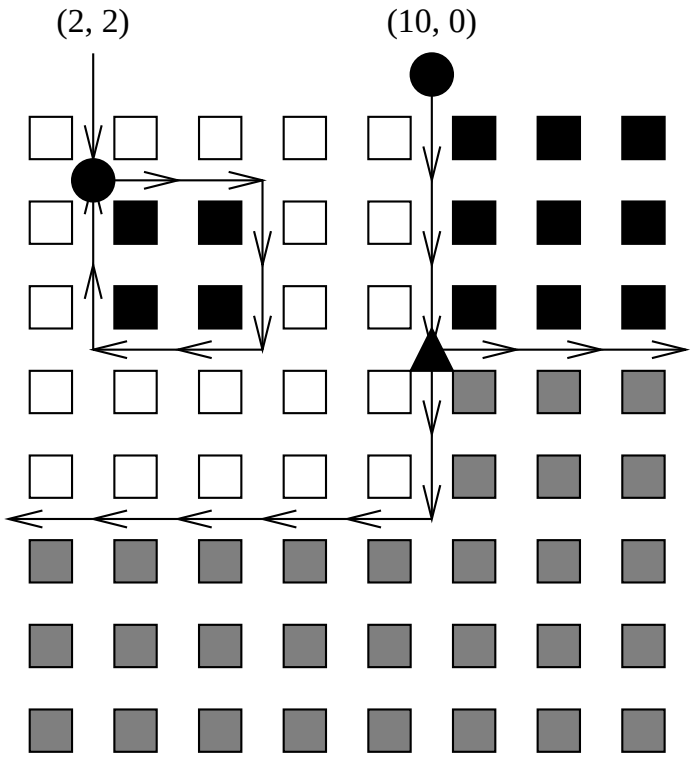


Figure 2.7: Modified Chain Code
איור 2.7: קוד שרשרת משופר

Chapter 3

Image Segmentation

In the introduction chapter, we stated that the purpose of second generation image coding is to partition the natural image into certain *homogeneous* regions as a human viewer does, so that the content information of the image can be fully exploited. Served as the prerequisite step, image segmentation nowadays has expanded into a wide range of study. Hundreds of methods can be found in the published literature in recent years. Unfortunately, at least according to our investigation, current available methods are quite *image-dependent*, and there is still a big gap in front of us to reach a generic image segmentation algorithm. In this chapter, we will discuss the image segmentation issue in detail. Even though there may be hundreds of methods which perform image segmentation, the three algorithms that we implemented use mathematical morphology tools. Two of these algorithms were selected from the literature and the third one is our own proposed algorithm. Therefore, background knowledge was provided in section 2.1 for readers who are unfamiliar with mathematical morphology. In the following sections, we describe the structure of the three implemented algorithms and discuss their performance.

3.1 Hierarchical Morphological Synthesis by Analysis Segmentation Algorithm

The hierarchical morphological synthesis by analysis segmentation algorithm is a quite widely used morphological image segmentation scheme [19, 23, 13]. Even though we did not get very satisfying simulation results, some of the techniques are quite useful for our following research. So we give a detailed description of the algorithm to simplify the explanation later of the other two segmentation schemes.

3.1.1 Algorithm Description

The block diagram of the algorithm is given in Fig. 3.1. A hierarchical structure is proposed: The algorithm first produces a simplified segmentation in the sense that it involves a reduced number of regions. Then, the segmentation is progressively improved by introducing more regions. A combined size-oriented and contrast-oriented criterion [23, 13] is used, so that in the final result, homogeneous regions with large size and small segments with high contrast are constructed. A typical scheme includes four segmentation levels. The first three levels are size-oriented, while the last level, a contrast-oriented segmentation is conducted. There are four separated steps within each level: *simplification, marker extraction, decision* and *modeling*.

- Simplification

The simplification step aims at filtering the noise present in the image, removing less important details, and reducing the number of regions in the segmentation output, while keeping the useful information. The simplification controls the nature and amount of information that is kept for segmentation at this level of the hierarchy.

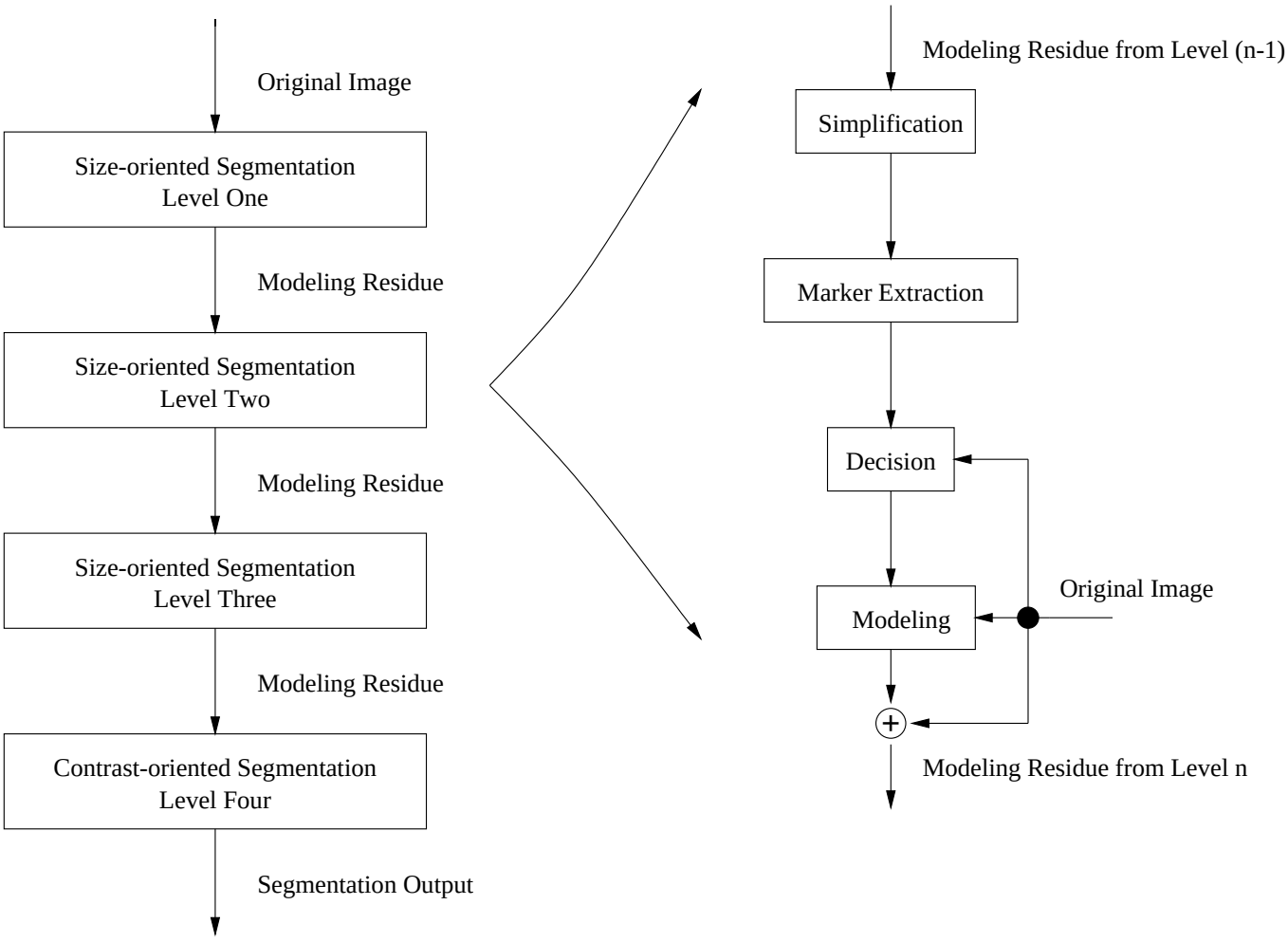


Figure 3.1: Hierarchical Morphological Synthesis by Analysis Segmentation Algorithm

איור 3.1: אלגוריתם סגמנטציה מורפולוגי היררכי המבוסס על סינתזה ע"י אנליזה

As explained above, morphological filters by reconstruction are very efficient for size simplification, morphological h-maxima or h-minima operators are particularly suitable for contrast simplification. Therefore, open-close by reconstruction morphological filters are used to fulfill the simplification task in the first three levels, and morphological h-maximum and h-minimum operators are used in the last level for contrast simplification. Moreover, this algorithm is a top-down scheme, new segments are introduced within the constraint of previous segmentation result in each level of hierarchy. Therefore, decreasing size morphological filters are used in each level to increase the amount of information left after simplification.

- Marker extraction

The goal of this step is to detect the presence of homogeneous regions. It produces markers identifying the interior of the regions that will be segmented. In practice, a marker is a connected component of the image labeled with a specific region number. The marker defines the set of pixels which surely belong to the region and in general, it defines the major part of the region interior. Corresponding to size-oriented simplification and contrast-oriented simplification, two kinds of marker extraction techniques are applied. Both methods should be performed under the constraint of the pre-level segmentation, that is, segmentation result from the previous level.

1. Marker extraction for homogeneous regions: This technique aims at finding flat zones in the simplified image which have size larger than a given lower limit. Theoretically, after morphological filtering by reconstruction, large size flat zones with constant gray-level are produces, we can use a technique which is similar to traditional *flat regions labelling* technique, called *constrained flat regions labelling* [23], to label these large size flat zones. The constrained labelling

technique should take into account that:

(a) The result of the previous segmentation step, that is, when labelling a flat region of the simplified image, we should check that it is not crossing a contour which has been defined by the previous segmentation steps.

(b) Small flat regions should have been removed by the size-oriented simplification. In fact, the simplification filter actually removes all small regions, but not the transitions between large regions. These transitions generally results in a very large number of small (even of area size one) regions. A simple way to handle them is to assign a specific label indicating that they do not represent meaningful regions but uncertainty areas.

2. Marker extraction for contrast regions: The same approach leads to the selection of flat zones that have a contrast higher than a limit defined by the simplification, i.e., a sufficiently large gray-level difference exist between these zones and their neighboring regions. After simplification by h_max and h_min operators, high contrast flat zone are produced. The marker extraction can simply be done by taking all flat zones of the simplified image where there exists at least a spatial position (x, y) such that: $h\_max(f) = f - h$, or $h\_min(h\_max(f)) = h\_max(f) + h$. All flat zones that are not selected are considered as uncertainty areas and the decision step (improved watershed algorithm) will be used to delimit the contours of the regions.

To implement this constraint flat regions labelling technique: We can use flat regions labelling technique to label all of the flat zones (marker candidates) exist in the simplified image, then discard all those that can not satisfy the constraint mentioned above. Labeled flat zones left are markers.

- Decision

Following marker extraction, the number of regions and their interior are known. However, a large number of pixels are yet not assigned to any region. These pixels correspond to uncertainty areas mainly concentrated around the contours of the regions. Assigning these pixels to a given region can be viewed as a decision process that precisely defines the partition. Because of the deficiency of classical watershed as a decision tool, we implemented an improved version of watershed according to the method mentioned in [19]. The idea of the authors of [19] is: Using the watershed algorithm directly on the original gray-level image instead of on the gradient image, and the set of markers is extended until they occupy all the available space. During the extension, pixels of the uncertainty areas are assigned to a given marker. A point is assigned to a specific region because it is in the neighborhood of at least one marker and is more similar (in the sense defined by a specific criterion) to this marker than to any other marker of its neighborhood. The similarity criterion is chosen to be the gray-level difference between the pixel under consideration and the mean of the pixels that have already been assigned to the region. We implemented the algorithm in a queue structure proposed by the authors in [19]. The algorithm consists of the following two steps:

1. Initialization: Put in the queue the location of all pixels corresponding to the interior of a region in the labeled marker (pixels not belonging to uncertainty areas). These pixels have the highest similarity to the respective regions. The uncertainty pixels are left outside of the queue, and are assigned to the regions they belong to by a flooding procedure.

2. Flooding: Assign pixels to regions following a region growing procedure. To

constrain the current segmentation to the segmentation obtained in previous levels, the algorithm checks whether the region and the pixel under consideration belong to the same partition class in the pre-level segmentation, that is, whether they are *compatible*. The flooding extracts a pixel from the queue. If the pixel does not belong yet to a region, we know that at least one of its compatible neighbors belongs to a region. Therefore, all compatible neighboring regions are examined, the distances between these neighboring regions and the current pixels are assessed, and the pixel is assigned to the region with the highest similarity. Once a new pixel has been assigned to a region, the mean gray-level of the region should be updated in order to accurately compute its difference with respect of new pixels. If the last extracted pixel, which has been assigned to some region **A**, has some compatible neighbors that do not belong to any region, these neighbors are put in the queue according to the order of similarity defined by their difference to the region **A**. The algorithm stops, when all unassigned pixels are assigned to some regions.

- Modeling

The last step in the current level is modeling, using an efficient way to represent the segment gray-level. The output of the decision step is a set of connected regions. One can use any content coding techniques to describe these regions (Since contour coding could be very costly, it is not suggested to perform contour coding at this stage. Nevertheless, the partition of the image is precisely known). We can choose a content coder from a large number of coders. In [13] a shape adaptive second-order polynomial coding algorithm is chosen, detailed discussion of this texture coding technique is provided in Chapter 5. In our implementation, we just filled the region with its

mean, because of the simplicity of mean calculation, while obtaining only a small difference as compared with a low order polynomial coding. The difference between the coded image and the original one, called the *modeling residue* is computed (at the first level, the original image is considered as a coding residue, and the whole image is considered as a single segment). Since this residue contains all the information about the poorly coded regions, it is used as the image to be segmented in the next level.

3.1.2 Discussion

The above scheme produces a reduced a number of regions as compared with the Morphological Simplification, Region Splitting and Merging segmentation algorithm described in the section 3.2. But there are several problems in its application. First, it is not very clear how to choose the proper size of the morphological filer without prior knowledge of the image, e.g., is the local characteristic of the region flat, texture or an edge, etc. Moreover, in some applications, additional information such as the approximate size of output segments is also needed as prior knowledge. Even with an hierarchical structure, the simplification step at the first level is very important, and there is a dilemma concerning the selection of the size of the morphological filers: Simplification with large size morphological filters may cause big distortion of the boundary of the regions besides the heavy calculation requirement. On the other hand, the image can not be efficiently simplified by small size morphological filters. Yet, the biggest barrier, that strongly restricts the application of this algorithm, is the decision tool used. Even the improved watershed algorithm can not well delimit the boundary of texture regions. And this innate deficiency limits the application to images with big flat homogeneous areas. Furthermore, within the hierarchy, the

sequential segmentation steps are conducted under strict constraint of previous partition results, so that the boundary distortion from the previous step can not be well refined by the following steps. Finally, the complexity of the algorithm is high. So, our conclusion is that this algorithm is not a very suitable candidate for our image segmentation task.

Hierarchical Morphological Synthesis by Analysis segmentation algorithm is a classical image segmentation algorithm, and it is the first algorithm we implemented. Although we finally abandoned its application for our segmentation task, terms and techniques such as labelling, modeling, queue structure etc., are the basis for the implementation of other algorithms. Its segmentation results are also used as a basis for comparison with the segmentation results of the other algorithms we implemented.

3.2 Morphological Simplification, Region Splitting & Merging Segmentation Algorithm

In [20], two basic segmentation algorithm structures are mentioned: Top-down structure versus bottom-up structure. The above "Hierarchical Morphological Synthesis by Analysis" segmentation algorithm, which we implemented, is a top-down scheme, and the author claims that this structure outperforms its rivals. But, since this top-down scheme did not give us very satisfying results, we turned to examine a bottom-up approach. "Morphological Simplification, Region Splitting & Merging Segmentation Algorithm" is a typical bottom-up algorithm [14].

The block diagram of the algorithm is shown in Fig. 3.2, and is described in detail in the following paragraphs.

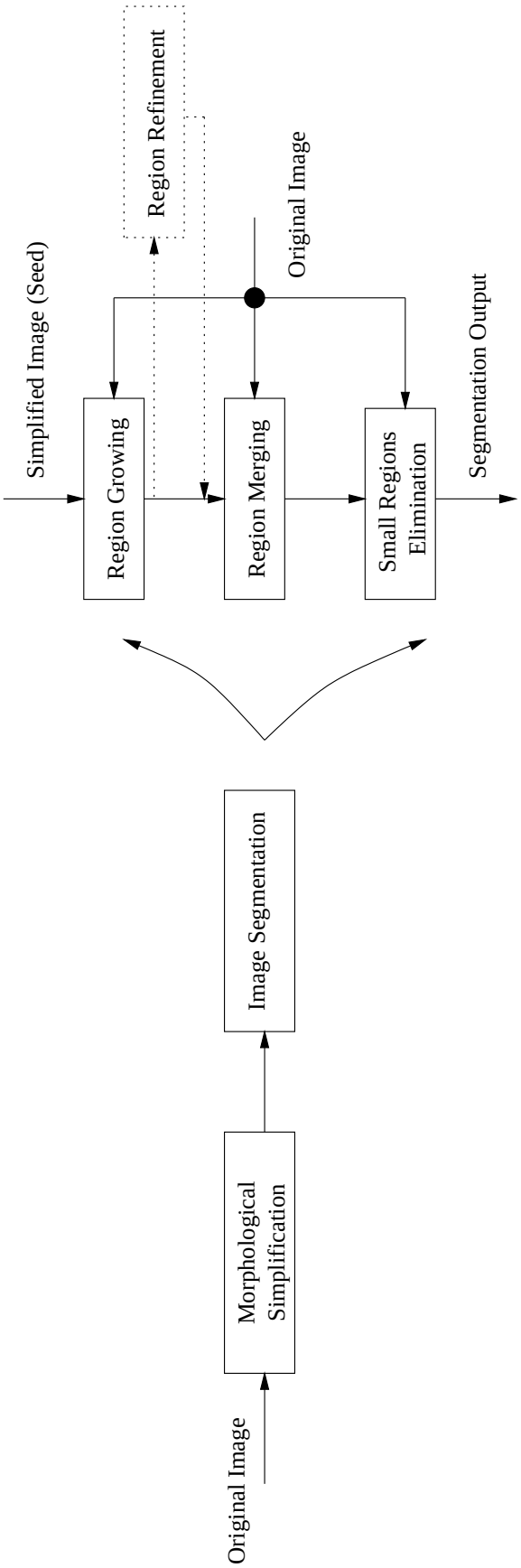


Figure 3.2: Morphological Simplification, Region Splitting & Merging Segmentation Algorithm

איור 3.2:

3.2.1 Morphological Multi-Scale Simplification

The main contribution of this algorithm is the simplification proposed in [14]. The authors point out the drawbacks of conventional morphological filters. First, that the contrast information of image components is not well handled, while to the HVS, a small component with a high contrast may be more important than a large one with a low contrast. Second, that the proper usage of morphological filters resides in the selection of the structuring element. A large structuring element will remove some perceptually sensitive details, while a small structuring element can not sufficiently simplify large image components. Hence a multi-scale simplification scheme is proposed, which will take into consideration not only the size but also contrast information. The multi-scale simplification is performed as follows:

Let three structuring elements B_0, B_1, B_2, satisfying $B_0 \supset B_1 \supset B_2$, and a threshold T be defined. Let $f(r)$ denote the original image which is first filtered using the open-closing by reconstruction with the largest structuring element B_0. In the filtered image, which is denoted by $\Psi(f)$, large homogeneous regions of $f(r)$ are transformed into flat regions and all the image components smaller than B_0 are removed. To simplify the image components which have high contrast and which are slightly smaller than B_0, the simplification residual $f(r) - \Psi(f)$ is first divided into a light residual G_1 and a dark residual G_2:

$$G_1(r) = \begin{cases} [f - \Psi(f)](r), & \text{if } [f - \Psi(f)](r) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$G_2(r) = \begin{cases} [\Psi(f) - f](r), & \text{if } [\Psi(f) - f](r) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

The high contrast components in the two residual images are detected by thresholding with T

$$g_i(x) = \begin{cases} G_i(r), & \text{if } G_i(r) > T \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2 \tag{3.3}$$

$g_i(x)$ are then simplified using the opening by reconstruction in which B_1 is taken as the structuring element and G_i as the reference

$$C_i = \gamma^{(rec)}[\gamma_{B_1}(g_i), G_i], \quad \text{for } i = 1, 2 \tag{3.4}$$

where the notation follows previous definitions in chapter 2, i.e. $\gamma_{B_1}(g_i)$ denotes the opening of g_i with B_1 and $\gamma^{(rec)}(X, Y)$ the reconstruction by dilation of X under reference Y. C_1 and C_2 are then combined with $\Psi(f)$ to improve the simplified image: $f_1 = \Psi(f) + C_1 - C_2$. Afterwards, C_1 and C_2 are respectively subtracted from G_1 and G_2, which produces two new residuals. Equations (3.3) and (3.4) are further applied to these new residuals in order to simplify the image components that are higher than the threshold T, smaller than B_1 but larger than B_2. At this step, B_1 in (3.4) is replaced by B_2, and the new filtered outputs are combined with f_1. The final simplified image can be denoted as $f_2(r)$. We implemented the multi-scale simplification with the values that the author suggested: Threshold T is equal to eight, and B_0, B_1, and B_2 are squares of size 12×12, 6×6 and 3×3, respectively.

3.2.2 Image Segmentation

Refering to the simulation section 3.5, Fig. 3.6, one can see from the simplified image that the multi-scale simplification produces flat zones, which are refered to as seeds in [14]. Thus image segmentation can be performed on this simplified image to define

the region boundary. It consists of four steps: *region growing*, *region refinement*, *region merging*, and *small region elimination*.

- Region Growing

The simplified image $f_2(r)$ contains many flat zones, which identify the inner part that belongs to a certain region, and can serve as seeds for region growing, which in turn finally define the region boundary. It is suggested in [14] that only flat zones that are equal to or larger than 3×3 pixels are candidates for seeds. It is observed that the definition of a "seed" used here coincides with the definition of a "marker" used in the previous algorithm. Hence, it is not hard to conclude that the flat regions labelling technique is also suitable for seed detection, i.e., the algorithm labels all of the flat zones and then deserts all those which are not seed candidates. The region growing procedure is performed by appending a neighboring pixels to a seed region if the absolute difference of gray-level between the pixel and the seed region is less than a specified threshold value. When all the pixels complying with this threshold have been appended, the threshold value is increased and the procedure is repeated until all pixels of the image are assigned.

- Region Refinement

The purpose of region refinement is to produce regions with better homogeneous quality. The segments which result from the region growing step are first approximated by their gray-level means, and forms a reconstructed image $f_3(r)$. If the absolute reconstruction error $|f_3(r) - f_2(r)|$ is lower than threshold value T for every pixel in a region, this region is considered to be homogeneous. Otherwise, inhomogeneous regions exist in the segmentation output, so we lower down the threshold value used

in region growing and redo it. Based on our simulations, this region refinement should be optional, since it may not sufficiently improve the quality of segmentation output, while increasing the computation time.

- Region Merging

The previous region growing procedure typically produces an oversegmentation of the image. A homogeneous region may be split into two or more segments. The step of region merging aims to merge these segments into one region by boundary elimination. Let b denote the boundary which separates regions R_1 and R_2. If the average contrast across the boundary is lower than $T/2$, boundary b is eliminated and regions R_1 and R_2 are merged into one region. Since the order in which regions are merged strongly influences the results of region merging, the region merging is carried out by successively eliminating the boundary which sits between regions with the lowest average contrast.

- Elimination of Small Regions

As we pointed out above, the segmented image given by region growing is over segmented; even after region merging is performed, some small regions may remain in the segmented image because of their high contrast. Consequently, it results in a complicated partition of the image. The complicated segmentation output highly increases the contour and texture coding cost. Hence, small regions should be eliminated in order to further reduce the number of regions. The author in [14], suggests that the threshold value should be set such that any region smaller than 0.04% of the total image area be merged into its most similar adjacent region (adjacent region with lowest contrast difference). After these steps are performed, a comparatively small

number of regions is obtained, and the contours of the segments are more accurately defined than the result from morphological watershed.

3.2.3 Discussion

The above segmentation algorithm is a typical bottom-up structure algorithm. Most of the regions which result from the segmentation are strictly "homogeneous"; that is, have low variation of pixel gray-level within the region. But the way by which human viewer groups pixels into regions does not strictly follow this simple model. At least, according to the several simulation results we got, the algorithm tends to split a region which a viewer would have identified as "homogeneous" into many small fragments. Too much complicated image partition increases the complexity and the cost for contour and texture coding. So, we propose in the next section an algorithm which draws on the above two segmentation algorithms but is better suited to the segmentation-based coding task.

3.3 Edge Detection, Local-Activity-Classification Segmentation Algorithm

Let's restate the image segmentation problem we are confronting. Current segmentation algorithms are basically image dependent. That is, the algorithms proposed are only suitable for some specific kind of images and there is no general acceptable ones. The "homogeneity criterion" used is different from that of a human viewer. In fact, the complexity of the HVS and the diversity of purposes for performing image segmentation make it almost impossible to define a generic criterion to judge the

segmentation result. From a coding point of view, an ideal segmentation algorithm should possess the property that it defines the object boundary as does the HVS. Human viewers tend to be sensitive to the *edge* information in the image, hence the algorithm should be able to group individual pixels into regions which are separated by the boundary defined by the edge. Within the defined region, certain defined local characteristics can be used to describe *homogeneous regions* or *texture regions*. A characteristic of an homogeneous region is: flat areas with very little pixel gray-level variation. Texture regions on the other hand are charactered by areas with frequent pixel gray-level variation, but the patterns of changes within the neighborhoods are similar to each other. Thanks to the progress achieved in relevant fields such as edge detection, methods are available for defining the local activity of each pixel. This can serve as a clue for clustering pixels having similar characteristics together. Our method of defining the local activity is based on [24] and is described next.

3.3.1 Local-Activity Classification

A modification of the Prewit operator has been employed in [24] to determine the local activity of a pixel under consideration. The four operator matrices used are depicted in Fig. 3.3.

The advantage of these operators is that all elements are either equal to 0, +1 or −1, so that only additions are required for the calculation. Let $f(x, y)$ be the gray-level at position (x, y) and let $p_{i,j}(k)$, $i, j = 1, \ldots, 5$; $k = 1, \ldots, 4$, be the elements of the operator matrices $p(k)$. The local activity, $A_{x,y}$, at the position (x, y) is then calculated by the following formula:

$$A_{x,y}(k) = \sum_{i=1}^{5} \sum_{j=1}^{5} f(x-3+i, y-3+j) p_{(i,j)}(k) \tag{3.5}$$

P(1)

0	0	0	0	0
0	1	0	-1	0
0	1	0	-1	0
0	1	0	-1	0
0	0	0	0	0

P(2)

0	0	1	0	0
0	1	0	0	0
1	0	0	0	-1
0	0	0	-1	0
0	0	-1	0	0

P(3)

0	0	0	0	0
0	1	1	1	0
0	0	0	0	0
0	-1	-1	-1	0
0	0	0	0	0

P(4)

0	0	1	0	0
0	0	0	1	0
-1	0	0	0	1
0	-1	0	0	0
0	0	-1	0	0

Figure 3.3: Operators for Local Activity Classification
איור 3.3: אופרטורים לסיווג פעילות מקומית

$$A_{x,y} = \max_{k=1,\dots,4}(|A_{x,y}(k)|) \tag{3.6}$$

Depending on the value of $A_{x,y}$, each pixel will be associated to one of four activity classes:

class 1: $0 \leq A_{x,y} < 32$: flat areas,

class 2: $32 \leq A_{x,y} < 64$: low structured regions,

class 3: $64 \leq A_{x,y} < 128$: high structured regions,

class 4: $128 \leq A_{x,y}$: edges.

We define *flat areas* as *homogeneous regions*, and the *structured areas* are *texture regions*. Hence, following this processing, a set of homogeneous regions and texture regions are formed. Meanwhile, new problems arise because of this rough partition. First, there may exist homogeneous areas inside texture regions; therefore, many pixels in class 1 can be found in structured regions. Second, the problem we confront is how to assign pixels classified as an edge to the regions they actually belong to. We were inspired to embed this scheme as a preprocessing tool for the original image simplification. The proposed algorithm is described in the next subsection. It still has a top-down structure, but different operations are inserted to improve the system performance.

3.3.2 Proposed Segmentation Algorithm

The proposed algorithm, depicted in Fig. 3.4, has the following steps:

- Simplification

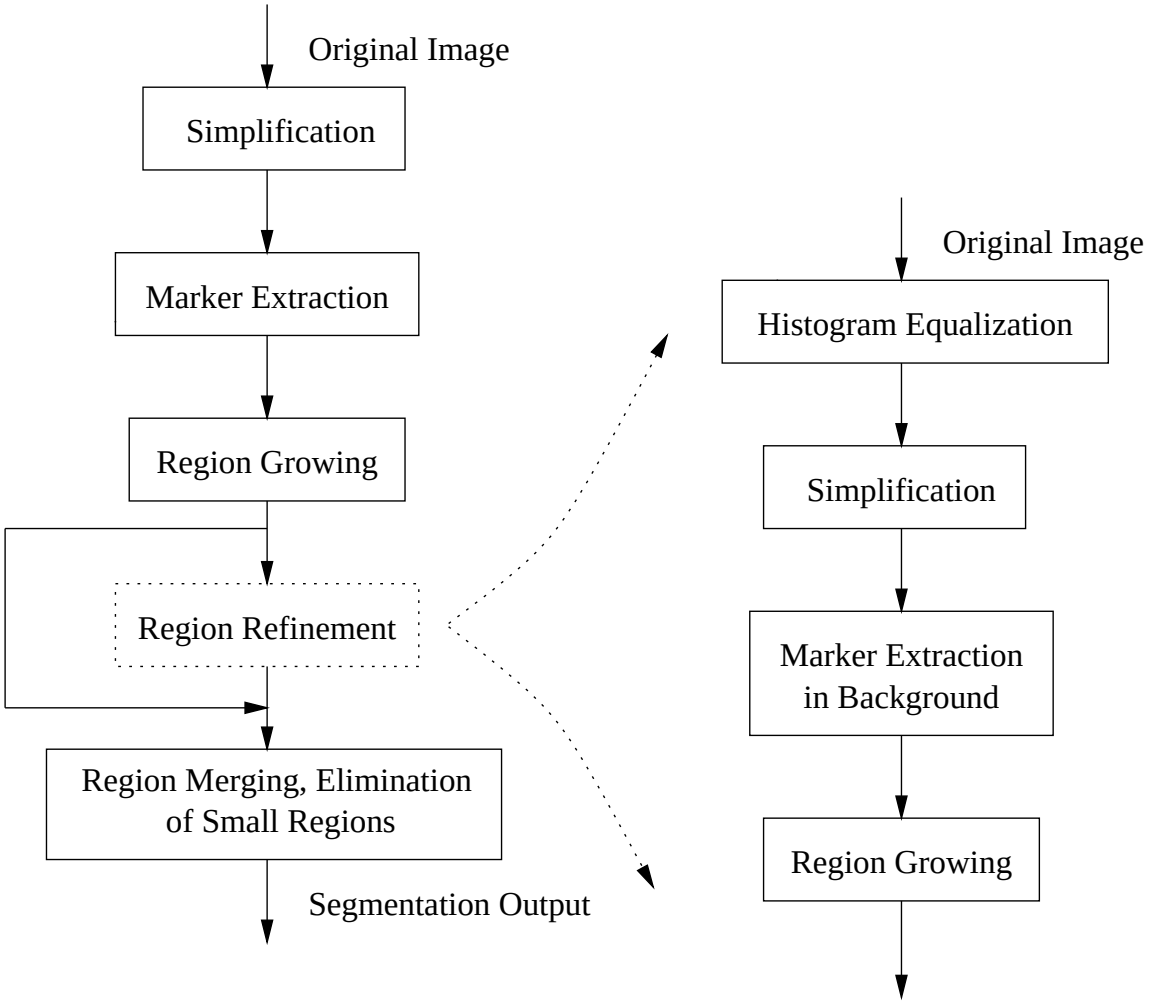


Figure 3.4: Edge Detection, Local-Activity Classification Segmentation Algorithm

איור 3.4: אלגוריתם סגמנטציה המבוסס על גלוי קצוות וסיווג פעילות מקומית

Three operations are executed in this step. The original image is first filtered by morphological opening-closing by reconstruction filters to remove noise and less important details. Then, the pixelwise local activity is calculated based on the simplified image; a new image, which may be called the *characteristic image*, and has only four gray-level values (We have used the values 255, 180, 100 and 0), to indicate the pixel local characteristic, is generated. This characteristic image is further filtered by morphological opening-closing by reconstruction filters with a smaller size, and the final output, the simplified characteristic image, can be used for marker extraction. Empirical results indicate that the size of the morphological filters can be chosen to be 5×5 in the initial step and 3×3 in the following step.

- Marker extraction

Referring to the demonstration in the simulation result (Fig. 3.7 in section 3.5), within the simplified characteristic image, a smaller number of regions having large size (flat zones) are produced. The boundaries of the regions are well defined, so that we can use the discussed marker extraction technique to detect the inner part of the regions, shown as white zones in Fig. 3.7.

- Region Growing

Because of the deficiency of the watershed algorithm as discussed earlier in section 3.1, a region growing algorithm is used as a decision tool instead. The algorithm implementation was described earlier in section 3.2. As seen in the demonstration in Fig. 3.7, the input image is partitioned into segments with quite precisely defined boundaries.

- Region Refinement

Adjacent segments with a perceived difference along their boundary are easily identified after the region growing. In some cases, however, the contrast between adjacent segments is not big enough to be identified, and it is therefore recommended to apply region refinement to this type of images. In the simplification step, because of their comparatively small size or low contrast, some important details could be eliminated and merged into a big segment, which in a typical situation is the background of the image. The purpose of region refinement is to retrieve these detail from the big segments or background. The refinement step consists of following sub-steps:

1. Histogram equalization [25]: To improve the contrast difference between regions with poor contrast.

2. Simplification: The processed image is simplified, as in the previous simplification step, with reduced size morphological filters.

3. Marker Extraction (in Background): Within a background region, as shown in Fig. 3.7, two kinds of markers are generated. One consists of the white flat zones mentioned above, which indicate homogeneous areas. The other consists of gray flat zones, which indicate texture areas. Thus, the marker extraction technique should be used to identify these two kinds of markers. The histogram equalization introduced helps in the generation of markers which identify the texture regions.

4. Region Growing: Region growing is redone to redefine the contours of the regions.

- Region Merging & Elimination of Small Regions

The purpose of this step is quite similar to the one described in section 3.2. Even though the number of segments generated is largely reduced, oversegmentation still exists. That is, a segmentation which visually appears as a single segment is split into several fragments. Thus, merging is performed according to the region merging scheme described in section 3.2. Due to contour simplification, a step which we describe later on, small regions as well as long narrow regions may result. Since from a coding point view, the existence of these regions just increases the coding cost, these regions are merged to their lowest contrast neighbors, the procedure is identical to the one described previously.

3.3.3 Further Discussion

Because of time limitation, some further study could not be carried out to improve this algorithm. The decision of whether to introduce the region refinement step for further segmentation is done by subjective judgment after the initial segmentation. We hope that an objective criterion could be found in the future to decide whether region refinement is needed. Besides, the region refinement itself could be further exploited. In our scheme, the small regions or low contrast regions are assumed to be embedded in the background which, in general, is the biggest segment obtained after the first level of the region growing step. But further studies are needed to devise a scheme which will detect other regions which need to be split further. Moreover, histogram equalization increase the whole system's sensitivity to noise, the question of denoising, is another problem one needs to cope with. If these problems are successfully solved, the proposed algorithm can be a promising candidate for a generic image segmentation algorithm needed for our coding application.

3.4 Contour Simplification Using a Majority Filter

The purpose of contour simplification is to reduce the number of contour points. As we stated before, the contour coding cost is quite considerable. Thus, one needs simple and short contours to save bits. On the other hand, contours represent the most sensitive information for the human visual system and should be described precisely. The segmentation algorithms we mentioned before do not have means for controlling the complexity of the contours produced. Due to the fact that various noises exist in natural images, the partition is also quite noisy. Hence, another issue, contour simplification is introduced, which aims to filter the noise and decrease the complexity of the image partition without severely deteriorating the original contours.

Classical methods for contour simplification include Phagocyte heuristic, Fourier descriptors, curve filtering by estimation of abscissa, geometrical curve approximation and morphological contour simplification. Generally, there are different drawbacks in these contour simplification tools as discussed in [20].

The contour simplification tool we used in the proposed segmentation algorithm is a majority filter based on [26], which is performed on the partition image. This nonlinear filter is designed to remove very small isolated regions and also to smooth all the boundaries with "sparks", i.e., rugged boundaries. The filter has the property that the filtered partition image is independent of the assignment of the labels in the partition image. It only relies on the shape of the contours. The structure of this majority filter is as follows:

Assume point x is the pixel under consideration, structuring element B indicates an area surrounding point x. Assume that there are k different labels $(L_1, L_2, \ldots, L_k)$

in the area contained by B. Each label L_i is associated with N_i pixels in B respectively. The original label in point x is $P(x)$. The *majority operator* M is defined as:

$$M(P(x)) = \begin{cases} L_i & N_i > N_j, 1 \leq i, j \leq k, j \neq i \\ P(x) & \text{otherwise} \end{cases} \tag{3.7}$$

That is, label the pixel under consideration with the majority label within the neighborhood; and keep its original label in case of equality. The *majority filter* σ_B is now defined as:

$$\sigma_B = M^{\infty}(P(x)) \tag{3.8}$$

Although the majority filter is defined as an infinitely repeated process, convergence is usually reached after just a few iterations. The size and shape of the structuring element B can be variable which provides a possible multi-scale simplification for different levels of smoothness. The author in[26] suggested a structuring element of 3×3 cross or square shaped. In our application, a 5×5 square shaped has also been used in some cases (refer to the simulation results in Fig 3.8).

Spark contours will disappear which results in a more pleasant smoothed contour image. Sometimes, small isolated regions or long narrow regions may be generated because of the contour simplification, and they are merged to their closest neighbors (as mentioned in section 3.3).

A more recent algorithm is proposed by V.A. Christopoulus et al. in [27]. It performs the simplification on the contour image instead of the traditional partition (label) image. As discussed in [27], this scheme requires less computation time than the majority filter.

3.5 Simulation Results

In this section, we illustrate simulation results obtained with the described algorithms. We also discuss the advantages and disadvantages of different methods, along with the practical problems that we encountered in our simulation.

We first present an example obtained from the first level segmentation of the Hierarchical Morphological Synthesis by Analysis Segmentation Algorithm of section 3.1. Looking at the simplified image in Fig. 3.5, it is obvious that large flat zones were produced, because a big size open-close by reconstruction filter is used. Using the marker extraction technique to detect these flat zones, eleven markers are identified. Finally, the improved version of watershed is performed to define the boundary of the regions. Problems still exist in the improved watershed. Implemented by a queue structure, whenever a pixel is extracted from the queue, its undecided neighbors need to be inserted into the queue, and the queue needs to be resorted. The procedure takes a considerable amount of time. Together with other problems which exist in the algorithm, such as false contours generated and huge computation time for simplification. This algorithm was finally abolished in our application, because of these shortcomings.

In the results from the second segmentation algorithm – Morphological Simplification, Region Splitting and Merging Segmentation Algorithm, one can see (Fig. 3.6) flat zones which are generated in the inner part of high contrast regions. The algorithm also enjoys the advantage of requiring less computations. But, unfortunately, the innate drawback of the algorithm is that the segmented image output is severely oversegmented. In the medical image shown in Fig. 3.6, it partitions a big segment into many small pieces, while for a HVS-like segmentation, one would like to keep the

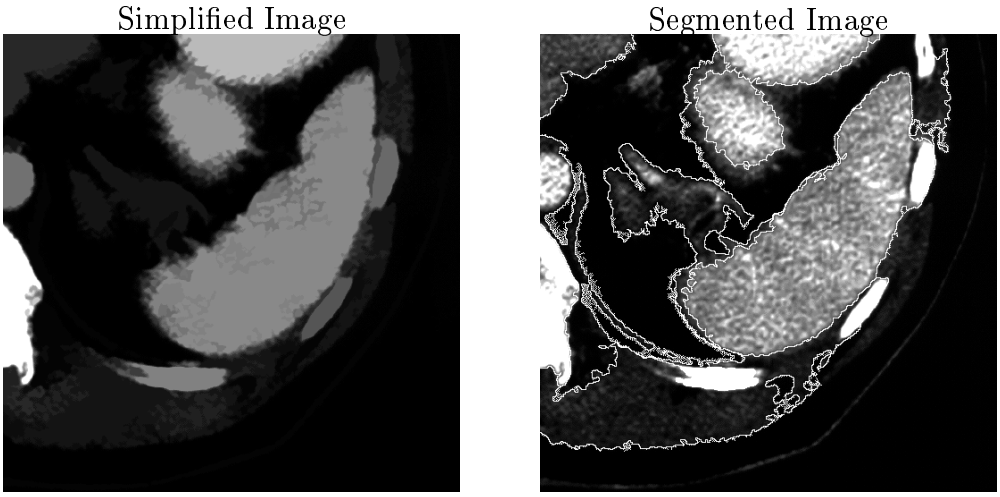


Figure 3.5: Segmentation Output of Hierarchical Morphological Synthesis by Analysis Segmentation Algorithm

איור 3.5: תוצאות סימולציה באמצעות אלגוריתם סגמנטציה מורפולוגי היררכי המבוסס על סינתזה ע"י אנליזה

whole thing as one segment of textured area. Even in the "Peppers" image (Fig. 3.6), while human viewers consider the original image as consisting of big flat homogeneous regions, the homogeneous regions are split into many pieces. So it does not seem that this algorithm is suitable for our application.

More acceptable results were obtained with our proposed Edge detection, Local-Activity-Classification segmentation algorithm. Adjacent regions are separated, if steep edges exist between them, while regions with similar characteristic are clustered into one segment. The advantages of the resulting partitions are: First, regions with similar characteristic are grouped into one, so that the following shape adaptive coding can fully exploit the specific characteristic of the segment to achieve better compression. Second, the contours generated in this algorithm are relatively simple, therefore the contour coding cost is reduced as we mentioned before. The examples

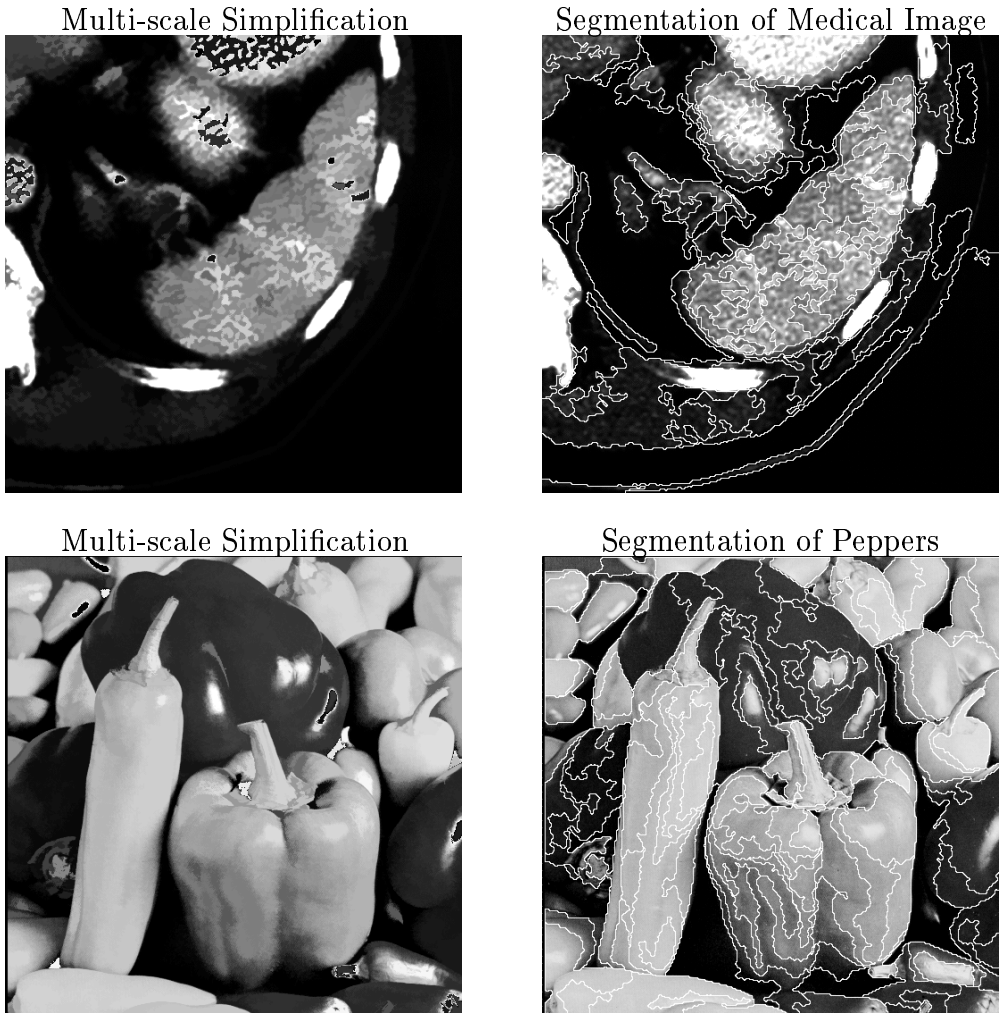


Figure 3.6: Segmentation Output of Morphological Simplification, Region Splitting & Merging Segmentation Algorithm

איור 3.6: תוצאות סימולציה באמצעות אלגוריתם המבוסס על פישוט מורפולוגי, פיצול ומיזוג אזורים

in Fig. 3.7 to Fig. 3.10 show that this algorithm matches a wide range of images, besides the advantages of a well defined segmentation output and a comparatively low computational load. We demonstrate now the complete processing procedure of the "Bike" image (Fig. 3.7). After the original image is simplified by the proposed procedure, white flat regions are produced to indicate the inner part of the regions. Most of the contours to which the HVS is sensitive to are identified after region growing. Region refinement is performed to further partition the roughly segmented regions. The final segmentation output is the refined segmentation filtered by a 3×3 majority filter. An additional segmented "House" image is shown in Fig. 3.8, on the left is the image obtained following segmentation without contour simplification. On the right the final segmentation result is shown. The majority filter used in this case is of size 5×5, and one can well see that smoother contours are generated because of the larger filter size. Some other segmentation results are shown in Fig. 3.9 and Fig. 3.10. Except for the medical image, in which the contour simplification is performed by a 5×5 majority filter, all the other contour simplifications are performed by a 3×3 majority filters.

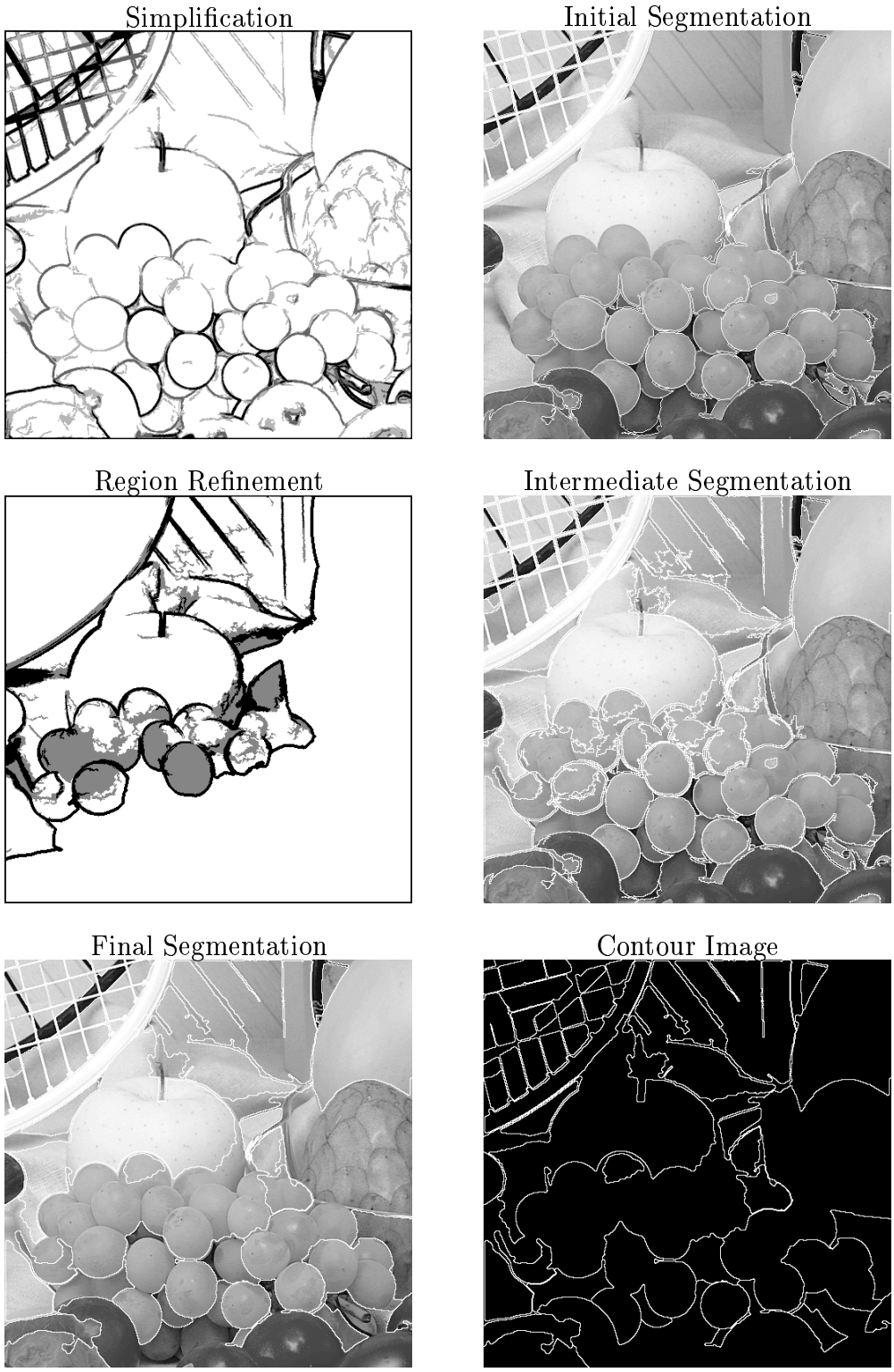


Figure 3.7: Segmentation Results of the Proposed Algorithm for the Image "Bike"
"Bike" איור 3.7: תוצאות סימולציה באמצעות האלגוריתם המוצע של התמונה

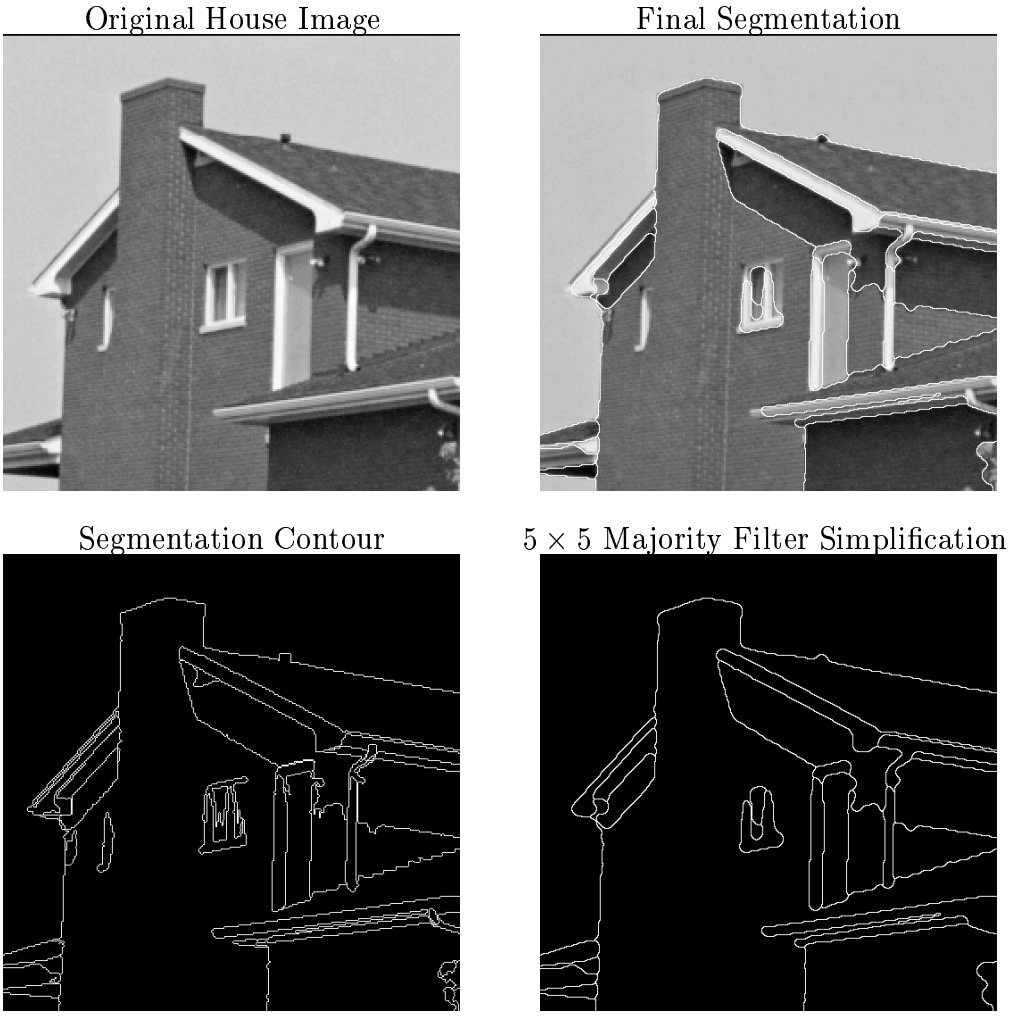


Figure 3.8: Segmentation Results of the Proposed Algorithm for the Image "House"
"House" איור 3.8: תוצאות סימולציה באמצעות האלגוריתם המוצע של התמונה

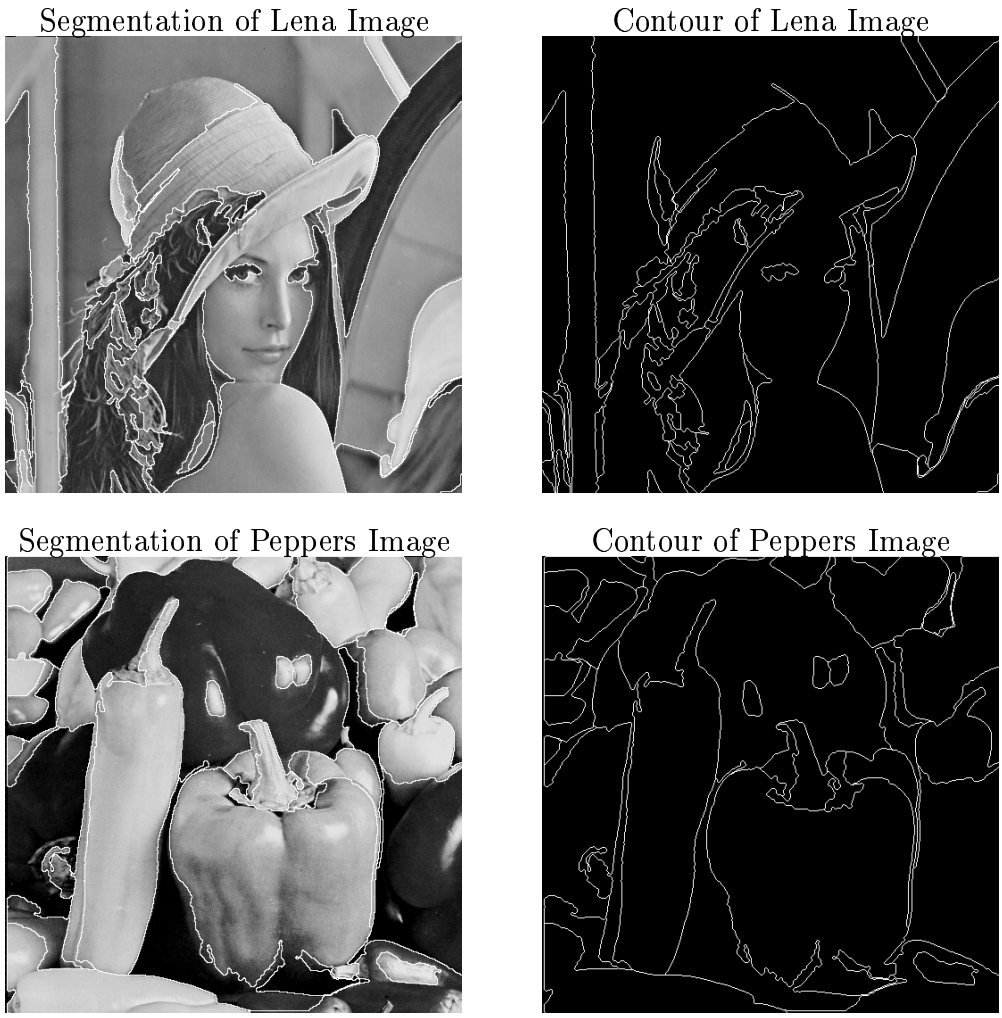


Figure 3.9: Segmentation Results of the Proposed Algorithm for the Images "Lena" and "Peppers"

איור 3.9: תוצאות סימולציה באמצעות האלגוריתם המוצע של התמונות "Lena" ו"Peppers"

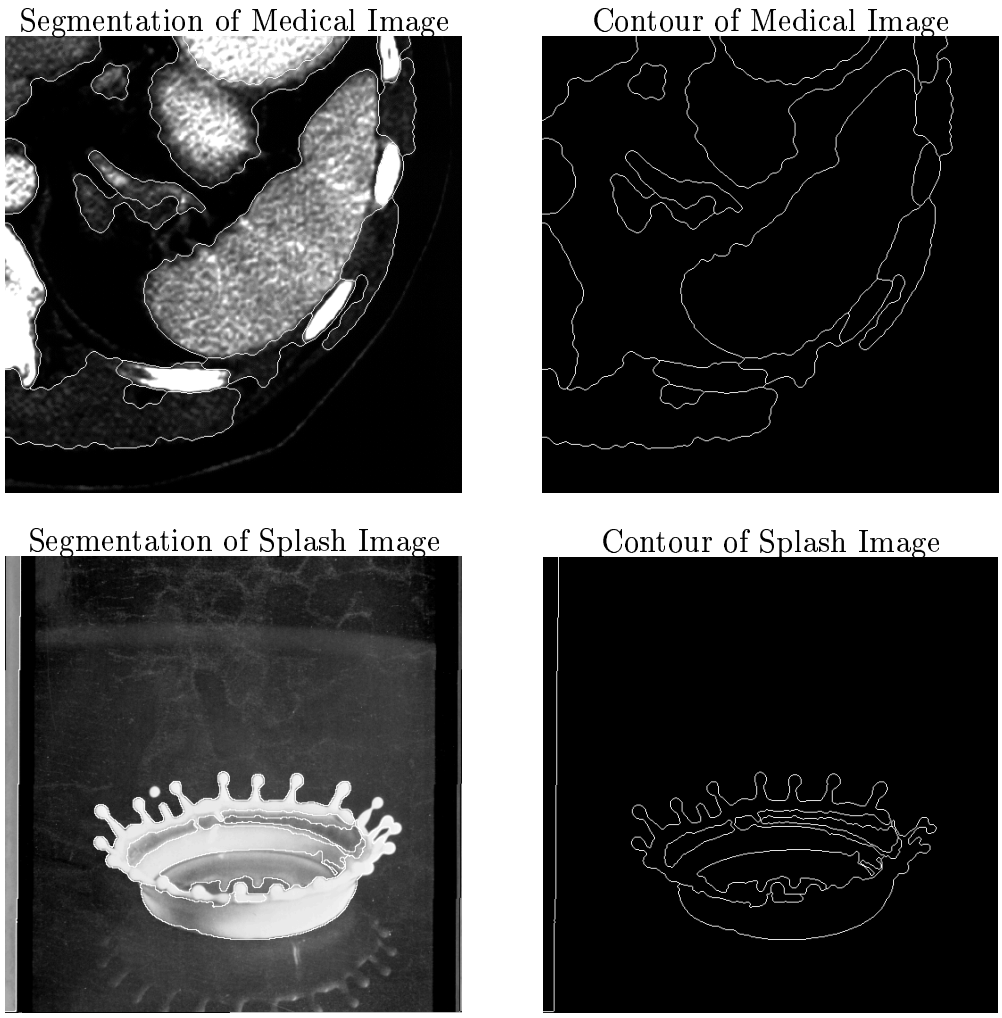


Figure 3.10: Segmentation Results of the Proposed Algorithm for the Images "Medical" and "Splash"

"Medcal" איור 3.10: תוצאות סימולציה באמצעות האלגוריתם המוצע של התמונות ו"Splash"

Chapter 4

Block-Based Shape-Adaptive Coding Techniques

The following two chapters cover various shape adaptive coding methods that were investigated in this work. The purpose of this chapter is to describe several block-based algorithms. In particular, Shape-Adaptive DCT (SADCT) [28, 7], Low Pass Extrapolation (LPE) padding [7], and our proposed optimal block extrapolation method (in a sense to be defined). A comparison of these coders to conventional block-based 2D-DCT transform coding, such as JPEG, is inevitable. The readers may refer to Appendix A or reference [29] for the JPEG coder details.

4.1 Low-Pass Extrapolation (LPE) Padding

Previously, we have discussed the purpose of segmentation-based image coding techniques. In order to encode the content of segments efficiently, different approaches have been exploited. Considering the problem of compatibility of potential coders

with current prominent JPEG (or MPEG, H.26x) coders, some researchers proposed extension of normal 2D 8×8 block DCT transform coding. *Shape-Adaptive DCT* (SADCT) and *Low-Pass Extrapolation* (LPE) Padding Algorithms are the most competent candidates in this context. Both of these algorithms have the advantage of low computation complexity (especially LPE, while SADCT has the advantage of higher compression ability in some applications). In a recent MPEG-4 release [7], both LPE padding and SADCT are proposed for intra mode texture coding.

To simplify the explanation, just one segment out of the segmented image is considered (this coincides with the notion of *foreground segment* (video object) in a video sequence coding application). The input source image is partitioned into 8×8 blocks as usual. As shown in Fig. 4.1, the partition results in three kinds of blocks:

1. Those that lie completely inside the considered segment – *inner blocks.*
2. Those that lie on the boundary of the segment, so that only part of the block belongs to the segment – *boundary blocks.*
3. Blocks which lie outside the segment – *outer blocks*

Only the coding of the first two kinds of blocks need to be considered for the current segment. Empirical results indicate that human vision is more sensitive to distortion along the segment boundary than to segment content [2]. For these blocks, the assumption that data samples vary slowly within the block is no longer valid, because of the presence of an edge. Consequently, JPEG type coders are usually inefficient in encoding such blocks. On the other hand, smooth inner blocks can still be coded efficiently with normal 8×8 2D-DCT. LPE padding was proposed to reduce the number of bits needed for coding boundary blocks without highly increasing the computation complexity of the system. The basic idea is to complete the part of the

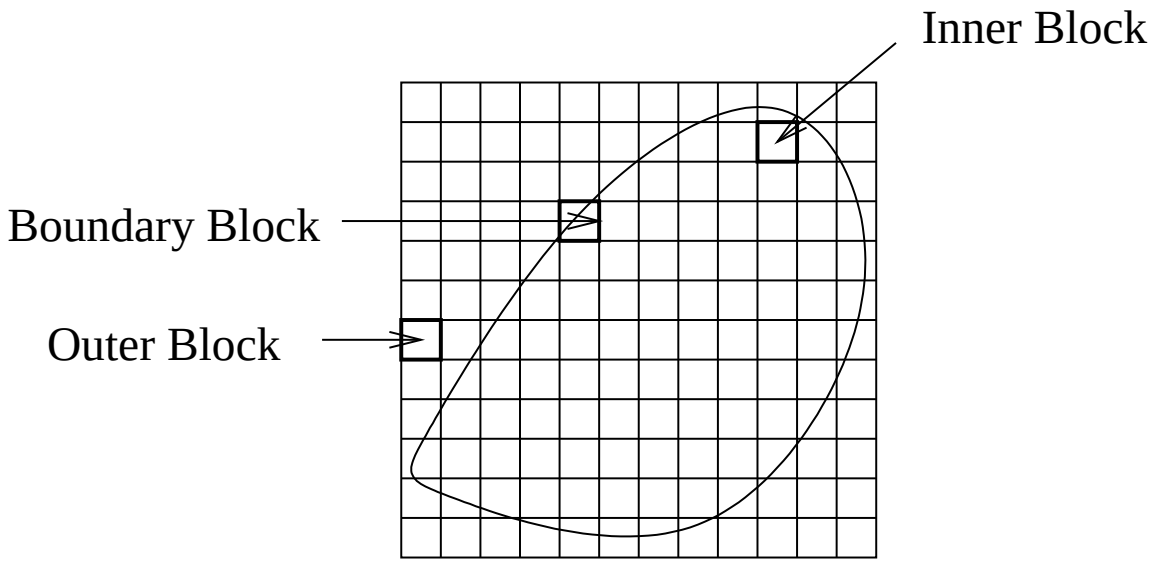


Figure 4.1: Block Classification
איור 4.1: סיווג בלוקים

boundary block which is outside the segment by pixel values which will bring about a reduction in the number of bits needed to code the block. A simple approach is to select a fixed gray-scale value such as 0, or the mean of the part of the block in the segment. Instead, LPE padding completes the outside part by performing the following steps for an 8×8 block:

1. Calculate the arithmetic mean value m of all those pixels (i, j) of a boundary block B, which are situated within the segment R, i.e., $(i, j) \in B \bigcap R$

$$m = (1/N) \sum_{(i,j) \in B \bigcap R} f(i, j) \tag{4.1}$$

   where N is the number of pixels located in $B \bigcap R$. Division by N is done by rounding to the nearest integer.

2. Assign m to each pixel in block B which lies outside of the region R, i.e.,

$f(i,j) = m$, $(i,j) \in B$ and $(i,j) \ni R$

3. Apply the following filtering operation to each pixel in block B outside of R: Starting from the top left corner of the block and proceeding row by row to the bottome right pixel, the value of $f(i,j)$ is replaced by

$$f(i,j) = [f(i,j-1) + f(i-1,j) + f(i,j+1) + f(i+1,j)]/4 \tag{4.2}$$

Division is done by rounding to the nearest integer.

If one or more of the four pixels used for filtering are outside of the block, the corresponding pixels are not included in the filtering operation and the division by 4 is changed accordingly. Consider, e.g. $i = 0$ and $j = 0$, in which case the above operation becomes

$$f(i,j) = [f(i,j+1) + f(i+1,j)]/2 \tag{4.3}$$

Step 3 is repeated until there is no data change is caused by one more step of iteration.

After this padding operation the resulting block is ready for DCT coding. We should notice that the LPE padding process is only performed at the encoder. On the decoder side, the extrapolated data should be discarded, which is easily done, since the shape information is already known. Obviously, the LPE padding method is a modified version of the simpler mean padding, expected to bring about a further reduction in bit-rate.

4.2 Shape-Adaptive DCT (SADCT)

Another approach to boundary block coding is *Shape-adaptive* DCT (SADCT), where SADCT basis functions are predefined orthogonal sets of DCT basis functions. An

illustration of the algorithm is shown in Fig. 4.2. The dark region within the 8×8 block, indicates pixels within the segment (Fig. 4.2.(a)), and we define these pixels as *inner pixels*. The algorithm consists of *vertical* DCT and *horizontal* DCT calculations.

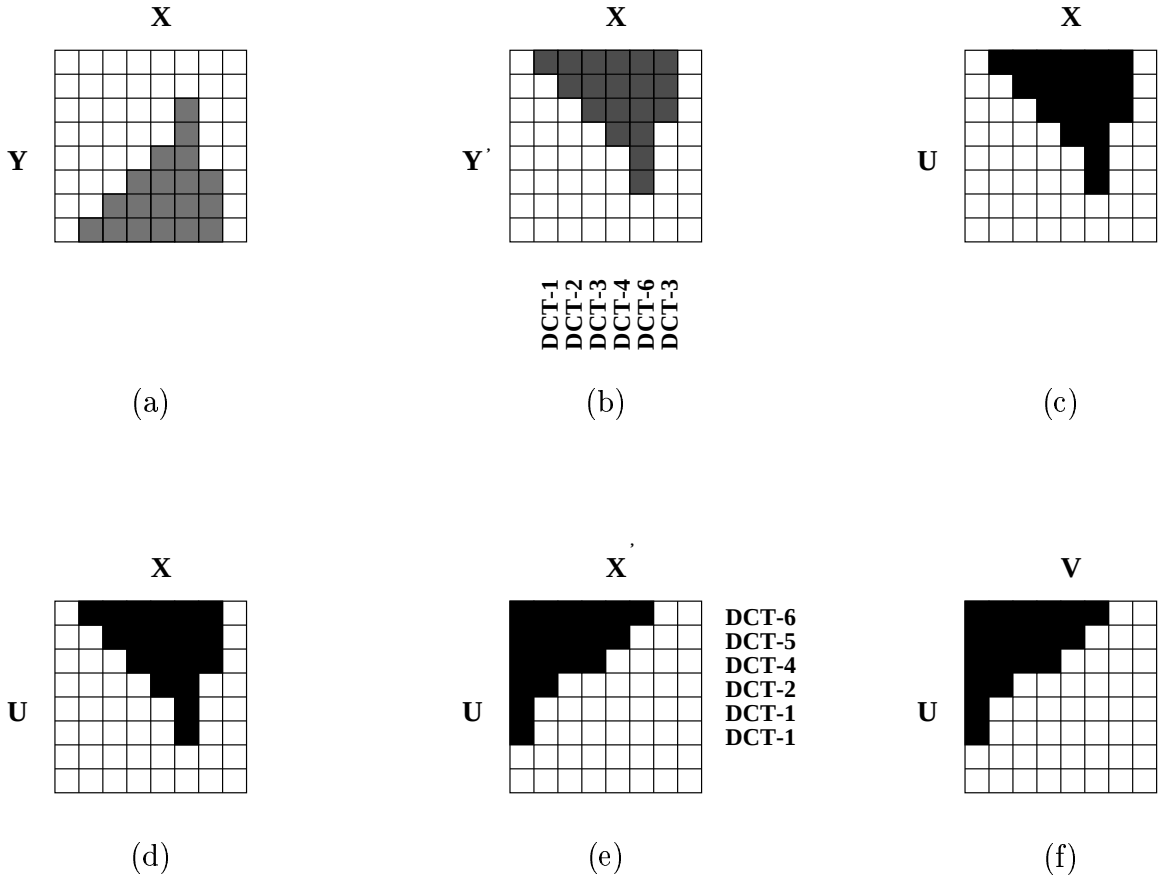


Figure 4.2: Successive Steps Involved in Performing SADCT on a Boundary Block
איור 4.2: רצף הפעולות הדרושות לביצוע SADCT על בלוק קצה

To perform a vertical SADCT transformation, the number of inner pixels contained in each column j $(1 \leq j \leq 8)$ is calculated, where the vector size is denoted by N, $1 \leq N \leq 8$, and the columns $\underline{x}_j$ $(1 \leq j \leq 8)$ are shifted and aligned to the upper border of the 8×8 reference block (Fig. 4.2.(b)). Depending on the vector size, N,

of each particular column, a DCT transform matrix DCT_N,

$$DCT\_N(p,k) = c_0 \cdot \cos\left[p(k+\frac{1}{2}) \cdot \frac{\pi}{N}\right], \quad 0 \leq k, p \leq N-1 \tag{4.4}$$

containing a set of N basis vectors is generated, where p denotes the pth DCT basis vector, and $c_0 = \begin{cases} \sqrt{1/2} & p = 0, \\ 1 & \text{otherwise} \end{cases}$.

The N vertical DCT_N coefficients $\underline{c}_j$ for each column $\underline{x}_j$ are calculated according to the formula:

$$\underline{c}_j = (\frac{2}{N}) \cdot DCT\_N \cdot \underline{x}_j \tag{4.5}$$

For example, in Fig. 4.2.(b) the right most column is transformed using DCT_3 basis vectors. After the SADCT transformation is calculated in the vertical direction, the lowest DCT coefficients for all the columns are positioned along the upper border of the 8×8 reference block. To perform the horizontal DCT transformation (Fig. 4.2.(e)), the length of each row is calculated, and the rows are shifted to the left border of the 8×8 reference block. A horizontal DCT adapted to the size of each row is then calculated using the same formulae (4.4) and (4.5). Note that horizontal SADCT transformation is performed along vertical SADCT coefficients with the same index (e.g., all vertical DC coefficients are grouped together and are SADCT transformed in the horizontal dimension). Fig. 4.2.(f) shows the final location of the resulting DCT coefficients in the 8×8 image block.

Since the contour of the segment is transmitted to the receiver, the decoder can perform the shape-adaptive inverse DCT in reverse order. The inverse transform matrix is DCT_N^T, and the inverse horizontal DCT is calculated by:

$$\underline{x}_j = DCT\_N^T \cdot \underline{c}_j \tag{4.6}$$

The output $\underline{x}_j$ are the vertical DCT coefficients. The reverse coefficient-shift operations are conducted, so that all the vertical DCT coefficients are placed at the proper positions (Fig. 4.2.(d)). Inverse vertical transform is conducted in the same manner as the inverse horizontal transform, with the output as the reconstructed data of SADCT. A reverse shift procedure to move these data to the original positions they belong to.

An example of SADCT transformation is illustrated in section 4.4, where various methods are discussed. In the remaining of this section, we discuss some characteristics of the SADCT.

First, in this algorithm, the final number of DCT coefficients is identical to the number of pixels contained in the image segment. In addition, the coefficients are located in comparable positions as in a standard 8×8 block. The DC coefficient is located in the upper left corner of the reference block, and, depending on the actual shape of the segment, the remaining coefficients are concentrated around the DC coefficient.

Second, the complexity of SADCT is moderately low compared with another shape adaptive coding method, BP, which will be described later. However, SADCT is not always efficient in representing boundary blocks, as discussed below.

As we pointed out before, it is well known that the DCT achieves a compact representation for highly correlated signals and especially, its performance is close to the performance of the optimum transform KL transform in most image coding applications – the fact that has made the DCT favorable for still image and video coding. Thus, the SADCT transformation in vertical dimension outlined in Fig. 4.2.(b) is expected to result in a compact vertical representation of the segment data. The

horizontal SADCT in Fig. 4.2.(e) is performed by grouping all vertical DCT coefficients with the same index (e.g., all vertical DC coefficients $c_1(0)$, $c_2(0)$,...) into one vector. DCT_N transformation is then performed using the coefficients contained in this vector and also achieves a compact DCT-domain representation of the signal in horizontal dimension. However, the efficient representation of the original 2D signal is valid only if horizontal SADCT is performed along the vertical SADCT coefficients that are highly correlated. In other word, the covariance $\sigma^2(c_l(j), c_k(i)) = [E\{\underline{c}_l \underline{c}_k^T\}]_{i,j}$ between the ith vertical DCT coefficient $c_k(i)$ of a column k and the jth vertical DCT coefficient $c_l(j)$ of a column l should be higher for $i = j$ (same index) than for $i \neq j$ (different index). This is the case for "smooth" segment borders. Experimental results indicate that the SADCT method as proposed in Fig. 4.2.(e), which performs horizontal transformation on vertical DCT coefficients with the same index, is a valid efficient procedure and that reasonable performance can be expected as long as the sizes of adjacent columns within a segment are not drastically different ("smooth" segment border). On the other hand, if the segment does not have this smooth characteristic, SADCT is not a competent candidate for boundary block coding. Another problem we may encounter is dealing with non-contiguous object pixels (e.g., holes within a segment). In this case, it is recommended to proceed as described in Fig. 4.2.(b) and Fig. 4.2.(e) by shifting all row or column pixels to the relevant border of the 8×8 reference block. Non-contiguities are eliminated by aligning all row or column pixels next to each other. Obviously, the procedure may cause severe distortions of correlation properties between pixels within the segments. Therefore, even though non-contiguous segments do not frequently occur in our segmentation output, the performance of SADCT may be degraded because of their existence.

In spite of the fact that the separation of the transform into two 1D-DCT with

vector shifts degrades the coding performance of SADCT, some references related to MPEG-4 [30] claim that compared to padding, this method gains between 1 dB and 3 dB in PSNR, measured over the boundary blocks for the same bit rate (with an increased complexity as a tradeoff). In our experiments, no such advantage was achieved. The reason could be that MPEG-4 applies an *adaptive scanning* technique of the non-zero coefficients, and uses a modified VLC table (modified to suit this change). In addition, in MPEG-4, mainly foreground segment (video object) coding is considered. Therefore, the application is different from ours. Our implementation of the SADCT coding system was done according to the proposed scheme in [6]. The main advantage of LPE padding and SADCT methods in shape-adaptive coding is their portability with existing DPCM/DCT hybrid coding system such as JPEG, MPEG (intra mode) and H.26x. In fact, SADCT and LPE padding methods were not only chosen to be standard texture coding techniques in MPEG-4 intra-mode coding, but also in inter-mode (coding inter-image differences), with a slight difference in definition. Since video inter-mode coding is not our goal, we do not discuss this aspect here.

4.3 Optimal Approaches to Data Extrapolation

The advantage of the LPE padding method is its simplicity. In the case of a boundary block, it fills in extrapolated data outside of the segment in a simple way. At the same time, DCT coefficients distribution in the frequency domain is more compact. What is in question is its efficiency as a data extrapolation method. In this section, we discuss different proposals to data extrapolation and introduce our version of data extrapolation, which is optimal in a sense to be defined.

4.3.1 Problem Statement

To simplify the explanation, a one dimensional signal is considered. The extension from a 1D signal to a 2D signal, which is an image block in the current application and an image segment in a later application, will be discussed later on.

It is well known that a 1D-DCT can be calculated by matrix multiplication.

$$\underline{a} = [C_{N\times N}]\underline{s} \tag{4.7}$$

where the $N \times 1$ column vector $\underline{s}$ denotes the input signal vector, and $\underline{a}$ denotes a $N \times 1$ vector of DCT coefficients. The cosine transform matrix $C_{N\times N}$ is defined by:

$$c(k,n) = \begin{cases} \frac{1}{\sqrt{N}} & k = 0, 0 \le n \le N-1 \\ \sqrt{\frac{2}{N}} \cos \frac{\pi(2n+1)k}{2N} & 1 \le k \le N-1, 0 \le n \le N-1 \end{cases} \tag{4.8}$$

The cosine transform matrix $C_{N\times N}$ is an orthonormal real matrix, that is, $CC^T = I$ (superscript ^{T} denotes transpose, and I denotes the identity matrix).

We can even consider a more general transform matrix, $\Psi_{N\times N}$, which we assume to be a real invertible matrix with an inverse matrix denoted as $\Phi_{N\times N}$. Under this premise, a unique solution, $\underline{s} = [\Phi_{N\times N}]\underline{a}$, exists for the set of equations $\underline{a} = [\Psi_{N\times N}]\underline{s}$. In our application, we wish to extrapolate unknown elements in $\underline{s}$ from the known elements in $\underline{s}$. Without loss of generality, we assume that

$$\underline{s} = \begin{bmatrix} \underline{\beta}_{m\times 1} \\ \underline{\theta}_{(N-m)\times 1} \end{bmatrix} \tag{4.9}$$

where $\underline{\beta}_{m\times 1}$ denotes the part of known data, and $\underline{\theta}_{(N-m)\times 1}$ denotes the remaining unknown data in $\underline{s}$. The task of extrapolation is to estimate the unknown data

$\underline{\theta}_{(N-m)\times 1}$, such that

$$\begin{bmatrix} \Phi' \\ \Phi'' \end{bmatrix}_{N\times N} \underline{a} = \begin{bmatrix} \underline{\beta} \\ \underline{\theta} \end{bmatrix}_{N\times 1} \tag{4.10}$$

Here the inverse transform matrix $\Phi = \begin{bmatrix} \Phi'_{m\times N} \\ \Phi''_{(N-m)\times N} \end{bmatrix}$, and $\underline{\theta} = \Phi''\underline{a}$. Since the only known data is $\underline{\beta}$, the system of equations that we actually want to solve (to find $\underline{a}$) is:

$$\Phi'\underline{a} = \underline{\beta} \tag{4.11}$$

Note that since Φ' (of size $m \times N$, $N > m$) is not a full rank matrix, the solution for $\underline{a}$ is nonunique. Constraints to be satisfied by the solution should be added for a unique representation of $\underline{a}$. Turning to available solutions of constrained optimization problems, we investigated the following two methods.

4.3.2 Method of Frames

The *Method of Frames* (MF) [31] picks out, among all solutions of (4.11), one whose coefficients have a minimum l_2 norm:

$$\min \|\underline{a}\|_2 \quad \text{subject to:} \quad \Phi'\underline{a} = \underline{\beta} \tag{4.12}$$

The solution of this problem is unique; define it as $\underline{a}^\dagger$. Geometrically, the collection of all solutions to (4.11) is an affine subspace in R^N, MF selects the element of this subspace closet to the origin. In optimization problems, it is sometimes called a minimum-length solution [31]. There exists a matrix $\Phi^\dagger$, the generalized inverse of Φ, which produces the minimum length solution to a system of linear equations in

(4.12):

$$\underline{a}^{\dagger} = \Phi^{\dagger}\underline{\beta} = (\Phi')^{T}_{N\times m}(\Phi'_{m\times N}(\Phi')^{T}_{N\times m})^{-1}\underline{\beta}_{m\times 1} \tag{4.13}$$

Specifically, for the Cosine transform, assume we separate the transform matrix C into C' and C'', the way we did with Φ, so that the size of C' is $m \times N$.

$$\begin{aligned} \underline{a}^{\dagger} &= (C')^{T}(C'(C')^{T})^{-1}\underline{\beta} \\ &= (C')^{T}I\underline{\beta} \\ &= (C')^{T}\underline{\beta} \end{aligned} \tag{4.14}$$

The solution satisfies the equation:

$$\begin{aligned} (\underline{a}^{\dagger})^{T}\underline{a}^{\dagger} &= \underline{\beta}^{T}C'(C')^{T}\underline{\beta} \\ &= \underline{\beta}^{T}\underline{\beta} \end{aligned} \tag{4.15}$$

it means that the extrapolated value of $\underline{\theta}$ here is $\underline{\theta} = \underline{\mathbf{0}}$. Obviously, in R^N space, $[\underline{\beta}\ \ \underline{\mathbf{0}}]^T$ is the one closest to the origin among all available solutions. From Parseval's Theorem, the same conclusion can also be easily derived.

4.3.3 Basis Pursuit

Obviously, data extrapolation in minimum l_2 sense does not provide us a satisfying solution. There are other methods people used in searching for the proper solution based on other criteria [31]. We consider an optimal approach which aims to minimize the l_1 norm of the transform coefficients. In [31], a technique named *Basis Pursuit* (BP) is designed for signal analysis. That is, BP chooses a proper set of basis functions to represent the signal from an overcomplete basis set, such that the transform

coefficients have the minimal l_1 norm value over all possible solutions. That is, BP solves the following problem.

$$\min \|\underline{a}\|_1 \ \text{ subject to } \ \Pi \underline{a} = \underline{s} \tag{4.16}$$

Since the matrix Π is representing an overcomplete basis set, it has more columns than rows.

Referring the problem posed in (4.12), but replacing the l_2 norm by the l_1 norm of $\underline{a}$, we see that the above BP solution to (4.16) can be used to solve our problem as well, i.e.,

$$\min \|\underline{a}\|_1 \ \text{ subject to } \ \Phi^{'} \underline{a} = \underline{\beta} \tag{4.17}$$

Even though the problem looks similar to the Method of Frames in its form, the basis pursuit method requires considerable more effort and sophistication for its solution. The connection between linear programming and basis pursuit is well known since the 1950's, so that the basis pursuit problems can actually be solved with less computation.

Let's now explain the connection between basis pursuit and linear programming. The standard linear programming form is a constrained optimization problem defined in terms of a variable $\underline{x} \in R^N$ by

$$\min M^T \underline{x} \ \text{ subject to } \ A\underline{x} = \underline{b}, \ \text{ and to } \underline{x} \geq \underline{\mathbf{0}} \tag{4.18}$$

where $M^T \underline{x}$ is the objective function, $A\underline{x} = \underline{b}$ is the collection of equality constraints, and $\underline{x} \geq \underline{\mathbf{0}}$ is a non-negativity constraint.

Now, if we define a $N \times 1$ vector $\underline{a} = \underline{u} - \underline{v}$, $\underline{u}, \underline{v} \geq \underline{\mathbf{0}}$, then,

$$\min \|\underline{a}\|_1 = \min \underline{\mathbf{1}}^T \underline{u} + \underline{\mathbf{1}}^T \underline{v} \tag{4.19}$$

The equivalence relies on the fact that at an optimal solution, only one of the two variables u_i or v_i can be nonzero [32]. Hence,

$$\Phi' \underline{a} = \Phi'(\underline{u} - \underline{v}) \tag{4.20}$$

$$= [\Phi' - \Phi'] \begin{bmatrix} \underline{u} \\ \underline{v} \end{bmatrix} \tag{4.21}$$

The basis pursuit problem then is finally reformulated into a standard linear programming problem defined in R^{2N}:

$$\min \underline{\mathbf{1}}^T \begin{bmatrix} \underline{u} \\ \underline{v} \end{bmatrix} \quad \text{subject to} \quad [\Phi' - \Phi'] \begin{bmatrix} \underline{u} \\ \underline{v} \end{bmatrix} = \underline{\beta}, \quad \underline{u}, \underline{v} \geq \underline{\mathbf{0}} \tag{4.22}$$

The solution of the BP problem through LP benefits from the fact that convenient tools are available for solving linear programming problems [33]. Besides, there is abundant progress in *large scale linear programming* in recent years [31], so that it is possible to solve large size problems. Finally, in our application, we are also interested in studying the innate properties of a LP solution.

1. It is well known that a linear programming problem with A in (4.18), a $m \times N$ matrix with $N > m$, always has a solution having the property that at most m entries in the optimal $\underline{x}$ are nonzero. Moreover, in the generic case, the solution is so-called nondegenerate, and there are exactly m nonzeros. In this case, the nonzero coefficients are associated with m columns of A; and these columns make up a basis of R^m. So, it means that to our formula (4.22), there is always a solution $[\underline{u} \ \ \underline{v}]^T$ with maximum m nonzero items. Moreover, suppose there are p nonzero components in $\underline{u}$ $(0 \leq p \leq m)$ and $m - p$ nonzero components in $\underline{v}$, respectively. Since our solution is $\underline{a} = \underline{u} - \underline{v}$, hence, the conclusion is that there exists a solution $\underline{a}$ that has at most m nonzero entries. When this technique is

applied for signal analysis, the original signal is precisely decomposed into its basis components, so, as stated above, the technique is named as Basis Pursuit.

2. Another interesting question is the predictability of the positions of these nonzero entries in the solution. Unfortunately, the answer is that their identity is, in general, not known in advance, but instead depends on the problem data.

- Block Extrapolation Using Basis Pursuit

Since an image segment is a 2D signal, the problem of implementing the 2D transformation via a 1D transformation should be solved before we can apply basis pursuit for optimal block extrapolation. Fortunately, the *separability* property of fast image transforms, like DFT, DCT, DST, Hadamard, Haar, etc. provides a simple solution.

An arbitrary 1D transformation $\underline{y} = [M_{N^2 \times N^2}]\underline{x}$ is called *separable* if $M_{N^2 \times N^2} = M'_{N \times N} \otimes M''_{N \times N}$, where $\otimes$ indicates a *Kronecker Product*. Correspondingly, the 1D transform is related to a separable 2D transform $Y = M'X(M'')^T$. Conversely, the 2D-DCT can be implemented by a 1D-DCT at the cost of increased complexity. The standard inverse 2D-DCT is defined as:

$$S_{N \times N} = C^T_{N \times N} A_{N \times N} C_{N \times N} \tag{4.23}$$

where C is the DCT transform matrix, S the reconstructed image and A the DCT coefficient matrix. Mapping matrices A and S into vectors $\underline{a}$ and $\underline{s}$ by row ordering, the 2D-IDCT can be calculated by a 1D-IDCT as follows:

$$\underline{s} = [\mathcal{C}_{N^2 \times N^2}]\underline{a} \tag{4.24}$$

where $\mathcal{C}_{N^2 \times N^2}$ is calculated by $\mathcal{C}_{N^2 \times N^2} = C^T_{N \times N} \otimes C^T_{N \times N}$.

Referring to Fig. 4.3, for illustration, the block size is chosen to be 4×4. The matrix in Fig. 4.3.(a) is mapped into a single column vector. Since in our previous assumptions, the positions of the known data is at the beginning of the signal, the column vector need to be adjusted. The aligning procedure is shown in Fig. 4.3.(b). Consequently, the same manner of row shifts should be applied to transform matrix $\mathcal{C}$, in order to keep the validity of the whole system of equations. For example, item $B_{(2,3)}$ in the original block is the twelveth component in the column vector, s_{11}. After order adjustment is performed, its position is changed to the eighth position, where s_7 was located. Correspondingly, row r_{11} of matrix $\mathcal{C}$ is moved to where r_7 was located.

4.4 Illustration

In previous sections, we discussed various boundary block coding schemes. The purpose of this section is to demonstrate their performance. A comparison is performed between the resulting transform coefficients in the DCT domain, and the advantages and drawbacks of each method is discussed.

In Fig. 4.4, a 8×8 block is taken from one of the boundary blocks generated by the segmentation algorithm. The intersection points indicate the pixel locations. Inner pixels are located at the left side of the block, and the flat right side indicates pixels outside of the boundary of the current segment. The methods to be illustrated here include: Method of Frames (MF), SADCT, LPE padding and Basis Pursuit (BP).

1. Method of Frames (MF): This method simply fills in pixels outside of the segment with zeros (Fig. 4.4). The resulting DCT transform coefficient matrix is shown in Fig. 4.5.

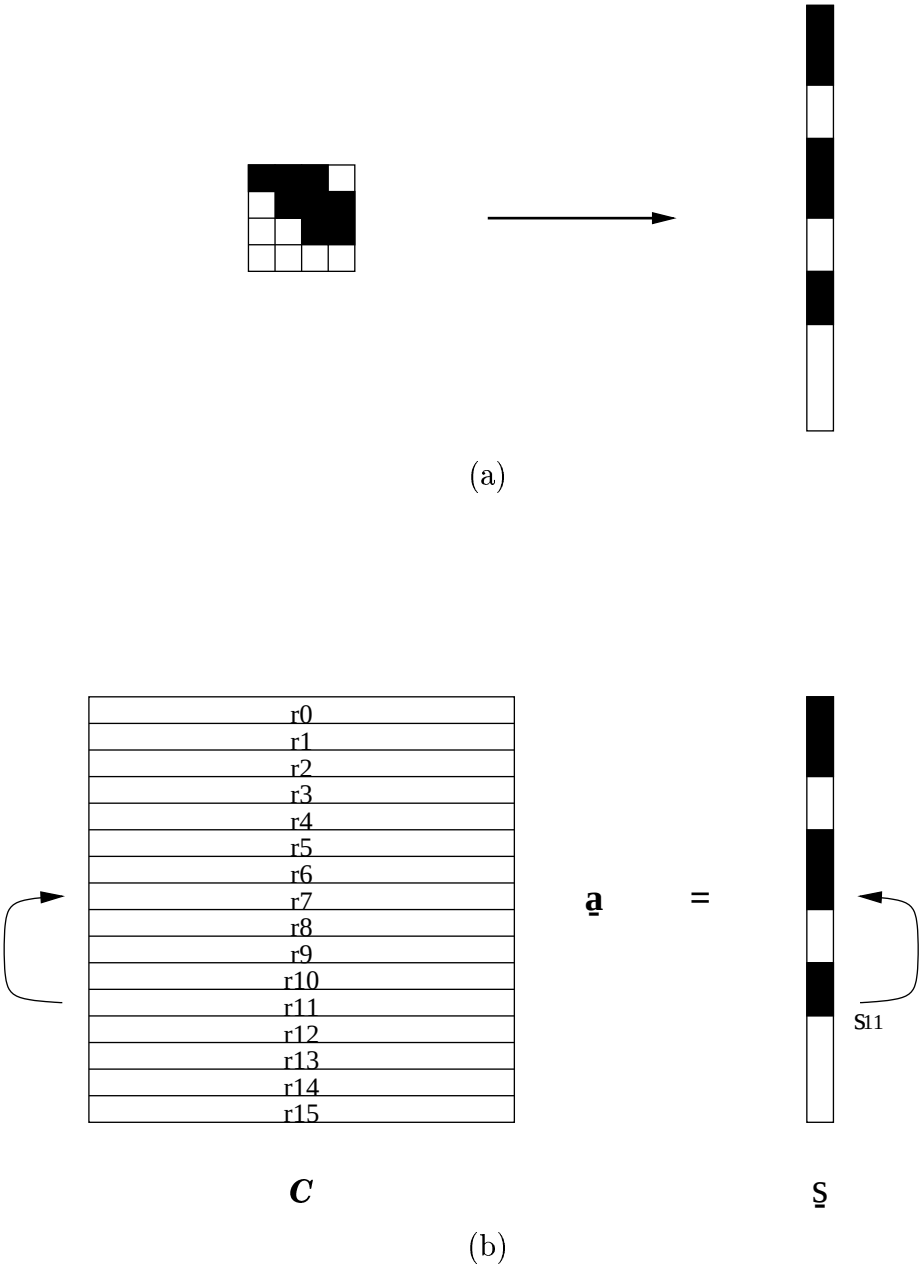


Figure 4.3: Connection between 2D-DCT & 1D-DCT Transforms
איור 4.3: קשר בין התמרות 2D-DCT ו-1D-DCT

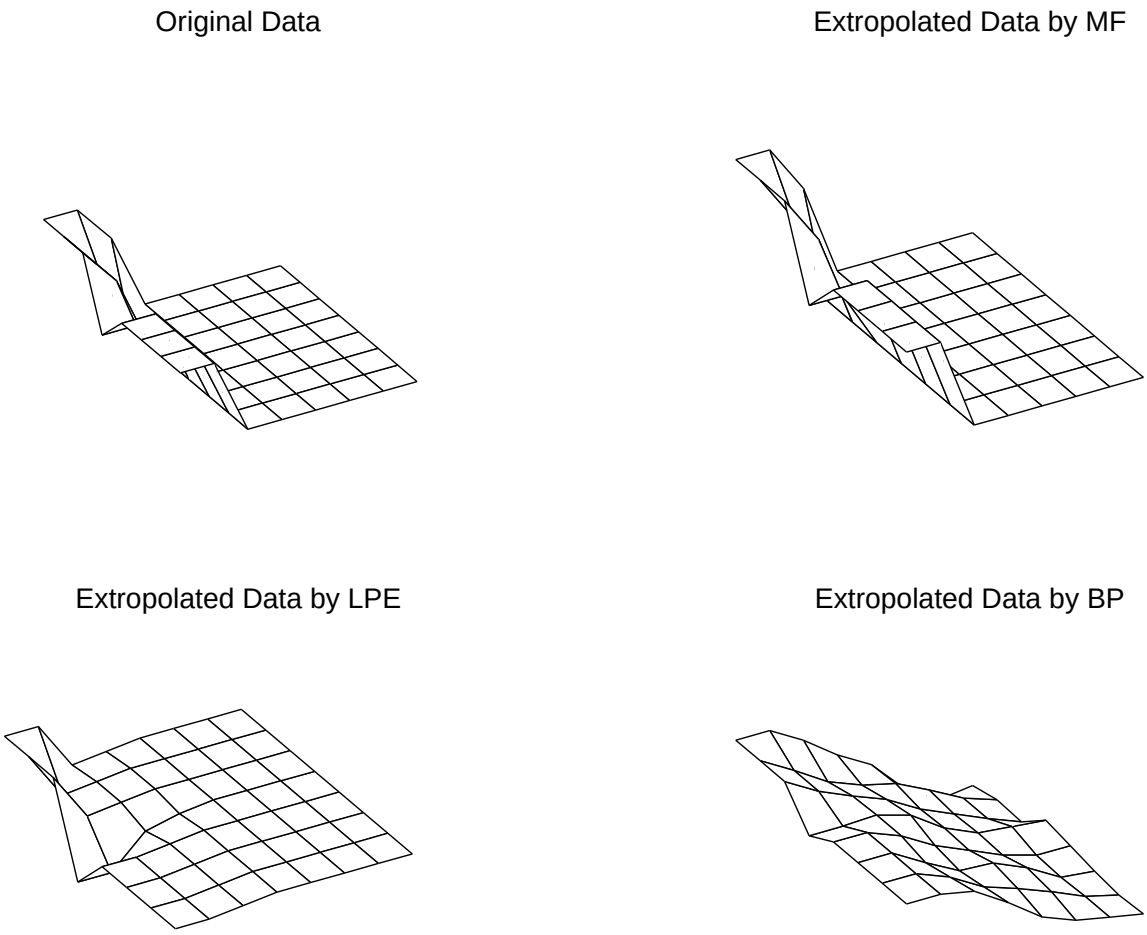


Figure 4.4: Boundary Block Extrapolation
איור 4.4: אקסטרפולציה של בלוקי קצה

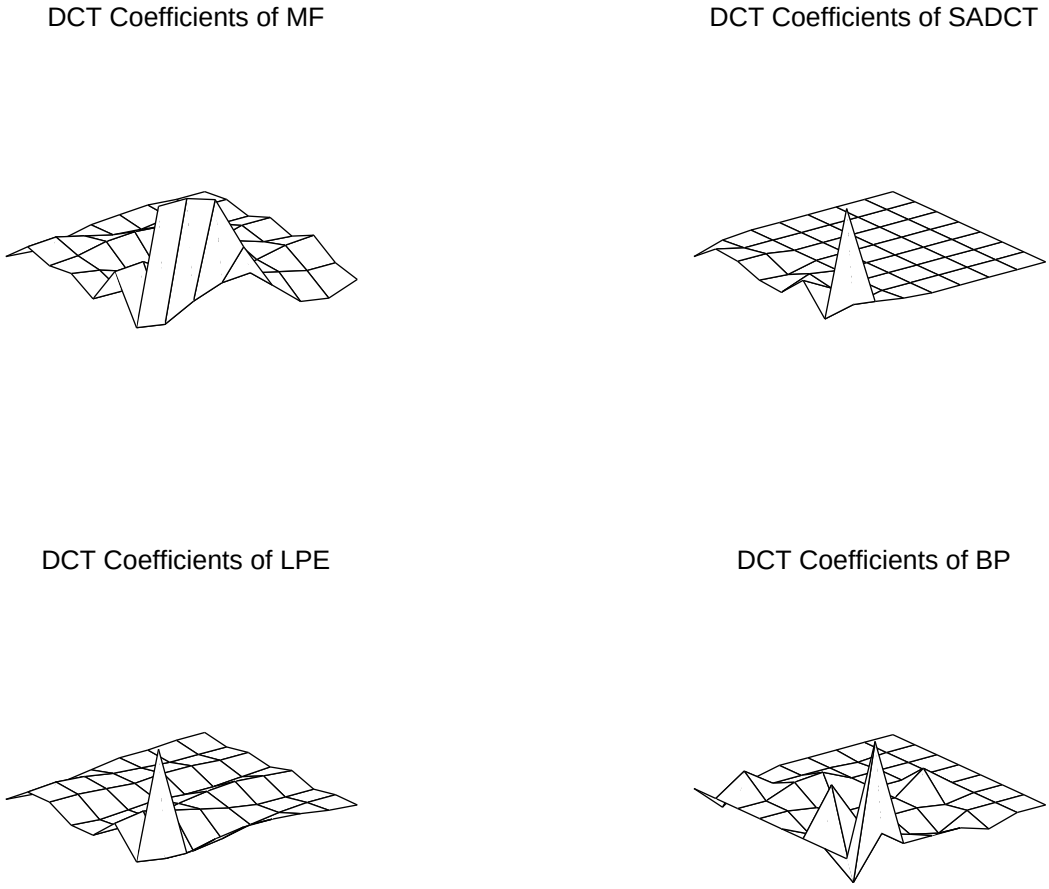


Figure 4.5: Boundary Block Coding
איור 4.5: קידוד בלוקי קצה

2. SADCT: This transform is performed following the steps specified in section 4.2. The final transform coefficient matrix is shown in Fig. 4.5.

3. LPE padding: A boundary block padded by LPE is illustrated Fig. 4.4. Subsequently, a regular 8×8 2D-DCT is calculated, and the DCT coefficients are shown in Fig. 4.5.

4. BP padding: Boundary blocks are padded by the Basis Pursuit method discussed above. For the example considered, the extrapolated output and the DCT transform coefficients, have different characteristics than those produced by all the other examined techniques, as seen in figures 4.4 and 4.5.

In an attempt to compare the compression quality of the different techniques for just the examined block, an entropy-like measure which is related to coefficients energy distribution [34] is defined as follows:

$$p_i = \frac{c_i^2}{\sum_k c_k^2} \tag{4.25}$$

$$E = -\sum_i p_i \log p_i, \quad i = 0, \ldots, N-1 \tag{4.26}$$

The entropies of MF, SADCT, LPE and BP coefficients in the above particular example are: 2.6489, 0.5630, 0.2181, 0.5129 respectively.

Since only one specific block is studied by the criterion, it is not global. Therefore, the numerical result supports only part of the conclusions below. Basicly, the conclusion should be drawn based on the coding results, i.e., the bit-rate, PSNR and perceptual quality of the reconstruction image. The next section provides this information.

Our conclusion of the shape-adaptive coding techniques discussed above is that:

1. We certainly do not recommend to choose the *Method of Frames* as the padding method for boundary block coding. Even though the scheme does not add redundancy to the original data. In the DCT domain, as seen in Fig. 4.5 and from the entropy, the energy obviously expands into high frequencies. Therefore, a low compression performance is expected.

2. *Low-Pass Extrapolation (LPE)* padding generates a gradually changing smooth surface in the image block (Fig. 4.4). Therefore, energy compaction in the DCT domain is better than by padding with Method of Frames.

3. To perform the sequential entropy coding, the transform coefficients calculated by *Basis Pursuit (BP)* method are more suitable than those calculated by the other methods. Because the transform coefficients calculated by BP, in Fig. 4.5, are fewer and their energy tends to concentrate on a very few, and results bit rate reduction, as shown later in section 4.5. But, since the positions of these nonzero coefficients are not predictable as in the other method, where they are mainly constrained to low frequencies. This property indicates that the default quantization table and the Huffman entropy coding scheme used in traditional JPEG coder are not suitable if BP is used.

4. The data shifts performed in *SADCT* guarantees that the output coefficients occupy the low frequencies corner. But, the procedure also decreases the compactness of energy as shown in Fig. 4.5 and by the numerical results. This disadvantage of the SADCT will further be discussed in the sequel.

4.5 Coding System and Simulation Results

As we stated before, one of the advantages of block-based coding methods are their compatibility with existing JPEG and other standard coding systems. Our main effort is invested in finding the proper substitute for the conventional 2D block-based DCT. Thus, we simply embedded the transform coding techniques that we discussed earlier into the JPEG coder, and compared the corresponding coding performance respectively.

4.5.1 Proposed Coding System

Fig. 4.6 shows a general block diagram of the proposed shape-adaptive image coding system. Since current methods are block-based, the texture coding in Fig. 4.6 includes boundary block coding and inner block coding as discussed earlier in section 4.1. That is, following the classical 8×8 block image partition, the sequential blocks fall into inner block or boundary block categories. Those which fall into the inner-block category are coded by a JPEG codec. Boundary blocks are coded by the methods we specify below. The reader should bear in mind that contour decoding is conducted before texture decoding, therefore, contour information is priorly known. Hence, the separation of inner blocks and boundary blocks can be done in the same manner as at the encoder side and no extra bits are needed for the block information.

There exist three alternative transform coding methods that we investigated. The Method of Frames (MF) padding is ignored, because of its obvious shortcomings. As we discussed early, the SADCT transform coefficients occupy positions corresponding to low frequencies in the 2D-DCT domain. It is clear that a slight change in

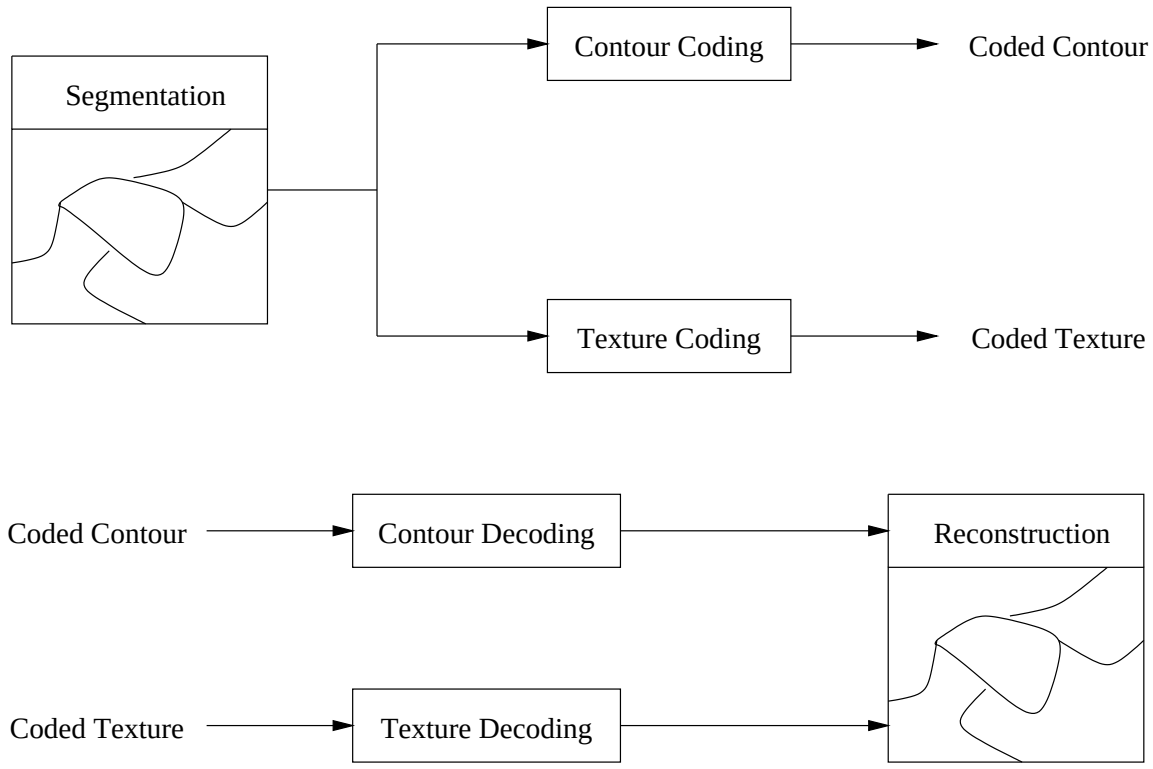


Figure 4.6: Segmentation-Based Shape-Adaptive Coding & Decoding System

איור 4.6: מערכת קידוד ופענוח מבוססת חלוקה ותלוית צורה

the JPEG baseline codec by substituting SADCT (SA-IDCT) for the 2D-DCT (2D-IDCT) routine, makes the whole system a suitable SADCT codec. Even though in a recent MPEG-4 release, adaptive scanning is adopted, and further compression may be achieved, our implementation is similar to the coder proposed in [6]. The same situation happens to DCT coefficients calculated by the LPE padding method, which also occupy the low frequencies in the DCT domain. Therefore, the extrapolated block can be treated as an inner block, and JPEG coding is applied directly after LPE padding. To code the transform coefficients calculated by the BP method, we apply a uniform quantizer, and the quantized coefficients are zig-zag scanned. Since the positions of the high energy coefficients is not predictable, a considerable number of experiments would be needed to obtain the coefficient statistic and build the corresponding VLC table. Another entropy coding option is arithmetic coding [35]. Based on the quantized coefficients sequence, a sequence, a_i, of the amplitudes of the nonzero coefficients, and a sequence, z_i, of the number of zeros before each nonzero coefficients are generated. Arithmetic coding is then applied to encode the sequences of a_i and z_i. On the decoding side, an inverse procedure is conducted to obtain the extrapolated data block. As we discussed earlier, the extrapolated data can now easily be discarded, since the shape information is priorly known.

4.5.2 Simulation Results & Discussion

This section demonstrates block-based image coding results of several 512×512 still images. We apply the modified chain coding method to code the contours, so that a comprehensive comparison between different shape-adaptive coding methods can be made. To achieve the optimal performance of shape-adaptive techniques, we applied rougher quantization of the transformation coefficients. The tables provided below

indicate an advantage (in bit-rate reduction) of the BP padding method over SADCT and LPE padding. This is particularly pronounced for the images "Peppers" and "House". As seen in the following two figures, the reconstructed images produced by JPEG and BP have almost the same PSNR and about the same perceptual quality, but lower rate for BP. The reason is that a lower reconstruction error is obtained by performing BP coding of the boundary blocks, while rougher quantization is used for inner blocks, as mentioned eariler, which compensate each other.

In general, block-based shape-adaptive coding techniques follow the conventional rectangular grid image partition context. The advantage thus is that they are quite compatible with current widely used image coding systems. On the other hand, since these coding methods can not exploit the global characteristic of a segment, the potential compression capability of segment-based coding is not fully exploited. In particular, in still image coding, a boundary block belongs to more than one segment, so that its must be done more than once, which increases the bit-rate. The simulation results shown below indicate that SADCT and LPE do not provide better compression than JPEG. Yet BP padding outperforms JPEG in some cases, Further studies are needed to identify the kind of images which are better matched to BP padding. Besides, the complexity of the BP extrapolation algorithm is still too high for general use.

Method	Texture Coding (bpp)	Contour Coding (bpp)	Total Bit-Rate	PSNR
JPEG	/	/	0.6400	35.760
	/	/	0.4124	33.666
BP Padding	0.5597	0.0671	0.6268	35.201
	0.3883	0.0671	0.4554	33.382
LPE Padding	0.7554	0.0671	0.8225	35.489
	0.4725	0.0671	0.5396	33.542
SADCT	0.7186	0.0671	0.7857	35.312
	0.4945	0.0671	0.5616	33.356

Table 4.1: Coding Results of the Image "Lena" by Block-Based Techniques

טבלה 4.1: תוצאות הקידוד של תמונת "lena" ע"י שיטות המבוססות על חלוקה לבלוקים

Method	Texture Coding (bpp)	Contour Coding (bpp)	Total Bit-Rate	PSNR
JPEG	/	/	0.4154	36.134
	/	/	0.2917	33.793
BP Padding	0.3692	0.0240	0.3932	35.973
	0.2660	0.0240	0.2900	33.794
LPE Padding	0.4609	0.0240	0.4849	36.171
	0.3222	0.0240	0.3462	33.962
SADCT	0.4923	0.0240	0.5163	35.630
	0.3435	0.0240	0.3675	33.831

Table 4.2: Coding Results of the Image "House" by Block-Based Techniques

טבלה 4.2: תוצאות הקידוד של תמונת "House" ע"י שיטות המבוססות על חלוקה לבלוקים

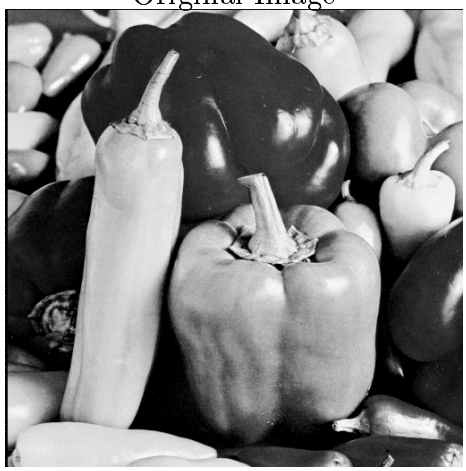
Method	Texture Coding (bpp)	Contour Coding (bpp)	Total Bit-Rate	PSNR
JPEG	/	/	0.7320	33.448
	/	/	0.4488	31.879
BP Padding	0.6172	0.0574	0.6746	33.324
	0.3835	0.0574	0.4409	31.682
LPE Padding	0.7282	0.0574	0.7856	33.093
	0.4463	0.0574	0.5037	31.725
SADCT	0.7336	0.0574	0.7910	33.117
	0.4415	0.0574	0.4989	31.309

Table 4.3: Coding Results of the Image "Peppers" by Block-Based Techniques
טבלה 4.3: תוצאות הקידוד של תמונת "Peppers" ע"י שיטות המבוססות על חלוקה לבלוקים

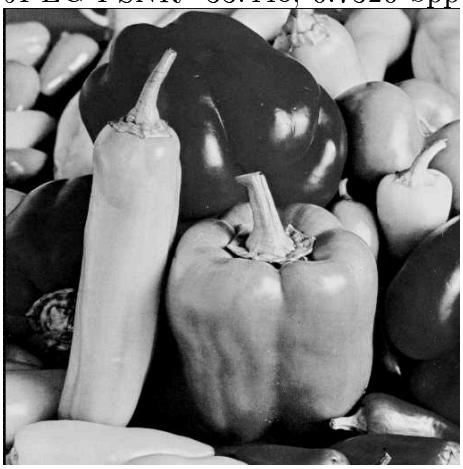
Method	Texture Coding (bpp)	Contour Coding (bpp)	Total Bit-Rate	PSNR
JPEG	/	/	0.6425	35.417
	/	/	0.3945	33.207
BP Padding	0.5848	0.0745	0.6593	35.405
	0.3702	0.0745	0.4447	33.251
LPE Padding	0.7363	0.0745	0.8108	35.332
	0.4416	0.0745	0.5161	33.354
SADCT	0.7132	0.0745	0.7877	35.240
	0.4499	0.0745	0.5244	33.052

Table 4.4: Coding Results of the Image "Bike" by Block-Based Techniques
טבלה 4.4: תוצאות הקידוד של תמונת "Bike" ע"י שיטות המבוססות על חלוקה לבלוקים

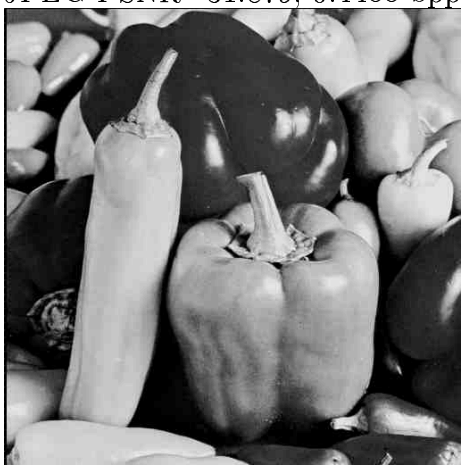
Original Image



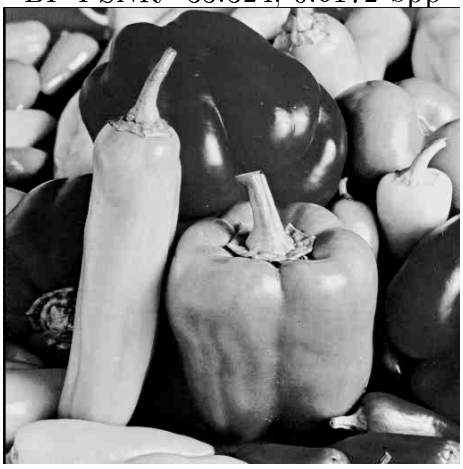
JPEG PSNR=33.448, 0.7320 bpp



JPEG PSNR=31.879, 0.4488 bpp



BP PSNR=33.324, 0.6172 bpp



BP PSNR=31.682, 0.3835 bpp

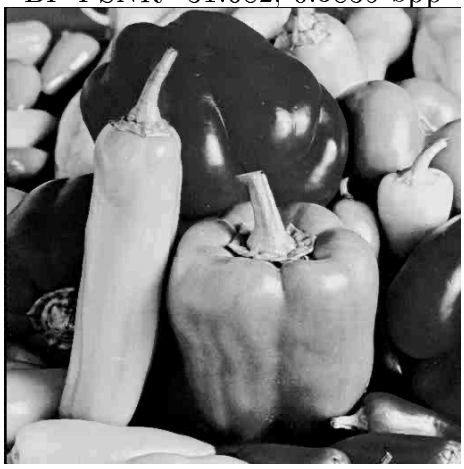
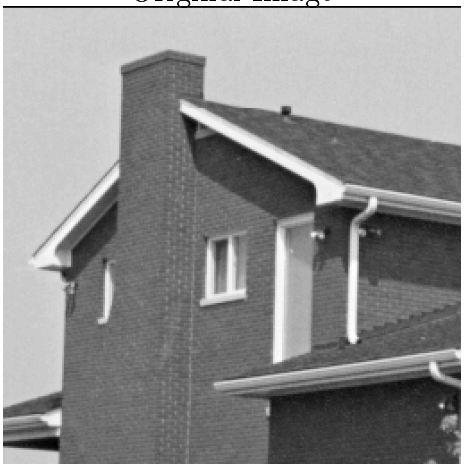


Figure 4.7: "Peppers" Image Coding Results
"Peppers" איור 4.7: תוצאות הקידוד של התמונה

Original Image



JPEG PSNR=36.134, 0.4154 bpp

JPEG PSNR=33.793, 0.2917 bpp

BP PSNR=35.973, 0.3692 bpp

BP PSNR=33.794, 0.2660 bpp

Figure 4.8: "House" Image Coding Results
"House" איור 4.8: תוצאות הקידוד של התמונה

Method	Texture Coding (bpp)	Contour Coding (bpp)	Total Bit-Rate	PSNR
JPEG	/	/	0.4152	35.068
	/	/	0.3572	34.223
BP Padding	0.4136	0.0290	0.4426	34.816
	0.3751	0.0290	0.4041	34.159
LPE Padding	0.4197	0.0290	0.4487	34.773
	0.3713	0.0290	0.4003	34.074
SADCT	0.4834	0.0290	0.5124	34.974
	0.4102	0.0290	0.4392	34.008

Table 4.5: Coding Results of the Image "Medical" by Coded by Block-Based Techniques

טבלה 4.5: תוצאות הקידוד של תמונת "Medical" ע"י שיטות המבוססות על חלוקה לבלוקים

Chapter 5

Region-Based Shape-Adaptive Coding Techniques

In the previous chapter, we discussed several shape-adaptive coding techniques which inherit the main properties of the conventional block-based DCT transform coding scheme. Thus, they are named as block-based shape adaptive coding methods. We name another category of techniques as region-based coding methods. These methods try to exploit the homogeneous character of the segments produced by the segmentation algorithm and find a compact representation for the original segment data. Such an approach is related to an approximation of the original region gray-levels. Therefore, a general background overview of approximation theory is necessary, and different algorithms that were investigated are described afterwards. Certainly, because both approaches aim at efficient coding, there must be some connection between these methods. But, at the same time, region-based methods may confront other problems, e.g., a much larger quantity of data needed to be dealt with; a more global characteristics of the data, etc. Readers may be curious to know if some of the

methods we discussed earlier can still be applied here, and vice versa. This is the last issue that is discussed in this chapter.

5.1 Related Issues in Approximation Theory

Given an image segment $f(x, y)$, the purpose of image compression is to find an approximation function $g(x, y)$, which approximates the original function $f(x, y)$ as close as possible, and in a more compact form. Therefore, principles from approximation theory, generalized moments and least squares (LS) approximation, strongly caught researchers' attention [5, 36]. In this section, a brief introduction of the two principles, generalized moments and normal equations for solving LS problems is provided.

The theory of generalized moments defines certain measurements or properties of both the original and the approximation functions. The theory of least squares approximation defines the criterion to measure the difference between original and approximation functions. Reference [5] shows that approximation functions that are determined independently by applying these two principles, eventually coincide.

5.1.1 Generalized Moments

From a mathematical point of view, a gray-level image segment $f(x, y)$ with arbitrary shape $\mathcal{D}$ is equivalent to a function $f(x, y)$ which has non-zero values only in a finite part of the x-y plane, namely the region $\mathcal{D}$ corresponding to the shape of the segment. We can define linear properties of the given function using the following operation

$$m_{ij} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_{ij}(x, y)f(x, y)dxdy \tag{5.1}$$

$$= \int_{\mathcal{D}}\int h_{ij}(x, y)f(x, y)dxdy \tag{5.2}$$

The measurements or properties m_{ij} are called *generalized moments* and $h_{ij}(x,y)$ is the measurement kernel. Under the assumption that the set of $h_{ij}(x,y)$ is complete, a uniqueness theorem [37] states that the sequence of moments $\{m_{ij}\} = \{m_{00}, m_{01}, \cdots, m_{kl}, \cdots\}$ is *uniquely* defined by $f(x,y)$, and conversely $f(x,y)$ is *uniquely* defined by its associated sequence of moments m_{ij}.

The specification of the measurement kernel $h_{ij}(x,y)$ depends on the application. The approach is general enough to show that different, commonly used measures, all involve the same mathematical context. For example, consider the kernel $h_{ij} = \exp[-j2\pi(u_i x + v_j y)]$. We then obtain m_{ij} as samples of the 2D Fourier-transform of the image at the spatial frequency (u_i, v_j). With the choice $h_{ij} = h(x_i - x, y_j - y)$, the measurements become samples of the convolution of the image with the point spread function h. Finally, we take $h_{ij} = x^i y^j$, and obtain m_{ij} the *geometrical moments* with the *order* of the moments denoted by $i + j$. The above definition has the form of a projection of the function $f(x,y)$ onto the monomial $x^i y^j$. This basis functional set $\{x^i y^j\}$ is complete, but not orthogonal.

Two problems arise, before we can employ the above definition:

1. Given a sequence of generalized moments, we need to find an inverse transform to retrieve $f(x,y)$ or at least an approximation of $f(x,y)$. In the case of Fourier transform we know an inverse exists, but does an inverse of the geometric moment representation exist?

2. Assuming we found an inverse operation, we are faced with the problem that the geometrical moments sequence is infinite. How many and which of the moments do we need for an approximation which satisfies a given error constraint?

Unfortunately, it has been shown that no direct inverse exists for the case of a finite

number of geometrical moments and a continuous function $f(x, y)$. Of course, for the discrete case, i.e., sampled image function, moment measurement can be written in a form in which the integrals are replaced by summations over the defined area of the given segment, $\mathcal{D}$,

$$m_{ij} = \sum_{(x,y)\in\mathcal{D}} x^i y^j f(x, y) \tag{5.3}$$

or in matrix form

$$\begin{bmatrix} m_{00} \\ m_{10} \\ \vdots \\ m_{ij} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_M \\ \vdots & \vdots & & \vdots \\ x_1^i y_1^j & x_2^i y_2^j & \cdots & x_M^i y_M^j \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} f(x_1, y_1) \\ f(x_2, y_2) \\ \vdots \\ f(x_k, y_k) \\ \vdots \end{bmatrix} \tag{5.4}$$

$$\underline{\mathbf{m}} = \Phi\underline{\mathbf{f}} \tag{5.5}$$

If we additionally assume that M is the number of given image samples, and we calculate exactly $K = M$ moments, the above equation has the form of a one to one transformation. We know that the inverse is given right away by $\underline{\mathbf{f}} = \Phi^{-1}\underline{\mathbf{m}}$, with Φ^{-1} as the inverse matrix of Φ. Unfortunately, inverting the matrix Φ is a mathematically ill-posed problem. Besides, we want less moments than the given number of pixels to suffice as a description of $f(x, y)$. As a solution out of this situation *moment matching* is introduced in the following, and the result nicely turns out to be equivalent to the method of least squares approximation.

Assume a finite set of K known moments m_{ij} is given, and we use a model function $g(x, y)$ which may be expressed in parameter form with N unknown parameters a_n, by

$$g(x, y) = a_1\phi_1(x, y) + a_2\phi_2(x, y) + \cdots + a_N\phi_N(x, y) \tag{5.6}$$

The functions $\phi_n(x,y)$, $n = 1, 2, \cdots, N, \cdots$, are called the *basis functional set*, and for the task of approximation the set must be complete. If we now require the moments M_{ij} of $g(x,y)$ to be identical to the given moments m_{ij}: $m_{ij} = M_{ij}$, the following system of equations is derived:

$$\sum_{(x,y)\in\mathcal{D}} h_{ij} f(x,y) = \sum_{(x,y)\in\mathcal{D}} h_{ij} g(x,y) = \sum_{(x,y)\in\mathcal{D}} h_{ij} \sum_{n=1}^{N} a_n \phi_n(x,y) \tag{5.7}$$

we get a coupled set of K equations. If the number of parameters N is a equal to the number of moments K, the set of equations has exactly *one solution* and we can solve for the vector $\underline{\mathbf{a}}$ (consisting of N components). The closeness of the approximation is determined by the number of moments used. The sequence of given moments approximates $f(x,y)$ by $g(x,y)$ which follows from the uniqueness theorem

$$f(x,y) \approx g(x,y) = \Phi\underline{\mathbf{a}} \tag{5.8}$$

Equality holds in the discrete case for $K = N \to M$, where M is the number of segment samples, or in the continuous case for $K = N \to \infty$.

With regard to image representation, researchers are interested in using geometrical moments generated by the measurement kernel $h_{ij} = x^i y^j$, and simple basis functions $\phi_i(x,y) = x^{k(i)} y^{l(i)}$, $k + l \leq N$ for representing the approximation function $g(x,y)$.

5.1.2 Least Squares Approximation and Normal Equations

Two kinds of errors are connected with an approximation of the original image region:

1. *Measurement errors.* Taking the samples (pixels) from the original continuous luminance function can be thought of as a measuring process. The obtained values differ from the true brightness values by the measurement error, e.g.,

camera noise. Denoting the true but unknown gray-level by $s(x, y)$ and the error term by $e(x, y)$, the following equation holds:

$$f(x, y) = s(x, y) + e(x, y) \tag{5.9}$$

2. *Modeling errors.* The approximation tries to fit a selected model to the measurements. Depending on the physical data generation process, the selected model is more or less suited for the description. Only in rare cases do we know the physics behind the data generating process which dictates the choice of the model. In the case of modeling the image intensities, little can be assumed about the nature of $s(x, y)$, except assuming that the function is positive and bounded. When we are considering the human observer, we can also assume that $s(x, y)$ is band-limited, otherwise it would contain information a human observer cannot perceive anyway. Summarizing, it can be said that the approximation error is dependent on the used model.

In order to find the "best" approximation for a given set of data we have to select not only a suited model but we also have to express "best" in a defined way. For a best approximation $g(x, y)$ of $f(x, y)$, in the least-squared-error sense, we have to use the Euclidean distance - also called sum of squared errors:

$$d(f, g) = \sum_{(x,y)\in\mathcal{D}} (g(x, y) - f(x, y))^2 \tag{5.10}$$

The least squares (LS) approximation enjoys the following advantages [38]:

1. Easy to implement.

2. Does not require an iterative solution.

3. Filters zero mean, finite variance, noise in an unbiased manner.

4. If the measurement error $e(x, y)$ has Gaussian distribution and if the measured data are considered to be realizations of a stochastic process, then the obtained estimate $g(x, y)$ using the least squares (reformulated as a minimum mean squared error) criterion yields the minimum variance unbiased estimate among all unbiased estimators. Furthermore, the estimate $g(x, y)$ is the maximum likelihood estimate.

The approximating function $g(x, y)$ can be written in a generalized form as in (5.6), $g(x, y) = a_1\phi_1(x, y) + a_2\phi_2(x, y) + \cdots + a_N\phi_N(x, y)$. With the basis functions $\phi_1, \phi_2, \cdots, \phi_N$ given through the selected model, e.g., polynomials, and $a_1, a_2, \cdots, a_N$ being the parameters which are to be determined in a way that will minimize the distance $d(f, g)$:

$$d(f, g) = \sum_{(x,y)\in\mathcal{D}} (\sum_{n=1}^{N} a_n\phi_n(x, y) - f(x, y))^2 \rightarrow \min \tag{5.11}$$

To find the vector $\underline{\mathbf{a}}$ which brings the sum of squared errors to a minimum, we differentiate (5.11) with respect to every component a_q of $\underline{\mathbf{a}}$ and require it to be zero:

$$\frac{\partial d}{\partial a_q} = 2 \sum_{(x,y)\in\mathcal{D}} (\sum_{n=1}^{N} a_n\phi_n(x, y) - f(x, y))\phi_q(x, y) = 0, \quad \text{for } q = 1, \cdots, N \tag{5.12}$$

Hence, the set of equations (5.13) is derived.

$$\sum_{n=1}^{N} a_n \sum_{(x,y)\in\mathcal{D}} \phi_n(x, y)\phi_q(x, y) = \sum_{(x,y)\in\mathcal{D}} f(x, y)\phi_q(x, y), \quad \text{for } q = 1, \cdots, N \tag{5.13}$$

If the measurement kernel h_{ij} is chosen to be $\phi_q(x, y)$, (5.7) is exactly equivalent to (5.13), therefore it proves our earlier statement. Equations (5.13) are called *normal equations* because in an N-dimensional Euclidean space the smallest error vector $e(x, y) = g(x, y) - f(x, y)$ is perpendicular (normal) to the hyper-plane spanned by

the basis functions $\phi_q(x, y)$. The maximal dimension of the Euclidean space is given by the number of pixels in the current segment, M in our case.

The N equations can be solved for the N values of $\underline{\mathbf{a}}$ minimizing the approximation error. Concerning the number of unknowns, N, and the number of given pixels, M, we can distinguish between three cases:

1. $M < N$: We have less data points than unknowns, and the system is said to be *underdetermined.*

2. $M = N$: We have as many data points (pixels) as parameters and the system is guaranteed to have exactly *one solution* if the columns of Φ are linearly independent. This solution can be viewed in two ways:

   (a) We are mapping the data values onto the basis functional set given by the matrix Φ. Obtaining $\underline{\mathbf{a}}$ is a *transformation* into another coordinate system and the values of $\underline{\mathbf{a}}$ are often called coefficients.

   (b) Another interpretation is that we now have an analytical function $g(x, y)$ which agrees with $f(x, y)$ on the nodes of the sampling grid. But unlike $f(x, y)$ we can compute from $g(x, y)$ all the values in-between sampling positions, a process called *interpolation.*

3. $M > N$: We have more data points than parameters which is what we want for data compression. But the distance between g and f usually cannot be zero for all M positions. The system of linear equations is said to be *overdetermined*, It can be solved by distributing the fitting error in a way that minimizes the sum of squared errors.

The third case $M > N$ in connection with image data compression offers the following advantages:

1. $f(x, y)$ is usually corrupted by measurement errors as stated above, therefore an exact reproduction of $f(x, y)$ requires not only the information but also the noise to be coded. A least squares approximation on the other hand usually reduces stochastic errors.

2. Approximating the function by less parameters than given points yields a compact description, which yields data compression (unless the representation of the fewer parameters requires more bits than the given pixels).

3. With the assumption in [5], of a slow varying surface within the image segment, the function g can be selected to give smooth spatial gray-level changes, e.g., by using polynomials.

The set of equations in (5.13) is coupled and, numerically, not necessarily stable. A solution for $\underline{\mathbf{a}}$ can be obtained through the Gauss-Jordan algorithm [36] which is computationally expensive. Another drawback makes the direct solution approach derived above impractical: In an attempt to improve the approximation quality by including more coefficients (this is necessary for example if we think about progressive transmission), all equations have to be solved again from the beginning, due to the coupled nature of the equations. Hence, some researcher thought that *orthogonialzing* the basis functions, and the using orthogonal basis functions will simplify the solution.

Summarizing this section, we have shown that the generalized moment representation and moment matching includes several other well known techniques as a special case, e.g., Fourier analysis. This method nicely turns out to be equivalent to a least squares approximation. Taking as many approximation coefficients as given segment

pixels results in a perfect error-free reconstruction. For a smaller set of coefficients, only an approximation can be recovered. Therefore, the data compression ratio depends on the basis functions used and the number of approximation coefficients. In the following section, we will discuss the theory and application of orthogonal basis functions, which serve as a classical tool for avoiding the heavy computational load involved in (5.13).

5.2 Orthogonal Basis Function Generation

One of the possible solutions to reduce the computation complexity involved in solving equations (5.13) is using orthogonal basis functions [5]. Hence, most of the research works that applied this approach later on tried to find a proper orthogonal basis functions generation method. In this section we introduce the concept of *orthogonal basis functions with respect to segment shape* and demonstrate the simplicity it brings in the calculation of the approximation coefficients $\underline{\mathbf{a}}$.

5.2.1 Orthogonal Basis Functions with respect to Segment Shape

The set of basis functions are called *orthogonal with respect to segment shape*, if the following condition holds

$$\sum_{(x,y)\in\mathcal{D}} \phi_n(x,y)\phi_q(x,y) = 0, \text{ for } n \neq q \tag{5.14}$$

The orthogonality property greatly simplifies the solution of (5.13). Now the system of equations is decoupled, yielding

$$a_q \sum_{(x,y)\in\mathcal{D}} \phi_q(x,y)\phi_q(x,y) = \sum_{(x,y)\in\mathcal{D}} f(x,y)\phi_q(x,y), \text{ for } q = 1,\cdots,N \tag{5.15}$$

And consequently, the approximation coefficients $\underline{\mathbf{a}}$ can be calculated by

$$a_q = \frac{\sum_{(x,y)\in\mathcal{D}} f(x,y)\phi_q(x,y)}{\sum_{(x,y)\in\mathcal{D}} \phi_q(x,y)\phi_q(x,y)}, \text{ for } q = 1,\cdots,N \tag{5.16}$$

If the set of basis functions is even orthonormal, i.e. $\sum_{(x,y)\in\mathcal{D}} \phi_q(x,y)\phi_q(x,y) = 1$. We can simplify the solution further

$$a_q = \sum_{(x,y)\in\mathcal{D}} f(x,y)\phi_q(x,y), \text{ for } q = 1,\cdots,N \tag{5.17}$$

As we see, the system of equations can be solved very easily for each of the approximation coefficients by just mapping the given image onto the respective basis function $\phi_q(x,y)$. Even including more coefficients does not necessitate the recalculation of previously computed coefficients. And this is very useful for progressive image built-up. The main advantage however connected with the orthogonal basis functions is the numerical stability and its simplicity as no matrix inversion or an iterative solution is needed. Instead, we get a well conditioned system of equations. Furthermore, the following theorem [39] guarantees the existence of the orthogonal basis functions for a given shape. The theorem states that:

> If the set of functions $\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{n-1}$ is linearly independent in a n-dimensional space A_n, then there exists a set of functions $\mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_{n-1}$ which are orthogonal with respect to the same space and moreover the functions $\mathbf{q}_i$, $i = 0,\cdots,n-1$, are linear combinations of the functions $\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{n-1}$.

It is clear that, in general, two given functions are not orthogonal with respect to several differently shaped segments. Therefore, it is necessary to find a specific set of orthogonal functions for each of the different segments given. In fact, the main computational burden comes from orthogonal basis function generation with respect to a specific image segment. In the next section, different algorithms are introduced to inform the reader of different approaches to efficient orthogonal basis function generation.

5.2.2 Gram-Schmidt Orthogonalization Scheme

As we mentioned before, with the introduction of orthogonal basis functions, the main source of the computation burden comes from the computation of basis functions which are orthogonal to a given shape. Since orthogonality is shape dependent and there are many different segment shapes, the requirement for an efficient orthogonal basis function generation algorithm is indispensable. Various works were conducted to generate a set of orthogonal basis functions that fully exploit the appealing theoretical compression potential, provided that computational load is kept low. In the sequel, four algorithms are introduced in chronological order, Gram-Schmidt orthogonization scheme, Householder Polynomials, Weakly separable basis functions and Shape-independent basis functions. Relevant references can be found in [5, 40, 41, 42, 43, 44]. We will discuss the practical problems that we encountered in their implementation later on.

Before we start to describe these algorithms, we need to introduce some related mathematical notation: Let $\mathcal{D}$ be the arbitrary-shaped image segment under consideration, and let $(x, y) \in \mathcal{D}$ denote the coordinates of the pixels of the region to be

coded. The *inner product* of functions $f(x,y)$ and $g(x,y)$ on $\mathcal{D}$ is defined as:

$$(f,g) \stackrel{\text{def}}{=} \sum_{(x,y)\in\mathcal{D}} f(x,y)g(x,y) \tag{5.18}$$

Therefore, $f(x,y)$ and $g(x,y)$ are *orthogonal* if $(f,g)=0$ holds, provided that none of the two functions are equal to zero. Moreover, the l_2 *norm* of function $f(x,y)$ can be expressed as $\|f\| \stackrel{\text{def}}{=} \sqrt{(f,f)}$. Notice that since only l_2 norm is considered here, sometimes, just the word "norm" is used for simplicity. The function $f(x,y)$ is said to be *normalized* if $\|f\| = \sqrt{(f,f)} = 1$.

Gram-Schmidt orthogonalization is a classical orthogonal functions generation algorithm. Starting from a linear independent set of functions $\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_n$ we generate the function

$$\mathbf{q}_1 = \mathbf{u}_1 - r_{01}\mathbf{n}_0, \quad \text{with} \quad \mathbf{n}_0 = \frac{\mathbf{q}_0}{\|\mathbf{q}_0\|} \quad \text{and} \quad \mathbf{q}_0 = \mathbf{u}_0 \tag{5.19}$$

By choosing

$$r_{01} = (\mathbf{u}_1, \mathbf{n}_0) \Rightarrow \mathbf{q}_1 = \mathbf{u}_1 - (\mathbf{u}_1, \mathbf{n}_0)\mathbf{n}_0 \tag{5.20}$$

we get that $\mathbf{q}_1$ is orthogonal to $\mathbf{q}_0$.

$$(\mathbf{q}_1, \mathbf{n}_0) = ((\mathbf{u}_1 - r_{01}\mathbf{n}_0), \mathbf{n}_0) = (\mathbf{u}_1, \mathbf{n}_0) - r_{01}(\mathbf{n}_0, \mathbf{n}_0) = 0 \tag{5.21}$$

The term $r_{01}\mathbf{n}_0$ yields the projection of $\mathbf{u}_1$ onto $\mathbf{q}_0$. The difference of $\mathbf{u}_1$ and its projection is normalized, yielding $\mathbf{n}_1 = \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|}$.

In general we can write

$$\mathbf{q}_k = \mathbf{u}_k - r_{0k}\mathbf{n}_0 - r_{1k}\mathbf{n}_1 - r_{2k}\mathbf{n}_2 - \cdots - r_{k-1k}\mathbf{n}_{k-1} \tag{5.22}$$

with $r_{lk} = (\mathbf{u}_k, \mathbf{n}_l)$ and $\mathbf{n}_l = \frac{\mathbf{q}_l}{\|\mathbf{q}_l\|}$.

The whole algorithm produces a set of *orthonormal* basis functions recursively. The number of orthogonal functions that we can generated is at most the number of linearly independent functions. On a discrete grid the maximum number of linear independent functions is equal to the number of the points within the segment.

5.2.3 Householder Polynomials

- 1D Householder Polynomials

Householder polynomials in the one-dimensional case are well known since the 50's [45]. These polynomials are widely used because the three-term recursive method shown below greatly simplifies the orthogonal polynomial generation procedure. Based on a set of data points $x_0, x_1, \cdots, x_n$, a set of orthogonal polynomials (Householder polynomials) $p_0(x), p_1(x), \cdots, p_n(x)$ can be generated as follows:

$$\begin{aligned} p_0(x) &= 1 \\ p_1(x) &= xp_0(x) - \alpha_1 p_0(x) \\ p_2(x) &= xp_1(x) - \alpha_2 p_1(x) - \beta_1 p_0(x) \\ \cdots &= \cdots \\ p_{i+1}(x) &= xp_i(x) - \alpha_{i+1} p_i(x) - \beta_i p_{i-1}(x), \quad i = 1, 2, \cdots, (n-1) \end{aligned} \tag{5.23}$$

Where,

$$\alpha_{i+1} = (xp_i, p_i)/(p_i, p_i), \quad \beta_i = (xp_i, p_{i-1})/(p_{i-1}, p_{i-1}) \tag{5.24}$$

(f, g) here denotes the corresponding inner product. With value of α and β computed by (5.24), it can be shown inductively that the Householder polynomials are orthogonal polynomials. The procedure can generate up to n, the number of

known data points, orthogonal polynomials. Moreover, since the polynomial $p_i(x)$ is a polynomial in x of degree i, the set of Householder polynomials produced is complete if the data points number $n \to \infty$.

- 2D Householder Polynomials

A *2D Householder Polynomials* generation algorithm is given in [40]. It imitates the three term recursive form of the 1D case. Following the previous assumptions, all of the mathematical operations are conducted within the constraint of a segment $\mathcal{D}$. The *order* of a polynomial $p(x, y)$ is the degree of $p(x, x)$. Let $q_{m,n} = x^m y^n$. A particular *sorting* of the polynomials indexed by (m, n) is defined via the relation: $(m, n) \prec (m', n')$ iff $(m + n) < (m' + n')$ or $(m + n) = (m' + n')$ and $n < n'$. The way the functions $q_{m,n}$ are sorted by increasing order, or by increasing powers of y, if the two functions have the same order. Let $S_{m,n}$ be the space spanned by the basis $Q_{m,n} = \{q_{i,j} : (i, j) \preceq (m, n)\}$. It is obvious that $S_{i,j} \subset S_{m,n}$ iff $(i, j) \prec (m, n)$. It is assumed that all $q_{i,j} \in Q_{m,n}$ are linearly independent. Consequently, the spaces $S_{i,j}$ are distinct. The following procedure produces 2D Householder polynomials, which are orthogonal basis functions of $S_{i,j}$, $i \geq 0$, $j \geq 0$:

$$\begin{aligned}
p_{0,0}(x, y) &= 1 \\
p_{m,0}(x, y) &= x p_{m-1,0}(x, y) - \sum_{(i,j)} \alpha_{i,j}^{m,0} p_{i,j}(x, y) \quad (m-2) \leq (i+j) < m \\
p_{m,n}(x, y) &= y p_{m,n-1}(x, y) - \sum_{(i,j)} \beta_{i,j}^{m,n} p_{i,j}(x, y) \quad (m+n-2) \leq (i+j) < (m+n) \\
\cdots &= \cdots
\end{aligned} \tag{5.25}$$

Where,

$$\alpha_{i,j}^{m,0} = \frac{(x p_{m-1,0}, p_{i,j})}{(p_{i,j}, p_{i,j})}, \quad \text{and} \quad \beta_{i,j}^{m,n} = \frac{(y p_{m,n-1}, p_{i,j})}{(p_{i,j}, p_{i,j})}, \quad m > 0,\ n > 0 \tag{5.26}$$

Fig, 5.1 below shows a computation comparison between Householder polynomial generation and Gram-Schmidt orthogonization algorithm. The figure on the left is corresponding to the generation of $p_{m,0}$, and the on the right to $p_{m,n}$, $n > 0$. The symbol **N** indicates the position of the orthogonal basis to be computed, a symbol **X** indicates that this inner product needs to be calculated to compute $p_{m,0}$ or $p_{m,n}$. But both the • and **X** symbols indicate inner products which are needed to be calculated when Gram-Schmit is used for orthogonizing the initial polynomials. Therefore, the inner products in the dots triangle which are needed to be calculated in Gram-Schmidt are not necessary in the calculation of the 2D Householder polynomials. Similar results can also be found in [41].

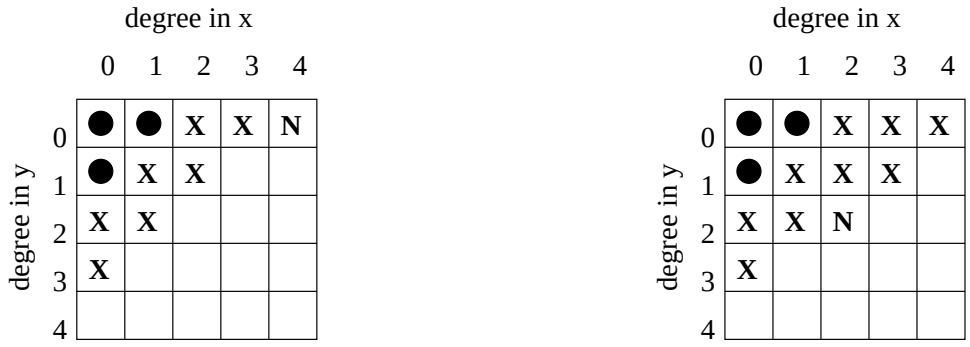


Figure 5.1: Inner products in Householder vs. Gram-Schmidt
איור 5.1: ביצוע מכפלות פנימיות באלגוריתמים Householder ו-Gram-Schmidt

5.2.4 Weakly Separable Basis Functions

The authors in [42] further proposed weakly separable basis functions, which also follows the three term recursive context. But, because of their *weak separability* property, which is discussed later on, the complexity of the orthogonal basis function generation process is further reduced, as compared with Householder Polynomials.

Since we did not have enough time to implement this algorithm. The description provided below is almost totally based on the above reference. As described in [42], the weakly separable functions $\{P_{m,n}(x,y)\ (m,n) \in I\}$ are generated to be *orthonormal* with each other on a certain image segment $\mathcal{D}$, containing p pixels. So, as we discussed before, $g(x,y) = \sum_{(m,n)\in I} a_{m,n} P_{m,n}$ approximates the original image intensity $f(x,y)$ in the LS sense. When the number of basis functions equals to the number of pixels, p, an error free reconstruction is obtained. The region $\mathcal{D}$ can be expressed as the union of n_r different rows $R_k \stackrel{\text{def}}{=} \{(x,y) \in \mathcal{D} : y = y_k\}$, $0 \le k < n_r$. Let r_k be the number of pixels in R_k and c_k the number of rows with at least $k+1$ pixels. Without loss of generality, we assume that the rows are indexed such that $r_i \le r_j$, if $i > j$. By performing orthogonization of the initial basis functions $x^m y^n$ with $0 \le n < n_r$ and $0 \le m < r_n$ ordered lexicographically as $1, y, y^2, \cdots, x, xy, xy^2, \cdots$, the generated weakly separable polynomials satisfy the following equations:

$$\begin{aligned} P_{0,0} &= 1/\mathcal{N}_{0,0} \\ P'_{m,n}(x,y) &= yP_{m,n-1}(x,y) - \sum_{l=1}^{2} \alpha^l_{m,n} P_{m,n-l}(x,y) \\ P_{m,n}(x,y) &= P'_{m,n}(x,y)/\mathcal{N}_{m,n} \end{aligned} \tag{5.27}$$

$$\text{where, } \alpha^l_{m,n} = \begin{cases} (yP_{m,n-1}, P_{m,n-l}) & l \le n \\ 0 & others \end{cases} \quad \text{and } \mathcal{N}_{m,n} = \|P'_{m,n}\| \tag{5.28}$$

A fast implementation to generate polynomials $P_{m,n}$ is possible. It relies on the weak separability property of the polynomials:

$$P_{m,n}(x,y) = P_{m,0}(x,y) Q_{m,n}(y) \tag{5.29}$$

Where $Q_{m,n}(y)$ is a polynomial of degree n in y, and the property is easily shown by induction on n. As shown in [42], $P_{m,0}(x,y_k)$ is proportional to $p_{m,k}(x)$ on R_k,

therefore,

$$P_{m,n}(x, y_k) = p_{m,k}(x) q_{m,n}(y_k) \tag{5.30}$$

Where, $q_{m,n}(y) = \sqrt{w_m(y)} Q_{m,n}(y)$.

The two terms $p_{m,k}$ and $q_{m,n}$ can be calculated recursively by the respective equations.

$$\begin{aligned} p_{0,0}(x) &= 1/N_{0,0} \\ p'_{m,k}(x) &= x p_{m-1,k}(x) - \sum_{l=1}^{2} \beta^l_{m,k} p_{m-l,k}(x) \\ p_{m,k}(x) &= p'_{m,k}(x)/N_{m,k} \end{aligned} \tag{5.31}$$

$$\begin{aligned} q_{m,0}(y) &= \sqrt{w_m(y)}/\mathcal{N}_{m,0} \\ q'_{m,n}(y) &= y q_{m,n-1}(y) - \sum_{l=1}^{2} \alpha^l_{m,n} q_{m,n-l}(y) \\ q_{m,n}(y) &= q'_{m,n}(y)/\mathcal{N}_{m,n} \end{aligned} \tag{5.32}$$

With $(f, g)_{r,k} \stackrel{\text{def}}{=} \sum_{x \in R_k} f(x) g(x)$ denoting a row inner product;

While the value for $\alpha^l_{m,n}$ and $\mathcal{N}_{m,n}$ is the same as shown in (5.27) and (5.28); $N_{m,k} = \|p'_{m,k}\|_{r,k}$.

$$\beta^l_{m,k} = \begin{cases} (x p_{m-1,k}, p_{m-l,k})_{r,k} & l \le m \\ 0 & \text{otherwise} \end{cases} \tag{5.33}$$

Here, $w_m(y) = s^2_{m,k}$, where $s_{m,k} = t_{m,k}/T_m$, $t_{m,k} = \Pi_{j=0}^{m} N_{j,k}$ and $T_m^2 = \sum_{k=0}^{c_m - 1} t^2_{m,k}$.

In general, to compute the same number of basis functions, the algorithm used to compute weakly separable basis functions is 9 to 60 times faster than Householder polynomial, and 15 to 250 times faster than Gram-Schmidt (segment dependent),

but the corresponding root-mean-square error between the original and approximation functions is increased by only 3% to 6% as compared to Gram-Schmidt or Householder methods. A detailed discussion can be found in [42].

5.2.5 Shape-Independent Basis Functions

This method uses basis functions which are linear independent (or orthogonal) on the rectangular area $\mathcal{R}$ that circumscribes the image segment $\mathcal{D}$. Previous methods tended to pad zero values for pixels outside of the image segment, so that a unique solution to the normal equations $g(x,y) = \sum_{m,n} a_{m,n}\phi_{m,n}(x,y)$ is obtained. $g(x,y)$ is the LS approximation to the original image function $f(x,y)$. $\phi_{m,n}(x,y)$ are linear independent basis functions on the rectangle $\mathcal{R}$. Thus, the approximation is heavily dependent on the rectangle, which is not very suitable for region-based image coding. The most recent published development we found [43] adopted an approach which is similar to the *Matching Pursuit* [46] algorithm developed for adaptive signal expansion. The algorithm is described as follows:

With a set of predefined basis functions $\phi_{m,n}(x,y)$, which are linear independent on the rectangle $\mathcal{R}$, the algorithm uses the criterion

$$\triangle E_{\mathcal{D}}^{(\ell)} = \frac{[\sum_{(x,y)\in\mathcal{D}} r^{(\ell)}(x,y)\phi_{u,v}(x,y)]^2}{\sum_{(x,y)\in\mathcal{D}} \phi_{u,v}^2(x,y)} \rightarrow \min \tag{5.34}$$

to select the specific basis function $\phi_{i,j}(x,y)$ in the current iteration (ℓ), and add it to the set $\mathcal{K}_\ell \stackrel{\text{def}}{=} \{\phi_{m,n} \text{ selected in iteration } \ell = 1, \cdots\}$. Where $r^\ell(x,y) \stackrel{\text{def}}{=} f(x,y) - g^\ell(x,y)$ denotes the residual error after ℓ iterations. Thus, $\mathcal{K}_{\ell+1} = \mathcal{K}_\ell \bigcup \{\phi_{m,n}\}$ if $\phi_{m,n} \notin$

$\mathcal{K}_\ell$. By demanding that the following set of equations holds:

$$\sum_{(m,n)\in\mathcal{K}_{\ell+1}} \triangle a_{m,n} \sum_{(x,y)\in\mathcal{D}} \phi_{m,n}(x,y)\phi_{k,l}(x,y) = \sum_{(x,y)\in\mathcal{D}} r^\ell(x,y)\phi_{k,l}(x,y) \;\forall (k,l) \in \mathcal{K}_{\ell+1} \tag{5.35}$$

the coefficients $\triangle a_{m,n}$ can be calculated. Hence the approximation coefficients $a^\ell_{m,n}$ are modified by:

$$a^{\ell+1}_{k,l} = \begin{cases} a^\ell_{k,l} + \triangle a_{k,l} & \text{for } \phi_{k,l} \in \mathcal{K}_\ell \\ a^\ell_{k,l} & \text{else} \end{cases} \tag{5.36}$$

Normally, the initial approximation $g^0(x,y) = 0 \;\; \forall (x,y) \in \mathcal{R}$ is chosen. The algorithm includes the constraint that the basis function selected in each iteration should be linearly independent of the already selected functions on image segment $\mathcal{D}$. Thus, the convergence of the iteration is guaranteed with the additional basis function definitely reduces the residual energy in each step.

As an algorithm that follows the matching pursuit context, we may assume it suffers the same drawbacks [31]. That is, as an iterative algorithm, it does not explicitly seek any overall goal, but merely applies a simple rule repeatedly, and it may fail to retrieve the best answer. Some other problems are the determination of the proper initial set of basis functions $\phi_{m,n}$; their linear independence property on $\mathcal{D}$, and proper initial approximation function $g(x,y)$, etc. Besides, the computational load is quite considerable if the basis functions are not orthogonal.

5.3 Simulation Results

Shown in Fig. 5.2 and Fig. 5.3 are examples of segmented images represented by low order orthogonal basis functions. Obviously, to generate images with acceptable perceptual quality, higher order orthogonal basis functions should be added. But higher order could not be generated because of numerical problems that we encountered. Fig. 5.4 shows the residual images. Clearly, there is still too much information left in them. This means that the residual image needs also to be coded to obtain a reconstructed image which is perceptually acceptable. The coding of the residual image is very costly [47].

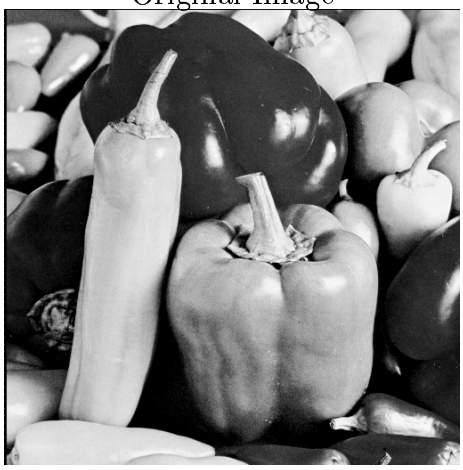
5.4 Discussion

5.4.1 Disadvantages of Shape-Adaptive Orthogonal Basis Functions

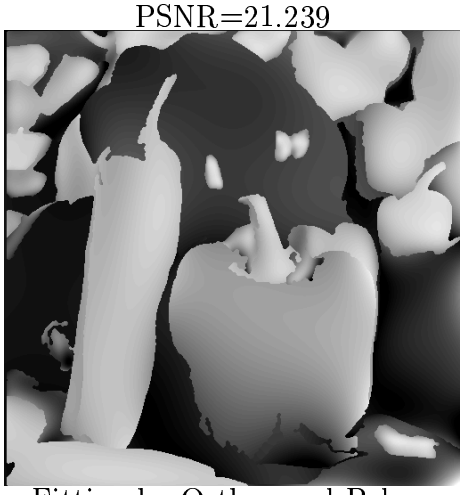
As we mentioned before, the region-based approach tends to exploit the global characteristics of the whole segment. Theoretically, it has a higher potential for compression. Moreover, because of the simplicity and the stability in the calculation of the approximation coefficients when orthogonal basis functions are used, it seems to be an appealing method for arbitrary shape image segment coding. In practice, unfortunately, currently available methods have not produced satisfying result, as shown in section 5.3. Similar results can be found in the general summary in [48].Their drawbacks are discussed as follows:

1. Complexity: The computational load is one of the most important factors that we have to keep an eye on in a real application. When orthogonal basis

Original Image

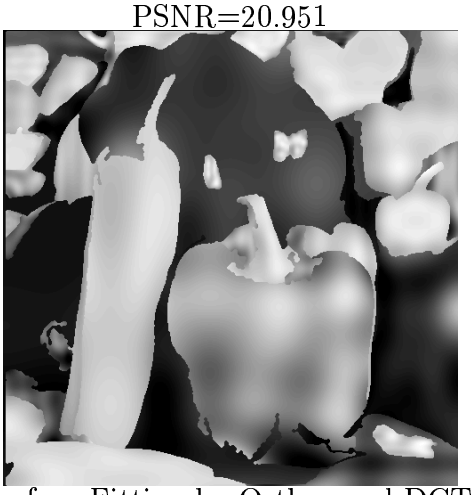


PSNR=21.239



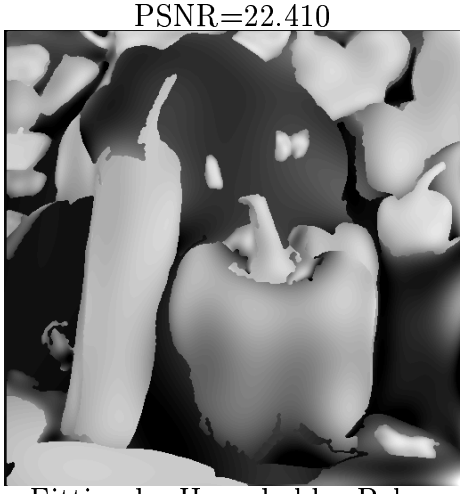
Surface Fitting by Orthogonal Polynomial

PSNR=20.951



Surface Fitting by Orthogonal DCT

PSNR=22.410



Surface Fitting by Householder Polynomial

PSNR=17.971

Surface Fitting by Region Mean

Figure 5.2: "Peppers" Image Coding Results

"Peppers" איור 5.2: תוצאות הקידוד של התמונה

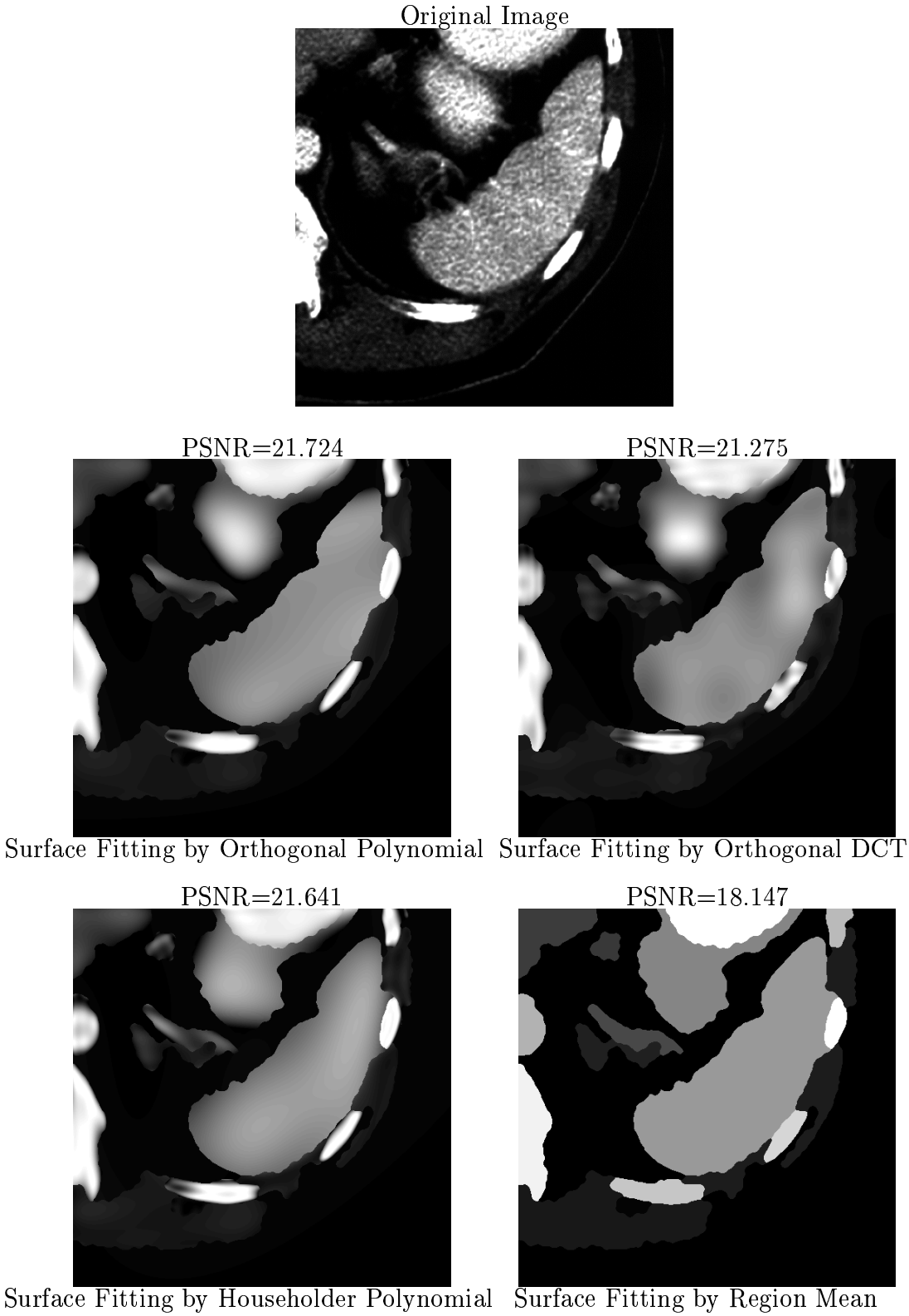


Figure 5.3: "Medical" Image Coding Results
"Medical" איור 5.3: תוצאות הקידוד של התמונה

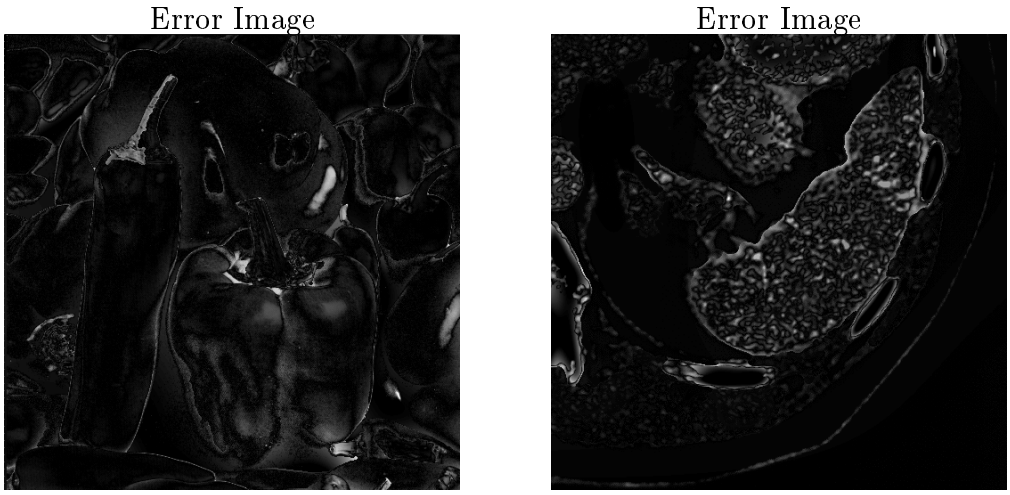


Figure 5.4: Absolute Difference between Orignal & Orthogonal Polynomial Surface Fitting

איור 5.4: ערך מוחלט של ההפרש בין התמונה המקורית והתמונות המקודדות ע"י פולינומים אורתוגונליים

functions are generated by the orthogonization of initial 2D basis functions, the procedure is computationally expensive both with respect to memory requirements and computation time needed. Additionally, for every image segment, the corresponding set of orthogonal basis functions needs to be recalculated, which may be tens or hundreds of times the computation demand corresponding to specific segment.

2. Improper generation caused by initial functions which are not linearly independent: It is assumed that the initial basis functions are linear independent on the specific image segment. However, in practice, this property is not always guaranteed. Hence, it is necessary to check the linear independence (orthogonality) before (after) the orthogonal basis function is generated and reject the improper basis functions. This, in turn, introduces extra computations.

3. Limited order basis functions can be generated: Besides all of the problems stated above, in practice, only low order basis functions can be generated by the Gram-schmidt orthogonalization method, because of the round-off calculation error. This means that a textured segment can not be well approximated because the number of basis functions used is not sufficient. Although the use of Householder polynomials (or Weakly separable polynomials) reduces the computation complexity and enables the generation of more basis functions (depending on the actual segment data). Yet, both the PSNR and the perceptual quality of the reconstructed segment drop with the use of these basis functions [42], as compared with those by Gram-Schmit. Thus, currently, orthogonal basis functions coding do not seem to be a promising shape-adaptive method.

5.4.2 Mixing of Block-Based and Region-Based Approaches

Another issue that we would like to address is the mixing of these two algorithm categories. One way is to apply region-based algorithms which use orthogonal basis functions to the boundary-block data. An experiment in which we applied this approach was not successful. We think that the reason is that the data within a 8×8 boundary block is too limited for an efficient representation. Another possible mixing of approaches is to do it the other way around. That is, applying a data extrapolation method, such as BP extrapolation, to the rectangle which circumscribes a whole image segment. Theoretically, a higher compression is achievable, since the extrapolation method exploits the global characteristics. Practically, the obstacle in applying BP is the high complexity of the linear programming solution. In our experiment, when the circumscribing rectangle is larger than 50×50, the BP extrapolation

method could not practically be solved.

Chapter 6

Conclusions & Further Studies

In this research, we examine three image segmentation schemes and two approaches to shape-adaptive image coding. The conclusions of our study are summarized below, followed by suggestions for further studies on these issues.

6.1 Conclusions

Image segmentation serves as an important preprocessing tool for shape-adaptive image coding. In this research, two mathematical morphology based segmentation algorithms [13, 14] are studied. However, simulation results show that these two image segmentation schemes do not produce good enough results for coding. Therefore, a new image segmentation scheme – *edge detection, local-characteristic classification*, is proposed. Segmentation results indicate that the proposed scheme produces adequate segmentation for the coding task. The original image is partitioned wherever a steep edge exists between adjacent regions, and regions with similar perceptual characteristics are clustered into one segment. The resulting segmentation tends to

reduce the following texture (content) and contour coding cost. Another advantage that the proposed segmentation algorithm achieves is a reduction in computation time, as compared to those in [13, 14].

The second part of this research focuses on the texture coding part of shape-adaptive image coding. Two approaches are studied, a block-based approach and a region-based approach. Studies are conducted on SADCT and LPE padding proposed in a MPEG-4 release [7]. Motivated by the data extrapolation idea of the LPE, we further propose a so called BP data extrapolation method. Simulation results show that available block-based shape-adaptive coding methods achieve higher compression ratios when they are applied to images characterized by steep edges between image segments. Block-based methods are compatible with existing 2D-DCT based coding systems, and the computation procedure involved is relatively simple. Therefore, current research on this topic is quite active [28, 49, 50, 51]. However, since this approach still follows conventional image coding techniques, the compression potential provided by segmentation is not fully exploited. The compression advantage of BP extrapolation over SADCT and LPE padding is demonstrated. But, since BP is based on a linear programming solution, it is quite more complex than the other techniques.

Theoretically, region-based methods, which use orthogonal basis functions for content description, have a higher compression potential. However, numerical problems that we encountered during our application, which are discussed in section 5.2, indicate that this technique needs to be improved before it becomes practical [44].

6.2 Further Studies

Segmentation-based image coding is a relatively new research field. To achieve the expected better reconstruction quality over conventional techniques such as JPEG, related techniques still need to be improved, as we discuss in the previous chapters. This research work achieved some positive results, and we suggest here some directions for further studies. The discussion is divided into two parts concerning extensions in both segmentation and shape-adaptive coding.

In segmentation, as we discussed in section 3.3, with the further study of region refinement, we can expect the proposed algorithm to become a generic approach to still image segmentation. Another more general extension is to video sequence segmentation. Since the amount of computation time of the available algorithm is greatly reduced, it can be used for the intra-frame image segmentation, by exploiting the connection of the same segment between consecutive frames, a video sequence image segmentation system could be generated.

In shape-adaptive image coding of still images, the achievements are not significant. The techniques that we discussed, SADCT and LPE, are actually more suitable for image sequence (video) coding. The reasons are: In video coding, each frame in the sequence is separated into foreground and background segments, so that only the coding of the foreground segments needs to be considered. Therefore, applying shape-adaptive techniques can achieve better results. Particularly, by using shape-adaptive methods to code the residual image generated after motion compensation, even more significant rate reduction is expected. This claim relies on the fact that the *relative* number of boundary blocks in the set of blocks that need to be coded is much larger, as compared with still image coding.

Another issue that could be addressed is the possibility of region-based BP extrapolation. Even though the experiment discussed in section 5.3 encountered numerical problems and hence was not successful, we believe that with the continuous progress in large scale linear programming, this application is to be possible. Following BP extrapolation the coding can be performed as in [43].

Appendix A

JPEG Baseline Codec

Two main reasons prompt us to give a brief review of the standard JPEG coding system. First, as a generally recognized standard, JPEG provides a reference bit-rate versus reconstruction quality coding output which can be used for comparison with other coding schemes examined in this work. In addition, the core of JPEG, the 8×8 DCT-based transform coding system can be used as a encoding/decoding system where the transform coding stage is substituted by the schemes we study. Generally speaking, JPEG, the abbreviation of "Joint Photographic Expert Group", has established to compress still, continuous-tone, monochrome and color images. Its possible bit rates and qualities are:

- $0.25 - 0.5$ bits/pixel: moderate to good quality, sufficient for some applications;
- $0.5 - 0.75$ bits/pixel: good to very good quality, sufficient for many applications;
- $0.75 - 1.5$ bits/pixel: excellent quality, sufficient for most applications;
- $1.5 - 2.0$ bits/pixel: usually indistinguishable from the original, sufficient for the most demanding applications.

There are four main processing modes, i.e. *sequential*, *progressive*, *lossless*, and *hierarchical* encoding modes. For our particular application, only the sequential encoding mode[1] is considered. Specifically, the JPEG baseline codec for gray-scale source image encoding. We skip several accessory functions, to concentrate on the crucial part. The general block diagram of a DCT-based transform coding scheme is given in Fig. A.1.

At the input side of the encoder, source image samples are grouped into 8×8 blocks, and sequential operations are conducted in a block-wise manner. For a specific block, data is transformed into the frequency domain by a forward DCT (FDCT), quantization is performed for data reduction. The encoding ends with lossless entropy coding, which further compresses the data transmitted to the receiver. The decoding is an inverse process of the encoding: entropy decoding, dequantization and IDCT are performed. From the application point of view, users need to specify tables used for quantizer and entropy coder. JPEG also provides default tables for general usage. To make things clear, let's take a deeper look at the operations we mentioned above.

A.1 8×8 FDCT & IDCT

The input source image block (8×8 matrix) is shifted from unsigned integers in the range $[0,\ 2^8 - 1]$ to signed integers in the range $[-2^{8-1},\ 2^{8-1} - 1]$, in order to eliminate the DC bias. Pixel-wise forward DCT (FDCT) is computed at the encoder, and inverse DCT (IDCT) at the decoder, correspondingly. The following equations are the mathematical definitions of the 8×8 FDCT and 8×8 IDCT.

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \qquad \text{(A.1)}$$

[1] Each image component is encoded in a single left-to-right, top-to-bottom scan.

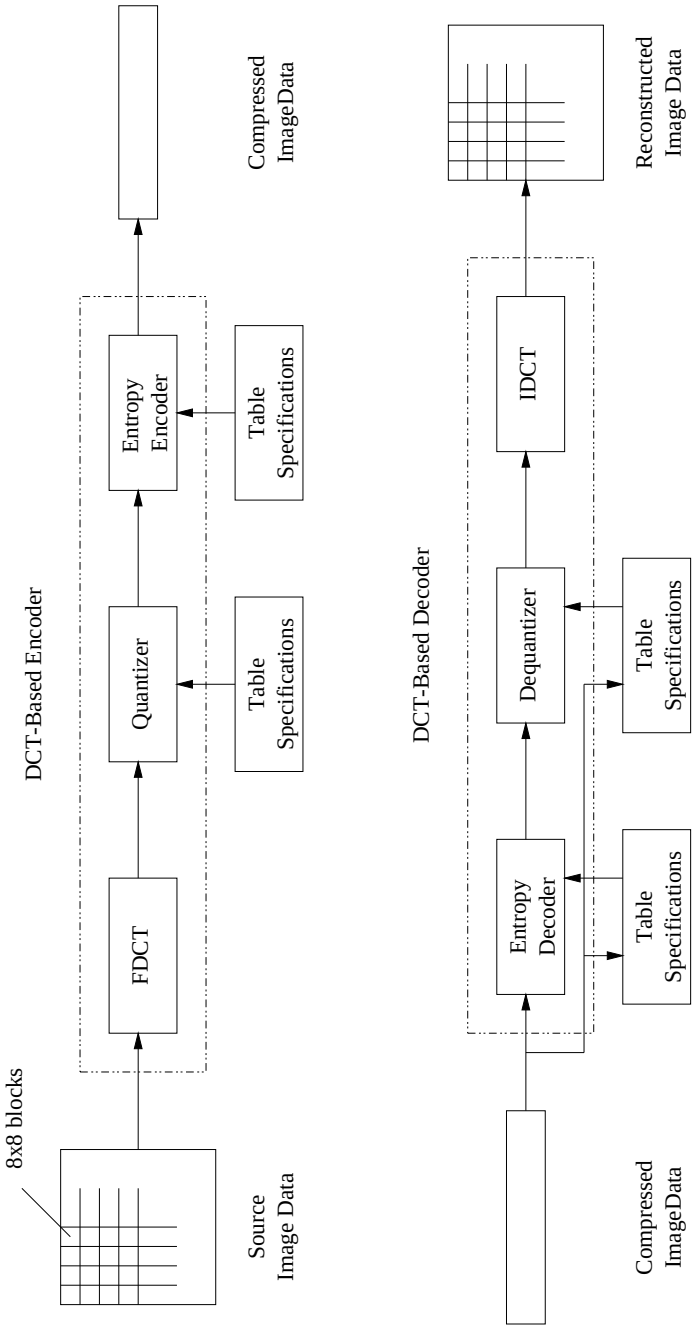


Figure A.1: DCT-Based Encoder & Decoder Processing Steps

איור 1.א: שלב העיבוד בקידוד/פענוח מבוסס DCT

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v) * \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \qquad \text{(A.2)}$$

where $\begin{cases} C(u), C(v) = 1/\sqrt{2} & \text{for u, v=0} \\ C(u), C(v) = 1 & \text{otherwise} \end{cases}$. The FDCT takes an 8×8 input signal, and decomposes it into 64 orthogonal basis signals. The output, DCT coefficients can thus be regarded as the relative amount of the 2D-DCT spatial frequencies contained in the 64-point input signal. Consequently, the coefficient with zero frequency in both dimensions is called the *DC coefficient* and the remaining 63 coefficients are called the *AC coefficients*. There are several advantages of 2D-DCT over other transforms such as DFT etc. The main advantages are: the transform is real, has a fast implementation and a high energy compaction in the spectral domain[2]. Therefore, the 2D-DCT is chosen as the transform method used in JPEG and in many other transform coding techniques.

A.2 Quantization

Because sample values typically vary slowly from point to point across an image, after the FDCT is performed, data compression is achieved by utilizing the property that most of the block energy is concentrated in the low spatial frequency. Most of the spatial frequencies have zero or near-zero amplitude and need not be encoded. The 64 DCT transform coefficients are quantized by uniform quantizers specified in advance. Quantization is the principal cause of lossiness in DCT-based encoders, so for high quality reconstruction, it is crucial to choose the "just noticeable difference"

[2]In fact, the performance of the 2D-DCT is generally less competitive than the KL transform, but it possesses the advantage of low computation requirement.

value as the quantization step. JPEG provides us with default tables which are based on empirical results of psycho-visual experiments. The quantization (respectively dequantization) is defined as follows:

$$F^Q(u,v) = \text{Integer Round}\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{A.3}$$

$$F^{Q'}(u,v) = F^Q(u,v) * Q(u,v) \tag{A.4}$$

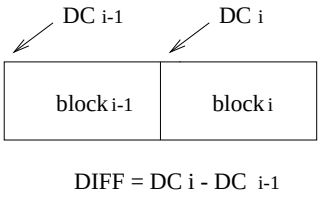
$F(u,v)$ is the input DCT coefficient, $Q(u,v)$ the quantization step and $F^Q(u,v)$, $F^{Q'}(u,v)$ are the output of quantizer and dequantizer, respectively.

A.3 DC Coding

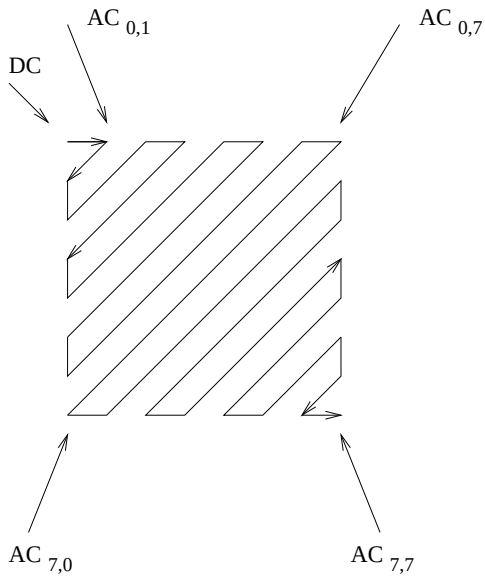
The DC coefficient indicates the average value of the 64 image samples, and usually there is a strong correlation between the DC coefficients of adjacent 8×8 blocks. Therefore, the quantized DC coefficient is encoded as the difference from the DC term of the previous block (DPCM encoding) in the encoding order shown on the left of Fig. A.2.

A.4 Zig-Zag Scan & Entropy Coding

The quantized coefficients are ordered by scanning the block in a zig-zag manner, as shown on the right of Fig. A.2. This ordering helps to facilitate entropy coding by grouping the low frequency coefficients (which are more likely to be nonzero) and placing them before high-frequency coefficients. The sequential entropy coding is lossless and achieves additional compression. The JPEG standard specifies two



Differential DC encoding



Zig-zag sequence

Figure A.2: Preparation of Quantized Coefficients for Entropy Coding
איור 2.א: הכנה לקידוד אנטרופיה של המקדמים לאחר קוונטיזציה

entropy coding methods: *Huffman Coding* and *Arithmetic Coding*. The baseline sequential codec uses *Huffman coding*.

Huffman coding requires that Huffman code tables be specified by the application. The same tables used to compress an image are needed for decompression. Huffman tables may be predefined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression. Concerning our application, the recommended Huffman table by JPEG is chosen (except for BP).

In this section, a very brief description of JPEG was given, interested readers may refer to the standard document to get detailed information about JPEG [1, 29]. Generally speaking, when the bit-rate is higher than 0.5 bits/pixel, using JPEG (or similar block-based methods) the reconstructed image quality is perceptually quite acceptable.

References

[1] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consumer Electronics*, vol. 38, pp. 18–34, Feb. 1992.

[2] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-generation image-coding techniques," *Procee. IEEE*, vol. 73, pp. 549–573, April 1985.

[3] R. Koenen, "Overview of the MPEG-4 standard," tech. rep., ISO/IEC JTC1/SC29/WG11, July 1998.

[4] P. Salembier, F. Marques, and M. Pardas, "Segmentation-based video coding system allowing the manipulation of objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 60–74, Feb. 1997.

[5] M. Gilge, T. Engelhardt, and R. Mehlan, "Coding of arbitraily shaped image segments based on a generalized orthogonal transform," *Signal Process. Image Commun.*, vol. 1, pp. 153–180, Oct. 1989.

[6] T. Sikora and B. Makai, "Shape-adaptive DCT for generic coding of video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 59–62, Feb. 1995.

[7] T. Ebrahimi, "MPEG-4 video verification model version 10.0," tech. rep., ISO/IEC JTC1/SC29/WG11, Feb. 1998.

[8] M. Tabb and N. Ahuja, "Multiscale image segmentation by integrated edge and region detection," *IEEE Trans. Image Process.*, vol. 6, pp. 642–655, May 1997.

[9] A. Kaup and T. Aach, "Stochastic model-based image segmentation using functional approximation," *IEICE Trans. Fund. Electron., Commun., Comput. Sci.*, vol. E77-A, pp. 1451–1456, Sept. 1994.

[10] A. Elmoataz, S. Schupp, R. Clouard, and P. Herlin, "Using active contours and mathematical morphology tools for quantification of immunohistochemical images," *Signal Process.*, vol. 71, pp. 215–226, 1998.

[11] J. Vaisey and A. Gersho, "Image compression with variable block size segmentation," *IEEE Trans. Signal Process.*, vol. 40, pp. 2040–2060, Aug. 1992.

[12] J. G. Choi, S.-W. Lee, and S.-D. Kim, "Spatio-temporal video segmentation using a joint similarity measure," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 279–286, Apr. 1997.

[13] P. Salembier, L. Torres, F. Meyer, and C. Gu, "Region-based video coding using mathematical morphology," *Proc. IEEE*, vol. 83, pp. 843–857, June 1995.

[14] D. Wang, C. Labit, and J. Ronsin, "Segmentation-based motion-compensated video coding using morphological filters," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 549–555, June 1997.

[15] J. M. Gauch, "Image segmentation and analysis via multiscale gradient watershed hierarchies," *IEEE Trans. Image Process.*, vol. 8, pp. 69–79, Jan. 1999.

[16] E. R. Dougherty, *An Introduction to Morphological Image Processing.* SPIE Optical Engineering Press, 1992.

[17] R. Kresch, *Morphological Image Representation for Coding Applications.* PhD thesis, Technion–Israel Institute of Technology, 1995.

[18] L. Vincent and P. Soille, "Watershed in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 583–598, June 1991.

[19] P. Salembier and M. Pardas, "Hierarchical morphological segmentation for image sequence coding," *IEEE Trans. Image Process.*, vol. 3, pp. 639–651, Sept. 1994.

[20] C. Gu, *Multivalued Morphology and Segmentation-Based Coding.* PhD thesis, EPFL, Lausanne, 1995.

[21] S. H. Lee, D.-S. Cho, Y.-S. Cho, and S. H. Son, "Binary shape coding using baseline-based method," *IEEE Trans. Circuits Syst. video Technol.*, vol. 9, pp. 44–58, February 1999.

[22] F. Marques, J. Sauleda, and A. Gasull, "Shape and location coding for contour images," *Picture Coding Symp.*, 1993.

[23] P. Salembier, "Multi-criterion segmentation for image coding," in *1st Workshop on Math. Morph. and its Appl. to Signal Process.*, pp. 43–45, May 1993.

[24] J. Pandel, "Variable bit-rate image sequence coding with adaptive quantization," *Signal Process. Image Commun.*, vol. 3, pp. 123–128, 1991.

[25] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall International, Inc., 1989.

[26] C. Gu and M. Kunt, "Contour simplification by a new nonlinear filter for region-based coding," in *VCIP-94*, vol. 2308, pp. 1180–1191, Sept. 1994.

[27] V. A. Christopoulos, P. D. Muynck, and J. Cornelis, "Contour simplification for segmented still image and video coding: Algorithms and experimental results," *Signal Process. Image Commun.*, vol. 14, pp. 335–357, 1999.

[28] H. Katata, N. Ito, T. Aono, and H. Kusao, "Object wavelet transform for coding of arbitrarily shaped image segments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 234–237, Feb. 1997.

[29] I.-T. R. T.81, "Information technology – Digital compression and coding of continuouse-tone still image: Requirements and guidelines," tech. rep., ISO/IEC JTC1 10918-1, 1994.

[30] J. Ostermann, E. S. Jang, J.-S. Shin, and T. Chen, "Coding of arbitrarily shaped video objects in MPEG-4," in *ICIP*, 1997.

[31] S. S. Chen, *Basis Pursuit.* PhD thesis, Stanford University, 1995.

[32] M. L. Puterman, *Markov Decision Process.* Wiley-inerscience. New York, 1994.

[33] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University Press, 1992.

[34] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Trans. Inform. Theory*, vol. 38, pp. 713–718, March 1992.

[35] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June 1987.

[36] M. Eden and M. Unser, "Polynomial representation of pictures," *Signal Process.*, vol. 10, pp. 385–393, June 1986.

[37] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inf. Theory*, pp. 179–187, 1962.

[38] M. Schlatter and J. Eichler, "An introduction to the Gaussian least squares approximation and its application in signal processing and system modeling," *Signal Process.*, vol. 1, pp. 211–225, July 1979.

[39] J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Domain.* American Mathematical Society, Providence, RI, 1965.

[40] M. Weisfeld, "Orthogonal polynomials in several variables," *Numerische Mathematik Bd.*, vol. 1, pp. 38–40, 1959.

[41] W. Philips, "A fast algorithm for the generation of orthogonal base functions on an arbitrarily shaped region," in *ICASSP*, pp. 421–424, 1992.

[42] W. Philips, "Fast coding of arbitrarily shaped image segments using weakly separable bases," *Opt. Eng.*, vol. 35, pp. 177–186, Jan. 1996.

[43] A. Kaup and T. Aach, "Coding of segmented images using shape-independent basis functions," *IEEE Trans. Image Process.*, vol. 7, pp. 937–947, July 1998.

[44] R. Stasinski and J. Konrad, "A new class of fast shape-adaptive orthogonal transforms and their application to region-based image compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 16–34, Feb. 1999.

[45] G. E. Forsythe, "Generation and use of orthogonal polynomials for data-fitting with a digital computer," *J. Soc. Indust. Appl. Math.*, vol. 5, pp. 74–88, June 1957.

[46] S. G. Mallat, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, pp. 3397–3415, Dec. 1993.

[47] M. J. Weinberger, J. Rissanen, and R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Process.*, vol. 5, pp. 575–586, Apr. 1996.

[48] C. A. Christopoulos, W. Philips, A. N. Skodras, and J. Cornelis, "Segmented image coding: Techniques and experimental results," *Signal Process. Image Commun.*, vol. 11, pp. 63–80, 1997.

[49] J.-H. Moon, J.-H. Kweon, and H.-K. Kim, "Boundary block-merging (bbm) technique for efficient texture coding of arbitrarily shaped object," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 35–43, 1999.

[50] P. Kauff and K. Schuur, "Shape-adaptive DCT with block-based dc separation and $\triangle$DC correction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 237–242, June 1998.

[51] J.-W. Yi, S.-J. Cho, W.-J. Kim, S.-D. Kim, and S.-J. Lee, "A new coding algorithm for arbitrarily shaped image segments," *Signal Process. Image Commun.*, vol. 12, pp. 231–242, 1998.

קידוד תמונות מבוסס-חלוקה ותלוי צורה

ג׳או יינג

קידוד תמונות מבוסס-חלוקה ותלוי צורה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת תואר

מגיסטר למדעים

בהנדסת חשמל

ג'או יינג

הוגש לסנט הטכניון — מכון טכנולוגי לישראל

אייר תשנ"ט חיפה מאי 1999

חיבור על מחקר נעשה בהדרכת פרופסור ד. מלאך

בפקולטה להנדסת חשמל

הכרת תודה

אני אסירת תודה לפרופ' דוד מלאך, עבור השגחתו המסורה, עצותיו הנבונות ועזרתו הנדיבה.

ברצוני להודות לצוות המעבדה לעיבוד אותות ותמונות ולחברי ללמוד-ים. תודה עבור האהבה והחברות, שליוו אותי במשך השנים בהן שהיתי בישראל.

ברצוני גם להודות למשפחתי עבור העידוד ובטחונם בי.

אני מודה לליידי דייביס ולמעבדה לעיבוד אותות ותמונות על התמיכה הכספית הנדיבה בהשתלמותי

מוקדש למשפחתי

תוכן עניינים

רשימת איורים

רשימת טבלאות

תקציר

עבודה זו עוסקת בקידוד תמונות בגישה המבוססת על חלוקת התמונה לאזורים (סגמנטים) וקידוד כל איזור בנפרד. גישה זו שייכת ל"דור השני של קידוד תמונות" (Second Generation Image Coding) והיא זוכה לתשומת לב מרובה בשנים האחרונות.

המוטיבציה העקרית לפיתוח התחום של "קידוד תלוי צורה" נובעת מהבעיה הבאה: בקידוד כגון JPEG, המיושם על-ידי 2D-DCT, כשנדרש קידוד בקצב סיביות נמוך, נדרשת קוונטיזציה גסה. עקב כך נגרמות אי רציפויות ב"רמות האפור" (gray-level) בין בלוקים סמוכים. הדבר מתבטא בתופעה הידועה כ"תופעת הבלוקיות" (blocking effect) בתמונה המשוחזרת.

על מנת להתגבר על "תופעת הבלוקיות", קידוד תמונות מבוסס-חלוקה ותלוי צורה מנצל את התכונה המיוחדת של מערכת הראייה האנושית (HVS) של מיון הפיקסלים בתמונה לקבוצות אשר מוגדרות כאיזורים או סגמנטים. כל איזור כזה יכול להיות שטוח (flat) , בעל מרקם (texture) או כולל שפות (edges). מערכת הראיה האנושית נוטה להיות רגישה להפרעות בשפות בעוד שהפרעות בתוך האיזור (עצם) מורגשות פחות. בהתאם לכך, העקרון של קידוד תמונות מבוסס-חלוקה ותלוי צורה הינו בראש ובראשונה ביצוע סגמנטציה אשר מחלקת את התמונה המקורית לאיזורים, באופן הדומה לחלוקה המתבצעת על-ידי העין האנושית.

המידע הכלול בתוך הסגמנט מקודד על-ידי טכניקה הנקראת "קידוד מרקם" (texture coding), כך שיתאפשר להציג מידע זה באופן קומפקטי. גם אם טכניקת הקידוד נשענת

על טכניקת החלוקה לבלוקים, ניתן עדיין להשיג הפחתה בקצב הסיביות על-ידי סווג לש-לושה סוגי בלוקים עיקריים: בלוקים הנמצאים בתוך הסגמנט, בלוקים על שפת הסגמנט, כך שרק חלק מכל בלוק נמצא בתוך הסגמנט, ובלוקים הנמצאים מחוץ לסגמנט. קידוד הבלוקים מהסוג הראשון מתבצע ע"י קוונטיזציה גסה של מקדמי ה-DCT, ואילו קידוד הבלוקים מהסוג השני, כלומר אלה שעל שפת הסגמנט , יפורט בהמשך התיזה. המידע או-דות השפות מקודד באופן בלתי תלוי ומשודר למקלט. אין, כמובן, צורך לקודד את הסוג השלישי (כיון שצורת הסגמנט מקודדת אף היא).

מחקר זה מתמקד בנושאים של סגמנטציה וקידוד אזורי מרקם. קידוד השפות נעשה בעבודה זו באמצעות קידוד שרשרת משופר (modified chain code).

במחקר זה יושמו שתי מערכות קיימות המבצעות סגמנטציה מורפולוגית וכן מערכת נו-ספת, אשר פותחה במהלך מחקר זה, שהיא בעלת יתרונות ביחס לשתי המערכות הקיימות בשימוש הנדון.

במערכות סגמנטציה נוטים להשתמש באופן נרחב במסננים ובאופרטורים מורפולוגים על מנת לפשט את התמונה. איזורים שטוחים מיוצרים על מנת לציין את החלקים הפנימיים של איזורים חשובים כגון איזור גדול ושטוח או איזור בעל ניגודיות (contrast) גבוהה.

ייצוג האיזורים השטוחים מאפשר שימוש בטכניקה סטנדרטית של סימון (labelling). תהליך זה נקרא (marker extraction). הפיקסלים בקרבת השפות אינם ממופים לתחום כלשהו לאחר ביצוע ה-marker extraction. אי לך יש צורך ליישם כלים לביצוע החלטה (decision tool) ביחס לסווג פיקסלים אלה לאזורים מתאימים.

תוצאות הסימולציה שהתקבלו משני האלגוריתמים הראשונים מראות שיש להם חסר-ונות מובנים כגון "סגמנטציית יתר" (oversegmentation), "שפות דמה" (false contour) וכן קבלת צורות חלוקה מורכבות (complicated partition), בהשוואה לתוצאות שהתקבלו עבור השיטה שפותחה במהלך מחקר זה. השיטה שפותחה היא סגמנטציה המבוססת על גילוי קצוות וסיווג פעילות מקומית (edge detection, local activity classification).

אלגוריתם זה מפשט את התמונה באופן משמעותי כך שניתן ליישם בנקל טכניקת סי-מון (labelling) שאחריה מבצעים region growing על מנת להגדיר את גבולות העצמים. תוצאות סימולציה מראות שבמקרים בהם קיימים קצוות חדים בין הסגמנטים הסמוכים, הם אכן מתגלים כשני סגמנטים שונים, ולהפך, במקרה שלסגמנטים סמוכים יש מאפיינים דומים הנ"ל ימוזגו לסגמנט משותף אחד. העובדה שסגמנטים, אשר התכונות הויזואליות שלהם דומות, מסווגים לסגמנט אחד מאפשרת ייצוג קומפקטי. יתרה מזאת, מתקבלת סגמנטציה פשוטה ומחיר קידוד קווי הגבול (contours) מופחת. לכן, תוצאת הסגמנטציה המתוארת מתאימה במיוחד למטרות קידוד. האלגוריתם המוצע נראה כגישה מבטיחה לקבלת מערכת סגמנטציה גנרית, המתאימה לתמונות מסוגים שונים, אם כי נדרש עדיין מחקר נוסף בנושא של region refinement.

בקידוד תמונה המבוסס על סגמנטציה מתייחסים לשתי קטגוריות עיקריות, האחת מבוססת על בלוקים (block-based) והשניה מבוססת על איזורים (region-based).

לאחר ביצוע הסגמנטציה בוחרים סגמנט אחד בכל פעם לצורך הקידוד. הגישה המבוססת על בלוקים מחלקת את הסגמנט הנבחר לבלוקים בגודל 8×8. כאמור לעיל הבלוקים המתקבלים שייכים לאחד משלש הקטגוריות הבאות: בלוקים פנימיים, בלוקי גבול ובלוקים חיצוניים.

ברור שאין כל צורך לקודד את הבלוקים החיצוניים, קידוד הבלוקים הפנימיים מתבצע בטכניקת JPEG סטנדרטית. קוונטיזציה גסה יותר יכולה להעשות למקדמי ההתמרה, שכן שגיאות בתוך תחום הסגמנט "נסבלות" יותר. קוונטיזציה גסה זו מאפשרת קידוד המרקם בקצב סיביות נמוך יותר. שימוש ב-2D-DCT 8×8 אינו יעיל עבור בלוקי גבול שכן בבלוקים הללו ישנן גבולות חדים. יש צורך לכן ביותר סיביות כדי למנוע הבחנה בעוותים ע"י העין האנושית.

שתי גישות שונות נבחנו על מנת לקודד את בלוקי הגבול .התמרת DCT תלוית צורה (shape-adaptive DCT - SADCT) ושיטת האקסטרפולציה של המידע בבלוק (block data extrapolation).

גישת ה-SADCT מזיזה את המידע בבלוק הגבול לעבר הצלע העליונה של הבלוק, עמודה אחר עמודה, ומחשבת את ה-1D-DCT בגודל המתאים לגודל העמודה. מקדמי ה-1D-DCT שחושבו מוזזים ומיושרים כעת לפינה השמאלית-עליונה של הבלוק שורה לאחר שורה. לאחר מכן מחשבים שוב את ה-1D-DCT לכל שורה באופן נפרד. התהליך שתואר הינו הפיך, כך שניתן לשחזר את גבולות הבלוק באופן מדוייק.

בגישת האקסטרפרלציה של המידע בבלוק מבצעים אקסטרפרלציה אל מחוץ לסגמנט כך שהגבולות שהיו חדים במקור הופכים להיות חלקים. ניתן להתייחס כעת לבלוק כזה, כ"בלוק פנימי" כלומר ניתן להשתמש ב-2D-DCT רגיל ולקודד את המקדמים שהתקבלו. במפנעח ניתן לבצע תהליך הפכי, שכן ניתן להתעלם מהמידע אשר מחוץ לסגמנט (גבולות הסגמנט ידועים בצד המפענח).

במחקר זה נבחנו בעיקר שתי גישות שונות לאקסטרפולציה של המידע בבלוק: גישת "ריפוד" ע"י אקסטרופולציה מעבירת נמוכים(low-pass extrapolation padding - LPE), המבוססת על MPEG-4 וגישה נוספת חדשה אשר פותחה במהלך מחקר זה - optimal) BP - basis pursuit) לביצוע אקסטרפולציה.

שיטת ה-LPE padding הינה אקסטרפולציה פשוטה (בעזרת מיצוע) של המידע בבלוק אך אינה מספקת מבחינת הורדת קצב הסיביות.

שיטת ה-BP הינה שיטה ששמשה למטרות אנליזה של אותות ומיושמת לראשונה במחקר זה לביצע אקסטרפולציה. השיטה מבוצעת ע"י מיזעור נורמת l_1 של מקדמי ההתמרה של הבלוק המעובד. מסתבר שקיים קשר בין בעית מזעור נורמת l_1 ו"תכנות לינארי" (linear programming) כך שניתן לפתור את בעית ה-BP בעזרת תכנות לינארי. מספר המקדמים השונים מאפס בהתמרת הבלוק המתאים שווה למספר הפיקסלים המכילים מידע בבלוק הגבול.

כאשר משתמשים בשיטת ה-SADCT או בשיטת ה-LPE מתקבל שמקדמי ההתמרה נו-טים להיות מרוכזים בתדרים הנמוכים של ה-DCT. כתוצאה מכך שיטת הקידוד המקובלת

ב-JPEG מתאימה גם למקרים הללו. כאשר משתמשים בשיטת ה-BP לצורך ביצוע האקסטר-פולציה, מקדמי ההתמרה המתקבלים אינם מרוכזים בתחום תדרים מסויים. אי לכך, טבלאות הקוונטיזציה וקידוד האנטרופיה של JPEG אינן מתאימות. הגישה שיושמה כאן הינה קוונטיזציה אחידה ושימוש בקידוד אריתמטי לקידוד המקדמים השונים מאפס וכן של סדרות האפסים ביניהם. תוצאות הסימולציה שהתקבלו מצביעות על יתרון לדחיסה על בסיס שיטת BP לעומת השיטות האחרות שצויינו. החסרון של שיטת BP הינו הסי-בוכיות הגבוהה שלה.

באופן כללי, שיטות המבוססות על בלוקים נהנות מיתרונות הפשטות והתאימות לגישות הסטנדרטיות, בהשוואה לשיטות המבוססות על איזורים (region-based). אך מכיוון שהאינ-פורמציה המעובדת בשיטות המבוססות על בלוקים היא מקומית, שיטות אלה אינן מספקות דחיסה גבוהה.

מבחינה תאורטית, גישה המבוססת על איזורים יכולה לנצל את התכונות הגלובליות של הסגמנט ויש בה פוטנציאל לדחיסה טובה יותר. שיטות אלו מנסות למצוא את פונקצית הקירוב הטובה ביותר במובן של ריבועים פחותים (LS) לפונקצית הסגמנט המקורי. מכיוון שפונקצית הקירוב יכולה להיות מיוצגת כשקלול של פונקציות בסיס, מתאפשרת דחיסה קומפקטית של מקדמי הפרוק כאשר בוחרים היטב את פונקציות הבסיס. חישוב מקד-מי ההתמרה הינו יציב ופשוט, אם משתמשים בפונקציות בסיס שהן אורתוגונליות ביחס לצורת סגמנט (shape-adaptive orthogonal basis functions). לכן, רב העבודות המדווחות מתמקדות בחישוב של בסיסים אורתוגונלים תלויי צורה לייצוג פונקצית הקירוב. שיטות מקובלות ליצירת פונקציות בסיס אורתוגונליות (מותאמות לצורה) שנבדקו במחקר זה הן:

אורתוגונליזציה לפי Gram-schmidt, שמוש בפולינומים של Householder, ושימוש בפונ-קציות בסיס שהן ספרביליות באופן חלש (weakly separable basis functions). חישוב הפולינומים של Householder ופונקציות בסיס ספרביליות באופן חלש הינו מהיר יותר מאשר בשיטת Gram-Schmidt. אך פונקציות הבסיס המתקבלות בשיטת Gram-shmidth מספקת בסיס המאפשר ייצוג טוב יותר של פונקצית הקירוב (במובן LS). העומס החישובי

של שלוש השיטות שצויינו לעיל הינו גדול יחסית, וכתוצאה מבעיות נומריות הצלחנו לחשב רק מספר מוגבל של פונקציות בסיס (כלומר מסדר נמוך). כתוצאה מכך, פונקציית הקירוב של הסגמנט המתקבלת משקלול פונקציות הבסיס הינה חלקה ומשתנה באיטיות. עובדה זו גוררת אחריה את הצורך לקודד, בנוסף למקדמי ההתמרה, גם את השארית בין הפונקציה המקורית לבין פונקצית הקירוב. ברור שקידוד השארית הופך את השיטה הנ"ל ללא כדאית שכן מחיר קידוד השארית דורש קצב סיביות גבוה. לפיכך, יישום שיטות המבוססות על קידוד איזורים אינו אטרקטיבי בשלב זה עבור תמונות טבעיות.