Image Compression using Piecewise Polynomials

Thesis submitted for the degree

Doctor of Philosophy

by

Roman Kazinnik

Under the supervision of Prof. Nira Dyn and Dr. Shai Dekel

Submitted to the Scientific Counsel of Tel Aviv University

Tel Aviv, Israel

February 1, 2008
Image compression with piecewise polynomials

PhD Thesis by Roman Kazinnik

Under guidance of Dr. Shai Dekel and Prof. Nira Dyn
Contents

I Introduction 9

II Background 13

1 Image coding 15

1.1 The importance of image coding . . . . . . . . . . . . . . . . . . . . . 15
1.2 Our goal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
1.3 'Beyond wavelets' modern research . . . . . . . . . . . . . . . . . . . . 16

2 Approximation 19

2.1 Approximation algorithms . . . . . . . . . . . . . . . . . . . . . . . . . 19
2.2 Linear and highly nonlinear approximation . . . . . . . . . . . . . . . 19
2.3 Nonlinear approximation with wavelets . . . . . . . . . . . . . . . . . 22

2.3.1 Besov-Sobolev smoothness spaces . . . . . . . . . . . . . . . . . . 24
2.3.2 Non-optimality of bivariate wavelet approximation . . . . . . . . 25
2.4 Examples of classical more adaptive approximation constructions . . 25

3 Piecewise polynomials 27

3.1 Evidence for the superiority of our approximation method . . . . . 27
3.2 Piecewise polynomials approximation over triangles . . . . . . . . . 30

3.2.1 \( \tilde{K} \)-functional and \( \tilde{B} \) smoothness spaces . . . . . . . . 30

3.3 Approximation with polygonal piecewise polynomials . . . . . . . . . 33

3.3.1 Definition of the approximation spaces \( A^{r,n,p}_{\alpha} \) . . . . . . . 33
3.3.2 A simplified approach to the estimation of the error . . . . . . . . 33

3.4 Approximation by a single polynomial in a complicated domain . . . 36

3.4.1 \( L_2 \) Error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
3.4.2 An instructive example . . . . . . . . . . . . . . . . . . . . . . . . 37
3.4.3 The distance defect ratio as a measure of domain singularity . . . 38
3.4.4 Measure of geometric singularity . . . . . . . . . . . . . . . . . . . 38

4 Wavelet-based image coding 41

4.1 Step-1: approximation with wavelets . . . . . . . . . . . . . . . . . . . 41
4.2 Step-2: wavelet thresholding and encoding . . . . . . . . . . . . . . . 42
4.3 Non-optimality of wavelet-based image coding . . . . . . . . . . . . . 43

III Geometric Piecewise Polynomials (GPP) image coding algorithm 47

5 GPP coding algorithm 49

5.1 Overview of other geometric algorithms . . . . . . . . . . . . . . . . . 49
5.2 GPP key concepts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
5.3 GPP outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
6 Image Segmentation
   6.1 Segmentation methods in image coding ........................................ 53
   6.1.1 Global segmentation: region growing .................................. 53
   6.1.2 Local segmentation: edge detection .................................. 57
   6.1.3 The Mumford-Shah functional and the $\tilde{K}$-functional ........ 59

7 The GPP Image Segmentation ................................................. 63
   7.1 GPP step-1: finding a partition .......................................... 63
   7.2 GPP step-2: creating the bands ........................................ 64
   7.3 GPP step-3: convex driven binary tree partitioning ............... 66
   7.3.1 An overview ...................................................... 66
   7.3.2 Geometry-driven binary tree partition ......................... 67
   7.3.3 Three examples ............................................... 68

8 Approximation and encoding in GPP .................................... 71
   8.1 1-D: Approximation and encoding of the partition curves .......... 71
   8.1.1 Lossless encoding using arithmetic coding and context modelling 71
   8.1.2 A new method of 1-D (lossy) encoding based on arithmetic coding and context modeling .............................................. 72
   8.2 2-D: Approximation and encoding in subdomains of smoothness ...... 75
   8.2.1 Scalar quantization ............................................. 75

9 Experimental results ......................................................... 77
   9.1 Demonstration of the GPP algorithm .................................. 77
   9.2 GPP algorithm vs. JPEG2000 ........................................ 77
   9.2.1 Budget allocation ............................................... 81
   9.3 Selecting GPP parameters ............................................. 81

IV Beyond GPP ................................................................. 83

10 New algorithm of dimension elevation .................................. 85
   10.1 Algorithm outline .................................................. 85

11 Measuring the visual quality of compressed images .................. 91
   11.1 PSNR ($L_2$) and Hausdorff distance ............................... 91
   11.1.1 Two counter examples ........................................ 91
   11.2 The Hausdorff distance definitions ................................ 93
   11.3 An idea for a measure of visual quality: combination of the Hausdorff distance and the $L_2$ norm ................................. 93
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Example of partition of $[0, 1]^2$, with subdomains ${\Omega_i}$ and curves ${b_i}$</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>(a) A smooth bivariate function, (b) ‘poor’ approximation with quadratic polynomials over the entire domain (PSNR=21.5dB), (c) approximation improves significantly once the domain is partitioned into ‘better’ subdomains (PSNR=33dB)</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>(a) Example of a subdomain in the initial partition of Cameraman image, (b) example of one geometry-driven partition with a straight line.</td>
<td>31</td>
</tr>
<tr>
<td>3.4</td>
<td>(a) Subdomain $\tilde{\Omega}$ generated by the partition in Figure 3.5, (b) its convex hull $H$, (c) the corresponding disjoint components ${C_i}$ of $H \setminus \tilde{\Omega}$.</td>
<td>39</td>
</tr>
<tr>
<td>3.5</td>
<td>(a) Example of a subdomain in the initial partition of Cameraman image, (b) example of one geometry-driven partition with a straight line.</td>
<td>39</td>
</tr>
<tr>
<td>3.6</td>
<td>(a) Segmentation with zero-crossing LoG of Cameraman image. (b) segmentation based on 29 of 112 edges in Figure 6.1(b), (c) its corresponding piecewise polynomial approximation, PSNR=19.4769 dB</td>
<td>51</td>
</tr>
<tr>
<td>3.7</td>
<td>(a) Final segmentation has 4 edges, (b) its corresponding approximation, PSNR=17.1735 dB</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>(a) Part of Cameraman image, (b) initial grid with 112 edges, (c) its corresponding piecewise polynomial approximation, PSNR=19.4769 dB</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>(a) Intermediate stage with segmentation based on 29 of 112 edges in Figure 6.1(b), (b) its corresponding piecewise polynomial approximation, PSNR=19.2255 dB</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>(a) Final segmentation has 4 edges, (b) its corresponding approximation, PSNR=17.1735 dB</td>
<td>55</td>
</tr>
<tr>
<td>4.4</td>
<td>(a) Segmentation with 29 of 112 edges, (b) corresponding approximation with PSNR=19.3343 (18.3142)</td>
<td>57</td>
</tr>
<tr>
<td>4.5</td>
<td>Segmentation of Lena image produced by minimization of the Mumford-Shah functional with the method of steepest descent iterations.</td>
<td>60</td>
</tr>
<tr>
<td>7.1</td>
<td>Pruning of a small width Gaussian Zero-crossing LoG of Cameraman image.</td>
<td>64</td>
</tr>
<tr>
<td>7.2</td>
<td>(a) Cameraman image, (b) its corresponding gradient image, (c) $\text{Zerocross(LoG)}$ partition curves with shown at the background Gradient values, (d) $\text{Zerocross(LoG)}$ partition curves after threshold with respect to Gradient</td>
<td>65</td>
</tr>
<tr>
<td>7.3</td>
<td>(a) Original curved edges (grey color) and its approximating polylines (black color), (b) wide bands (grey color) appended to polylines envelope original edge curves (black color)</td>
<td>65</td>
</tr>
<tr>
<td>7.4</td>
<td>(a) Cameraman image, (b) example of segmentation curves, (c) complicated domains generated by the segmentation in (b).</td>
<td>67</td>
</tr>
</tbody>
</table>
7.5 (a) Cameraman image, (b) example of segmentation curves, (c) complicated domains generated by the segmentation in (b) ............................................. 67

7.6 A subset of $H \setminus \tilde{\Omega}$ lying inside $\tilde{\Omega}$, with two casted rays (in black) . . . 69

7.7 (a) One component of $H \setminus \tilde{\Omega}$ lying inside $\tilde{\Omega}$, (b) the corresponding splitting of $\tilde{\Omega}$ by one ray ................................. 69

8.1 (a) Pruned curve portions of Lena, (b) the same curve portions after six-fold down-sampling ............................................................... 73

8.2 (a) Up-sampling curves (black color) with original partition curves (grey color), (b) smoothed up-sampling curves (black color) employing linear prediction approximate better the original curve portions (grey color) .............................................. 74

8.3 Bands appended to smooth up-sampled curves (grey color) enveloping original partition curves (black color) .......................... 74

9.1 (a) Pruning of the initial segmentation of Lena, notice the sorting out of the jagged curve portions ...................................................... 78

9.2 (a) 1-D approximation employs downsampling before lossless encoding, (b) the corresponding ‘jaggy’ reconstruction (upsampling) after decoding, (c) smoothing prediction is subsequently applied to produce smooth curves out of the ‘jaggy’ reconstruction ...................................................... 78

9.3 (a) Adding bands to the decoded lossy segmentation in Figure 9.2(c), (b) the initial subdomains of smoothness ........................................ 78

9.4 (a) The final partition obtained with the convex-based dyadic partitioning, (b) its corresponding polynomial approximation, (c) the linear interpolation at the bands ................................. 79

9.5 Comparison of artifacts in one region of the Cameraman image for GPP and JPEG2000 .............................................................. 79

9.6 R-D performance comparison of the GPP algorithm and JPEG2000 for the Chess, Cameraman and Lena images ......................................... 79

9.7 Artificial and real-life images coding examples with J2K (JPEG2000) versus GPP. Notice how GPP improves the coding of geometry as the bit-rate increases. ................................................................. 80

9.8 Budget allocation of the three components in the three images: Eggmean, Cameraman, and Lena ................................. 81

10.1 (a) Domain with one domain singularity, (b) the domain in 3-D resulting from the continuous mapping of the planar domain ................................. 86

10.2 (a) A domain with three domain singularities, (b) approximation with one 5-variate linear polynomial using continuous 5-D mapping achieves PSNR=28.6dB, (c) approximation using one bivariate linear polynomial produces PSNR=16.9dB. ................. 86

10.3 Comparison of the two algorithms, approximating the smooth function $f(\rho, \theta) = \rho \cdot \theta$ in a domain with one singularity component. (a) eight subdomains are required to approximate by piecewise linear (bivariate) polynomials, (b) the piecewise linear polynomial approximant on the eight subdomains approximates with PSNR of 25.6 dB, (c) similar approximation error is achieved with one tri-variate linear polynomial using our mapping. ............................................. 88
10.4 (a) A domain with one singularity, (b) approximation using 3-D mapping with 3-D linear polynomial produces PSNR=24.6dB, (c) approximation using one bivariate linear polynomial produces PSNR=13dB. Notice how well the 3-D mapping based algorithm captures the function along its domain singularity in (b), in comparison with the bivariate polynomial approximation in (c).

11.1 (a) Partition constructed by the GPP algorithm, (b) GPP image coding (at 0.05 bpp) corresponding to (a).

11.2 (a),(b) Partition and corresponding GPP image coding with less edges than in Figure 11.1. The PSNR of the image in 11.1(b) is lower than the PSNR of the image in (b) yet the amount of information in 11.1(b) is remarkably bigger. (Notice the tower, the handle of the camera and the hand of the cameraman in Figure 11.1(b) that are missing in (b)).
Abstract

This work presents a new image coding algorithm, the Geometric Piecewise Polynomials (GPP) method, that draws on recent developments in the theory of adaptive multivariate piecewise polynomials approximation. We regard an image as a bivariate function, with discontinuities along 1D smooth curves, which is smooth otherwise, and we develop new 2D and 1D approximation techniques.

The three main stages in the GPP algorithm are segmentation of the image, multiscale encoding of 1D edges, and polynomial approximation in complicated planar domains. The goal of the segmentation stage is to reduce a functional that is conceptually similar to the Mumford-Shah functional except that it measures the smoothness of the segmentation instead of its length. An initial segmentation, obtained by the zero crossing of the Laplacian-Gaussian of the image, is ‘pruned’ accordingly, and the remaining curve portions are lossy encoded. The image is then approximated by low order polynomials in the ”smoothness domains”. We developed an algorithm for polynomial approximation of smooth bivariate functions in highly complicated planar domains, with small encoding budget. The algorithm partitions a complicated domain into simpler subdomains in a recursive binary way. The function is approximated in each subdomain by a low-degree polynomial. The partition is based on both the geometry of the subdomains and the quality of the approximation there. The algorithm is based on a proposed measure of domain singularity, and is aimed at decreasing it.

The GPP algorithm significantly outperforms wavelet based coding methods on graphic and cartoon images. We show examples of real life images where the GPP method outperforms the state-of-the-art wavelet coding, in the low bit-rate range. For example, at 0.05 bits per pixel the GPP algorithm achieves on the test image Cameraman, which has a geometric structure, a PSNR of 21.5 dB, while the JPEG2000 Kakadu software obtains PSNR of 20 dB. For the test image Lena at 0.03 bpp the GPP algorithm obtains the same PSNR as JPEG2000, but with better visual quality.

A review of the theoretical results, related to the GPP method, precedes the presentation of the GPP algorithm. The thesis ends with two miscellaneous topics, an algorithm for the construction of a continuous approximation in a complicated planar domain, and a suggested measure for visual quality of compressed images.
Part I

Introduction
From the mid 80s, there have been many attempts to design ‘Second Generation’ image coding techniques that exploit the geometry of edge singularities of an image. The reader may consult the survey [61]. To this day, almost all of the proposed ‘Second Generation’ algorithms are not competitive with the state of the art (dyadic) wavelet coding [63], [70], [75]. In one of the outstanding ‘Second Generation’ methods [33], Froment and Mallat constructed multi-scale wavelet-like edge detectors and showed how to reconstruct a function from the responses of a sparse collection of these detectors. They reported good coding results at low bit-rates. There are coding algorithms that are geometric enhancements of existing wavelet transformed-based methods, where wavelet coefficients are coded using geometric context modeling [79]. In a recent work [65], the authors enhance classical wavelet coding by detecting and coding the strong edges separately and then using wavelets to code a residual image. Candès and Donoho [8] constructed Curvelets, a bivariate transform designed to provide sparse representations with local multi-scale directional information. Do and Vetterli’s construction of Contourlets [28] is similar but is a purely discrete construction. Cohen and Matei [11] also showed a discrete construction of an edge-adapted transform that is closely related to nonlinear Lifting [10]. All of these constructions are redundant, i.e., the output of the discrete transform implementations produces more coefficients than the original input data. The possibility to use these new transforms to outperform wavelet coding is still an on-going research. LePennec and Mallat [44] recently applied their ‘Bandelets’ algorithm to image coding, where a warped-wavelet transform is computed to align with the geometric flow in the image and the edge singularities are coded using one-dimensional wavelet-type approximations. Previous work that we find to be the closest to ours are the papers by Shukla, Dragotti, Do and Vetterli [71] (see also [60]), Dekel and Leviatan [18], and Demaret, Dyn and Iske [21].

Our algorithm, the Geometric Piecewise Polynomials (GPP) [42], departs from the framework of harmonic analysis, which is the theoretical basis for transform based methods and even from the more general framework of multi-scale geometric processing. The main difference between our algorithm and recent work is that we re-vise the ‘Second Generation’ approach and directly apply segmentation methods. In this respect, there are similarities between the GPP and the Bandelets algorithm that computes as a first stage the ‘geometric flow’ of the image. However, the Bandelets algorithm imposes a ‘wavelet’ structure on the computed geometric flow, while our algorithm applies the conceptually simpler piecewise polynomial approximation scheme in the ‘domains of smoothness’ of the image determined in the segmentation.

For $f \in L^2([0, 1]^2)$, $r > 0$, piecewise polynomial approximation of order $r$ on a polygonal domains has the form

$$\sum_{k=1}^{n} 1_{\Delta_k} P_k,$$

where $\{\Delta_k\}$ are polygonal domains with disjoint interiors, such that $\bigcup_{k=1}^{n} \Delta_k = [0, 1]^2$, and $P_k, 1 \leq k \leq n$, are polynomials of total degree $r - 1$. We note that there are computational variants of piecewise polynomial approximation where the domains are polygonal and satisfy a nesting property (see e.g. [18], [41]). In comparison, our approach can be regarded as a ‘higher order’ method, since the domains over which eventually polynomial approximation is performed, can have ‘curved’ portions (see Figure 7.1(c)).

It is known that in the univariate case, wavelets and piecewise polynomials have the same (theoretical) performance, since their corresponding approximation spaces are identical [26]. However, in the multivariate case this is no longer true and at least theoretically, piecewise polynomials outperform wavelets whenever the approximated
function has some ‘structure’, i.e., edge singularities that are smooth in some weak sense [19], [41].

In our GPP algorithm we model an image as a piecewise smooth bivariate function with curve singularities of weak type smoothness. We apply a segmentation algorithm aimed at reducing the $\tilde{K}$-functional of [19], which determines "smoothness domains" of the image.

This leads to the problem of approximating a smooth function in highly complicated planar domains [29]. Since the adaptive approximation method is aimed at image compression, an important property required from the approximation in the complicated domains is a low encoding budget, namely that the approximation is determined by a small number of parameters.

For the GPP method, we designed an algorithm which approximates the function by piecewise polynomials. The algorithm generates a partition of the complicated domain to a small number of less complicated subdomains, where low-degree polynomial approximation is good enough. The partition is a binary space partition (BSP), driven by the geometry of the domain and is encoded with a small budget. The algorithm is based on a new measure of domain singularity, concluded from case studies, showing that in complicated domains the smoothness of the function is not equivalent to the approximation error, as is the case in convex domains [17], but that the quality of the approximation depends also on geometric properties of the domain.

Here is the outline of the thesis. Chapters 1–4 review the theoretical approximation issues related to the GPP algorithm and to wavelets-based methods. The GPP method is presented in detail in Chapters 5–9. Chapter 5 gives an outline of the various steps in the GPP algorithm. Chapter 6 reviews relevant results on segmentation of images, while Chapter 7 presents the image segmentation by the GPP method. In Chapter 8, the lossy encoding of the edges, obtained in the segmentation step, are described, as well as the approximation by a low degree polynomial in a smoothness domain of the segmentation and its encoding. Chapter 9 brings comparison results between the performance of GPP and JPEG2000 on several test images, and compares the budget allocation in the various steps of the GPP. The last two chapters present two miscellaneous topics. One is a second algorithm for approximation in complicated planar domains, and the second is a measure for visual quality.
Part II

Background
Chapter 1

Image coding

1.1 The importance of image coding

Once personal computers gained the capacity to display sophisticated pictures as digital images, people started to seek methods for efficient representations for those digital pictures in order to save disk space. At this point Image coding became very important and highly applicable, and a lot of studies have been conducted since then. The need for image coding is even more pronounced due to second generation cellular phones. Since the wireless communication allows only very low bit traffic, it is essential to employ extremely efficient encoding algorithms to allow delivering images to cellular phones using very few bits, and obtaining reasonable quality for small images.

The main goal of image coding is to reduce the space required to store digital images. In the first step of compression the digital image \(I\) is converted into some data file \(D\), that occupies less disk space. During the second step of decompression the smaller \(D\) is converted back into a digital image \(\tilde{I}\). Lossless compression requires the exact equality of \(I\) and \(\tilde{I}\), and typically achieves space reduction around 50%. To the contrary, lossy compression does not provide the exact reconstruction, and may succeed in preserving as much as 95% of the information of \(I\) with as little as 5% of the original size of \(I\).

The Image size of \(I\) is defined as the physical space occupied by pixels of \(I\) on the disc. The error of lossy compression is commonly measured by the mean square error (MSE) \(\|I - \tilde{I}\|_2 = \left(\frac{1}{M^2} \sum |P_I - P_{\tilde{I}}|^2\right)^{1/2}\), where the digital images \(I\) and \(\tilde{I}\) are sets of the pixel values \(P_I\) and \(P_{\tilde{I}}\), respectively, over a common rectangular domain consisting of \(M\) pixels.

Lossy image coding is often based on continuous functional representation, where ones assume a functional model behind the image. In this case there are two main components. The first critical component is the transformation of the original digital image pixel values into a set of coefficients in its functional representation, subject to the a-priory chosen functional model. This stage usually involves no compression yet, but the coefficients themselves hold properties that are employed further, such as a large number of small near-zero coefficients. By pruning them away, or thresholding, the representation does not change much in the \(L_2\)-norm. In addition, the coefficients are well localized, i.e. they are statistically correlated subject to their placement in the functional representation, whereas this correlation can be employed in statistical encoding. Such thresholding and statistical encoding comprise the second component of the lossy image coding.
1.2 Our goal

There are many different task-driven aspects in image coding: lossless and lossy, progressive, real-time for low-power devices, colored images, and image coding for specific classes of images, such as medical ultra-sound images, cartoons, or artificial images of computer graphics. This work challenges the main image compression problem, that is how to reduce the amount of disk space as much as possible, while at the same time to minimize the compression error. Although it is possible to extend this work to other issues such as time efficiency, progressive coding and color images, our efforts at this time are channelled towards and focused on the above most pertinent problem of image coding.

In our work it is especially important to compare our results with wavelet-based image coding algorithms. First of all, because wavelets became an important part of the classical analysis of nonlinear approximation, whereas in this work we go one step further towards the highly nonlinear approximation. In addition, as wavelets present a powerful mathematical tool, they became popular in a wide range of modern applications, including image coding. The new image coding JPEG2000 standard, is based on wavelets. In essence, we are challenging the latest cutting-edge image coding algorithm EBCOT, which is the basis of the JPEG2000 standard and which provides excellent results in terms of rate-distortion performance, with our new image coding algorithm based on the paradigm of bivariate piecewise polynomials.

1.3 'Beyond wavelets' modern research

Today, the state-of-the-art image coding algorithms are transformed-based coders (Section 4). The examples of such image coders are the popular Fourier transform based JPEG image coding tool, which is extensively used in many applications, and the recent highly competitive JPEG2000 standard, based on the Wavelet transform. While providing excellent results in terms of rate-distortion compression, the transform-based coding methods do not take an advantage of the underlying geometry of the edge singularities in an image (if it exists), as explained in Section 4.3. Therefore, the recent research activity aims at developing geometric sparse representation and coding methods.

Many new methods employing the geometry of singularities have been already reported. However, only few of these geometric methods report competitive results for complex real life images over the compression range where the image coding algorithms are usually used in applications, compared with the transform based image coding methods.

Let us review several distinct geometric image coding approaches and methods. First of all, it is important to recall that while the modern research activity in geometric coding started some half a dozen years ago, a number of geometric image coding algorithms have been already presented more than twenty years ago. These methods received the name of the second generation methods (see [61]). One of its most popular approaches, known as the region-based image coding, considers a real life image as a set of distinct objects, where each object is assumed to have its own lightning characteristics. Several algorithms segmenting an image into disjoint objects (regions) were proposed ([72], [64]). Region-based methods encode images by partitioning the image into objects and describing these objects with a small set of parameters. Another approach utilizes edges of images and develops a new contour-based image representation. In [31], [32] and [33] image coding schemes are proposed that compute the representation of image based on information recorded on edges. Elder and Zucker in [31] find edges and blur information, such that the image is even-
tually encoded by the areas around its significant edges. The two main components in this representation are the intensity and an estimate of the blur at edge locations in the image. As far as we know, none of the second generation methods reported competitive with the established image coders results, such as JPEG.

In a recent paper [71], Shukla et. apply the least-squares piecewise polynomial approximation and a sophisticated method of partitioning an image domain into subdomains. The partitioning of the image domain is obtained as the result of a complex segmentation process, which utilizes prune-and-join quad-tree axes aligned partitioning, whereas the smallest quad-tree regions are subdivided also diagonally. This allows to this algorithm to gain more adaptivity to the geometry of singularities in the image. In the obtained subregions the image are approximated by orthonormal bivariate polynomials. The work [71] reports competitive results in low bit-rate both for artificial and real-life images, comparable with the wavelet coding methods. Although the resulting partition in [71] is more powerful than the dyadic squares partition, it is still not completely adaptive to any geometry of edges. There are examples of images for which the proposed model does not take full advantage of the geometry of edges.

Bandelets, introduced in [44], decompose the image along multiscale vectors that are elongated in the direction of a geometric flow. The geometric flow indicates direction in which the image grey levels have regular variations. Bandelet bases are obtained by a bandelization of warped wavelet bases, which takes advantage of the image regularity along the geometric flow. The geometric flow is linked intrinsically with the edges of image, and by warping the wavelet functions the method tries to cope with the image singularities by adapting to its geometry. Finally, Bandelets method encodes both the warped wavelet representation and the edges of an image.

Candes and Donoho ([8]) introduce a bivariate Curvelet transform designed to capture local multi-scale directional information. The transform employs Curvelet families of functions, which are elongated and rotated univariate wavelet functions at different scales. Such Curvelet families are specifically constructed as a frame in $L_2$. The Curvelets representation of an image is capable to take advantage of arbitrary geometric regularity in an image. After Curvelets multiscale image representation is computed, it is thresholded and encoded by regular means.

A proposed projection pursuit algorithm in [49] finds the best basis for an image from a large and redundant library of bases. The algorithm is iterative and maintains a list of basis functions, its corresponding projection and the reminder between the projection and the original functional model of an image. At every step the algorithm determines a function of the redundant library, most correlated with the reminder, to add the list of basis functions, and computes the projection and the reminder corresponding to the new basis. The projection pursuit is a greedy algorithm, therefore it does not guaranty finding the optimal basis and may have a slow convergence rate. However, for some specific subset of image [49] illustrates that the projection pursuit algorithm can produce comparatively high compression results.

With the hypothesis that an image can be decomposed into a sum of mainly two components, [52] presents a multilayered representation technique for image coding. An image is decomposed into a superposition of two coherent layers: a layer of piecewise smooth regions and a texture layer. The piecewise smooth layer keeps the piecewise smooth component of the image and is represented with Ridgelets, which can be adapted to linear edges in an image, whereas the texture layer representation employs the local cosine transform. The two layers are encoded independently. On a set of known real-life images this image coder reports compression results comparative to wavelet coders.

The authors of [20] approximate an image by linear splines over an adapted De-
launay triangulation $D(Y)$ of a small set of significant pixels $Y$. The linear spline is so constructed to minimize the mean square error to the image. The significant pixels in $Y$ are selected by an adaptive thinning algorithm, which recursively removes less significant pixels in a greedy way. A sophisticated scattered data coding scheme was developed to efficiently encode the subset $Y$ and the linear interpolation spline. The proposed compression method outperforms an algorithm of JPEG2000 on geometric images and reports competitive results on a set of known real life images.
Chapter 2
Approximation

2.1 Approximation algorithms

In this Section we first explain one of the subject matters of approximation - a comparative analysis of different algorithms of approximation, performed in terms of the rate of convergence. Specifically, we discuss the theoretical issues of comparing the two approximation algorithms - first in terms of wavelets and second in terms of piecewise polynomials.

Approximation theory issues arises in algorithms that compute a simpler representation of a complex functional model. Assume the functional model be a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, and let the approximating algorithm $M$ computes for $f$ a finite set of coefficients $S = \{S_i\}_{i=1}^n$, $n < \infty$, in order to construct an approximant $f_n[S]$ for the given function $f$ using the set $S$. Denote by $\sigma_n$ the error of the approximation of the original function $f$ by the approximant $f_n$, which is measured in some functional norm (or quasi-norm) $\sigma_n = \|f - f_n\|$, therefore $\sigma_n = 0$ when $f_n = f$. The approximation order $r$ is defined asymptotically by the equality $\sigma_n = O\left(\frac{1}{n^r}\right)$ as $n \to \infty$.

Let us illustrate how approximation theory provides a way to analytically compare different approximation algorithms. Firstly, for a given algorithm $M$ the theory tries to find a characterization of its asymptotic quantity $r (n \to \infty)$ in terms of some known properties of $f$, namely, to find a functional space $V$ for instance $C^r$, such that:

$$\sigma_n(f) = O\left(\frac{1}{n^r}\right) \iff f \in V$$

Assume now that there are two approximation algorithms $M_1$ and $M_2$, and assume we have found their characterizations in terms of some known classical linear spaces $V_1^r$ and $V_2^r$ respectively, for all functions of interest $f$. If there exists a partial ordering such that $V_1^r \supset V_2^r$ for all $r > 0$, one draws the conclusion that the algorithm $M_1$ is more efficient than $M_2$, since for any function $f$ the algorithm $M_1$ obtains the same or a higher convergence rate as the algorithm $M_2$.

2.2 Linear and highly nonlinear approximation

Let us recall some definitions from approximation theory. For $\Omega \subset \mathbb{R}^d$ define an approximation family of functions in $L_p(\Omega)$ to be a sequence of nested and not necessarily linear subspaces $\{A_n\}$, $n \in \mathbb{N}$, $A_n \subset A_{n+1}$, such that $\bigcup A_n = L_p(\Omega)$. The approximation error of $f \in L_p(\Omega)$, relative to $A_n$, is

$$\sigma_n(f)_p = \inf_{y \in A_n} \|f - y\|_{L_p(\Omega)} ,$$
where the infimum is attained at the best approximant $f^*_n \in A_n$ (if it exists). The approximation spaces $A^\alpha_{p,q}$ are subsequently defined as:

$$A^\alpha_{p,q} = \left\{ f \in L^p(\Omega) \text{ s.t. } \{\sigma_n(f)\}_n \subset \ell_{\alpha,q} \right\},$$

where $\ell_{\alpha,q} = \left\{ \{x_n\}_{n=1}^\infty \text{ s.t. } (\alpha^n x_n) \in \ell_q^* \right\}$, $\ell_q^* = l_q(N, \frac{1}{n})$ denotes the weighted $l_q(N)$ space with the weights $\{\frac{1}{n}\}$: $l_q(N, \frac{1}{n}) = \left\{ \{x_n\}_{n=1}^\infty \text{ s.t. } \sum \frac{1}{n} \|x\|^q < \infty \right\}$.

We assume $p$ and $q$ to be fixed and denote $A_\alpha = A^\alpha_{p,q}$. Although the ultimate goal is to develop approximation methods that compute the best approximant, it is often feasible to find only a near-best approximant $f_n \in A_n$ to a given $f$, that satisfies the inequality:

$$\|f - f_n\|_{L^p(\Omega)} \leq C \sigma_n(f)_p,$$

where $C > 0$ is independent of $n$ and $f$ (for instance, see Section 2.3). Approximation theory also characterizes the approximation space $A_\alpha$ in terms of known functional spaces. Such characterization can be exploited for a comparison between different approximation methods (as explained in Section 2.1). However, the above two goals are rather connected since finding an upper bound estimation of the approximation error, such as $\sigma_n(f)_p \leq C \cdot (n^{-r})$, eventually requires the construction of an approximation method that attains the specified rate of approximation $O(n^{-r})$.

### Approximation complexity

We emphasize the importance that the index $n$ in $A_n$ exhibits the complexity of the functions in $A_n$. Defining the complexity of $f \in A_n$ as the amount of physical space needed to store a function $f$ in a computer memory (we make a practical assumption that an integer index and a real number occupy a fixed amount of a computer memory), we require:

$$\text{complexity}(f) = O(n),$$

for all $f \in A_n$.

The requirement (2.2) imposes a constraint on the approximation family $\{A_n\}$, and is commonly satisfied by known approximation families. Consider, for example, a bivariate approximation with piecewise polynomials over triangles, where $A_n$ denotes a nonlinear space of all bivariate functions constructed by $n$ polynomials of a fixed degree $k$ over $n$ disjoint triangles. The complexity of any function $f \in A_n$ is clearly proportional to $O(n)$, as it is specified by a set of $3 \cdot n$ vertices and $k \cdot n$ coefficients of the polynomials. This example can be extended to the approximation by piecewise polynomials over $n$-polygons, assuming the number of edges of the polygons is bounded. The complexity requirement (2.2) is also maintained by the wavelet based method of approximation, where $\{A_n\}$ denotes the nonlinear space of all linear combinations of any $n$ wavelets (at arbitrarily scale) from a given wavelet basis. The complexity of any function represented with $n$ wavelets is $O(n)$, as such function is defined with $n$ wavelet coefficients in the wavelet representation and $n$ integer indices of the corresponding wavelets (Section 2.3).

The complexity equivalence requirement (2.2) helps to eliminate some odd methods from consideration. For example, this requirement does not hold when $\{A_n\}$ denotes approximation with $n$ polynomials over $n$ subdomains with boundaries defined as free-form continuous curves. In addition, the complexity requirement (2.2) eliminates the trivial method with $A_1$ defined simply as a set of all functions, which approximates any function $f$ immediately by selecting the best approximant $f_1 \in A_1$ equals $f$. 
Approximation spaces

Linear approximation denotes the case when the approximation family \( \{A_n\} \) consists of linear spaces. Nonlinear approximation corresponds to the case when the approximation family consists of sublinear spaces \( A_n \):

\[
A_n + A_n \subseteq A_{kn},
\]

where \( k > 0 \) is a real constant. Highly nonlinear approximation dismisses the above assumptions of linearity or sublinearity and eventually corresponds to the most arbitrary sequence \( \{A_n\} \).

Let us show that in both cases of linear and nonlinear approximation the approximation space \( A_r \) in (2.1) is a linear space. Given \( x, y \in A_r \), assume the best approximants can be obtained for all \( n > 0 \) (i.e., inf is achieved). Denote the approximants of \( x \) and \( y \) as \( \{x_n\} \) and \( \{y_n\} \) respectively. The approximants \( x_n, y_n \in A_n \) satisfy:

\[
\{\sigma_n(x)\} = \{\|x - x_n\|\} \in l_{r,q}, \{\sigma_n(y)\} = \{\|y - y_n\|\} \in l_{r,q}.
\]

Now, since \( x_n + y_n \in A_{kn} \):

\[
\sigma_{kn}(x + y) \leq \|x + y - (x_n + y_n)\| \leq \|x - x_n\| + \|y - y_n\|.
\]

Thus

\[
(k \cdot n)^r \sigma_{kn}(x + y) \leq (k \cdot n)^r (\sigma_n(x) + \sigma_n(y)) \leq k^r (n^r \sigma_n(x) + n^r \sigma_n(y))
\]

since \( \{\sigma_n(x)\}, \{\sigma_n(y)\} \in l_{r,q}, \) and we conclude that

\[
\{(k \cdot n)^r \sigma_{kn}(x + y)\} \in l_{r,q}^r \leftrightarrow \{\sigma_m(x + y)\} \in l_{r,q} \leftrightarrow x + y \in A_r.
\]

Such a classification into linear and nonlinear approximation spaces \( A_r \) makes sense only under the complexity requirement (2.2). Otherwise, without the requirement (2.2) any sequence \( \{A_n\} \) can be turned into a sublinear sequence by an appropriate re-indexing. For example, assume that a sequence \( \{A_n\} \) is not sublinear and satisfies the most arbitrary condition \( A_n + A_n \subseteq A_{kn^2} \). Define a new index \( m = 2^n \) and select the subsequence of spaces \( \{A_m\} \). Such subsequence is sublinear, since for \( x, y \in A_m = A_{2^n} \):

\[
x + y \in A_{k \cdot 2^n}^2 = A_{2^n \cdot 2 \log_2 k} = A_{k \log_2 k \cdot m},
\]

i.e. \( \{A_m\} \) is a sublinear approximation sequence. Obviously, such re-indexing does not change the computational capability of the approximation family \( \{A_n\} \). Therefore, by requiring (2.2) we dismiss the trivial case of re-indexing, since the re-indexed subsequence \( \{A_m\} \) does not satisfy the requirement (2.2). Finally, the distinction between linear and nonlinear spaces \( A_r \) is essential, because one eventually seeks for a characterization of \( A_r \) by known linear spaces (such as \( C^r \), see Section 2.1).

Approximation methods with wavelets and piecewise polynomials

Image coding methods employ the best \( n \)-term wavelet approximation, that corresponds to nonlinear approximation since the approximation family \( \{A_n\} \) of the \( n \)-term wavelet is sublinear: \( A_n + A_n \subseteq A_{2n} \). Because any two functions \( x, y \in A_n \) are represented by \( n \) wavelets, therefore the function \( x + y \) consists of at most \( 2 \cdot n \) wavelets, i.e., \( x + y \in A_{2n} \). The method of approximation with piecewise polynomials over dyadic rings corresponds to nonlinear approximation as well. In both methods the \( n \)-term approximant has complexity \( O(n) \).
A useful theoretical result states an equivalence between the two methods: for spaces \( \{ A_n \} \) defined by wavelets \( f \in A_n \) if and only if \( f \in A_n \) for piecewise polynomials over diadic rings. This equivalence is evident in the special case of B-spline wavelets, which are finite linear combinations of B-splines (see [43]), since uniform B-splines can be represented with piecewise polynomials over dyadic rings. A general wavelet function may have no polynomial representation (for a comprehensive review of this topic see [23]).

The approximation family with piecewise polynomials over arbitrary disjoint triangles is more general than the approximation over dyadic rings, and satisfies the condition \( A_n + A_n \subset A_{k,n^2} \), which is more relaxed than the sublinear condition. Such a nonlinear quadratic condition follows the fact that inserting one triangle to a given triangulation may create \( c \cdot n \) new triangles. Therefore, the resulting triangulation produced from two triangulations consisting of \( n \)-triangles may consist of \( c \cdot n^2 \) triangles. Such quadratic condition appears to be the most relaxed condition for any approximation families \( \{ A_n \} \) employing \( n \)-term linear combinations. We are not aware of an example of approximation family that would be capable to produce an amount of terms larger than \( O(n^2) \) in a sum of two \( n \)-term combinations. A more general approximation family is based on piecewise polynomials in disjoint polygons, comprising a partition of the given domain. In order to satisfy the requirement (2.2), such polygons should have a bounded number of vertices.

### 2.3 Nonlinear approximation with wavelets

**Introduction**

Wavelets is a powerful mathematical tool thanks to such properties as compact support and multiscale hierarchy (multiresolution). Wavelets are now indispensable for a wide range of applications, including image coding. Even though the compression results of the wavelet based methods are comparable with the results of DCT (Difital Cosine Transform), the above properties of wavelets are essential for progressive streaming and yield more visual pleasing compressed images ([75]). In addition, the time-frequency localization property of wavelets is highly important theoretically since it allows to describe the Besov-Sobolev smoothness spaces in terms of wavelet coefficients (Section 2.3.1).

Below is a review of some theoretical results of wavelet approximation, relevant to the goals of image coding. Then the curse of dimensionality is explained, which is inherent to any tensor product bases and in turn explains the non-optimality of bivariate wavelet approximation used in image coding. The definitions in this section are in the general notation from approximation theory, while later in Section 4, discussing the application of approximation to image coding, it is assumed the number of dimensions \( d = 2 \) and \( p = 2 \) for the functional space \( L_p \) and the approximation error \( \sigma_n(f)_p \).

**Properties**

Many applications in image coding use wavelet bases or redundant wavelet families, such as wavelet frames. Let us review some basic definition of wavelet dyadic bases and multiresolution analysis, as it is presented in [14].

A multiresolution analysis consists of a sequence of successive approximation closed subspaces \( V_j \):

\[
\ldots V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots
\]
2.3. NONLINEAR APPROXIMATION WITH WAVELETS

with

\[ \bigcup_{j \in \mathbb{Z}} V_j = L^2(\mathbb{R}) , \]

\[ \bigcap_{j \in \mathbb{Z}} V_j = \{0\} , \]

and the additional requirement:

\[ f \in V_j \iff f(2^j \cdot) \in V_0 . \]

There exists also \( \phi \in V_0 \), often called a scaling function, so that \( \{ \phi_{0,k} = \phi(\cdot - k) ; k \in \mathbb{Z} \} \) is an orthonormal basis in \( V_0 \). It follows also that \( \phi_{j,k}(x) = 2^{-j/2} \phi(2^{-j}x - k), k \in \mathbb{Z} \) is an orthonormal basis for \( V_j \) for all \( j \in \mathbb{Z} \).

Define \( W_j \) to be the orthogonal complement of \( V_j \) in \( V_{j-1} \)

\[ V_{j-1} = V_j \bigoplus W_j , \quad j \in \mathbb{Z} . \]

We require also that there exists an orthonormal wavelet basis \( \{ \psi_{0,k}(x) = \psi_0(x - k) , k \in \mathbb{Z} \} \) of \( W_0 \), and define \( \psi_{j,k} = 2^{-j/2} \psi_0(2^{-j}x - k) \), then for fixed \( j \) the system \( \{ \psi_{j,k} ; k \in \mathbb{Z} \} \) constitutes an orthonormal basis for \( W_j \), and it also follows

\[ L^2(\mathbb{R}) = \bigoplus_{j \in \mathbb{Z}} W_j . \]

We call \( \psi_{j,k} \) a wavelet function, and the wavelet dyadic system generated by \( \psi_0 \) is the family \( \Psi = \{ \psi_{j,k} : j, k \in \mathbb{Z} \} \). Notice that \( \Psi \) is an orthonormal system and holds the perfect reconstruction property

\[ f = \sum_{\psi \in \Psi} < f, \psi > \psi , \text{ for all } f \in L^2(\mathbb{R}) . \]

This requirement on \( \Psi \) to be an orthonormal basis can be relaxed and replaced with biorthonormality ([14]), such that

\[ f = \sum_{\psi \in \Psi} < f, \tilde{\psi} > \psi , \text{ for all } f \in L^2(\mathbb{R}) , \]

where the function \( \tilde{\psi} \) is the dual wavelet to \( \psi \): \(< \tilde{\psi}_{j_1,k_1}, \psi_{j_2,k_2} > = \delta(j_1,j_2)\delta(k_1,k_2) \), (here \( \delta \) is the Kronecker symbol).

Wavelet systems are characterized by several desirable properties which may compete with each other and could be grouped in several categories (see [14] and the references there in):

(1) **The invertibility and redundancy of representation.** The system is required to be either orthonormal, bi-orthonormal, a tight frame, or a frame. Also the existence of a fast algorithm to compute the decomposition and the reconstruction is require.

(2) **The space-frequency localization of the system.** If the wavelets are compactly supported, one measures smoothness of the wavelets in \( \Psi \) and the size of the support.

(3) **Approximation properties of \( \Psi \).** This corresponds to the three notions, which are the approximation order of the underlying MRA, the number of vanishing moments of the wavelets \( \Psi \), and the approximation order of the wavelet system \( \Psi \). For instance, the number of vanishing moments of the wavelet corresponds to the approximation order of the wavelet system.
CHAPTER 2. APPROXIMATION

(4) Miscellaneous properties, which are dictated by the applications and include the symmetry of the wavelets, and the translation-invariance of the system $Ψ$.

We define wavelet approximation family and $n$-term wavelet approximation as a sequence of spaces $A_n$, $n \geq 0$, where $A_n$ is a space of linear combinations of $n$ wavelet functions. In the linear approximation the $n$ wavelets in $A_n$ are chosen a-priory at the $n$th scale $A_n = \text{span}\{ψ_i,−n\}_{i=0}^{n−1}$. The corresponding $n$-term approximant is the projection onto the $n$th multiresolution subspace $A_n$.

Nonlinear approximation was first investigated by Peetre and Sparr in 1972 [58], and Brudnyi in 1976 [6] (one can see the review of nonlinear approximation in [23]). The introduction of Besov spaces into linear approximation theory proves the superiority of nonlinear approximation over linear approximation. The superiority follows from the fact that linear approximation is characterized by Sobolev spaces, while nonlinear approximation is characterized by the more general Besov spaces.

2.3.1 Besov-Sobolev smoothness spaces

In nonlinear wavelet approximation $A_n$ consists of $n$ wavelets at different scales and locations, that means the best $n$-term wavelet approximant is chosen from a nonlinear finite subspace $A_n$, $\dim A_n = n$. The equivalence results for the wavelet nonlinear approximation are proven by DeVore, Jawerth, and Popov in [24] (see also [26]).

There is a number of equivalent definitions of the Besov spaces. Following the notation in [26], the Besov smoothness space $\mathcal{B}_q^α$ is defined via a characterization of the rate of wavelet approximation of a function $f \in L^p(\mathbb{R}^d)$:

$$\mathcal{B}_q^α(L^p) \equiv A_{p,q}^{α/d}.$$  

Here $A_{p,q}^{α/d}$ is an approximation space, $\mathcal{B}_p^α$ is the Besov smoothness space, and $\frac{1}{q} = \frac{α}{d} + \frac{1}{p}$. Namely:

$$\sum_{n=1}^{∞} \left[ n^{α/d} σ_n(f)_{L^p(\mathbb{R}^d)} \right]^q \frac{1}{n} < ∞ \leftrightarrow f \in \mathcal{B}_{p,q}^α(\mathbb{R}^d).$$  

However, in another definition of the Besov space $B_{p,q}^α$, for $p, q ≥ 1$, one can clearly see that the Besov space extents the Sobolev space $W^α_p = W^α(L^p(\mathbb{R}^d))$ having the $γ = α + β$ weak derivative, $0 < β < 1$:

$$f \in W^α_p(\mathbb{R}^d) \text{ and } w_1(f^{(α)}; t)_p = ε(t)t^β,$$

where $||ε(t)||_q^* < ∞$. The Besov space semi-norm is then:

$$|f|_{W^α_p(\mathbb{R}^d)} = \sum_{|k|=|α|} ||D^k f||_{L^p(\mathbb{R}^d)}$$

The Besov space $\mathcal{B}_{q,p}^α(\mathbb{R}^d) = \mathcal{B}_q^α(L^p(\mathbb{R}^d))$ denotes the functions of the Sobolev space $W^α_p = W^α(L^p(\mathbb{R}^d))$ having the $γ = α + β$ weak derivative, $0 < β < 1$:

where $w_1, p$ is a moduli of continuity (also known as modulus of smoothness, defined in Section 3.4), $1 ≤ p, q ≤ ∞$, and $|| · ||_q^*$ is the norm in the weighted $L_q$-space $L_q([0, ∞)^d, \frac{dt}{t})$. 
Here is the complete formal definition of Besov spaces as it appears in p.55 in [25], which extends for all positive $p, q$. Let $\alpha > 0$ and $0 < p \leq \infty$. Denote $r := [\alpha] + 1$, and the Besov semi-norm is defined for $0 < q < \infty$:

$$|f|^{q}_{B^\alpha_q(L^p(\mathbb{R}^d))} = \int_0^{\infty} \left( \frac{w_r(f, t)}{t^{\alpha}} \right)^q \frac{dt}{t},$$

and for $q = \infty$:

$$|f|^{B^\alpha_\infty(L^p(\mathbb{R}^d))} = \sup_{t>0} \frac{w_r(f, t)}{t^{\alpha}}.$$

The equivalence (2.3) is derived by establishing the Jackson-Bernstein inequalities (2.5), (2.6), which provide a general procedure to characterize wavelet approximation in terms of the smoothness spaces (see [26], [25]).

$$\sigma_n(f)_p \leq C n^{-\alpha/d} |f|_{W^{\alpha}_p}, \quad f \in W^{\alpha}_p \quad \text{(2.5)}$$

$$\|S\|_{W^{\alpha}_p} \leq C n^{\alpha/d} \|S\|_{L^p(\mathbb{R}^d)}, \quad S \in A_n \quad \text{(2.6)}$$

The derivation of the Jackson-Bernstein Equalities (2.5) and (2.6) is shown in ([27]) and [24].

### 2.3.2 Non-optimality of bivariate wavelet approximation

Multivariate $d$-dimensional wavelet is commonly constructed employing the tensor product approach, obtained as product of $2^d - 1$ univariate wavelets.

Equation (2.3) shows that the $\gamma$-smooth functions in $B^\gamma_{p,q}$ reside in the approximation space $A_{\gamma/d}^{p,q}$ where the corresponding rate of approximation is reduced by the factor $d$. Such $d$-times reduction of the approximation rate, known as the curse of dimensionality, imply that the computational cost increases exponentially with the dimension of the problem (in other words, a higher dimension should be compensated by a higher degree of smoothness). In the image coding applications $d = 2$, therefore the rate of approximation is halved.

The curse of dimensionality is explained from various perspectives in works aiming to surmount the curse of dimensionality ([44], [49], [28], [8]). For example, one can argue that the classical pointwise discontinuity of the Besov-Sobolev spaces is not general enough in higher dimensions and does not recognize the discontinuities along the lower dimensional manifolds, such as discontinuities along curves - edges - in the bivariate case $d = 2$. Another explanation points to the isotropic property of the tensor basis functions (Section 2.4).

The context of approximation with piecewise polynomials explains the non-optimality of bivariate wavelets in terms of partition of the planar domains. Namely, approximation with bivariate tensor product wavelets is proven to be equivalent to approximation with polynomials over dyadic squares partition, and such dyadic squares partition is not optimal. For example, a very large amount of dyadic squares is required, in order to approximate an indicator-like function of a domain with smooth boundaries (as shown in Section 3.2).

### 2.4 Examples of adaptive constructions

Multivariate classical wavelet approximation methods employ tensor product constructions of univariate wavelet functions and assumes equal smoothness in all the parameter directions. In view of the non-optimality of the multivariate classical wavelet approximation, more adaptive constructions, such as anisotropic functional
spaces, free-knots splines, and ridge-functions were investigated. This Section shortly describes these constructions.

We emphasize that these constructions are still classified as nonlinear approximation, which is essentially less optimal compared to the highly nonlinear approximation with piecewise polynomials in freeform domain. Therefore, a further developing of more arbitrary bases would require to depart from the classical tensor product. The next least restrictive and very fundamental representation is the piecewise polynomials in freeform domains.

Anisotropic bivariate Sobolev spaces were introduced by Nikol’skii in [56] and allow different smoothness in two different parameter directions, in contrast to the classical Sobolev $W^\alpha(L_p(\mathbb{R}^d))$ and Besov $B^\gamma_q(L_p(\mathbb{R}^d))$ smoothness spaces, which are isotropic as they characterize such functions that have the same smoothness ($\gamma$ and $\alpha$) in both parameter directions. The anisotropic Sobolev space $W^\alpha_1,\alpha_2$ is defined for $d = 2$

$$W^\alpha_1,\alpha_2 = \{ f(x_1, x_2) \text{ s.t. } \int \int \left| \frac{\partial^{k+m} f}{\partial x_1^k \partial x_2^m} \right|^p dx_1 dx_2 < \infty , 0 \leq k \leq \alpha_1 , 0 \leq m \leq \alpha_2 \},$$

and $d$-variate anisotropic Sobolev spaces can be defined similarly. Applications of such constructions are shown in [55]. However, as such anisotropic generalization still employs the tensor product construction of univariate wavelets, its approximation capability is still equivalent to the classical approximation with piecewise polynomials on dyadic squares.

Another anisotropic multivariate construction employs ridge functions. In its simplest form, a ridge function is a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ of the form $f(\overline{x}) = g(<\overline{a}, \overline{x}>)$ where $g : \mathbb{R} \to \mathbb{R}$ and $\overline{a}, \overline{x} \in \mathbb{R}^n$. The vector $\overline{a}$ is generally called the direction. In other words, a ridge function is a multivariate function constant on the parallel hyperplanes $<\overline{a}, \overline{x}>= c$, $c \in \mathbb{R}$.

Approximation by ridge functions of a function $f$ has the form

$$\sum_{k=0}^n g_k(<\overline{a}_k, \overline{x}>)$$

where the directions $\overline{a}_k \in \mathbb{R}^n$ and the functions $g_k$ are the variables. The idea here is to "reduce dimension" and thus bypass the curse of dimensionality. A solution to the $d$-dimensional problem of approximation is replaced with $k$ univariate approximations by introducing a projection $<\overline{a}_k, \overline{x}>$ of $\overline{x}$. The directions $\overline{a}_k$ are chosen to "pick up the salient features". See [59] and references therein for a survey on some approximation-theoretic questions concerning ridge functions.

Free-knot splines are also known as non-uniform B-splines (see [43] and references therein). A non-uniform $d$-variate B-spline basis function is defined with $d$ uniform non-uniform univariate B-spline basis functions using a tensor product construction. Essentially, as follows from its construction, the non-uniform representation is a linear combination of piecewise polynomials in an axes-aligned partition. Its corresponding approximation family is clearly sub-linear, and by that the approximation is non-linear and the corresponding approximation spaces are linear. That indicated that the free-knots splines based approximation is less general than the approximation with piecewise polynomials on arbitrary domains.
Chapter 3

Piecewise polynomials

3.1 Evidence for the superiority of our approximation method

In this Section we provide a detailed explanation of the two approximation approaches, which are central to this work, one based on approximation by wavelets and one on approximation with piecewise polynomials, and explain shortly a theoretical comparison between the two algorithms in terms of their corresponding approximation spaces. It is the theoretical evidence concluded from the comparison between these approximation spaces that encourages us to develop a new approximation algorithm for image coding, that would outperform wavelet-based algorithms.

Theoretical evidence

Before diving into all the details, we provide here an intuitive theoretical evidence that approximation, which is based on piecewise polynomials in freeform domains, is superior to the wavelet based approximation. Firstly, in Lemma 3.1.2 we show that any dyadic \( n \)-term partition can be turned into \( C \cdot n \)-term disjoint rectangular partition, where \( C \leq 9 \). That means, that any \( n \)-term B-spline function is a linear combination of at most \( C \cdot n \) piecewise polynomials defined in disjoint subdomains. Secondly, considering the fact that a B-wavelet function is a linear combination of \( M \) B-splines (for example, see [43]), where \( M > 0 \) is a constant for a given wavelet family, it follows that any \( n \)-term B-wavelet approximant can be represented with \( M \cdot C \cdot n \) piecewise polynomials. Finally, there is an equivalence, based on Besov-Sobolev characterization ([25]), between wavelet approximation spaces corresponding to different wavelets. All these imply that the approximation space corresponding to piecewise polynomials contains the approximation space corresponding to wavelets.

**Definition 3.1.1** We call \( \Pi_n = \{ b_i, \ i \in \mathbb{N}, i \leq n \} \) a dyadic partition of \( \Omega = [0,1]^2 \) if the following conditions are fulfilled:

(a) Every \( b_i \subseteq \Omega \) is a dyadic box, that is, there are exist \( k, l, m \in \mathbb{N} \) such that

\[
\begin{align*}
    b &= \left[ \frac{k}{2^m}, \frac{k+1}{2^m} \right] \times \left[ \frac{l}{2^m}, \frac{l+1}{2^m} \right].
\end{align*}
\]

(b) \( \bigcup_{i=1}^n b_i = \Omega \), and \( b_i \neq b_j \) if \( i \neq j \).

Notice that the dyadic boxes in \( \Pi_n \) can have intersections.

**Lemma 3.1.2** For any dyadic partition \( \Pi_n \) it can be found a refined partition \( \Pi'' \) of \( C \cdot n \) disjoint rectangles, that is, for any \( b \in \Pi_n \) there exists a subset \( \Pi_b \subseteq \Pi'' \) such that

\[
\begin{align*}
    b &= \bigcup_{b'' \in \Pi_b} b''.
\end{align*}
\]
Proof of lemma: here is an algorithm that constructs a rectangular partition $\Pi''$ with a number of rectangles bounded by $9 \cdot n$. For simplicity we assume $\Pi_n$ contains the box $\Omega$.

1. Notice that any two dyadic boxes in $\Pi_n$ can be either disjoint or nested.
2. The relation in (1) defines a unique tree with nodes corresponding to the boxes in $\Pi_n$, $\Theta = \{\text{node}_a, \text{edge}_{a,b}\}_{a,b\in\Pi}$, where $\text{edge}_{a,b}$ is defined for the box $a$ (child) and the smallest box $b \in \Pi_n$ (parent) that contains $a$. We employ the common tree notations: child, parent, grand child, grand parent, subtree, leaf (see for example [1] for in detail definitions).
3. For any $a, b \in \Pi_n$ we have:
   $$a \subset b \iff \text{node}_a \in \text{subtree}(\text{node}_b),$$
   and for any $\text{node}_a \in \Theta$ all its children are disjoint sets. The example of such tree is given in Figure 3.1.
4. Each box $a$ is assigned an initial partition $\Pi''_a = \{a\}$. The algorithm proceeds iteratively, at each step merging one parent with all its leaves in the following way.
   We define an initial set $B_0$ of boundaries consisting of the boundaries of the parent box $b$ and of its $s$ children boxes. In addition we define a set of initial vertices, consisting of the vertices of the $s$ children boxes and the parent. The merging of the parent with its children is done iteratively. After $t < s$ steps of merging $t$ leaves we are given a set of boundaries $B_t$ and a set of vertices $V_t$. The $(t+1)$th leaf is merged by extending its four boundaries in both directions, inside the box $b$, until an intersection producing a vertex not in $V_t$ occurs. The extended boundaries are added to $B_t$ and the new vertices are added to $V_t$ to produce $B_{t+1}$ and $V_{t+1}$.
   After merging all $s$ sons, the boundaries $B_s$ determine a rectangular partition $\Pi''_b$ of the parent $b$. By construction, this partition consists of at most $9 \cdot s$ disjoint rectangles and contains all the $s$ leaves.
5. The algorithm proceeds until all the boxes are inserted into the box $\Omega$ and resumes with $\Pi'' := \Pi''_\Omega$. Since in each basic merging step each merged box generates at most $9$ rectangles, the total number of rectangles in $\Pi''$ is at most $9 \cdot n$. Figure 3.2 illustrates the algorithm.

**Approximation component**

In spite of this clear theoretical evidence, the task of image coding imposes several problems. Assume a functional model behind an image, namely that a digital image
3.1. Evidence for the Superiority of Our Approximation Method

Figure 3.2:

is sampled from some bivariate function \( f : \mathbb{R}^2 \rightarrow \mathbb{R} \). As we show in Section 3.2, even a simple class of artificial piecewise constant images is associated with Besov-Sobolev linear functional space of a very low smoothness. In practice, the way to show the superiority of the piecewise polynomials approximation is implementing both algorithms and testing them on an acknowledged set of real-life images (the wavelet based algorithm has been already implemented in JPEG2000).

Statistical encoding component

The second practical problem we face in image coding is that the evidence of the superiority of piecewise polynomials is based solely on the approximation theoretical results. However, approximation is just one of the two main components in image coding, whereas the second component of thresholding and statistical encoding (Section 4.2) is crucial as well. Moreover, the development of statistical encoding algorithms specifically aimed at wavelet representations has been in the center of the research effort over the last ten years. The development of new encoding algorithms for the wavelet based image coding has steadily progressed and eventually produced the fourth generation highly sophisticated wavelet coders. Therefore, in order to outperform the modern wavelet based image coding algorithms, we need strong motivation, great enthusiasm and sound expertise.

Conclusion

This work can be considered as a testimony to the belief that a significant part of real-life images can benefit more with piecewise polynomial approximation than with the
wavelet approximation. Section 3.3.1 shows that the piecewise polynomials approximation performs by far more efficiently than the wavelets approximation methods for the special class of piecewise constant functions with smooth edges, since the latter method does not exploit edge singularities. We believe that taking a full advantage of edge singularities allows us to surpass even the most sophisticated wavelet based image coding algorithms.

3.2 Piecewise polynomials approximation over triangles

This Section describes a constructive proof for the upper bound estimate of the rate of approximation with piecewise polynomials over a partition by disjoint triangles. In Section 2.2 we illustrated that any method of highly nonlinear approximation is associated with nonlinear functional spaces and can not be characterized by linear smoothness spaces, such as Sobolev and Besov. We derive in this Section a classification of the highly nonlinear approximation spaces in terms of known approximation terms rather than smoothness terms. An example of such approximation-based classification approach can be found in [2] and [47]. The goal of this section is to introduce a new short and intuitive derivation of the classification. The conclusions provide an explanation for the key ideas of our algorithm of segmentation (described in Section 6.1.2 and 6.1.1). Particularly, this Section explains the idea of combined smoothness in bivariate approximation, based on classical definitions of 1-D and 2-D smoothness.

3.2.1 $\tilde{K}$-functional and $\tilde{B}$ smoothness spaces

Background

It has been discussed that while approximation with wavelets and with piecewise polynomials in dyadic squares correspond to linear approximation spaces (Section 2.3.2), the more adaptive highly nonlinear methods define nonlinear approximation spaces and cannot be characterized by classical linear smoothness spaces. Dekel, Leviatan and Sharir propose in [19] a new bivariate measure of smoothness that incorporates measures of smoothness in 1-D and 2-D dimensions. Such combined measure quantifies the amount of low-dimensional structure presented in the bivariate function. Let us now explain in more details the key results of [19], that include a new definition for a $K$-functional for the upper bound estimate of the piecewise polynomials approximation error in the Jackson-type inequality.

We present now the new smoothness measure in terms of a $\tilde{K}$-functional as it is defined in [19], and explain its two key properties. First of all, the new $\tilde{K}$-functional yields for any bivariate function the classical approximation rate corresponding to its Besov-Sobolev smoothness. Secondly, for the specific class of piecewise constant bivariate functions $f = \sum a_i 1_{\Omega_i}$, where $\Omega_i \subset [0,1]^2$ is a domain with a smooth boundary, the $\tilde{K}$-functional yields the optimal rate of $O(n^{-1})$. Roughly speaking, this means that if we were given a ‘magic’ solver, which can compute the infimum in the definition of the $\tilde{K}$-functional, then the approximant generated by such a ‘solver’ would be essentially more efficient than the wavelet approximant.

Definition of the $\tilde{K}$-functional

Let $f \in L_p(\Omega)$, $1 \leq p < \infty$, and $t > 0$. Take any partition $\Lambda$ of $[0,1]^2$ to be obtained with a set of continuous curves $b_j : [0,1] \to [0,1]^2$, $1 \leq j \leq n_E(\Lambda)$, each of finite
Figure 3.3: Example of partition of $[0, 1]^2$, with subdomains $\{\Omega_i\}$ and curves $\{b_i\}$

length, denoted by $\text{length}(b_j)$, which may intersect only at endpoints, such that the curves partition the image domain $\Omega = [0, 1]^2$ into open domains, $\Omega_k$, $1 \leq k \leq n_F(Λ)$ and a subset of the curves composes the boundary of $\Omega$ (see example in Figure 3.3). We define by $\Lambda$ both the set of curves $\{b_j\}_{j=1}^{n_E}$ and the domains $\{\Omega_k\}_{k=1}^{n_F}$.

**Definition 3.2.1** We attach to each curve $b_j$ a weight $t_j$, $1 \geq t > 0$ and we say that the partition $Λ$ is in $Λ(t)$ if $\sum_{j=1}^{n_E(Λ)} t_j < t - 1$ (in particularly it implies $n_E < t - 1$).

For a function $f \in L^p([0, 1]^2), 1 \leq p < \infty$, and $t > 0$, the $\widetilde{K}$-functional of order $r \in \mathbb{N}$ is defined as,

$$\widetilde{K}_r(f, t)_p := \inf_{Λ \in Λ(t)} ^{n_E(Λ)} \sum_{j=1} length(b_j) K_2(b_j, t_j^2)_{∞,1} + \sum_{k=1}^{n_F(Λ)} K_r(f, \Omega_k)^p_1/p,$$  (3.1)

where $K_2(b_j, \cdot)_{∞,1}$ is a $K$-functional measuring the (weak-type) smoothness of curves and $K_r(f, \Omega)_p$ is a $K$-functional measuring the (weak-type) smoothness of a bivariate function (see e.g. [25] for applications of $K$-functionals in approximation theory). For a curve $b : [0, 1] \rightarrow \Omega$, the quantity $K_2(b_j, \cdot)_{∞,1}$ is small if the curve is smooth, for example, if the $L_1$-norm of its second derivative is small. Also, this norm is small if the curve is only piecewise smooth, with a ‘small’ number of pieces and it is identically zero for a line segment. The quantity $K_r(f, \Omega)_p$ is small if $f$ is smooth in $\Omega$ and is identically zero if the function is a bivariate polynomial of degree $< r$ in $\Omega$. Thus in case that for a sufficiently small $t$, the curves $b_j$ align with the singularities of the function $f$, and in the partition subdomains $\Omega_k$ the function $f$ is smooth, then $K_r(f, \Omega)_p$ is small.

The novelty of the $\widetilde{K}$-functional is the way it combines the smoothness gauges of the curves and bivariate functions to give a geometric generalization of the classical $K$-functional. Let us emphasize two key advantages of the $\widetilde{K}$-functional. Firstly, having a more general form than the classical $K$-functional, the $\widetilde{K}$-functional associated approximation spaces $\widetilde{B}$ are in turn at least as precise as the classical (wavelet) approximation spaces. Secondly, the class of piecewise constant functions belongs to the $\widetilde{B}$ smoothness space with the optimal approximation rate $O(n^{-1})$, in contrast to the classical (wavelet) approximation rate $O(n^{-\frac{1}{2}})$ (see Section 2.3.2). For such functions it is shown in [19] that the quantity $K_r(f, n^{-1})_p$ is equivalent to the optimal approximation error $\sigma_{r,n}(f)_p$. Lastly, it is important to note the close relation between the $\widetilde{K}$-functional and the well-known Mumford-Shah functional [19], [54] (we also explain it in Section 6.1.3).
\( \tilde{A}_{r,p,q} \) approximation spaces

Following [19] we define now a highly nonlinear approximation with piecewise polynomials \textit{over triangles} for fixed polynomial degree \( r - 1 \geq 0 \) and a given polygonal domain \( \Omega \subset \mathbb{R}^2 \). We define the collection \( A_n = \{ \sum_{k=1}^{n} 1_{\Delta_k} P_k \} \subset L_p(\Omega) \), where \( \Delta_k \) are triangles with disjoint interiors such that \( \bigcup_{k=1}^{n} \Delta_k = \Omega \), \( P_k \) are polynomials of total degree \( r - 1 \), and \( 1_{\Delta_k} \) denotes the characteristic function of \( \Delta_k \). Denote for \( f \in L_p(\Omega) \)

\[
\sigma_{r,n}(f)_p = \inf_{S \in \Lambda_n} \| f - S \|_{L_p(\Omega)}. 
\]

We denote the corresponding approximation spaces by

\[
\tilde{A}_{r,p,q} = \{ f \in L_p(\Omega) \text{ s.t. } \{ \sigma_{r,n}(f)_p \}_{n=1}^{\infty} \in l_{a,q} \},
\]

here \( \{ x_n \} \in l_{a,q} \Leftrightarrow \sum \frac{1}{n}(n^ax_n)^q < \infty \) for \( q < \infty \), and for \( q = \infty \) \( \max_n n^ax_n < \infty \).

Jackson inequality and \( \tilde{B}_{r,n,p} \) smoothness spaces

Characterization of approximation spaces employs the procedure of Jackson-Bernstein inequalities (Section 2.3). A corresponding Jackson-type inequality for triangles is derived in [19] in the following form:

\[
\sigma_{r,n}(f)_p \leq C_1(p,r) \max(||f||_{L_\infty(\Omega)}, 1) \tilde{K}_r(f, C_2n^{-1})_p, \tag{3.2}
\]

here \( 1 \leq p < \infty \), \( f \in L_\infty(\Omega) \), and \( n \geq 1 \). Using the \( \tilde{K} \)-functional, [19] defines the new approximation classes \( \tilde{B}_{r,p,q} \):

\[
\tilde{B}_{r,p,q} = \{ f \in L_p(\Omega) \text{ s.t. } \tilde{K}_r(f, n^{-1})_p \in l_{a,q} \},
\]

whereas this Jackson inequality 3.2 establishes the equivalence between the \( \tilde{B}_{r,n,p} \) spaces and the approximation spaces \( \tilde{A}_{r,n,p} \), by providing an upper bound of \( \sigma_{r,n} \).

The difference between the Besov spaces and the \( \tilde{B} \)-spaces is that the \( \tilde{B} \)-spaces exhibit the most significant singularities along curves penalized by a measure of lower-dimensional smoothness of those curves. If a function \( f \in \tilde{B} \) represents an image, then the curve singularities of \( f \) are the edges in the image and the image is said to reveal a ‘low-dimensional structure’, what eventually constitutes the partition \( \Lambda \).

Applications of \( \tilde{K} \)-functional

Nonlinear approximation in dyadic squares is characterized by linear Sobolev-Besov smoothness spaces and employs the classical \( K \)-functional ([25]). It follows from its definition that the univariate \( K \)-functional \( K_2(b,s)_{\infty} \) vanishes for dyadic square partitions and by that the \( \tilde{K} \)-functional turns into the classical bivariate \( K \)-functional. But such dyadic square partitions constitute only a small part of the larger set of arbitrary partitions \( \Lambda(t) \). Therefore, the \textit{infimum} in (3.1) provides a more precise upper bound for the estimation of the error \( \sigma_{r,n}(f)_p \) than the classical \( K \)-functional.

As a result, the \( \tilde{B} \) smoothness classes, defined with the \( \tilde{K} \)-functional, characterize piecewise polynomials approximation more precisely than the classical Besov spaces, defined with the \( K \)-functional. However, we cannot expect any bivariate function to have any lower-dimensional geometric structure, and for ”non-structured” functions we don’t expect that piecewise polynomials approximation perform better than the classical approximation.

Computation of a close to optimal partition in Equation (3.1) is hard. The set of parameters in the infimum expression is not only unbounded but also uncountable.
Thus, it can not be computed in a reasonable time. All we know is that a near optimal partition $\Lambda(t = n^{-1})$ approximates the low-dimensional structure of a function. Our aim is to construct an approximation method that efficiently computes a partition, which is good enough to outperform the classical dyadic partition. We believe that the functions corresponding to images have sufficient amount of geometric structure and essentially can be better approximated by our method.

### 3.3 Approximation with polygonal piecewise polynomials

#### 3.3.1 Definition of the approximation spaces $A_{r,n,p}^α$

Bivariate approximation with piecewise polynomials of degree $r - 1 > 0$ in a planar domain $\Omega \subset \mathbb{R}^2$ repeats the definitions in Section 2.3 with the new collection defined as $A_n = \{ \sum_{k=1}^{n} \Delta_k P_k \}$, where the polygons $\{ \Delta_k \}$ have the total number of edges bounded, $\bigcup_{k=1}^{n} \Delta_k = \Omega$, $P_k \in \Pi$, are bivariate polynomials of degree not exceeding $r - 1$, and $1_{\Delta_k}$ denotes the characteristic function of $\Delta_k$. The approximation family $\{ A_n \}$ is a sequence of nonlinear spaces, and the error of approximation of $f \in L_p(\Omega)$ in $A_n$ is $\sigma_n(f)_p = \inf_{S \in A_n} \| f - S \|_{L_p(\Omega)}$. Consider the piecewise constant functions to be the indicator functions $f = 1_D$, where the subdomain $D \subset \Omega$ has a smooth boundary $\partial D = (\gamma_x(t), \gamma_y(t), \ t \in [0, 1]) \in W^1_2[0, 1]$. Approximation with a polyline $D_n = (\gamma_{x,n}(t), \gamma_{y,n}(t), \ t \in [0, 1])$ of the smooth boundary curve $\partial D$ is described by the univariate linear approximation

$$
\sigma_n^{1D}(\partial D)_\infty := \max_{t} \| \partial D - D_n \| = \max_{t} \sqrt{(\gamma_x(t) - \gamma_{x,n}(t))^2 + (\gamma_y(t) - \gamma_{y,n}(t))^2} \leq \sqrt{\| \gamma_x(t) - \gamma_{x,n}(t) \|_\infty^2 + \| \gamma_y(t) - \gamma_{y,n}(t) \|_\infty^2} \leq C n^{-2}.
$$

Assuming $f_n = 1_{D_n}$ the approximation of $f$ can be essentially estimated as follows,

$$
\sigma_n(f)_2^2 \leq \| f - f_n \|_2^2 \leq 2 \cdot \| f \|_\infty^2 \cdot \text{length}(\partial D) \cdot \sigma_n^{1D}(\partial D)_\infty \leq \tilde{C} n^{-2}, \tag{3.3}
$$

where $\text{length}(\partial D)$ is assumed to be bounded for the whole class of such piecewise constant functions, and $\| f \|_\infty = 1$. From $f_n \in A_n$ we conclude that the piecewise polynomials approximation attains the rate $O(n^{-1})$. For the class of piecewise constant functions the rate $O(n^{-1})$ is commonly known as the optimal rate (see Section 2.3.2). In the following Section we construct a method of approximation with piecewise polynomials of functions in $L_p(\Omega) \cap L_\infty(\Omega)$, which obtains an optimal rate for the subclass of piecewise constant functions.

#### 3.3.2 A simplified approach to the estimation of the error

For the approximation family $\{ A_n \}$ with the corresponding error $\sigma_n(f)_p$, denote the approximation spaces $A_{p,q}^α$, following the definitions in Section 3.3.1. Let the terms of classical approximation be $\{ \tilde{A}_n \}$, $\tilde{\sigma}_n(f)_p$ and $\tilde{A}_{p,q}^α$, which correspond to the classical approximation with piecewise polynomials over dyadic squares or to wavelets. Such classical approximation is characterized in terms of the Sobolev-Besov smoothness spaces (Section 2.2). Approximation with polynomials in arbitrary polygons is more general since approximation in dyadic squares is its special case. Therefore,

$$
\sigma_n(f)_p \leq \tilde{\sigma}_n(f)_p
$$
for any $f \in L_p(\Omega)$, and consequently by definition,

$$\tilde{A}_{p,q}^\alpha \subseteq A_{p,q}^\alpha,$$

for $p, q > 0$ and $\alpha > 0$.

Such estimation of the approximation spaces $A_{p,q}^\alpha$ in terms of the classical linear spaces $\tilde{A}_{p,q}^\alpha$ is too rough and is very crude in some cases. Therefore, we construct here a new method of approximation with piecewise polynomials that has two distinctive properties. Firstly, this method attains the optimal rate $\sigma_n(f) = O(n^{-1})$ for indicator functions of domains with a smooth boundary, i.e. $f = 1_D$, $\partial D \in W^1_2(0,1)$. Secondly, our method provides for all functions in $L_p(\Omega)$ at least the rate of classical approximation, which is equivalent to the rate corresponding to the wavelet approximation. Our method provides a constructive derivation of an upper-bound estimate for $\tilde{\sigma}_n$. The class of piecewise constant functions with smooth boundary singularities belongs to $A_{p,q}^\alpha$ for $\alpha = 1$ and $q = \infty$, whereas the classical approximation attains the lower rate $O(n^{-\frac{1}{2}})$.

Finally we draw practical conclusions for our algorithm of approximation with piecewise polynomials. The above characterization of the approximation spaces $A_{p,q}^\alpha$ is derived without employing the smoothness terms such as $K$-functional and modulus of smoothness. An example of such approach is described in [7] (see Section 4 and references therein, more examples can be found also in [2] and [47]).

**Approximation of functions in the $C-$class**

We describe our approach in several steps, applied each time to a more general class of solutions. Let the $C-$class denote the functions with one discontinuity along a continuous curve $C(t) : [0,1] \to \Omega$, here $C(t) \in W^1_2[0,1]$ splits $\Omega$ into two subdomains $\Omega_1$ and $\Omega_2$.

Now we introduce a simple $C$-method that looks for the discontinuity $C(t)$ in some given bivariate function $f \in L_p(\Omega)$. For the $C-$class functions, the $C$-method proceeds by performing in each of the two subdomains a dyadic squares polynomial approximation.

Let $\tilde{\sigma}_{nD}(C(t))_\infty$ denote the rate of univariate approximation in the $L_\infty$-norm with a polyline $P_n(t)$ with $n$ segments. For $C(t) \in W^1_2[0,1]$, $\tilde{\sigma}_{nD}(C(t))_\infty = O(n^{-2})$. Now we cover the curve $C(t)$ with a band (Section 7.2) having the width $\tilde{\sigma}_{nD}(C(t))_\infty$. Then the partition of $\Omega$ into three regions is obtained

$$\tilde{\Omega}_1 \subset \Omega_1, \tilde{\Omega}_2 \subset \Omega_2, \text{and} \quad \Omega_B = \Omega \setminus \{\tilde{\Omega}_1 \cup \tilde{\Omega}_2\},$$

(3.4)

here $\Omega_B$ denotes the band. The reason for introducing the band is to prevent the subdomains $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$ from overlapping, by keeping the discontinuity $C(t)$ inside the band $\Omega_B$. The $C$-method proceeds gradually, simultaneously refining the polyline and the corresponding subdomain $\Omega_B$, and performing the dyadic squares approximation in the two subdomains $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$. In the 2-D domain $\Omega \in \mathbb{R}^2$ a procedure of dyadic partitioning is defined by splitting the domain at its center into four squares at each step, and the $n$-term approximant consists of $n$-polynomials and $n$ bits representing the dyadic partition. In order to obtain a better approximation we should think of more arbitrary partitioning.

The approximant in the subdomain $\Omega_B$ interpolates linearly values from the neighboring $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$. Assume (without proof) that the approximation of a curve $C(t)$ with an $n$-segment polyline produces a $2n$-polygon band $\Omega_B$ (in the case of a self-intersection free polyline such an assumption is obvious), therefore we require the total
3.3. APPROXIMATION WITH POLYGONAL PIECEWISE POLYNOMIALS

number of polygons in the subdomains $\Omega_B, \Omega_1, \Omega_2$ to be bounded by $3 \cdot n$. Eventually, the $C$-method leads to the estimate

$$\sigma_{3 \cdot n}(f, \Omega)^2 \leq \inf_{\Omega_1 \cup \Omega_2 = \Omega} \{ \sigma_n(f, \Omega_1)^2 + \sigma_n(f, \Omega_2)^2 + \|f\|_\infty^2 \text{Area}(\Omega_B) \} .$$

By 3.4 and by

$$\text{Area}(\Omega_B) \leq \text{length}(C(t)) \tilde{\sigma}_n^{D}(C(t))_\infty = O(n^{-2}) ,$$

we get

$$\sigma_{3 \cdot n}(f, \Omega)^2 \leq \inf_{\Omega_1 \cup \Omega_2 = \Omega} \{ \sigma_n(f, \Omega_1 \cup \Omega_2)^2$$

$$+ \|f\|_\infty^2 \text{length}(C(t)) \tilde{\sigma}_n^{D}(C(t))_\infty \} , \quad (3.5)$$

where the first term decay faster than the last term.

Note that Equation (3.5) illustrates a combined characterization of $\sigma_n(f, \Omega)_p$ in terms of a univariate approximation $\tilde{\sigma}_n^{D}(C(t))_\infty$ and a bivariate approximation $\tilde{\sigma}_n(f, \Omega_k)_p$. Finally, the $C$-method attains the goal as it obtains the optimal rate $O(n^{-1})$ for functions in the $C$-class.

Approximation of functions with curved discontinuities

We introduce a more general $(n_F, n_E)$-class of piecewise smooth functions with smooth discontinuities $\{C_j\}_{j=1}^{n_E}$ and subdomains $\{\Omega_k\}_{k=1}^{n_F}$. We extend our $C$-method to apply to a set of smooth edges $\{C_j\}, C_j \in W^1_2([0, 1] \to \Omega)$. Each curve $C_j$ is approximated with an $m_j$-segment polyline $C_{j,m_j}$ in $L_\infty([0, 1])$ norm. We cover $C_{j,m_j}$ with a band $\Omega_{B_j}$ which is a polygon with $2m_j$-edges. Such a band $\Omega_{B_j}$ separates the neighboring subdomains from each other.

This method of approximation employs 1D approximation to construct the polylines and a classical 2D approximation outside the bands in order to construct the approximant. Then, the approximation error can be estimated by

$$\sigma_{3 \cdot n}(f, \Omega)^2 \leq \inf_{\sum_{i=1}^{n_F} \sum_{m_j=m}^{n} \Omega_k = \Omega} \{ \sigma_n(f, \bigcup_{k=1}^{n_F} \Omega_k)^2$$

$$+ \|f\|_\infty^2 \sum_{j=1}^{n_E} \text{length}(C_j) \tilde{\sigma}_n^{D}(C_j)_\infty \} . \quad (3.6)$$

Here we employed the assumption that for $\Omega_B = \bigcup_{j=1}^{n_E} \Omega_{B_j}$

$$\text{Area}(\Omega_B) \leq \sum_{j=1}^{n_E} \text{length}(C_j) \tilde{\sigma}_n^{D}(C_j)_\infty$$

and that every polyline $P_j(t)$ defined with $m_j$-segments is covered by a polygonal band with $2m_j$-sides. Since $\sum_{j=1}^{n_E} \text{length}(C_j) \tilde{\sigma}_n^{D}(C_j)_\infty = O(n^{-2})$ and the first term decays faster then the second, it can be seen that our method attains the optimal rate $O(n^{-1})$ for functions in the $(n_F, n_E)$-class.
Approximation of functions in the general case

So far our method and the corresponding estimate in Equation (3.6) target the specific class of functions \( f \) having a well defined partition, i.e. a set of discontinuities \( \{ C_j(t) \}_{j=1}^{n_E} \), and subdomains \( \{ \Omega_k \}_{k=1}^{n_F} \). Therefore, we can naturally extend the estimate in (3.6), by bringing the search for the partition under the infimum condition. Thus, given a function \( f \in L_2(\Omega) \bigcap L_\infty(\Omega) \), our final estimation inequality holds:

\[
\sigma_{3-n}(f, \Omega)^2 \leq \inf_{(c_j)_{j=1}^{n_E} s.t. \sum_{j=1}^{n_E} m_j = n} \{ \tilde{\sigma}_n(f, \bigcup_{i=1}^{n_F} \Omega_i)^2 \}
\]

\[
+ \| f \|_\infty^2 \sum_{j=1}^{n_E} \text{length}(C_j) \tilde{\sigma}_{m_j}^{1D}(C_j)_\infty \}
\]

where \( n_E \) denotes the number of curves \( C_j \) which determine a partition of the domain \( \Omega \) into \( n_F \) domains \( \Omega_j \). Here the optimal solution of the infimum is sought over all partitions \( \{ C_j \} \) and all univariate approximations of \( C_j \) with \( m_j \)-segment polylines and bivariate approximations of \( f \) in \( \bigcup \Omega_i \) by \( n \) dyadic squares. Notice that for the special case of \( n_E = 0 \) (implying \( n_F = 1 \)) Equation (3.7) yields the upper bound estimate \( \sigma_n \geq \tilde{\sigma}_n(f, \Omega) \). That means that the estimate in Equation (3.7) is better than the classical estimate with \( \tilde{\sigma}_n \), and the that corresponding approximation rate is faster than the classical approximation rate.

Equation (3.7) contains many unknowns \( \{ C_j(t) \}, n_E, \{ m_j \} \), and it is hard to estimate the infimum over an uncountable set of parameters. Still, two practical conclusions can be drawn. The first conclusion is that Equation (3.7) still yields the optimal approximation rate for functions \( f \) having smooth discontinuities \( C_j(t) \) while being pointwise smooth away from \( C_j \). Such functions can be approximated with the optimal rate using the constructive method leading to Equation (3.6). The second conclusion points out that in approximating a general function with piecewise polynomials over subdomains \( \{ \Omega_k \} \), a satisfactory partition should contain smooth curves \( \{ C_j \} \), having a small error of polyline approximation \( \tilde{\sigma}_n^{1D}(C_j)_\infty \), weighted with \( \text{length}(C_j(t)) \), where the approximation error by polylines is bounded by \( \tilde{\sigma}_n^{1D}(C_j)_\infty \leq \| C_j \|_{W^2[0,1]} \). Following (3.7) the partition curves \( \{ C_j \} \) should coincide with the discontinuities of \( f \) in order to maintain \( f \) smooth in the subdomains \( \{ \Omega_k \} \), by that improving the classical approximation error \( \tilde{\sigma}_n(f, \Omega_k) \).

3.4 Approximation by a single polynomial in a complicated domain

In our work it is an important task to construct a good polynomial approximation inside subdomains of an image partition. This section reviews relevant results on \( L_2 \) bivariate polynomial approximation in planar domains. By analyzing an example of a family of polynomial approximation problems, we arrive at an understanding of the nature of domain singularities for approximation by polynomials. This understanding is the basis for the measure of domain singularity proposed in the next section, and used later in our GPP algorithm.

3.4.1 \( L_2 \) Error

The \( L_2 \) error of bivariate polynomial approximation in a convex planar domain \( \Omega \subset \mathbb{R}^2 \), can be characterized by the smoothness of the function in the domain (it also
applies for 'almost-convex' domains, see definitions and details in [16], [17]). These results can be formulated in terms of the moduli of continuity/smoothness of the approximated function, or of its weak derivatives. Here we cite results on general bounded domains.

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain and let $f \in L_2(\Omega)$. For $m \in \mathbb{N}$, the $m$th difference operator is:

$$\Delta^m_h(f, \Omega)(x) = \begin{cases} \sum_{k=0}^{m} (-1)^{m+k} \binom{m}{k} f(x + kh), & [x, x + kh] \subset \Omega, \\ 0, & \text{otherwise}, \end{cases}$$

where $h \in \mathbb{R}^2$, and $[x, y]$ denotes the line segment connecting the two points $x, y \in \mathbb{R}^2$. The $m$th order $L_2(\Omega)$ modulus of smoothness is defined for $t > 0$ as

$$\omega_m(f, t, \Omega)_2 = \sup_{|h| < t} \| \Delta^m_h(f, \Omega) \|_{L_2(\Omega)}, \quad (3.8)$$

with $|h|$ the Euclidian norm of $h \in \mathbb{R}^2$.

Denote by $\Pi_n$ the linear space of bivariate polynomials of total degree $n$, then the approximation error is defined as

$$E_n(f, \Omega)_2 = \inf_{p \in \Pi_n} \| f - p \|_{L_2(\Omega)}. \quad (3.9)$$

This quantity is equivalent in convex domains to the modulus of smoothness of $f$, namely there exist $C_1, C_2 > 0$ such that

$$C_1\omega_n(f, diam(\Omega), \Omega)_2 \leq E_n(f, \Omega)_2 \leq C_2\omega_n(f, diam(\Omega), \Omega)_2 \quad (3.10)$$

(see [17] for further details). While the constants $C_1$ and $C_2$ in (3.10) depend on $n$, for a general domain $\Omega$ the constant $C_2$ may also depend on the geometry of $\Omega$. In particular, the Bramble-Hilbert lemma states that for $f \in W^m_2(\Omega)$, $m \in \mathbb{N}$, where $W^m_2(\Omega)$ is the Sobolev space of functions with all weak derivatives of order $m$ in $L_2(\Omega)$, there exists a polynomial $p_n \in \Pi_n$ for which

$$|f - p_n|_{m, 2} \leq C(n, m, \gamma)(diam(\Omega))^{m-k}|f|_{m, 2}, \quad k = 0, 1, \ldots, m,$$

where $|\cdot|_{m, 2}$ the Sobolev semi-norm. It is important to note that in [16] the dependence on the geometry of $\Omega$ in case of convex domains is eliminated.

When the geometry of the domain is complicated then the smoothness of the function inside the domain does not guarantee the quality of the approximation. Figure 3.4 shows an example of a smooth function in a highly non-convex domain which is poorly approximated there.

### 3.4.2 An instructive example

Here we show that (3.10) cannot hold with a constant $C_2$ independent of the domain, by an example that "blows-up" the constant $C_2$. For this example we construct a smooth function $f$ and a family of planar domains $\{\Omega_{\epsilon}\}$, such that for any $n > 0$, $\omega_n(f, \Omega_{\epsilon})_2 \to 0$ as $\epsilon \to 0$, while $E_n(f, \Omega_{\epsilon})_2 = O(1)$.

Let $S$ denote the interior of the square with vertices $(\pm 1, \pm 1)$, and let $R_\epsilon$ denote the rectangle with vertices $(\pm(1 - \epsilon), \pm \frac{1}{2})$. The domains of approximation are $\{\Omega_{\epsilon} = S \setminus R_\epsilon\}$. The function $f$ is defined by

$$f(x) = \begin{cases} 1, & x \in S \cap \{x : x_2 > \frac{1}{2}\}, \\ 0, & x \in S \cap \{x : x_2 < -\frac{1}{2}\}, \end{cases}$$
where \( x = (x_1, x_2) \), and \( f \) is smooth in \( S \).

It is easy to verify that \( \omega_n(f, t, \Omega_\epsilon)_2 \to 0 \) as \( \epsilon \to 0 \). We claim that \( E_n(f, \Omega_\epsilon)_2 \) for small \( \epsilon \) is bounded below by a positive constant. To prove the claim assume that it is false. Then there exists a sequence \( \{\epsilon_k\} \), tending to zero, such that \( E_n(f, \Omega_\epsilon_k)_2 \to 0 \). Denote by \( p_k \) the polynomial satisfying \( E_n(f, \Omega_\epsilon_k)_2 = \| f - p_k \|_{L_2(\Omega_\epsilon_k)} \). Since there is a convergent subsequence of \( \{p_k\} \), with a limit denoted by \( p^* \), then \( \| f - p^* \|_{L_2(\Omega_\epsilon_k \setminus R_0)} = 0 \), which is impossible.

The relevant conclusion from this example is that the quality of bivariate polynomial approximation depends on the smoothness of the approximated function and also on the geometry of the domain. This example is valid for \( 1 \leq p < \infty \), and for \( p = \infty \) there can be constructed another complicated example, which we leave beyond the scope of this thesis.

### 3.4.3 The distance defect ratio as a measure of domain singularity

A possible measure of the non-convexity of a domain, that is used in our GPP algorithm, is based on the distance defect ratio of two points in a domain. Let the \textit{distance defect ratio} of \( x, y \in \text{cl}(\Omega) = \Omega \cup \partial \Omega \) (with \( \partial \Omega \) the boundary of \( \Omega \)) be

\[
\mu(x, y)_\Omega = \frac{\rho(x, y)_\Omega}{|x - y|}
\]

where \( \rho(x, y)_\Omega \) is the length of the shortest path inside \( \text{cl}(\Omega) \) between \( x \) and \( y \).

Note that in convex domains the distance defect ratio is 1, while in arbitrary domains it is greater than 1. Indeed, we observe that in the domains \( \{\Omega_\epsilon\} \) of the example above, and also in the example in Figure 3.4(c) there exist pairs of points with distance defect ratio greater than 1.

### 3.4.4 Measure of geometric singularity

Here we propose a measure of the \textit{singularity} of a domain, assuming that convex domains have no singularity. Later, we present two algorithms which aim at reducing
3.4. APPROXIMATION BY A SINGLE POLYNOMIAL IN A COMPLICATED DOMAIN

Figure 3.5: (a) Example of a subdomain in the initial partition of Cameraman image, (b) example of one geometry-driven partition with a straight line.

Figure 3.6: (a) Subdomain $\tilde{\Omega}$ generated by the partition in Figure 3.5, (b) its convex hull $H$, (c) the corresponding disjoint components $\{C_i\}$ of $H \setminus \Omega$.

the singularities of the domain where the function is approximated; one by partitioning it into subdomains with smaller singularities, and the other by mapping it into a less singular domain in higher dimension. For our GPP algorithm we use the first algorithm.

The measure of domain singularity we propose, is defined for a domain $\Omega$, such that $\rho(x, y)_\Omega < \infty$, for any $x, y \in \partial \Omega$. Denote the convex hull of $\Omega$ by $H$, and the complement of $\Omega$ in $H$ by

$$ C = H \setminus \Omega. $$

The set $C$ may consist of a number of disjoint components $C = \bigcup C_i$.

A complicated planar domain $\tilde{\Omega}$, the corresponding sets $H$ and $C$, the latter consisting of several disjoint components $\{C_i\}$, are shown in Figure 3.6.

Note that each $C_i$ can potentially impede the polynomial approximation, independently of the other components, as is indicated by the example in Section 3.4.1. For a component $C_i$ we define its corresponding measure of geometric singularity relative to $\Omega$ by

$$ \mu(C_i)_\Omega = \max_{x, y \in \partial C_i \cap \partial \Omega} \mu(x, y)_\Omega, \quad (3.12) $$

with $\mu(x, y)_\Omega$ the distance defect ratio defined in (3.11). We denote by $\{P_1^i, P_2^i\}$ a pair of points at which the maximum in (3.12) is attained and $\rho(P_1^i, P_2^i)$ is maximal (Figure 3.7). We propose the following measure of geometric singularity of the domain $\Omega$

$$ \mu(\Omega) = \max_i \mu(C_i)_\Omega. $$
We refer to the \( \text{ith} \) (geometric) singularity component of the domain \( \Omega \) as the triplet: the singularity component \( C_i \), the distance defect ratio \( \mu(C_i)_{\Omega} \), and the singularity pair of points \( \{P_i^1, P_i^2\} \). The domain singularity is defined then as a set of all the singularity components.
Chapter 4

Wavelet-based image coding

4.1 Wavelet image coding as modern application of nonlinear approximation

Approximation is the first of the two main steps in image coding, where the original image is transformed into a new set of coefficients in some functional representation. Those coefficients undergo quantization (also known as thresholding or shrinkage), which maps these coefficients into a sequence of integer indices and essentially introduces distortion. At the second stage of encoding, the ordered sets of quantized indices are statistically encoded to form the resulting bit-stream. This step of encoding is invertible and introduces no distortion.

Let’s see now how approximation theory helps to deal with image coding. Namely, to compute an approximant that has a lower complexity, compared to the complexity of a functional model of the image. We illustrate this using a functional model based on wavelets, since wavelets present a common mathematical tool in approximation. For example, the Sobolev-Besov smoothness spaces can be characterized using the wavelet coefficients of a function ([25]).

Let a digital image $I$ be a finite matrix of pixel values (which are usually given as integer values in the range $0 \ldots 255$). Assign the integer $m$ to the highest original resolution and denote by $D_m$ the indexed set of the pixel squares of $I$. At the $k-1$th resolution the size of the pixel length is doubled relative to the $k$th resolution, thus the pixel areas are related as $|I_{k-1}| = |I_k|/4$, here $I_{k-1} \in D_{k-1}$, $I_k \in D_k$.

Let $\{\phi\}$ and $\{\tilde{\phi}\}$ be a scaling function system and its dual, and $\{\psi\}$ and $\{\tilde{\psi}\}$ is a corresponding wavelet system and its dual wavelet system. Then we assume that the pixel values are produced from averaging of some unknown real-valued function $f(x)$:

$$P_I = < f, \tilde{\phi}_I >, I \in D_m.$$  

where $I$ is the squared region corresponding to a pixel from $D_m$. Essentially we employ a bivariate functional model to represent images with $I$ denoting the bivariate indexing, compared to the scalar indexing in the univariate setting in Section 2.

In the above functional model we associate with the image $I$ a real-valued function $f$:

$$f_m = \sum_{I \in D_m} P_I \phi_I = \sum_{I \in D_m} < f_m, \tilde{\phi}_I > \phi_I$$

where $\phi_I = 2^m \phi_0(2^m \cdot -(i, j))$ is a bivariate scaling function corresponding to pixel $I$ with center $(i, j)$. Notice that the function $f$ is essentially approximating (but not interpolating) the pixel values $P_I$.

In practical computational procedures with wavelets it is often required the domain of the digital image $I$ to be extended beyond its rectangular boundaries such
that the image continuity to be preserved (a.k.a. continuous boundary condition), for example by padding or by reflection of its values around the boundary, which helps to decrease the number of computations.

The function $\phi$ is selected so that its smoothness is higher than the assumed smoothness of the image function $f(x)$. Selecting $\phi$ too smooth results in a higher computation time ([14]), therefore in practical image processing applications usually a simple $\phi$ is used. In the classical multiresolution approach based on wavelets and the two-scale relation, $f$ can be represented in the form:

$$f = f_0 + \sum_{k=0}^{m-1} \sum_{l \in D_k} \sum_{e \in \Gamma} a_k^e \psi_k^e,$$

(4.1)

where $\Gamma$ is a set of some three bivariate wavelet functions $\psi_k^e$, and the lowest resolution version $f_0$ is:

$$f_0 = \sum_{i \in D_0} P_0^i \phi_0^i.$$

Here $\phi_0^i = \phi_0(1 \cdot -(i, j))$ where $\phi_0$ is the scaling function at the lowest resolution, and $P_0^i$ are assumed to be the pixel values of the image at the lowest resolution. Notice again, that the low resolution version of the image is the set of the pixel values $P_i^0$, and not values of the function $f_0$.

Following definitions given in Section 2.2, a wavelet approximation family is a sequence of spaces $\{A_n\}_{n \in \mathbb{N}}$, where $A_n$ is the union of all the linear spans of any $n$-term wavelet functions:

$$A_0 \subset A_1 \subset ... A_n \subset ... ,$$

and

$$A_n + A_n \subseteq A_{2n},$$

here the space $A_0$ contains only the function 0. In the best $n$-term wavelet approximation one selects $f_n$ with those $n$ terms $A_n^f \subset \{\psi_k^e\}$, that minimize the error $\sigma_n(f)_p = \|f - f_n\|_{L_p}$:

$$f_n = f_0 + \sum_{\psi_k^e \in A_n^f} a_k^e \psi_k^e.$$

Obtaining the approximant $f_n$ constitutes the first stage of an image coding algorithm. Eventually, we construct a sequence of approximants $\{f_n\}$, $f_n \in A_n$.

In case of orthonormal wavelets $\{\psi_k^e\}$, the best $n$-term approximant $f_n$ for $p = 2$ is obtained by selecting the $n$ terms $a_k^e \psi_k^e$ corresponding to $n$ biggest $a_k^e$ from the wavelet representation in Equation (4.1). The selection of the biggest terms requires sorting and essentially has an above linear algorithm complexity. However, linear complexity can be achieved by using the wavelet thresholding method (explained in Section 4.2).

### 4.2 Step-2: wavelet thresholding and encoding

The best $n$-term wavelet method (Section 4.1) presents the nonlinear wavelet approximation. In image coding algorithms, the wavelet coefficients are then quantized and finally the locations and the values of non-zero quantized coefficients are encoded. The quantization can be done by a standard uniform quantization, or perhaps with a slightly enlarged dead-zone around zero; this results in more zero symbols being generated, leading to better compression with limited increase in distortion. The encoding uses either Huffman or arithmetic coding. In practical applications of image coding there are further ingenuity in quantizing the coefficients. Advanced approaches
are also employed in leveraging the set of wavelet coefficients, which provide a bet-
ter compression, although at the cost of complexity (for in-depth explanations and
examples of encoding see [76], [75]).

Here we illustrate the thresholding operation. The common method of a hard
thresholding is defined as

$$\theta(a, t) = \begin{cases} 
0 & \text{if } |a| < t \\
\frac{a}{t} & \text{else}
\end{cases}$$

and a soft thresholding has the form:

$$\theta(a, t) = \begin{cases} 
0 & \text{if } |a| < t \\
2 \cdot (|a| - t) \cdot \text{sign}(a) & \text{if } t \leq |a| < 2 \cdot t \\
\frac{a}{t} & \text{else}
\end{cases}$$

(4.2)

with $x_+ = x$ if $x > 0$ and zero otherwise. Local thresholding assumes that the coeffi-
cients of the wavelet representation in Equation (4.1) are thresholded independently
of each other. Than, the local-threshold wavelet approximant $f^*$ in the representa-
tion of Equation (4.1) has instead of $a^x_I$:

$$f^* = f_0 + \sum_{k=0}^{m-1} \sum_{I \in D_k} \sum_{e \in \Gamma} \theta_{k,I}(a^x_I, t) \psi^e_I.$$

(4.3)

The choice of the threshold parameter $t$ in $\theta_{k,I}$ in Equation (4.3) may also depend
on the level and the wavelet, and is dictated by the asymptotic behavior of the approximant $f^*$ ([37], Section 11).

JPEG2000 format employs encoding and ordering techniques based on the algo-
rithm of Embedded Block Coding with Optimized Truncation (EBCOT) (see [75]).
In contrast to the zero-tree algorithm [70], which encodes the entire set of coeffi-
cients, EBCOT divides each resolution layer $D_k$ into fixed-sized blocks of coefficients
$\{B^k_l\}_{l=1}^{L_k}$, $D_k = \bigcup_{l=1}^{L_k} B^k_l$, and encodes thresholded coefficients in each block $B^k_l$ inde-
pendently. Such independent block encoding doesn’t exploit the redundancy between
different blocks within the same resolution layer $B^k_l$, $B^s_l$, $l \neq s$. Neither it exploits
the correlation between different resolution layers, that is in the basis of the zero-tree
encoding schemes.

Surprisingly, these drawbacks of the EBCOT algorithm are greatly compensated
by the fact that the contribution of each code-block $B^k_l$ to the final approximant
$f_n$ may be optimized independently. Even though individual blocks are coded more
efficiently when exploiting parent-child relationships, this imposes a constraint on the
quality level $\theta_s(\cdot)$ of parent and child blocks. The simpler EBCOT algorithm without
such inter-layer dependency yields superior compression results. Since the blocks
$B^k_l$ are encoded independently, we are free to select a block $B^k_l$ at any quality level
$\theta_{s_{k,l}}(\cdot)$ in order to construct the approximant $f_n$ that minimizes the overall distortion
$\|f - f_n\|_{L_p}$:

$$f_n = f_0 + \sum_{(k,l) \in B^k_l} \sum_{i \in \Gamma} \theta_{s_{k,l}}(a^x_i) \psi^e_i.$$

4.3 Non-optimality of transform-based image cod-
ing

In spite of all the efforts to develop new bases and representations for images, industry
still relies solely on transform based methods which have a linear-time complexity,
such as Fast Fourier Transform and Wavelet Transform. Wavelets have become popular in image coding due to development of wavelet methods of approximation with linear-time complexity (Section 4.2). Methods of wavelet-based image coding have been a subject for intensive research over the last decade and have been increasingly improving and becoming more sophisticated. At this time, the method of wavelet-based image coding of Taubmann ([75]) reports one of the best compression results and serves as a prototype for the JPEG2000 standard.

Linear transform-based image coding relies on a tensor-product functional model for image representation, and a sublinear approximation family. For example, a wavelet-based image coding method employs tensor-product construction of a univariate wavelet. As a result, any method of linear and nonlinear approximation operates with a sublinear approximation family, which is redundant (and eventually less powerful) compared to an arbitrary approximation family (Section 3.1). Such redundancy of wavelet-based approximation is explained by the equivalence of approximation with wavelets and with piecewise polynomials over dyadic squares. The dyadic squares axes-aligned partitioning is far from optimal in approximating a simple indicator function, as the number of squares is large compared to a partition with arbitrary polygons (as shown in Figure 4.1 reproduced from [4]). Also, approximation with wavelets is not optimal since the wavelet linear approximation spaces constitute a subset of the more general nonlinear approximation spaces associated with piecewise polynomials over arbitrary partition (Section 2.2). We phrase this non-optimality by saying that the wavelet approximation does not exploit the geometrical low-structure singularities in images.
Figure 4.1: Dyadic partition and arbitrary partitions with triangles. In the arbitrary partition (right) the polygons are aligned along the boundary, which helps to reduce their amount, compared to the amount of polygons in the dyadic partition (left)
Part III

Geometric Piecewise Polynomials (GPP) image coding algorithm
Chapter 5

GPP coding algorithm

5.1 Overview of other geometric algorithms

A dozen of years ago an attempt was made to challenge the classical transformed-based image coding with geometry-based algorithms. The new algorithms exploited a partition of image into subdomains by planar curves (segmentation), and received the name of second generation coding. In this section we illustrate by one example of such a geometric algorithm, the two major problems that prevented from the methods of second generation coding to attain competitive results. Subsequently, we present our ideas for solving these problems, that we have applied in developing the GPP algorithm.

The geometric-based algorithm of image coding in [22] employs edge-based representation of the image by constructing pixel chains with the attached intensity data at their two sides, whereas these pixel chains are assumed to coincide with the jumps in image intensities (edges). The intensity values outside the pixel chains are interpolated from the data at the sides of surrounding pixel chains. The pixel chains are found using Sobel edge detection operator, after that they are transformed with a loss from 8-connected to a simpler 4-connected pixel chains and eventually lossless encoded with arithmetic chain coding (Section 8.1.1 has a detailed description of the chain encoding algorithm). The attached to the contours intensity data is encoded at fixed-length pieces as averages.

The major problem that prevented from the geometric algorithm in [22] to attain competitive results for real-life images was the large encoding cost of 1-D geometry (the pixel chains). Considering this problem, we point out that our approach assumes the bivariate function to have essentially 2-D and 1-D components with the corresponding 2-D and 1-D smoothness. As a result, we consider in our GPP algorithm 1-D geometry (curved singularities in the image) to be a subject for 1-D approximation. Lossy compression of the edges significantly reduces the amount of data in the encoding, whereas the error of approximation is compensated by covering the edges with wide bands (detailed description is in Section 5.3 and 7.2). The numerical results in Section 8.1.1 and 8.1.2 show a significant reduction in the encoding cost of the 1-D geometry.

The second idea helps us to significantly improve the quality of the 2-D approximation inside the subdomains of the image partition, which we attain at the cost of the simplest dyadic squares partitioning. Our approach to image partition relies on the theory of function approximation in planar domains (Section 3.4). Eventually, we developed a segmentation method that proceeds with further partitioning of the initial image segmentation (obtained by the edge singularities in the image) in order to remove the geometric singularities of the segmentation subdomains. Such a geometric singularity is shown to obscure the efficiency of 2-D approximation (as illustrated in Section 3.4).
5.2 GPP key concepts

We now highlight three key concepts of our GPP algorithm. First, we model an image as a piecewise smooth bivariate function with curve singularities of weak type smoothness. We apply a segmentation algorithm derived from the $\tilde{K}$-functional of [19]. We then approximate the detected edge singularities, where the distortion is resolved with bands (Section 7.2). As a result, the GPP algorithm attains the optimal approximation rate for piecewise constant images (Section 3.2.1). Finally, we address the inefficiency of polynomial approximation of smooth functions over nonconvex domains. In planar nonconvex domains we regard the singularity to be twofold, whereas in addition to the singularity of the function there is also the geometric singularity of the planar domain, as it is explained in Section 3.4.

5.3 GPP outline

Here is a general description of our GPP algorithm, which is explained in detail in the following sections.

Segmentation: finding edges Given an image we first apply the segmentation algorithm of Section 7.1 that captures the significant edges in the images.

Segmentation: edge pruning Given a target bit-rate, the edges are then pruned using the $\tilde{K}$-functional model.

Edge encoding The pruned edges are then encoded using the lossy adaptive algorithm of Section 8.1.2, which is closely related to adaptive polygonal curve approximation.

Finding partition In Section 7.2 we describe how the pruned edges are allocated some width, so that they become ‘bands’. After that, to better fit the $\tilde{K}$-functional model, we apply further partitioning of the segmentation domains. This step, detailed in Section 7.3.2, is almost a ‘pure-geometric’ algorithm, that the decoder can apply without requiring too much information from the encoder.

2-D encoding In Section 8.2 we give details of the last step of the algorithm where polynomial approximation, quantization and coding is performed over each of the final subdomains.

In Section 9.2 we provide experimental results and compare the performance of the GPP algorithm with Kakadu implementation [74] of the EBCOT algorithm [73].

GPP segmentation

The first step of our coding algorithm is a segmentation procedure whose goal is to approximate the solution of the $\tilde{K}$-functional introduced in [19]. Therefore, since a segmentation which is a near-minimizer of (3.1), leads to a construction of a good piecewise polynomial approximation, we may conclude that designing an algorithm that tries to find such a segmentation is the key to good performance of our coding algorithm. This task is still an on-going research project, where we try to draw upon the close relationship between the $\tilde{K}$and the Mumford-Shah functionals, and the fact that there is vast literature on techniques that find approximate minimizers of Mumford-Shah-type functionals.

In Section 6.1.2 we describe our segmentation algorithm. For the purpose of low-bit rate coding, we found that the following simple heuristic algorithm works sufficiently well. Figure 5.1 provides an example of segmentation produced by the initial step of GPP algorithm. One can view the output of the segmentation algorithm as a bit-map, where pixels that are assigned the value 1 are the segmentation pixels. One of the main properties of the zero-crossing algorithm is its connectivity, where
the segmentation pixels form continuous segmentation curves. With the minimization of the $\bar{K}$-functional (3.1) in mind, we now prune the collection of segmentation curve portions (we assume each curve to consist of 50-pixel length curve portions) based on the following criteria: smoothness of the curve portions (normalized by their length) and the behavior of the gradient of the target function (image) on these curve portions. Obviously, the amount of pruning relates to the target bit-rate we wish to achieve in our coding algorithm.

The pruned segmentation can serve as a good basis for coding of geometric images, for example, graphic-art images that are simple piecewise constant with spline edge singularities. However, it is not sufficient when one tries to encode real-life images, because until now, we have not fully taken into account (3.1) or alternatively, the performance of polynomial approximation on the segmentation domains. In Section 7.3.2, we elaborate on further partitioning of the obtained segmentation with the pruned curve portions, so that the final segmentation provides a sufficiently good approximation to a near-minimizer of (3.1).

Following the algorithm description it is important to notice that the algorithm achieves the optimal $\frac{1}{n}$ approximation rate for pure geometric images. It is easily seen from the step-by-step description, just ”feed” the geometric image into the algorithm flow and see that it actually resembles 1-D approximation along the 1-D contours.
Chapter 6

Image Segmentation

Segmentation plays an important role in our geometry-based image coding algorithm, because our method employs partitioning, which in turn is seen as a segmentation of an image into subdomains (as it is illustrated in Section 1.3 and 5.1).

Segmentation aimed at image coding finds its explanations in approximation theory. Smoothness spaces $\tilde{B}$ (Section 3.2), defined by means of the $\tilde{K}$-functional, characterize highly nonlinear approximation with piecewise polynomials. A solution to the $\tilde{K}$-functional $\tilde{K}_r(f, \frac{1}{n})_p$ is a partition $\Lambda$ having a complexity $O(n)$ and comprised of $\Lambda_E$-edges and $\Lambda_F$-subdomains (Section 3.2.1). In contrast to the classical $K$-functional, there is no closed form solution $\Lambda$ of the $\tilde{K}$-functional, and solving for $\Lambda$ is essentially a nonlinear problem (Section 3.3.2). Were we given a ‘magic-box’ able to find an optimal solution of $\tilde{K}_r(f, \frac{1}{n})_p$ for each $n$ and any bivariate function $f$ and we could accomplish an optimal method of approximation with piecewise polynomials. Such method attains the optimal approximation rate $O(n^{-1})$ for piecewise constant functions $f$ and by that is more efficient than wavelet approximation for functions $f$ having a geometric structure. As long as we are not granted such a ‘magic-box’, we are forced to search for a partition $\Lambda$ on our own. As it is hard to estimate ‘optimality’ of the solution besides the case of piecewise constant functions $f$, we consider a partition $\Lambda$ sufficiently optimal once its corresponding approximants attain results in image coding, competitive with the best known image coders.

We tried several approaches when solving the $\tilde{K}$-functional. Section 6.1.3 illustrates the link between $\tilde{K}$-functional and the well-known Mumford-Shah functional, and explains the variational solution of Mumford-Shah functional. In Section 6.1.1 we describe another region growing segmentation algorithm, which is essentially a greedy algorithm driven by the heuristic function derived from the $\tilde{K}$-functional. Greedy algorithms are known to be suboptimal as they do not guarantee to find the optimal solution. Yet there are numerous improvement methods turning a given greedy algorithm into ‘less greedy’. Finally, in Section 6.1.2 and 7.3.2 we present a segmentation algorithm that we employ in our image coding algorithm.

6.1 Segmentation methods in image coding

6.1.1 Global segmentation: region growing

The approaches in image segmentation may be divided into global and local varieties. For example, Section 6.1.2 recalls a local approach based on the the gradient operator or Laplacian operator, whereas Section 6.1.3 depicts the global approach to image segmentation based on functional minimization.

This section describes a global segmentation algorithm of region growing, that divides an image domain into subdomains, until some criteria are satisfied. Region
Algorithm of region growing

Approximation using similarity criteria

In some simplified settings the zero-area edge (curved discontinuity) \( e \) has two adjacent regions \( d_1 \) and \( d_2 \) with the corresponding number of pixels \( n_1 \) and \( n_2 \). Let \( p_1, p_2 \) be the approximating polynomials in the regions \( d_1 \) and \( d_2 \), and let \( \sigma_1 \) and \( \sigma_2 \) denote the corresponding \( l_2 \)-errors of the approximation with \( p_1 \) and \( p_2 \).

Let \( d = d_1 \cup d_2, n = n_1 + n_2, \) and \( \sigma \) denote the \( l_2 \)-error of approximation (variance) of a given sequence \( f = \{ f_i \} \) with a polynomial \( p \) in the region \( d \):

\[
\sigma^2 = \| f - p \|^2_{l_2(d)} = \frac{1}{n} \sum_{i \in d} (f_i - p(x_i, y_i))^2 ,
\]

(6.1)

where \( f = \{ f_i \} \) denote the values at the pixels \( (x_i, y_i) \in d \). Approximation with one polynomial in the region \( d \) has a larger error by \( \Delta(e) \), compared to approximation with two polynomials \( p_1 \) and \( p_2 \):

\[
\Delta(e) = n\sigma^2 - (n_1\sigma^2_1 + n_2\sigma^2_2)
\]

the distance \( \text{dist}(e, d_1, d_2) \) denotes the relative loss in the approximation error as the result of merging of the two subdomains \( d_1 \) and \( d_2 \) into one subdomain \( d \):

\[
\text{dist}(e, d_1, d_2) = \frac{\Delta(e)}{n_1\sigma^2_1 + n_2\sigma^2_2 + \epsilon} = \frac{n\sigma^2}{n_1\sigma^2_1 + n_2\sigma^2_2 + \epsilon} - 1 ,
\]

(6.2)

where the regularization constant \( \epsilon > 0 \) helps to handle the case of \( \sigma_1 \) and \( \sigma_2 \) being equal or near zero. In the more general settings the edge \( e \) may have a non-zero area (the number of pixels \( n_e > 0 \)):

\[
\Delta(e) = \sigma^2_{p_1 \cup p_2 \cup e} - (n_1\sigma^2_1 + n_2\sigma^2_2 + n_e\sigma^2_e)
\]

where approximation of a value in the edge \( e \) is computed by linear interpolation of the nearest values from the adjacent regions \( d_1 \) and \( d_2 \) and the \( l_2 \)-error of such approximation is denoted \( \sigma^2_e \).

Algorithm of region growing

The algorithm for region growing is greedy and global. It starts with most detailed partition, and at each step it seeks in the whole image domain for two candidate subdomains to be merged. The most detailed partition into subdomains is shown in Figure 6.1(b). The algorithm starts with a construction of the neighborhood graph \( G(E, D, V) \) of the edges \( E \), subdomains \( D \) and vertices \( V \), and the relations: for each subdomain \( d \in D \) each edge \( e \in E \), and each vertex \( v \in V \) the following subsets are constructed: \( D_d, D_e, D_v \subset D, E_d, E_e, E_v \subset E, \) and \( V_d, V_e, V_v \subset V \). Here \( D_v \) denote all the domains that contain the vertex \( v \), \( E_d \) is a set of all the edges that ‘touch’ with two its vertices the domain \( d \), and so on.

After that, for each edge \( e \in E \) its cost is computed, for example the corresponding distance \( \Delta(e) \). Finally, we construct a sorted list of all the edges subject to the cost \( \Delta(e) \).

At each step, the algorithm removes the least useful edge \( e^* \) from the tail of the cost-sorted list, and deletes \( e^* \) from the graph \( G(E, D, V) \). In order to delete the edge \( e^* \), it is needed to recalculate the new polynomial approximant \( p_d \) in the one-neighborhood of \( e^*: d = \bigcup d_i, \) here \( d_i \in D_{e^*} \). After that, the costs \( \Delta(e') \) are recalculated, \( e' \in E_{d_i} \), for all \( d_i \). The intermediate stage of the algorithm after...
6.1. SEGMENTATION METHODS IN IMAGE CODING

Figure 6.1: (a) Part of Cameraman image, (b) initial grid with 112 edges, (c) its corresponding piecewise polynomial approximation, PSNR=19.4769 dB

Figure 6.2: (a) Intermediate stage with segmentation based on 29 of 112 edges in Figure 6.1(b), (b) its corresponding piecewise polynomial approximation, PSNR=19.2255 dB

several steps of edge removal is shown in Figure 6.2(a). The algorithm will stop after meeting a given condition, such as a threshold on the total $l_2$ approximation error or the number of subdomains (Figure 6.3(a)).

Region growing is a robust algorithm that produces a clear final partition with disjoint subdomains and linked edges. However, it does not converge to the optimal solution, which is the best piecewise polynomial approximant, defined over the optimal partition. Yet, there are various ways of improving the greedy algorithms into ‘less’ greedy, such as employing the more sophisticated heuristics (cost functions), or making several steps at once by removing several edges at each iteration.

New region growing algorithm based on combined 1-D and 2-D criteria

The merging criteria in Equation 6.2 is solely based on the 2-D approximation error $\sigma$ in Equation 6.1. Following the discussion in Section 3.2, we offer a new algorithm of region growing that takes into account the 1-D geometry smoothness in order to construct the optimal segmentation aimed for image coding.

We have shown in Section 3.2 that approximation with piecewise polynomials in arbitrary domains seeks for a reasonable partition $\Lambda$ of the image domain into subdomains of smoothness (where the function is smooth) with smooth partition curves. Therefore, we should direct our region growing algorithm towards constructing subdomains of smoothness with smooth boundary edges $e$ in order to produce a reasonable
CHAPTER 6. IMAGE SEGMENTATION

(a) Final segmentation has 4 edges, (b) its corresponding approximation, PSNR=17.1735 dB

Figure 6.3: (a) Final segmentation has 4 edges, (b) its corresponding approximation, PSNR=17.1735 dB

partition.

We add an additional weight to the criteria \( \Delta(e) \) in Equation (6.2) that exhibits the 1-D smoothness of the curved edge \( e : [0, 1] \to \mathbb{R}^2 \). We can estimate the 1-D approximation error of \( e(t) = (e_x(t), e_y(t)) \) with an approximating \( n \)-segment polyline \( P_n(t) = (e_{n,x}(t), e_{n,y}(t)) \) using the inequality:

\[
\| |e(t) - P(t)| \|_\infty \leq \| |e''(t)| \|_{L_1[0,1]} \frac{1}{n^2},
\]

where \( e''(t) = (e''_x(t), e''_y(t)) \). In practice, \( e(t) \) is a pixel chain and at each pixel the value \( |e''(t)| \) can be estimated with the inverse radii through the three adjacent pixels \( e(t-1), e(t), e(t+1) \). The \( L_1 \) norm \( \| |e''(t)| \|_1 \) is estimated then as a sum of such inverse radii for all \( t \). In practice, we found more stable to compute an estimation of the value of \( |e''(t)| \) as average of several radii, where each radii is computed for the three pixels \( e(t-k), e(t), e(t+k) \), for several values of \( k > 0 \).

The 2-D approximation criteria \( \Delta^{2D}(e) \) in Equation 6.2 has a direct relation between the value of \( \Delta(e) \) and the retention of \( e \), i.e., the smaller is the value \( \Delta^{2D}(e) \) the less likely that \( e \) will be kept. Furthermore, we define the 1D criteria \( \Delta^{1D}(e) \) to denote the 1D geometric cost of the edge \( e \):

\[
\Delta^{1D}(e) = \frac{1}{\| |e''(t)| \|_{L_1}}, \\
\Delta(e) = \Delta^{2D}(e) + \alpha \Delta^{1D}(e) = \frac{1}{\| |e''(t)| \|_{L_1}},
\]

where \( \alpha \geq 0 \) is a predefined parameter that can give more weight for bringing more 'geometry' (edges) into final encoding.

Figure 6.4 illustrates an example of partition produced with the combined 1-D and 2-D criteria \( \Delta(e) \). One can see that such a partition has smoother curved edges, compared with the 'zig-zag' edges of the partition in Figure 6.2(a).

Summary

The described greedy algorithm has large complexity, as it requires to manage nine large data sets of the graph \( G(E, D, V) \). We eventually selected for the GPP encoding one of the segmentation algorithms based on edge detection (described further in the text), which has a linear running time complexity and is also simple for implementation.
6.1. SEGMENTATION METHODS IN IMAGE CODING

6.1.2 Local segmentation: edge detection

We provide in this section a brief introduction to some fundamental edge detection techniques and explain the reason we decided on the local second-order edge detection to be employed in the partition step of our GPP algorithm (described in Section 7.1).

Edge detection is an important component of image segmentation, as partitioning curves in a segmentation essentially coincide with the curved singularities (edges) in the image. The term ‘edge’ refers to the specific points where an image intensity is distinctly transformed from low to high value. In the functional based model an edge corresponds to a function discontinuity of the function. There are variations in edges, in widths and shapes (straight lines or curves). There are several edge-detection techniques, each one has its own strong points as well as its shortcomings. Experiments are vital for determining what is the best edge detection technique for a given application.

First-order edge detection

Most local edge detectors are based on measuring the intensities gradient at a particular bounded location in the image, where the gradient operator is

$$||\nabla|| = ||\frac{\partial}{\partial x}, \frac{\partial}{\partial y}||.$$  

The corresponding discrete operator employs the finite difference approximation $\frac{\partial f}{\partial x} \sim f(i+1,j) - f(i,j)$ and $\frac{\partial f}{\partial y} \sim f(i,j+1) - f(i,j)$:

$$\text{Grad} f = \left(\frac{\partial}{\partial y}, \frac{\partial}{\partial x}\right) f,$$

where the edges correspond to the magnitude value in $\text{Grad} f$. Local edge-detection methods may employ other difference operators, such as Roberts, Kirsch, Prewitt, and Sobel ([50], [57], [36], [78]).

Second-order edge detection

The edges in real-life images are often not sharp drop-offs but gradual transitions of the intensity. Usually gradient magnitude rises up to some peak and then gradually falls.
CHAPTER 6. IMAGE SEGMENTATION

There are several operators that find directly the peaks in the gradient magnitude. In the one-dimensional case, the peak is located at the point where the derivative is a local maximum or minimum. Therefore, differentiating the gradient magnitude and finding places where it is zero requires finding places where the second derivative vanishes.

These zeroes of the second derivative rarely fall exactly on a pixel in practice but typically between pixels. Zero crossings locates these zeroes by finding places where one neighbor pixel is positive and another is negative. The important property of zero crossing is that this approach provides closed paths (except where the path extends outside the image border). The well-known problem of the zero-crossing methods is that they are extremely sensitive to noise.

The Laplacian operator computes a second derivative:

\[(\nabla)^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2},\]

and the edges locations in the image \(I\) are found at the zero-crossings \((\nabla)^2 I = 0\).

Zero-crossing of Laplacian of Gaussian

ZeroCross(LoG) - ZeroCross of Laplacian-Gaussian - is a known segmentation method that locates edges (curved singularities) in images as closed curves. The method applies to discrete images, whereas its derivation can be done in continuous settings.

Now we find the ZeroCross(LoG) operator. Given a continuous \(I\), a convolution filter is employed to approximate the second derivative (Laplacian operator). Second derivative measurements are exceedingly sensitive to the noise, which correspond to small but sharp intensity changes. Therefore, such noise is compensated with a Gaussian smoothing, applied prior to the Laplacian operator. The resulting edge detector of Laplacian of Gaussian(LoG) encompasses two convolutions with Gaussian and Laplacian filters consecutively. Since the convolution operation is associative, the Gaussian can be convolved with the Laplacian filter, producing a merged filter, i.e., the whole LoG operator can be precalculated in advance. Than, only one convolution (the filter pass) needs to be performed with the image \(I\), where the computation time of LoG is linear in the image size. Finally, the operation of zero-crossing determines the locations in the image where its Laplacian’s sign changes (crosses zero). Such locations correspond to the edges in the image, and the whole algorithm of ZeroCross(LoG) produces edges as closed curves.

In the discrete case both Laplacian and Gaussian operators are final-length convolution operators, and

\[\text{LoG}(I) = L(G(I)) = L * (G * I) = (L * G) * I.\]

Consider now a continuous functional setting with \(I : \Omega \rightarrow \mathbb{R}\), where \(\Omega = [0, 1]^2\) is the image domain. The result of segmentation is image partition, which consists of univariate curves and disjoint subdomains.

Since the ZeroCross(LoG\(_\sigma\)) operator is extremely sensitive to variation of the image Gradient, rather than to the variation of the intensities \(I(x, y)\) (the image values) (see [57], [36]), the \(\sigma\) parameter is determined for each image \(I\). A small value of \(\sigma\) in LoG\(_\sigma\) results in smoothing with a narrow Gaussian, and \(\sigma < 0.5\) pixels has no effect on a discrete grid, i.e., the LoG\(_\sigma\) filter becomes Laplacian. On the other hand, large \(\sigma\) dismisses the undesired noise, and also makes the zero-contours smoother, moving them away from the original location of the edges in the image.

Automatic selection of the parameter \(\sigma\) is an important practical problem, in the GPP algorithm we determine \(\sigma\) from the target compression rate (Section 7.1).
6.1.3 The Mumford-Shah functional and the $\tilde{\kappa}$-functional

In [53] and [54] Mumford and Shah describe an approach for segmentation of a bivariate function in order to obtain a ‘compact’ representation for functions having some lower-dimensional structure. Assume a given image is represented by a bivariate function $g : \Omega \rightarrow \mathbb{R}$, $\Omega = [0,1]^2$. It is possible that $g$ does not have a derivative, however $g$ can be approximated by a piecewise smooth function $u : \Omega \rightarrow \mathbb{R}$ and a set of univariate curves $E$, which correspond to the minimum of the Mumford-Shah functional:

$$\mathcal{F}(u, E) = w_1 \int_{\Omega - E} (u - g)^2 + w_2 \int_{\Omega - E} |\nabla u|^2 + \int_E ds ,$$

here $E = \bigcup b_k$ denotes the curved edges $b_k : [0,1] \rightarrow \Omega$ in the image $u$, and the last term denotes the total length of the edges $E$. The positive values $w_1$, $w_2$ determine the desired mutual relation between the three terms: the approximation error $\|g - u\|_{L^2(\Omega - E)}$, the smoothness of the approximating function $\|\nabla u\|_{L^2(\Omega - E)}$, and the total length of the edges $E$ in $u$, whereas it is assumed that the function $u$ is $C^1$-smooth away from the edges $E$: $u \in C^1(\Omega - E)$.

The simpler form of the Mumford-Shah functional is obtained for the case $\|\nabla u\|_{L^2(\Omega - E)} = 0$, which corresponds to approximation with a piecewise-constant bivariate function $u$:

$$\tilde{\mathcal{F}}(u, E) = w_1 \int_{\Omega - E} (u - g)^2 + w_2 \int_E ds ,$$

(6.3) and the corresponding minimization solution of $\tilde{\mathcal{F}}(u, E)$ is described in [54]. However, minimization of the general Mumford-Shah functional $\mathcal{F}(u, E)$ imposes a difficult numerical problem, with the functional $\mathcal{F}(u, E)$ containing 2-D and 1-D parameters (in [34] there are descriptions of other such 2-D and 1-D combined functionals).

The main difficulty that hampers attempts to minimize the Mumford-Shah functional $\mathcal{F}(u, E)$ numerically is the necessity to develop a representation for the set $E$, whereas the number of possible sets of curved edges is enormous.

Solution based on approximation of the Mumford-Shah functional

Consider the task of approximating the Mumford-Shah functional by a ‘simpler’ functional. Let $u$ be $C^1$-continuous away of the curved edges in $E$. In [3] a new approximation functional, that employs a 2-D parameter $v : \Omega \rightarrow \mathbb{R}$ instead of the edges $E$, is introduced,

$$\mathcal{F}_\epsilon(u, v) = \int_\Omega (w_1 v^2 |\nabla u|^2 + w_2 ((v - 1)^2 / 4\epsilon + \epsilon |\nabla v|^2) + w_3 |u - g|^2)ds ,$$

(6.4) where $w_1$, $w_2$, $w_3$ are positive constants, and $v$ approximates $1 - \chi_E$ in the task of the minimization of $\mathcal{F}_\epsilon$ as it is nearly 0 where $|\nabla u|$ is large (presumably at the edges $E$), and close to 1 where $u$ is smooth. Then, the approximating functional $\mathcal{F}_\epsilon(u, v)$ is an elliptic functional, and can be minimized numerically. We denote

$$\mathcal{F}_\epsilon(u, v) = \int_\Omega F(u, u_x, u_y, v, v_x, v_y)dxdy ,$$

here $u = u(x,y)$ and $v = v(x,y)$. The corresponding Euler-Lagrange variational equations are:

$$\delta \mathcal{F}_\epsilon(\delta u) = 0 , \delta \mathcal{F}_\epsilon(\delta v) = 0 .$$

Assuming $F_u, F_{u_x}, F_{u_y}, F_v, F_{v_x}, F_{v_y}$ exist we have:

$$F_u - \frac{d}{dx} F_{u_x} - \frac{d}{dy} F_{u_y} = 0$$
CHAPTER 6. IMAGE SEGMENTATION

Figure 6.5: Segmentation of Lena image produced by minimization of the Mumford-Shah functional with the method of steepest descent iterations

\[ F_v - \frac{d}{dx} F_{vx} - \frac{d}{dy} F_{vy} = 0 \]

with \( F_u = \frac{\partial F}{\partial u}, F_{ux} = \frac{\partial^2 F}{\partial u^2}, \) etc.

The numerical solution employs the iterative method of steepest descent: a time variable \( t \) is formally added to the functions \( u \) and \( v \), and two more gradient descent equations are exploited:

\[
\begin{align*}
    u_t(t, x, y) &= -C_u \frac{\partial F}{\partial u} \\
    v_t(t, x, y) &= -C_v \frac{\partial F}{\partial v}
\end{align*}
\]

here \( C_u, C_v \) are positive constants.

One can see that the solutions \( u \) and \( v \) are gradually developed in the \( t \)-direction of decreasing of \( F \) until the minimum of \( F \) is attained. The steepest descent equations are derived explicitly for Equation 6.4 (see for example [5]):

\[
\begin{align*}
    u_t &= -C_u (2w_3(u - g) - w_1 \text{div}(2\nu^2 \nabla u)) = \\
    &\quad -2C_u (w_3(u - g) - w_1 \nu^2 \Delta u - w_1 2\nu \nabla v \cdot \nabla u), \\
    v_t &= -C_v (w_2 \nu |\nabla u|^2 - w_2 \left( \frac{v - 1}{2\epsilon} + 2\epsilon \Delta v \right)).
\end{align*}
\]

One can see that \( u \) solves essentially a diffusion-type equation \((u_t = k \cdot u_{xx})\), and \( v \) solves the Euler-type ODE \((v_t = -kv)\). Solving the Mumford-Shah functional by approximating functionals presents a considerable research study. In [5], [3], and [51] other forms of approximating functionals are presented, with corresponding numerical solutions.

Example of a numerical solution

The functional approach to image segmentation initiated a substantial research work, which in turn lead to many other applications, such as denoising and contour evaluation. An example of a numerical solution using approximation functionals ([5], [3], [51]) is shown in Figure 6.5. The iterative process in Equation 6.5 continues until \( u_t \) and \( v_t \) are close to zero, indicating that the local minimum is achieved. The segmentation information about the curved edges \( E \) is contained in \( v = 1 - \chi_E \), as shown in Figure 6.5.
6.1. SEGMENTATION METHODS IN IMAGE CODING

Such a procedure is not sufficient in the framework of our algorithm, since we seek a partition of $\Omega$ into subdomains $\{\Omega_i\}$ by curves $\{b_k\}$. While the variational solution produces two bivariate functions $u$ and $v$, where the function $v$ is essentially a set of pixels of different intensity, with the higher intensity corresponding to the higher probability of a pixel to belong to $E$. A large amount of postprocessing is still required, in order to determine the curved edges and turn these curves into closed contours.

Other works solve the Mumford-Shah functional with active contours and snakes (see for example [77]). This approach produces the edges $E$ as connected curves only for specific tasks, such as single object extraction or recognition, but they are not immediately applicable for the general segmentation problem of real-life images.

The relation between the Mumford-Shah functional and the $\tilde{K}$-functional

Take a partition $\Lambda$ of $[0,1]^2$, that is defined with a set of continuous curves $b_j : [0, 1] \to [0, 1]^2$, $1 \leq j \leq n_E(\Lambda)$, each of finite length, denoted by $\text{length}(b_j)$, which may intersect only at endpoints, such that the curves partition the image domain $\Omega = [0, 1]^2$ into open domains, $\Omega_k$, $1 \leq k \leq n_E(\Lambda)$. We denote by $\Lambda$ both the set of curves $\{b_j\}_{j=1}^{n_E}$ and the domains $\{\Omega_k\}_{k=1}^{n_E}$ (see example in Figure 7.1(c)).

Let $g \in L_2(\Omega)$ be such that $g \in H^1(\Omega_k)$, $1 \leq k \leq n_E(\Lambda)$, where $H^r(\Omega_k)$ is the Sobolev space. Then for any $f \in L_2(\Omega)$ define the energy gauge

$$E(f, \Lambda, g) = \sum_{k=1}^{n_E(\Lambda)} \|f - g\|_{L_2(\Omega_k)}^2 + \mu_1 \sum_{k=1}^{n_E(\Lambda)} |\nabla g|_{H^1(\Omega_k)}^2 + \mu_2 \sum_{j=1}^{n_E(\Lambda)} \text{length}(b_j), \quad (6.6)$$

which is the error of the approximation of $f$ by $g$, combined with two penalty terms, one measuring the piecewise smoothness of $g$ and the second, the total length of the segmentation curves of $\Lambda$. The Mumford-Shah functional seeks to minimize (6.6) over all partitions $\Lambda$ and piecewise smooth functions $g$, where the weights $\mu_1, \mu_2$ control the balance between approximation, smoothness, and the ‘amount’ of geometry one expects in the minimizing solution.

We now attach to each curve $b_j$ a weight $t_j$ and we say that the partition $\Lambda$ is in $\Lambda(t)$ if $\sum_{j=1}^{n_E(\Lambda)} t_j^{-1} < t^{-1}$ (in particularly it implies $n_E < t^{-1}$). Advancing towards the $\tilde{K}$-functional, a modified version of the Mumford-Shah functional can be expressed as the infimum of

$$\tilde{E}(f, \Lambda, g, t) = \sum_{k=1}^{n_E(\Lambda)} \|f - g\|_{L_2(\Omega_k)}^2 + \sum_{k=1}^{n_E(\Lambda)} |\nabla g|_{H^1(\Omega_k)}^2 + \sum_{j=1}^{n_E(\Lambda)} \text{length}(b_j) t_j^2 \|b_j\|_{L_1([0,1])}^2. \quad (6.7)$$

Comparing (6.7) with (6.6) we see that the main difference between $E$ and $\tilde{E}$ lies in the different notion of the lower-dimensional ‘structure’. The energy gauge $E$ uses only the length as a measure of lower-dimensional structure and does not distinguish for example between a straight line and a circle, both of the same length. Obviously, from an approximation theoretical point of view, the circle is more complex, its approximation involves a set of polylines, the corresponding value of $\tilde{E}$ is greater then the $E$ value, which illustrates that $\tilde{E}$ more precisely reflects the approximation complexity then $E$. As a result, for a practical approximation application the value $\tilde{E}$ is more reliable then the value $E$, and therefore approximation methods should seek minimization of $\tilde{E}$ (over admitted partitions).

We remind that the $\tilde{K}$-functional of order $r \in \mathbb{N}$ is defined as

$$\tilde{K}_r(f, t)_2 := \inf_{\Lambda \in \Lambda(t)} \{ \sum_{j=1}^{n_E(\Lambda)} \text{length}(b_j) K_2(b_j, t_j^2)_{\infty,1} + \sum_{k=1}^{n_E(\Lambda)} K_r(f, \Omega_k)_2^2 \}^{1/2}, \quad (6.8)$$
where
\[
K_2(b, t)_{\infty, 1} = \inf_{g \in C^2[0,1]} \| |b - g| \|_{\infty} + t \| |g''| \|_{L_1},
\]
measures the (weak-type) smoothness of the segmentation curves (where a univariate planar curve \( f = \{ \gamma_x, \gamma_y \} \) and the norm \( \| |f| \|_{\infty} := \max_t \sqrt{\| \gamma_x(t) \|^2 + \| \gamma_y(t) \|^2} \), and
\[
K_r(f, \Omega)_2 = \inf_{g \in H^r(\Omega)} \| f - g \|_{L_2(\Omega)} + \text{diam}(\Omega)^r \| g \|_{H^r(\Omega)},
\]
measures the (weak-type) smoothness of the surface pieces.

For a curve \( b : [0, 1] \rightarrow [0, 1]^2 \), the quantity \( K_2(b_j, \cdot)_{\infty, 1} \) is small if the curve is smooth, for example, if the \( L^1 \)-norm of its second derivative is small. It is also small if the curve is only piecewise smooth, with a ‘small’ number of pieces and it is identically zero for a line segment. The quantity \( K_r(f, \Omega)_2 \) ([25]) is small if \( f \) is smooth in \( \Omega \) and is identically zero if the function is a bivariate polynomial of degree \(< r \) in \( \Omega \). The reader should have in mind the case where for a sufficiently small \( t \), the curves \( b_j \), of a near-optimal partition \( \Lambda \in \Lambda(t) \), align alone the curve singularities of the function, and where over the segmentation domains, \( \Omega_k \), the function is smooth.

The novelty of the \( \tilde{K} \)-functional is the way it combines the smoothness gauges of the curve and surface to give a geometric generalization of the classical \( K \)-functional. More importantly, in [19] it is shown that, roughly speaking, the quantity \( \tilde{K}_r(f, n^{-1})_2 \) is equivalent to the approximation error \( \sigma_{n,r}(f)_2 \).

Therefore, since a segmentation which is a near-minimizer of (3.1) leads to a construction of a good piecewise polynomial approximation, we may conclude that designing an algorithm that tries to find such a segmentation is the key to a good performance of our coding algorithm. This task is still an on-going research project, however, for the purpose of low-bit rate coding, we found a simple and heuristic algorithm that works sufficiently well (see Section 7.1).
Chapter 7

The GPP Image Segmentation

7.1 GPP step-1: finding a partition

For the purpose of low-bit rate coding, we found that the following simple and heuristic algorithm for constructing a partition of an image works sufficiently well. First we apply the well known Gaussian zero-crossing segmentation algorithm, also known as the ‘Laplacian-Gaussian’ (LoG), with a relatively high width of the Gaussian kernel (Section 6.1.2 describes LoG in details). The idea here is to pick up the edge singularities of the image that are not noise or high-frequency texture, which indeed, the Gaussian kernel smoothes out sufficiently well. One of the main properties of the zero-crossing algorithm is its connectivity: the partition pixels form connected partition curves.

Observe, that in correlation with the $\tilde{K}$-functional the LoG algorithm of segmentation produces a partition with smooth curves that approximate wiggly edge singularities. In Figure 7.1(b), we see the partition produced at this initial step. Also, according to the theory of $\tilde{K}$-functional, the target compression rate dictates the parameter value of $\sigma$ in LoG: in the low bit-rate compression $\sigma$ is assigned with a relatively high value and the edge singularities are obtained as very smooth curves, whereas in the higher bit-rate the $\sigma$ value decreases and the obtained in the LoG curves in turn will correspond more precisely to the edges but will have a less smooth shape.

The next step guided by the minimization of the $\tilde{K}$-functional, is to prune the collection of segmentation curves. The goal is to identify and remove the non significant portions of the curves and to quantize and encode only the ‘significant’ segmentation pixels. To this end we subdivide the segmentation curves to short curve segments of fixed ‘length’ (we used 50 pixels in most of our experiments). We then prune away the curve segments where the norm of the image gradient is below some threshold, since these curve portions do not represent significant edges of the image. After that, we sort the remaining curve portions based on their smoothness, using the following formula

$$\text{len}(b) \|b''\|_1,$$

which is a simplification of the ‘curve smoothness’ term appearing in (3.1). Thus, we prune away the curve portions whose relative higher curvature impact the $\tilde{K}$-functional the most and equivalently, whose encoding requires a higher bit allocation budget. Notice that in such a procedure the straight line segments receive highest priorities and are not pruned away, which is in accordance with our practical goal, since encoding of a straight segment of any length takes equally small amount of information. In our practical application a small value is assigned to the straight line segment which helps to prevent the case of too many short straight line segments.
CHAPTER 7. THE GPP IMAGE SEGMENTATION

64

Figure 7.1: Pruning of a small width Gaussian Zero-crossing LoG of Cameraman image

In Figure 7.1(b) we see the initial segmentation of the Cameraman image and in Figure 7.1(c) the most ‘significant’ segmentation portions that survived the pruning process. As can be seen, after the pruning, we are left with a ‘quasi-segmentation’ with open contours and ‘cracks’.

Obviously, the amount of pruning relates to the target bit-rate we wish to achieve in our coding algorithm. The pruned segmentation can serve as a good basis for coding of geometric images, for example, graphic-art images that are piecewise constant with spline edge singularities. At this stage of the algorithm two main issues still need to be dealt with. First, we have not fully taken into account the minimization of ‘surface smoothness’ term in (3.1). Also, after pruning we are left with a ‘quasi-segmentation’ with open contours. These two problems are solved by the partitioning step, as explained in Section 7.3.2.

The operator $\text{Zerocross}(\text{LoG})$ produces a partition which has one-pixel width closed curves. However, this partition still needs to be improved in order to better meet the image compression objectives.

At the first stage, the partition curves are divided into short 50-pixel length portions, which are sorted and thresholded subject to the corresponding gradient value of image. Figure 7.2(b) illustrates the partition curves of $\text{Zerocross}(\text{LoG})$ together with the corresponding gradient values. We assign to every portion of the partition curve the average gradient value corresponding to its location. Some of these portions will correspond to strong edges (high values of the gradient), while others will have low gradient values (see Figure 7.2(c)). The curve portions, which correspond to the strongest gradient values, are most likely to belong to the optimal partition and should remain (Figure 7.2(d)).

7.2 GPP step-2: creating the bands

The main goal of bands is to deal with the distortion introduced by the 1-D approximation of the partition curves. The concept of bands also appears in [82] and [44], where the authors justify the bands by claiming that the edges in real-life images are "...most often ill-defined...".

Indeed, after we approximate the curve portions, computed in step-1 (see also Section 8.1), we allocate some width to the approximating segments, thereby creating
7.2. **GPP STEP-2: CREATING THE BANDS**

(a) Cameraman image  
(b) Gradient image  
(c) Edges and Gradient image  
(d) Pruned edges image

Figure 7.2: (a) Cameraman image, (b) its corresponding gradient image, (c) \textit{Zero-cross}(LoG) partition curves with shown at the background Gradient values, (d) \textit{Zero-cross}(LoG) partition curves after threshold with respect to Gradient.

(a)  
(b)

Figure 7.3: (a) Original curved edges (grey color) and its approximating polylines (black color), (b) wide bands (grey color) appended to polylines envelope original edge curves (black color)

edge bands. Denote the band’s width size (in pixels) as $\epsilon_B \in \mathbb{N}$ distance. We set $\epsilon_B$ using the two known parameters: $\epsilon_G$ - the Gaussian window width of the step in Section 7.1, and $\epsilon_C$ - the 1-D approximation error upper bound of the $L_1$-norm of the second derivative. If these two parameters are small, then we allocate small width to the bands and visa-versa. An example of such bands can be seen in Figures 7.3 and 9.3(a).

We then mark each pixel with $\epsilon_B$ distance from the approximating polylines pixel as a band pixel. The polyline in the GPP algorithm is constructed during the 1-D encoding-decoding. In our algorithm, the values at band pixels are never encoded, since they are in the vicinity of an edge singularity, whose exact location is unknown to the decoder. Instead, the decoder reconstructs the values of band pixels by linearly interpolating the two closest pixel values in the two adjacent subdomains of smoothness.
7.3 GPP algorithm step-3: convex driven binary tree partitioning

7.3.1 An overview

We recall that the GPP algorithm first creates a quasi-partition in step-1 (Section 7.1), and approximates the planar curve portions of the partition with polylines, which are further allocated with bands (Section 7.2). In considerations of Section 7.1 the second sun in Equation 3.1 is not considered.

The purpose of this step is to improve the partition with the bands obtained by the first two steps, by further partitioning of the subdomains into ‘more convex’ subdomains. In principal, this step complements the step in Section 7.1 that did not fully take into account the $\tilde{K}$-functional in Equation (3.1). Also, the pruning of the segmentation portions creates open contours that we should attempt to reconnect.

We propose here a remedy to these two shortcomings using a method that is very cost-effective from the viewpoint of rate distortion optimization. Section 3.4 shows that the quality of polynomial approximation over multivariate domains is determined by the smoothness of the function and by the domain’s geometry. It is difficult to approximate a non-smooth function by a low-order polynomials, but it is also difficult to approximate a smooth function over a highly non-convex domain.

One possible solution to the partitioning of a highly nonconvex domain into a set of ‘better’ domains is by triangulating it, subsequently approximating a given function separately in each triangle. Triangulation of the entire domain theoretically resolves that problem, since the total amount of triangles in a triangulation is proved to be bounded ([19]), and in each triangle the smoothness of the function determines uniquely the error of the 2-D approximation. The theoretical conclusion is that the error of approximation with piecewise polynomials over a partition with nonconvex domains can be estimated with the classical smoothness measure (such as $K$-functional) in the triangulated partition.

However, triangulating of the partition produces excessively large amount of triangles. In practice, reasonable approximation can be achieved at nonconvex domains also, and there is no point in forcing a partition into strictly convex subdomains. Section 3.4 illustrates the idea that ‘convex is good’ when the approximation tool at hand is piecewise polynomials. We employ this idea in our algorithm of convex-driven binary tree partitioning of the subdomains of smoothness.

Our algorithm of convex-driven binary tree partitioning recursively subdivides the partition subdomains we get from the step in Section 7.2, until these subdomains are partitioned into ‘more convex’ subdomains with a satisfactory polynomial approximation. Since the convex-driven binary partition needs to be encoded, it is advantageous that the partitioning algorithm is purely geometric. Then, once the decoder receives a ‘subdivide’ bit from the encoder, for a given subdomain, it uses the reconstructed partition of the approximating polylines, and applies a pure-geometric subdivision algorithm without any further information from the encoder. This is closely related to the dyadic square partitioning, that is very common in many applications. For example, in [71] once the decoder is ‘informed’ by the encoder that a dyadic square must be subdivided, it immediately knows that it must be subdivided into four smaller dyadic squares. The extra power of nonlinear approximation gained, by arbitrary partitioning of a domain $\Omega$ by planar curves (example of such curves for a real-life bivariate image function can be seen in Figure 7.4), imposes a new obstruction for approximation that does not exist in 1-D. Namely, the singularity becomes a two-fold quantity. Smoothness of a function in nonconvex domain doesn’t guarantee fast approximation. Section 3.4.4 discusses the two kinds of (function and domain) singularities.
7.3. GPP STEP-3: CONVEX DRIVEN BINARY TREE PARTITIONING

We describe further in Section 7.3.2 a new geometry-driven algorithm confronting the domain singularity, that has been developed specially targeting at image compression. The algorithm constructs a partition of a given domain at a very low cost, proportional to the cost of the simple dyadic splits. The new subdomains are constructed recursively, gradually discarding the geometric singularities of the domain (as defined in Section 3.4.4) and improving the overall quality of the approximant.

7.3.2 Geometry-driven binary tree partition

We presently describe the geometry-driven binary partition algorithm for approximating a function in complicated domains. We demonstrate the application of the algorithm on a planar domain from the segmentation of the cameraman image, as shown in Figure 7.4(c), and on a domain with one domain singularity, as shown in Figure 10.3(a), and Figure 10.3(b).

Our algorithm employs the measure of domain singularity introduced in Section 3.4.4, and produces geometry-driven partition of a complicated domain, which targets at efficient piecewise polynomial approximation with low-budget encoding cost. The algorithm constructs recursively a binary space partition (BSP) tree, improving gradually the corresponding piecewise polynomial approximation and discarding the domain singularities. The decisions taken during the performance of the algorithm are based on both the quality of the approximation and the measure of
Description of the Algorithm

The algorithm constructs the binary tree partition recursively. The root of the tree is the initial domain $\Omega$, and its nodes are subdomains of $\Omega$. The leaves of the tree are subdomains where the polynomial approximation is good enough. For a subdomain $\hat{\Omega} \subset \Omega$ at a node of the binary tree, first a least-squares polynomial approximation to the given function is constructed. If the approximation error is below the prescribed allowed error, then the node becomes a leaf. If not, then the domain $\hat{\Omega}$ is partitioned.

The partitioning step: the algorithm constructs the components $\{\hat{C}_i\}$ of the complement of $\hat{\Omega}$ in its convex hull, and selects $\hat{C}_i$ with the largest $\mu(\hat{C}_i)$ of $\hat{\Omega}$. Then the algorithm partitions $\hat{\Omega}$ with a ray, which is a straight line perpendicular to $\partial \hat{C}_i$, cast from the point $P \in \partial \hat{C}_i \cap \partial \hat{\Omega}$, chosen such that $\rho(P, P_1^i) = \rho(P, P_2^i)$, where $P_1^i, P_2^i$ are the pair of points of the singularity component $\hat{C}_i$, as defined in Section 3.4.4. We favor the partition along a straight line since a straight line does not create new non-convexities and is coded with a small budget. By this partition we discard the worst domain singularity (with the largest distance defect ratio).

It may happen that $\hat{C}_i$ lies entirely inside $\hat{\Omega}$. Then two rays in two directions are needed in order to partition $\hat{\Omega}$ in a way that eliminates the singularity of $\hat{C}_i$. Those two rays are perpendicular to $\partial \hat{C}_i \cap \partial \hat{\Omega}$ at two points $A, B \in \partial \hat{C}_i \cap \partial \hat{\Omega}$, satisfying $\rho(A, B)_{\hat{\Omega}} \geq \rho(P, Q)_{\hat{\Omega}}$, for any pair of points $P, Q \in \partial \hat{C}_i \cap \partial \hat{\Omega}$.

Furthermore, let us consider the case of a 'cracked’ domain, i.e., a domain generated by the removal of a portion of a boundary. The crack is a curve segment with one end point at the boundary and one internal end point. In this case the connected component corresponding to the crack has a very large measure of geometric singularity very large. Therefore, our algorithm will try to subdivide the domain by a ray casted from the internal end point of the 'crack’. Thus, the algorithm tends to split a 'cracked’ subdomain that was created by the pruning of an edge portion into two subdomains, and to eliminate the ‘crack’

A Modification of the Partitioning Step

Here is a small modification of the partitioning step of our algorithm that we find to be rather efficient. We select a small number $(2^k - 1)$ with $1 < k \leq 3$ of components $\{C_i\}$, having the largest $\{\mu(C_i)\}$, prompt the partitioning procedure for each of the selected components, and compute the resulting piecewise polynomial approximation. For the actual partitioning step, we select the component corresponding to the maximal reduction in the error of approximation. Thus, the algorithm performs dyadic partitions, based both on the measure of geometric singularity and on the quality of the approximation. This modification is encoded with $k$ extra bits.

7.3.3 Three examples

In this section we demonstrate the performance of the algorithm on three examples. The first example shows the first steps in the performance of the algorithm on the domain $\Omega$ in Figure 3.5 (a). Figure 3.5 (b) illustrates the first partition of the domain, generating two subdomains. Next we consider the generated subdomain $\hat{\Omega}$ shown in Figure 3.6 (a), its convex hull $H$, shown in Figure 3.6 (b), and the components $\{C_i\}$ of $H \setminus \hat{\Omega}$, shown in Figure 3.6 (c). The algorithm further partitions $\hat{\Omega}$, in order to reduce its measure of singularity and to improve the piecewise polynomial approximation (Figure 7.7).
Figure 7.6: A subset of $H\setminus\hat{\Omega}$ lying inside $\hat{\Omega}$, with two casted rays (in black)

Figure 7.7: (a) One component of $H\setminus\hat{\Omega}$ lying inside $\hat{\Omega}$, (b) the corresponding splitting of $\hat{\Omega}$ by one ray

The second example demonstrates the case when a component $C_i$ lies entirely inside the subdomain, and two rays have to be casted, as it is illustrated in Figure 7.6. Finally, Figure 10.3(a), 10.3(b) demonstrate a partition of a domain with one singularity, and the corresponding piecewise polynomial approximation.
Chapter 8

Approximation and encoding in GPP

In the GPP image coding algorithm the final representation of the compressed images consists of 1-D and 2-D components. The 1-D component (the geometry) is the image partition, which includes the representation of the partition curves and the binary partition tree into non-singular subdomains of smoothness (described in Section 7.3.2). The 2-D component is a list of the approximating bivariate polynomials of the image values in the subdomains of the final partition (the construction of these polynomials is described in Section 8.2). This chapter describes the methods of approximation and of statistical encoding we developed in order to represent efficiently the 1-D and 2-D components in the GPP algorithm.

8.1 1-D: Approximation and encoding of the partition curves

In our geometry-based image coding method, the approximation of the 1-D geometry (the image partition curves) and its efficient encoding is a crucial step, as it takes the largest part of the compression budget. Hence, even a small improvement in the efficiency of the 1-D approximation and its encoding causes a substantial improvement in the final compression results. This fact can be seen as follows: for a given image partition with \( n \) curves (in fact, \( n \) pixel chains), which has total length \( P \) (number of pixels), it can be reasonably assumed that \( P \gg n \). Now, the number of the subdomains of smoothness is of the order of \( n \), and for each subdomain there is one polynomial with \( k \) coefficients (here \( k \) is small). Thus, the total budget of the image representation is of the order of \( k \cdot n + P \), and depends by large on the efficient encoding of the \( P \) pixels of the partition curves.

8.1.1 Lossless encoding using arithmetic coding and context modelling

Chain coding, where each contour is represented as a starting point and a sequence of travel directions, was suggested in [46] as a method for encoding of curves given as pixel chains. K. P. Horn later proposed a method for lossy contour coding ([22]), where the 8-connected contour map was replaced with the 4-connected (the general algorithm of chain coding is described in Section 8.1.2). After that, the statistic coding is employed to encode the sequence of travel directions. Employing this approach, a bit-rate of 1.3 bits per contour pixel is reported in [9]. In our work we demonstrate an algorithm that attains a final bit-rate of 0.3-0.1 bits/pixel.
It is important to notice that approximation aimed at minimizing the normal-offset error of planar 1-D curves is more important, than the minimization of the $L_\infty$ or the $L_2$ error. In a recent work of Daubechies et. al. [13] an approximation algorithm for the minimization of the normal-offset error of planar continuous curves is described. This method is less efficient when employed for the approximation of discrete curves (pixel chains), which are often far from being infinitesimally continuous. Therefore, we developed a new algorithm that aims at approximation and encoding of discrete curves (pixel chains), with respect to the normal-offset error.

8.1.2 A new method of 1-D (lossy) encoding based on arithmetic coding and context modeling

Here we present a method that combines the lossless chain encoding and the lossy pixel-based curve approximation, which is more stable than continuous curve approximation. Chain coding interpolates a given pixel chain. However, with bands (see Section 5.3 and 7.2) the lossy approximation can be employed and tolerated (rather than the exact interpolation), with a significant reduction in the encoding cost.

Once we have computed the pruned segmentation in step-1, we need to efficiently encode it. Recall that the second generation methods [61] did not fully succeed in solving this problem and therefore were not able to compete with the more conventional wavelet-based image compression algorithms.

Algorithm of 1-D encoding

Step-1: downsample segmentation We approximate the pruned segmentation by downsampling it. We fix a parameter $\epsilon_C \in \mathbb{N}$ as the maximum error of the approximation in pixel-distance. Choosing a small $\epsilon_C$ gives a very good approximation to the segmentation curves, but leads to a higher bit budget. We downsample the pruned segmentation at the rate $\epsilon_C$ and generate its lower resolution, as shown at Figure 8.1 and 9.2(a). The procedure of downsampling of a pixel chain is quantization of the pixel integer indices: $i_{\text{downsampled}} := \lfloor \frac{i_{\text{upsampled}}}{\epsilon_C} + 0.5 \rfloor$.

Step-2: reconstruction by upsampling From the downsampled segmentation we reconstruct ‘the jaggy’ segmentation (see Figure 9.2(b) and 8.2(a)) with a maximal error of $0.5 \cdot \epsilon_C$ by upsampling it back to the original resolution of the image $i_{\text{upsampled}} := \lfloor i_{\text{downsampled}} \cdot \epsilon_C + 0.5 \rfloor$.

The downsampled segmentation pixel locations are encoded using a high order chain coding algorithm [39],[46] where each downsampled segmentation curve is represented as a starting point and a sequence of travel directions. In our algorithm, we encode the location of the next pixel in the curve, by encoding one of eight possible directions from the previous point (see [22] for a simpler lossy version). We use arithmetic coding combined with context modeling, where each context is determined by the previous four pixels in the chain, or equivalently, previous three travel directions. In some sense, this operation is equivalent to predicting the next point by extrapolating a cubic curve segment that interpolates the previous four points.

Step-3: smoothing by spline approximation Finally, the obtained upsampled pixel chains are smoothed using cubic spline least squares approximation (see Figure 8.2(b) and 9.2(c)).

1-D encoding experimental results

The lossy curve encoding of [9] obtains on average a bit-rate of 1.3 bit/(curve point) with a maximal error of 1 pixel, whereas we would like to have the possibility to go beyond that, i.e. to lower the bit-rate at the cost of a higher error (typically 2-4
Figure 8.1: (a) Pruned curve portions of Lena, (b) the same curve portions after six-fold down-sampling.

Our multiresolution approach employs the 8-connected contour encoding at different scales and obtains, on average for different images and compression rates, a bit-rate of 0.1-0.4 bit/(curve point), which is a significant improvement of the results reported in [45]. In practice, we found that the downsampling factor of the original segmentation is selected a proportional to the target bit-rate of the encoded downsampled segmentation.

Here we present numerical results for the example of approximation and encoding the partition of Lena test image:

the number of pixel-chains in the pruned partition = 33

the total number of pixels in the pixel chains = 7164

the total number of pixels in the down-sampled partition = 1503

the encoding of the 33 initial pixels takes 79 bytes

the encoding of the 1503 directions takes 306 Bytes

→ the efficiency of the encoding with down-sampling is 18.7539 (0.43 bits per pixel)

In the above example the curve-point bit-rate $\frac{79 + 306}{7164} = 0.43$, is slightly above the average interval of $0.1 - 0.4$ bpp, which we obtained for several different images. The above example of the rate 0.43 was produced for the real-life Lena image, which has many extremely non-smooth edges.

It is important to note that the segmentation, down-sampling and chain-coding combination detailed above can be considered a variant of adaptive polygonal curve approximation. For example, if a curve segment is almost a line, a good adaptive approximation algorithm will allocate only one polygonal segment, whereas the linear prediction in the chain-coding algorithm will be so successful that the arithmetic encoder will spend very few bits. On the other hand, pixel-based encoding techniques are robust in comparison with methods developed for the approximation of continuous functions.
Figure 8.2: (a) Up-sampling curves (black color) with original partition curves (grey color), (b) smoothed up-sampling curves (black color) employing linear prediction approximate better the original curve portions (grey color)

Figure 8.3: Bands appended to smooth up-sampled curves (grey color) enveloping original partition curves (black color)
8.2 2-D: Approximation and encoding in subdomains of smoothness

At the segmentation stage, the GPP image coding algorithm finds the partition, which is then 1-D approximated and encoded. Once the partition into the disjoint subdomains of smoothness is found, the GPP algorithm needs to compute the approximating polynomials in the subdomains. In each created subdomain \( \Omega \) (as described in Section 3.4) we compute a low-order polynomial approximation \( P_\Omega \) of the target function using the least-squares technique. The degree of the polynomials is fixed for all domains. To ensure stability of the quantization of the polynomial’s coefficients, we first compute a representation of \( P_\Omega \) in an orthonormal basis of \( \Pi_{r-1} \cap L^2(\Omega) \).

For example, in the case of bivariate linear polynomials, this simply amounts to transforming the standard polynomial basis \{1, x, y\} using a Graham-Schmidt process to an orthonormal basis \{\( u_1, u_2, u_3 \)\} of \( \Pi_1 \cap L^2(\Omega) \). This gives a representation \( P_\Omega = \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 \), with \( \alpha_i = < P_\Omega, u_i >_{L^2(\Omega)}, i = 1, 2, 3 \).

Since at the time of decoding the decoder has full knowledge of the geometry of \( \Omega \), the Graham-Schmidt process is carried out by the decoder in exactly the same way as the encoder. The polynomial \( P_\Omega \) is subsequently quantized and encoded in this stable basis. After all of the coefficients \( \{\alpha_i\} \) for all the subdomains are computed and quantized, they are encoded separately for each \( 1 \leq i \leq 3 \) using the well known Variable Length Coding (VLC) technique ([12], [81], [48]), combined with arithmetic encoding. At the decoder, the quantized coefficients of the polynomials are decoded and the polynomials are reconstructed. We note that this algorithm compresses, on average, the representation of each bivariate linear polynomial (determined by three real numbers) to 1.8 bytes.

8.2.1 Scalar quantization

For a given \( \epsilon \), we are looking for an efficient representation of the real value coefficients of the approximating polynomials in the subdomains of the partition from Section 3.4, that takes least amount of computer space for an \( \epsilon \)-bounded distortion.

Such procedure is known as the \( \epsilon \)-quantization. The scalar quantization \( F(x) \) maps an interval of the real line into an element from that interval. For instance, consider \( N \) disjoint intervals \([a_n, a_{n+1})\), \( n = 1, \ldots, N \), where \( \{a_n\} \) is an increasing sequence, and select any value \( \tilde{a}_n \) within the \( n \)th interval. Then, the scalar quantizer \( F \) is defined as a mapping of \( \bigcup_{n=1}^{N}[a_n, a_{n+1}) \) to the index \( n \in \{1, \ldots, N\} \), and the inverse dequantizer \( F^{-1} \) maps the index \( n \) to some value \( \tilde{a}_n \) in \([a_n, a_{n+1})\). Namely, the \( \epsilon \)-quantization is the \( F^{-1}F \) mapping:

\[
F(a) = n, \quad F^{-1}(n) = \tilde{a}_n \rightarrow F^{-1}F(a) = \tilde{a}_n,
\]

here \( a \in [a_n, a_{n+1}) \). The index \( n \in \mathbb{Z} \) provides a shorter representation of the real value \( a \), but introduces a distortion \(|\tilde{a} - a|\). Eventually, the quantization procedure \( FF^{-1} \) reduces the computer space needed to represent a set of real numbers. We assume a smooth distribution of the real values \( \{a_n\} \), (equivalently, we can assume that there is no favorite distribution). It is shown in [35] that the optimal scalar quantization is uniform. In this case, we divide uniformly the real line \( \mathbb{R} \) into equal length intervals \([a_n, a_{n+1})\), \( |a_{n+1} - a_n| = \epsilon \) for all \( n \), and define the uniform quantizer \( F \) and dequantizer \( F^{-1} \) as follows:

\[
n = F(a) = \text{sgn}(a) \left\lfloor \frac{|a|}{\epsilon} \right\rfloor,
\]

\[
\tilde{a}_n = F^{-1}(n) = \text{sgn}(n)(|n| + \frac{1}{2})\epsilon
\]
here \( \text{sgn}(a) \in \{-1, 0, 1\} \). Then, the maximum error \(|\tilde{a} - a|\) in the 0-containing subinterval (deadzone) is \(2\epsilon\), and beyond the deadzone the maximum error is \( \epsilon \).

After the real value coefficients of the polynomial approximation are computed, they undergo the \( \epsilon \)-quantization, mapping them to the \( \{\tilde{a}_k\} \) set by computing a set of the integer indices \( \{n_k\} \). These indices are statistically encoded, as we assume a statistical correlation in the approximating polynomials in adjacent subdomains. Therefore, additional data compression can be achieved by employing the context arithmetic encoding of the set \( \{n_k\} \).
Chapter 9

Experimental results

9.1 Demonstration of the GPP algorithm

We have chosen Lena image to illustrate the steps of the GPP algorithm in Figures 9.1, 9.2, 9.3, and 9.4.

9.2 GPP algorithm vs. JPEG2000

We applied our GPP algorithm to known test images at low-bit rates and compared the performance of GPP to the best wavelet image coding algorithm known to us: Taubman’s Kakadu implementation [74] of the JPEG2000 standard [75]. It should be noted that not all JPEG2000 compression algorithms provide the same coding performance, and therefore one should take care and reference the correct JPEG2000 implementation.

The GPP algorithm efficiently encodes geometric images, both artificial and real-life. We see that at low bit-rate both the visual quality and rate-distortion performance of the GPP algorithm are substantially superior to JPEG2000 from Figure 9.7. In fact, at low bit-rates, the GPP algorithm encodes images with relatively good visual quality and preserves the key geometric features, as can be seen also in Figure 9.5. One can observe in Figure 9.7 (k)-(l) and (o)-(p), where J2K and GPP encoded with the same bit-rate and PSNR, that GPP obtains perhaps better visual quality.

It is not surprising that the performance of the GPP coding algorithm depends on the amount of ‘geometric structure’ found in the image. We see on the Chess and Cameraman images, that have relatively more ‘geometric structure’ than other real-life images, that the GPP algorithm outperforms JPEG2000 by 5 dB at 0.08 bpp for the chess image, and 1.5 dB at 0.05 bpp for the cameraman image. However, on the Lena image that contains texture elements and a relatively small amount of geometry, the rate-distortion performance of GPP is similar to JPEG2000, with perhaps better visual quality of the GPP algorithm.

R-D comparison experiments are summarized in Table 9.1 and Figure 9.6, showing that GPP consistently outperforms JPEG2000 in the very low-bit rate. However, the GPP algorithm clearly underperforms JPEG2000 in the high bit-rates. In high bit-rates no new edges are added to the segmentation and the encoding inside the subdomains becomes most crucial. Thus, sophisticated approximation and encoding in the subdomains is required in the higher bit-rates, such as approximation by high degree polynomials or by wavelets.
CHAPTER 9. EXPERIMENTAL RESULTS

Figure 9.1: (a) Pruning of the initial segmentation of Lena, notice the sorting out of the jagged curve portions

Figure 9.2: (a) 1-D approximation employs downscaling before lossless encoding, (b) the corresponding ‘jaggy’ reconstruction (upscaling) after decoding, (c) smoothing prediction is subsequently applied to produce smooth curves out of the ‘jaggy’ reconstruction

Figure 9.3: (a) Adding bands to the decoded lossy segmentation in Figure 9.2(c), (b) the initial subdomains of smoothness
Figure 9.4: (a) The final partition obtained with the convex-based dyadic partitioning, (b) its corresponding polynomial approximation, (c) the linear interpolation at the bands

Figure 9.5: Comparison of artifacts in one region of the Cameraman image for GPP and JPEG2000

Figure 9.6: R-D performance comparison of the GPP algorithm and JPEG2000 for the Chess, Cameraman and Lena images
CHAPTER 9. EXPERIMENTAL RESULTS

(a) The artificial Eggmean [30]  (b) J2K coding at 0.01 bpp, PSNR=24.4dB  
(c) GPP coding at 0.01 bpp, PSNR=29.7dB  

d) The real-life Chess  
(e) J2K 0.08 bpp, 15 dB  (f) GPP 0.08 bpp, 20.7dB  (g) J2K 0.15 bpp, 22 dB  (h) GPP 0.12 bpp, 22 dB  
(i) J2K 0.05 bpp, 19.94 dB  (j) GPP 0.046 bpp, 21.5 dB  (k) J2K 0.07 bpp, 21.6 dB  (l) GPP 0.07 bpp, 21.6 dB  
(m) J2K 0.015 bpp, 23.65 dB  (n) GPP 0.015 bpp, 24 dB  (o) J2K 0.030 bpp, 26 dB  (p) GPP 0.030 bpp, 26.1 dB

Figure 9.7: Artificial and real-life images coding examples with J2K (JPEG2000) versus GPP. Notice how GPP improves the coding of geometry as the bit-rate increases.
9.3. SELECTING GPP PARAMETERS

9.2.1 Budget allocation

In Figure 9.8 we show the relative distribution of the total budget of encoding in the three test images Eggmean, Cameraman, and Lena. The three main components of the encoding budget in our GPP algorithm are: geometry (the pixel chains), polynomials, and the binary tree of the convex driven binary space partitioning algorithm. We see from Figure 9.8 that the more geometry the image has, the higher is the budget allocated for the geometry. In all three images the budget allocated for the binary tree is relatively small.

9.3 Selecting GPP parameters

There is a set of parameters the GPP algorithm operates with in order to produce an approximate of a given image with geometric polynomials and encode the obtained representation. The total number of the parameters and their precise role in our algorithm is a rather involved implementation-specific issue. In our application, the two main subsets of the parameters are the parameter set that governs the 1-D edge

<table>
<thead>
<tr>
<th></th>
<th>GPP</th>
<th>JPEG2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit-rate(bpp)</td>
<td>PSNR(dB)</td>
</tr>
<tr>
<td>Chess</td>
<td>0.056</td>
<td>20.1</td>
</tr>
<tr>
<td></td>
<td>0.12</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>22.6</td>
</tr>
<tr>
<td>Cameraman</td>
<td>0.044</td>
<td>21.2</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>21.6</td>
</tr>
<tr>
<td></td>
<td>0.09</td>
<td>21.8</td>
</tr>
<tr>
<td>Lena</td>
<td>0.011</td>
<td>22.6</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>25.2</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>26.1</td>
</tr>
</tbody>
</table>
detection and encoding, and the set handling the 2-D approximation and encoding.

For a given total encoding budget, the GPP algorithm decides first on the 1-D encoding budget and chooses the most significant edges encoded within this budget. For geometric artificial images we decided on the proportion of the 1-D budget to be about ninety percent of the total budget, and for the real-life images with few geometrical structure the proportion of 1-D budget to be about a half. The amount of encoded edges is governed by the $\sigma$ parameter in $LoG$, which controls the amount and the smoothness of obtained edges, and another edge pruning parameter, that thresholds subject to the edge smoothness. The rule of thumb in finding the budget and the amount of 1-D geometry to encode is the following. In geometrical artificial images all the edges have almost equivalent significance in terms of the two mentioned parameters, while the edges in the real-life images are clearly sorted subject to the different values of the above two parameters. We try to retain all the edges in the geometrical images, and in the real-life images the 1-D budget is determined by a gradual dilation strategy, by pruning away the edges until the 1-D encoding budget reaches half of the total budget. In our implementation there are more 1-D parameters, which govern the width of bands and the approximation of the edges, which we setup according to the target bit-rate.

In the first step the image partition is obtained. After that, having decided on the 1-D partition and its 1-D budget, the binary tree partitioning is performed, which iteratively subdivides the domains and constructs the approximating polynomials, until the rest of the target budget is reached. The GPP parameters include the parameters of quantization of the polynomial coefficients and heuristic parameters that govern creating the binary partitioning tree.
Part IV

Beyond GPP
Chapter 10

Polynomial approximation over planar domains using dimension elevation

Approximation of a smooth function in 2-D \( f : D \rightarrow \mathbb{R} \), where \( D \subset [0, 1]^2 \) imposes a new problem of domain singularity that does not exist in case of 1-D (univariate approximation). Similar to function singularity, we think of domain singularity as a property that reduces ”approximability” of a function in the domain. Such domain singularity we define as sharp, thin, long non-convex regions (see Section 3.4.4). Partitioning those ”bad” (far from convex) domains into ”better” subdomains and defining in each of them an approximating polynomial is a well-known idea, and we demonstrated our approach of improving nonconvex domains by convexity-driven partitioning in Section 7.3.2. We now introduce a novel approach to 2-D approximation in complicated domains, which is not based on partitioning of the domain. This algorithm challenges the problem of finding continuous approximants which can be encoded with a small budget. This algorithm is not used in our GPP method, but might be useful for high bit-rates, where the introduction of discontinuities by the convexity driven, binary space partition cannot be tolerated.

10.1 Algorithm outline

The basic idea

We explain the main idea on a domain \( \Omega \) with one domain singularity component \( C \), and later extend it to the case of multiple components. Roughly speaking, we suggest to raise up one side of the subset \( C \) along the additional dimension axis, in order to increase its Euclidian distance from the other side and by that to improve the distance defect ratio. This is demonstrated in Figure 10.1(b).

Once the domain \( \Omega \) is continuously mapped to the 3-D domain \( \hat{\Omega} = \Phi(\Omega) \) such that the distance defect ratio of the singularity component (as defined in Section 3.4.4) turns from arbitrary large to small bounded by 2, the given function \( f \) is mapped to the tri-variate function \( f(\Phi^{-1}(\cdot)) \) defined on \( \hat{\Omega} \), which is approximated by a tri-variate polynomial \( P \). The approximating polynomial \( P \), is computed in terms of orthonormal tri-variate polynomials relative to \( \hat{\Omega} \). The approximant of \( f \) in \( \Omega \) is \( P \circ \Phi \).
CHAPTER 10. NEW ALGORITHM OF DIMENSION ELEVATION

Figure 10.1: (a) Domain with one domain singularity, (b) the domain in 3-D resulting from the continuous mapping of the planar domain.

The dimension-elevation mapping

For a planar domain Ω with one domain singularity, the algorithm employs a continuous one-to-one mapping Φ : Ω → ̃Ω, Ω ⊂ R², ̃Ω ⊂ R³, such that for any two points in Φ(Ω) the distance inside the domain is of the same magnitude as the Euclidian distance.

The continuous mapping we use is so designed to eliminate the singularity of the singularity pair \{P_1, P_2\}. The mapping Φ(P), for \( P = (P_x, P_y) \in \Omega \) is

\[
\Phi(P) = (P_x, P_y, h(P)),
\]

with \( h(P) = \rho(P, P_C)_{\Omega} \), where \( P_C \) is one of the pair of points of the domain singularity corresponding to the unique singularity component \( C = H \setminus \Omega \). Note that the mapping is continuous and one-to-one. An algorithm for the computation of \( \rho(P_1, P_2)_{\Omega} \), based on the idea of a view frustum ([15]) is presented later in this Section.

For a domain with multiple domain singularities, we employ \( N \) additional di-
dimensions to resolve $N$ domain singularities $\{\mathcal{C}_i, i = 1, \ldots, N\}$. Each $\mathcal{C}_i$ contributes a pair of points $P_1^i, P_2^i \in \partial \mathcal{C}_i$ where the distance defect ratio has a maximum. Then, for each domain singularity $\mathcal{C}_i$ we construct independently a mapping $\Phi_i(P) = (x_i, y_i, h_i(P))$, $i = 1, \ldots, N$, where in the definition of $\Phi_i$ we ignore the other components $\mathcal{C}_j, j \neq i$, and regard $\mathcal{C}_i$ as a unique domain singularity. The resulting mapping $\Phi : \Omega \to \tilde{\Omega}, \Omega \subset \mathbb{R}^2, \tilde{\Omega} \subset \mathbb{R}^{2+N}$, is defined as

$$\Phi(P) = \{P_x, P_y, h_1(P), \ldots, h_N(P)\}.$$

After the construction of the mapping $\Phi$, we compute the best $(N+2)$-variate polynomial approximation to $f \circ \Phi^{-1}$, in the $L_2(\Phi(\Omega))$-norm. In case of a linear polynomial approximation, the approximating polynomial has $N$ more coefficients than a linear bivariate polynomial. For coding purposes only these coefficients have to be encoded, since the mapping $\Phi$ is determined by the geometry of $\Omega$, which is known to the decoder. Note that by this construction the approximant is continuous, but is not a polynomial.

**Algorithm for the construction of $\rho(P_1, P_2)_\Omega$**

Here is an algorithm for constructing the function $h(P) = \rho(P, P_C)$ for a given planar domain $\Omega$, where the point $P_C \in \Omega$ is a point of singularity at the boundary $\partial \mathcal{C} \cap \partial \Omega$. Intuitively, we employ the idea of a *view frustum* ([15]) centered at the source point $P_C$, and we regard the domain singularity subset $\mathcal{C}$ as a *view obstacle*. We start mapping the domain $\Omega$ by a view frustum from a point $P_C$ to produce the first subset $V_1 \subset \Omega$. We assign to every point $P \in V_1$ a value $h(P) = \|P - P_C\|$. The next view frustum we locate at the nearest point to $P_C$ from $\partial \mathcal{C} \cap V_1$:

$$P_F \in V_1 \bigcap \partial \mathcal{C} \text{ s.t. } \|P_F - P_C\| = \min_{P \in \partial \mathcal{C} \cap V_1} \|P - P_C\|.$$

Notice, that the internal path from a point in $\Omega \setminus V_1$ to $P_C$ goes through the point $P_F$.

Let the new view frustum centered at $P_F$ cover a set $V_2 \subset \Omega \setminus V_1$. We assign to a point $P \in V_2$ the value $\|P - P_F\| + h(P_F)$. We continue further this sequence of view frustum mappings until the whole domain $\Omega$ is covered completely. In practical implementation of this algorithm, we approximate a smooth boundary $\partial \mathcal{C}$ with a $k$-segment polyline, which guarantees that the final mapping $h(P)$ can be constructed using $k$ view frustum mappings corresponding to the $k$-segment polyline.

**3-D Mapping of a domain with multiple geometric singularities**

Our 3-D mapping algorithm can be immediately extended to the case of multiple domain singularities by employing $N$ additional dimensions to resolve $N$ domain singularities $\mathcal{C}_i$. Each $\mathcal{C}_i$ contributes a pair of points $P_1, P_2 \in \partial \mathcal{C}_i$ where the domain defect ratio $\rho_i = \rho_i$ has a local maximum. We have constructed previously a 3-D mapping $\Phi$ that discards one domain singularity. Then, for each domain singularity $\mathcal{C}_i$ we construct independently a mapping $\Phi_i$, $i = 1, \ldots, N$. When we select a domain singularity subset $\mathcal{C}_i$ we ignore the other subsets $\mathcal{C}_j, j \neq i$ by adding them back to the original planar domain $D$ and constructing the corresponding $\Phi_i$ as explained in Section 10.1 for the new domain $D'_i = D \cup \mathcal{C}_j, j \neq i$. Then, the domain $D'_i$ has one domain singularity, and we employ the mapping $\Phi_i$ to discard it.

The resulting mapping $\Phi : D \to D', D' \subset \mathbb{R}^2$, and $\tilde{D} \subset \mathbb{R}^{2+N}$ is defined as the $N + 2$-components: $\Phi(x, y) = \{x, y, \Phi_1(x, y), \ldots, \Phi_N(x, y)\}$. Finally we employ the regular $N + 2$-dimensional approximation with orthonormal polynomials in the new $N + 2$-dimensional domain, where in case of linear polynomial approximation the
new approximating polynomial will include $N$ additional terms. Every additional component introduces an additional dimension in the approximating polynomial, by that increasing its complexity.

There are domains for which it is possible to construct the mapping that resolves $N$ singularities yet employs one additional dimension, rather than $N$. If the domain $D$ has a partition into subdomains $D_i$, such that each $D_i$ contains only one convex domain singularity $C_i$, we can treat $C_i$ independently one from another, and construct a single piecewise constant mapping $\Phi$ as a collection of mappings $\Phi_i : D_i \rightarrow \tilde{D}_i$. However, the general case of arbitrary singularities $C_i$ doesn’t necessarily allows to localize each domain singularities within a separate convex domain. In Figure 10.2 we demonstrate the operation of our algorithm in case of three domain singularities. This example indicates that the approximant generated by the dimension-elevation algorithm is superior to the bivariate polynomial approximation, in particular along the boundaries of the domain singularities. Similar conclusions can be drawn from Figure 10.4.

Figure 10.3 displays an example, showing that the approximant generated by the dimension-elevation algorithm is better than the approximant generated by the geometry-driven binary partition algorithm, and that it has a better visual quality, mainly by avoiding the introduction of the artificial discontinuities along the partition lines.
Figure 10.4: (a) A domain with one singularity, (b) approximation using 3-D mapping with 3-D linear polynomial produces PSNR=24.6dB, (c) approximation using one bivariate linear polynomial produces PSNR=13dB. Notice how well the 3-D mapping based algorithm captures the function along its domain singularity in (b), in comparison with the bivariate polynomial approximation in (c).
Chapter 11

Measuring the visual quality of compressed images

11.1 PSNR ($L_2$) and Hausdorff distance

In this Section we draw attention to our observation that the PSNR measure based on the $L_2$ norm is not good enough for error measurements, and in some cases coding procedures aiming at maximization of the PSNR (which corresponds to minimization of the $L_2$ error) are forced to ‘damage’ the low-dimensional geometric structure. We illustrate this fact in Figure 11.2. Both coding results in Figure 11.1(b) and in Figure 11.2(b) produced at the same bit rate 0.05 bpp. GPP coding in Figure 11.1(b) captures more 1-D geometry since the partition in Figure 11.1(a) retains more edges than the partition in Figure 11.2(a). However, GPP coding in Figure 11.1(b) obtains a larger PSNR.

As we believe that it is the low-dimensional structure that carries the vital information for the human perception system, the visual quality of compressed images is linked to the quality of encoding of the original 1-D geometry in the image. Although the only commonly used measure of the quality of image coding is based on the PSNR measure, in fact, in the industry of image and video encoding, such as cable and satellite TV, the quality is measured not only with PSNR. The decision about the superiority of some coding algorithm is eventually taken by specialists after they watch its encoded images. However, there is still a lack of a commonly used automatic measure for measuring the visual quality of image coders.

11.1.1 Two counter examples

Here are two very clear and simple examples for ‘non-optimality’ of the $L_2$ or $L_\infty$ norm as a visual metric. First, consider an image, that has a small object on a zero value background $I = 1_\Delta$, $\text{area}(\Delta) \ll \text{area}(I) = 1$. If the approximant is simply an empty image $\tilde{I} \equiv 0$, as if we would have dismissed the object $\Delta$, the corresponding $L_2$ error between the original image and the approximant is insignificantly small $\|I - \tilde{I}\|_2^2 = \text{area}(\Delta) \ll 1$. This is a very unsatisfactory error margin, since it assigns a small error to the approximant that reproduces virtually zero information of the original image. To its contrary, the Hausdorff distance and the $L_\infty$ norm would signal a maximum error in this case, indicating that the above approximant is inferior.

Neither are we satisfied with the $L_\infty$ norm. Consider the example where the image has an object over a zero background $I = 1_\Delta$, and let the approximant be the same object $\Delta$, just shifted a little from its original location. In this case the $L_\infty$ norm $\|I - \tilde{I}\|_{L_\infty}$ is maximal, while the original image and the approximant show very little visual difference, and the corresponding Hausdorff distance is also small.
Figure 11.1: (a) Partition constructed by the GPP algorithm, (b) GPP image coding (at 0.05 bpp) corresponding to (a).

Figure 11.2: (a),(b) Partition and corresponding GPP image coding with less edges than in Figure 11.1. The PSNR of the image in 11.1(b) is lower than the PSNR of the image in (b) yet the amount of information in 11.1(b) is remarkably bigger. (Notice the tower, the handle of the camera and the hand of the cameraman in Figure 11.1(b) that are missing in (b)).
11.2 A new measure for visual quality of images

It has been already discussed in the past using other metrics, instead of the common $L_2$ norm, for practical applications in image processing. For example, it is a reasonable assumption that the human visual perception system is more sensitive to the Sobolev norm which considers image discontinuities (for example, see [80], and [40]). It was also advocated that the $L_1$ norm is better adapted for practical applications than $L_2$ (appropriate wavelet transform is proposed in [26], p. 58). In this section we suggest a generalization of the Hausdorff distance towards a more general geometric measure. Based on our practical experience, we suggest that measuring the quality of image coding should combine the $L_2$ norm and some geometric metric, therefore we generalize the Hausdorff distance employing the ideas of the classical $L_2$ norm. The Hausdorff distance is already employed in the problem of image matching and recognition ([38], [62]), and integral variants of the Hausdorff distance in approximation was done by Bl. Sendov in 1966 [66] (see also [67], [68], [69], and references therein).

Here we describe the digitalized $L_2$ extension of the Hausdorff distance specifically constructed for measuring the quality of approximation in images. Let $f$ and $g$ be two images, whose graphs $G(f)$, $G(g)$ present two finite sets in $3-D$. The definition of the Hausdorff distance is based on the Euclidian distance between every point of $G(f)$ to the set of $G(g)$:

$$\text{dist}(x, G(g)) = \min_{y \in G(g)} \| x - y \|$$

Then, the distance from the set $G(f)$ to the set $G(g)$ is:

$$\text{dist}(G(f), G(g)) = \max_{x \in G(f)} \text{dist}(x, G(g)),$$

and the Hausdorff distance is defined as:

$$\text{haus}(f, g) = \max\{\text{dist}(G(f), G(g)), \text{dist}(G(g), G(f))\} \quad (11.1)$$

There are several ways to extend 11.1). First of all, we consider the 3-D complete graph corresponding to the images $f$ and $g$. The complete graph $G(f)$ of $f$ is the boundary of $3D$ volume generated by columns of height $f(p)$ over the pixels $p$. We suggest the following measure between two images $f$, $g$:

$$\text{meas}(f, g)_2^2 = \max\left\{ \sum_{x \in G(f)} (\text{dist}(x, G(g))^2, \sum_{x \in G(g)} (\text{dist}(x, G(f))^2) \right\}$$

where $\text{dist}(x, G(g) = \min_{y \in G(g)} \| x - y \|$, with $\| \cdot \|$ the Euclidian norm.

A possible improvement in the GPP algorithm is to find a way to replace best approximation by polynomials in the $L_2$-norm by best approximation by polynomials in the $\text{meas}_2$.

11.3 An idea for a measure of visual quality: combination of the Hausdorff distance and the $L_2$ norm

Measuring the visual quality by the Hausdorff distance can be extremely advantageous in some cases, compared to the parametrization-dependant $L_2$ norm. Working on the GPP algorithm and observing different examples of images, we have gained an intuition that measuring the visual quality for the geometric artificial images should be based solely on the Hausdorff distance, while the less is the amount of geometry in the image, the more reasonable measuring the quality using the $L_2$ norm turns to be.
Therefore, we suggest that measuring the visual quality should be based both on the Hausdorff distance and the $L_2$ norm. The Hausdorff distance carries the information about the approximation of the geometric structure in the image (the edges), and may be computed either in the whole image, or at the locations of the edges. The $L_2$ norm should measure the quality of approximation in corresponding parts of the image. Those parts are decided for each task separately. For example, it is widely accepted that the $L_2$ norm does not apply well for measuring the quality of approximation in the region of texture (assumed to contain no geometry), but it can be a task-specific issue. However, the $L_2$ norm is expected to be very relevant in the regions of smoothness.

The GPP algorithm gives an opportunity to measure the visual quality of image coding with such a combined measure. We can employ the $L_2$ norm in the corresponding subdomains of smoothness, and apply the Hausdorff distance in the bands. To the contrary, any functional model, which does not represent explicitly the geometric structure in the image, but deprived of the ability to support such a combined error. The open problem with our combined measure is the issue of the exact definitions of the regions of smoothness and the edges. Such a problem might be settled in one specific task, but can be hardly resolved for all image coding applications.
Bibliography


