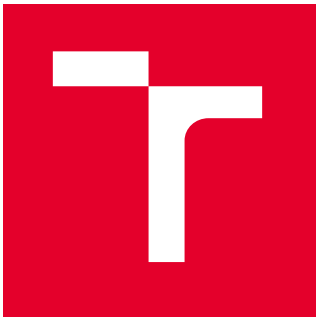


BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

IMAGE EDGE DETECTION USING CONVEX OPTIMISATION

DETEKCE HRAN V OBRAZE POMOCÍ KONVEXNÍ OPTIMALIZACE

DOCTORAL THESIS
DIZERTAČNÍ PRÁCE

AUTHOR — Ing. Michaela Novosadová
AUTOR PRÁCE

SUPERVISOR — prof. Mgr. Pavel Rajmic, Ph.D.
ŠKOLITEL

BRNO 2023

ABSTRACT

Image edge detection is one of the most important techniques in digital image processing. It is used, among other things, as the first step of image segmentation. Therefore, it remains an area of interest for researchers trying to develop ever-better detection approaches. The main objective of this Thesis is to find a suitable method for image edge detection using convex optimisation. The proposed method is based on sparse modelling, and its main part is formulated as a convex optimisation problem solved by proximal algorithms. For defining the optimisation problem, it is assumed that the signal can be modelled as an over-parametrised, piecewise-polynomial signal that consists of disjoint segments. The number of these segments is significantly smaller than the number of signal samples, which encourages the use of sparsity. The formulation of a suitable optimisation problem is first performed on one-dimensional signals since the implementation and comparison of the different algorithms is significantly easier and less time-consuming for one-dimensional signals than two-dimensional ones.

The first part of the Thesis introduces the basic theory in signal processing, sparsity, convex optimisation and proximal algorithms. It also presents a cross-section of the methods used for image edge detection. The second part of the Thesis focuses on the formulation and the subsequent evaluation of individual optimisation problems for the segmentation of one-dimensional synthetic signals corrupted by noise. The evaluation is conducted in terms of both denoising and breakpoint detection accuracy. The last part of the Thesis is dedicated to expanding the best-performing approach for breakpoint detection in one-dimensional signals for the application to image edge detection. The proposed approach is tested on a standardised dataset of images containing manually labelled edges of several subjects. The results of the proposed method are evaluated using precision-recall curves and their maximum F-measure score, and then compared with other edge detection methods.

KEYWORDS

Signal segmentation, image edge detection, convex optimisation, proximal splitting algorithm, proximal operator, sparsity, total variation, gradient

ABSTRAKT

Detekce hran v obraze je jednou z nejdůležitějších technik v oblasti digitálního zpracování obrazu. Bývá používána, mimo jiné, jako první krok segmentace obrazu. I proto stále zůstává v oblasti zájmu vědců, kteří se snaží vyvíjet stále lepší detekční přístupy. Hlavním cílem této práce je nalezení vhodné metody detekce hran v obraze pomocí konvexní optimalizace. Navržená metoda je založená na řídkém modelování, a její hlavní část je formulována jako konvexní optimalizační problém, který je řešen pomocí proximálních algoritmů. Pro definici optimalizačního problému se předpokládá, že signál může být modelován jako přeparametrizovaný po částech polynomiální signál, který se skládá z disjunktních segmentů. Počet těchto segmentů je výrazně menší než je počet vzorků signálu, což vybízí k použití řídkosti. Návrh vhodného optimalizačního problému nejdříve probíhá na jednorozměrných signálech, jelikož implementace a porovnání jednotlivých algoritmů je pro jednorozměrné signály výrazně jednodušší a časově méně náročná, než pro dvojrozměrné.

První část práce se věnuje představení základní teorie z oblasti zpracování signálu, řídkosti, konvexní optimalizace a proximálních algoritmů, a dále prezentuje průřez používanými metodami pro hranovou detekci v obraze. Druhá část práce se zaměřuje na návrh a následné vyhodnocení jednotlivých optimalizačních problémů pro segmentaci jednorozměrných syntetických signálů, které jsou poškozeny šumem. Vyhodnocení je provedeno jak z pohledu přesnosti detekce skoků tak i odšumění. Poslední část práce je věnována rozšíření nejlépe fungujícího přístupu k detekci skoků v jednorozměrném signálu pro použití na detekci hran v obraze. V této části je navržený přístup testován na standardizovaném datasetu obrázků, který obsahuje manuálně označené hrany od několika subjektů. Výsledky navržené metody jsou vyhodnoceny pomocí precision–recall křivek a jejich maximálního F skóre a následně porovnány s ostatními metodami hranové detekce.

KLÍČOVÁ SLOVA

Segmentace signálů, detekce hran v obraze, konvexní optimalizace, proximální algoritmy, proximální operátory, řídkost, totální variace, gradient

NOVOSADOVÁ, Michaela. *Image edge detection using convex optimisation*. Brno, 2023, 179 p. Doctoral thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications. Advised by Prof. Mgr. Pavel Rajmic, Ph.D.

Vysázeno pomocí balíčku `thesis` verze 2.63; `http://latex.feec.vutbr.cz`

DECLARATION

I declare that I have written the Doctoral Thesis titled "Image edge detection using convex optimisation" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Doctoral Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

During my studies, I have attended several conferences and learnt a lot of new things, and most importantly, I've met a lot of great people who are part of an excellent university environment at the Faculty of Computer Science at the Brno University of Technology, specifically the Department of Telecommunications, where the atmosphere is truly warm and friendly.

For starters, I would like to mention two professors, thanks to whom I learnt about the possibility of studying at this institute and who first gave me the impulse to start my doctoral studies. Specifically, professor Zdeněk Smékal and Kamil Vrba. Thank you.

Furthermore, I must thank my supervisor Pavel Rajmic for guiding me through the entire study and for the valuable advice he gave me throughout the process. And most importantly, for the support, I have always found in him during my many years of study. Thank you for not giving up on me.

Among other things, I have to thank my family, who have been my support system throughout my life. And as they say, you don't choose your family, but even if I had the choice, I wouldn't have chosen another. Special thanks goes to my sister Eliška, who gave me energy and a good mood, especially during the long, sometimes mainly monologue phone calls.

Same time last year, my boyfriend Pavel Záviška was writing the acknowledgements in his dissertation, and he mentioned several times that he was looking forward to reading the acknowledgements in my dissertation. That time has just come – a year to the day. Thanks to his support and encouragement, I was able to complete the Thesis. He supported me to the point where I could not engage in everyone's favourite activity of thesis writers – procrastination. He even gave up playing our favourite board games together so that I could focus on my writing. And that was a big sacrifice for him; fortunately, he didn't have to give up playing the guitar. Truth be told, this Thesis wouldn't look the way it does now without his valuable advice on formatting in LaTeX and proofreading the language. For all of this, I owe him an incredible debt of gratitude.

Finally, I must thank our unborn daughter, who provided me with 24/7 internal support while completing my dissertation. Thanks to her, it turned out that the work could get finished even without caffeine. And when I needed a kick up the backside, she took care of it herself.

This work was supported by the GAČR project number 16-13830S.

PODĚKOVÁNÍ

V rámci svého studia jsem se naučila spoustu nových věcí, zúčastnila se několika konferencí, a hlavně potkala spoustu skvělých lidí, kteří jsou součástí úžasného univerzitního prostředí na FEKT VUT v Brně, konkrétně Ústavu telekomunikací, kde panuje velmi přátelská atmosféra.

Na začátku bych chtěla zmínit dva pány profesory, díky kterým jsem se o možnosti studia na tomto ústavu dozvěděla a kteří mi jako první dali impuls k tomu, abych se do doktorského studia pustila. Konkrétně se jedná o pana profesora Zdeňka Smékala a Kamila Vrbu. Děkuji.

Dále musím poděkovat svému školiteli Pavlu Rajmicovi, za provedení celým studiem a za jeho cenné rady, které mi po celou dobu poskytoval. A hlavně za podporu, kterou jsem u něj vždy našla, během mého dlouholetého studia. Děkuji, že jsi to se mnou nevzdal.

Mimo jiné je potřeba poděkovat mé rodině, která je mi celý život oporou ve všech aspektech. A jak se říká, rodinu si člověk nevybírá, ale já i kdybych tu možnost měla, tak bych si nevybrala jinou. Speciální díky patří mé sestře Elišce, která mi dodávala energii a dobrou náladu především během dlouhých někdy z velké části monologových telefonních hovorů.

Loni touto dobou psal poděkování do své dizertace můj přítel Pavel Záviška, a u toho několikrát zmínil, že se těší až si bude moci přečíst poděkování v mé dizertaci. Ten čas právě nastal – do roka a do dne. Díky jeho podpoře a povzbuzování se mi podařilo práci dokončit. Podporoval mě až do takové míry, že jsem se nemohla věnovat oblíbené činnosti všech pisatelů závěrečných prací – prokrastinaci. Dokonce se vzdal i společného hraní oblíbených deskových her, abych se mohla psaní věnovat. A to pro něj byla velká oběť, naštěstí, hraní na kytaru se vzdávat nemusel. Pravdou je, že bez jeho cenných rad ohledně formátování v LaTeXu a korektury jazyka, by tato práce rozhodně nevypadla tak, jak vypadá teď. Za to vše mu patří neskutečné díky.

Na závěr musím poděkovat naší ještě nenarozené dceři, která mi při dokončování dizertace poskytovala vnitřní podporu 24 hodin denně. Díky ní se ukázalo, že i bez kofeinu se dá dílo dokončit. A když jsem potřebovala nakopnout, tak se o to sama postarala.

Tato práce byla podpořena z projektu GAČR číslo 16-13830S.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

INTRODUCTION

Sparse representation has proven to be a very powerful tool in a variety of applications, especially in signal processing, image processing, machine learning, and computer vision. The list of specific tasks concerning images, for which sparse representation offers a huge potential, contains image inpainting, image denoising, image segmentation, visual tracking, and more. Another example of a popular topic in the last few years, where the use of sparse representation is definitely beneficial and favourable, is image classification. Sparse representation are directly related to compressed sensing, which can be seen as a method for recovering a signal from a small number of linear measurements.

Specific problems of the mentioned topics, which need to be solved, can be often represented via convex optimisation problems. Searching for the sparse representation contributed to the development of numerical methods for solving convex optimisation problems – specifically the proximal algorithms, which are iterative algorithms based on the evaluation of the proximal operators associated with the optimised function. Optimisation problems can be solved using approximation algorithms, which can be divided into three categories: the greedy algorithms, relaxation algorithms (which include also the proximal algorithms) and "hybrid" algorithms, combining different approaches.

The image edge detection is one of the most used techniques in digital image processing, computer and robot vision. It finds application in many topics such as object tracking, motion detection, pattern recognition, image segmentation, medical data processing, etc. An edge in the image is usually defined as a position where the intensity of an image changes significantly. However, each application requires a different estimation of what a significant edge is and thus there are several different approaches suited for each application.

Image edge detection can be viewed as the first step of image segmentation, therefore, even nowadays, the edge detection is in the focus of researchers who are still trying to develop better edge detection techniques.

Image segmentation is one of the most important applications in digital signal processing. Segmentation divides an image into areas (segments) which have similar features or form logical parts. These segments are disjoint and cover the entire image. A very interesting and important area of image segmentation is a segmentation of medical images.

The most frequently used imaging modalities for anatomical structure imaging in medicine are Computed Tomography (CT), Magnetic Resonance (MR) and Ultrasound. Image segmentation is most often applied to CT and MR images. These imaging modalities are used for diagnostic and treatment planning in various med-

ical disciplines. In the time of developing 3D printing, the importance of image segmentation is increasing. Doctors can print segmented organs on a 3D printer and plan the operation on anatomical models of a particular patient.

This Thesis works with an assumption that an image can be modelled as an overcomplete piecewise-polynomial image, such as the image consisting of disjoint piecewise-polynomial patches/segments. The number of the signal segments S is considerably lower than the number of the signal samples N ($S \ll N$), which motivates to measure and optimise sparsity in parametrisation images. Based on the above-mentioned assumptions, the convex recovery problem can be formulated.

Our approach was inspired by authors of [11], who used a greedy approach to solve the optimisation problem for signal segmentation. The presented approach is using the over-parametrised signal model and ℓ_1-based convex relaxation methods. To the best of our knowledge, it is the first time when the ℓ_1-minimisation-based approach is used for signal segmentation/edge detection.

First, the convex optimisation problems for one-dimensional signal segmentation and denoising are formulated since it is easier to implement appropriate algorithms able to solve such formulated problems, and determine their advantages and disadvantages for one-dimensional signals than for two-dimensional signals, i.e. images. Afterwards, the formulation of the convex optimisation problem for the image edge detection, later solved by the most promising algorithm from the 1D segmentation phase, takes place.

This Thesis is divided into several chapters. In Chapter 1, a basic overview, the notation used, and preliminary knowledge concerning sparse representation, convex optimisation, and proximal splitting algorithms is presented. Chapter 2 briefly introduces selected topics from signal processing, such as image edge detection and segmentation. This is followed by Chapter 3, which gives an overview of the image edge detection techniques used. The aims and objectives of the Thesis are presented in Chapter 4. The entire Chapter 5 is dedicated to the solution of the 1D signal segmentation and denoising problem. It defines the formulation of the 1D signal model, proposes a general methodology for solving the 1D signal segmentation and denoising problem, and presents several formulations of the convex optimisation problem, as well as its solution and evaluation. The most promising method from Chapter 5 is extended and applied to image edge detection in Chapter 6. In this chapter, a description of the 2D signal model is presented and several experiments are performed. Various edge detection techniques discussed in Chapter 3 are compared with the final form of the proposed solution. The last part of this Thesis is the Conclusion, which summarises the whole Thesis and presents further possible research directions in this area.

1 PRELIMINARY KNOWLEDGE

In order to cover topics in this Thesis, this chapter provides a basic overview and preliminary knowledge. In Sec. 1.1, the basic concepts and notation are introduced. Sec. 1.2 deals with an explanation of sparse signal representation. Finally, convex optimisation including the description of proximal splitting algorithms can be found in Sec. 1.3.

1.1 Concepts and notation

This section presents the notation used, introduces vector and matrix norms, and presents definitions of vector spaces and bases.

1.1.1 Notation

Scalar variables are marked in italics, i.e. m, N, and vectors are denoted in bold, i.e. $\mathbf{x}, \mathbf{y}$. Unless stated otherwise, the finite-dimensional vectors are considered to be column vectors. The indexing of the elements of vectors starts with one, and the index is indicated either in square brackets or as a subscript, i.e. $\mathbf{y} = [y[1], \ldots, y[N]]^\top$, $\mathbf{x} = [x_1, \ldots, x_N]^\top$. The cardinality (the number of elements in a set) is denoted in the same way as absolute value, i.e. $|4, -6, 0, 8, 8, -5| = 6$. The support of the vector $\mathrm{supp}(\mathbf{x}) = \{i \mid x_i \neq 0\}$ is a set of its indexes, where the vector has non-zero values, i.e. for $\mathbf{x} = [x_1, \ldots, x_6]^\top = [0, 5, 0, 0, -4, 0]^\top$ its support is $\mathrm{supp}(\mathbf{x}) = \{2, 5\}$ and $|\mathrm{supp}(\mathbf{x})| = 2$.

Matrices are also marked in bold, but with capital letters, e.g. $\mathbf{A}, \mathbf{X}$. Elements of matrices are indexed by the respective lower-case letters, i.e. $a_{i,j}, x_{i,j}$. The row i of the matrix $\mathbf{A}$ is denoted as $\mathbf{a}_{i,:}$, similarly $\mathbf{a}_{:,j}$ represents the column j. The symbol $*$ denotes the so-called Hermitian transpose: $\mathbf{A}^*$ is the matrix formed by the composition of the transpose of the matrix and the complex conjugate of each of its elements. It holds $(\mathbf{AB})^* = \mathbf{B}^*\mathbf{A}^*$. Identity matrix is denoted as $\mathbf{I}$.

The vectorisation of the matrix is done by the operator $\mathrm{vec}(\cdot)$, which stacks the columns of the matrix one below the other into a single column vector.

Vector spaces are marked as $\mathbb{R}^N, \mathbb{C}^N$, where the upper index denotes the dimensions of the vector space.

Linear operators are marked in italics using capital letters as L. Each linear operator $L\colon \mathbb{R}^N \to \mathbb{R}^M$, can be always represented by a matrix $\mathbf{A}$ of a size $M \times N$, the operation then works like matrix multiplication $\mathbf{y} = \mathbf{Ax}$. The identity operator is marked as Id.

1.1.2 Vector norms

The vector norm is a function that assigns a non-negative real number to a vector, representing the "length" of the vector. The ℓ_p-norm of the vector is defined as follows [12]:

$$\begin{aligned} \|\mathbf{x}\|_p &:= \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p} \quad \text{for } 1 \leq p < \infty, \\ \|\mathbf{x}\|_p &:= \sum_{i=1}^{N} |x_i|^p \text{ for } 0 < p < 1. \end{aligned} \tag{1.1}$$

Strictly speaking, it is a norm only in the case of $1 \leq p < \infty$. However, the ℓ_p-norm will be used uniformly for all p for simplicity.

For the limit cases of (1.1), where $p = 0$ and $p = \infty$, it holds

$$\begin{aligned} \|\mathbf{x}\|_0 &:= |\text{supp}(\mathbf{x})|, \\ \|\mathbf{x}\|_\infty &:= \max_i |(x_i)|. \end{aligned} \tag{1.2}$$

The ℓ_0-norm represents the number of non-zero elements of the vector $\mathbf{x}$, and the ℓ_∞-norm is the maximum absolute value of the vector.

The most commonly used norm is ℓ_2-norm, the so-called Euclidean norm, which is defined as

$$\|\mathbf{x}\|_2 := \sqrt{\sum_i |(x_i)|^2}, \tag{1.3}$$

and if the simplified notation $\|\cdot\|$ is used, it will mean $\|\cdot\|_2$.

In this Thesis, the ℓ_1-norm, which represents the sum of the absolute values of the vector, formally

$$\|\mathbf{x}\|_1 := \sum_i |(x_i)|, \tag{1.4}$$

will be used.

1.1.3 Matrix norms

It is also possible to apply a norm to a matrix. The easiest way to apply a norm to a matrix is to vectorise it and apply the vector norm to the result.

The most commonly used matrix norm is the Frobenius norm, which is defined as the energy of the elements of the matrix $\mathbf{A}$ [12]:

$$\|\mathbf{A}\|_{\text{F}} = \|\,\text{vec}(\mathbf{A})\|_2 = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} |(a_{i,j})|^2}. \tag{1.5}$$

The Frobenius norm is a special case of general (p, q)-mixed norm where $p, q = 2$. A general (p, q)-mixed norm if $p, q \geq 1$ is defined by

$$\|\mathbf{A}\|_{pq} = \left(\sum_{j=1}^{N} \left(\sum_{i=1}^{M} |(a_{i,j})|^p \right)^{\frac{q}{p}} \right)^{\frac{1}{q}}. \tag{1.6}$$

It can be seen as an application of ℓ_q-norm to every column of the matrix $\mathbf{A}$ and then an application of ℓ_p-norm to the resulting vector.

The $\|\cdot\|_{21}$ is ℓ_{21}-norm whose input is a matrix $\mathbf{Z}$ of size $p \times q$ and it is formally defined as follows:

$$\|\mathbf{Z}\|_{21} = \|[\|\mathbf{z}_{1,:}\|_2, \|\mathbf{z}_{2,:}\|_2, \ldots, \|\mathbf{z}_{p,:}\|_2]\|_1 = \|\mathbf{z}_{1,:}\|_2 + \cdots + \|\mathbf{z}_{p,:}\|_2, \tag{1.7}$$

i.e. the ℓ_2-norm is applied to the individual rows of the matrix $\mathbf{Z}$, which results in a vector of size $1 \times q$. The resulting vector is evaluated by the ℓ_1-norm. The ℓ_1-norm is used as a convex substitute for the true sparsity measure [13, 14].

1.1.4 Operator norm

Let L be a linear operator between Hilbert spaces, its operator(spectral) norm is defined as follows:

$$\|L\| = \|L\|_{\mathrm{OP}} = \sup_{\mathbf{x} \in \mathbb{C}^N, \mathbf{x} \neq \mathbf{0}} \frac{\|L\mathbf{x}\|_2}{\|\mathbf{x}\|_2}. \tag{1.8}$$

For the operator/spectral norm it holds $\|L\|^2 = \|L^*L\| = \|LL^*\|$, where L^* is an adjoint operator of L, and $\|L\|^2$ is equal to the largest singular value of the operator L^*L [12].

1.1.5 Vector spaces and bases

A vector space is an algebraic structure satisfying the well-known axioms and its elements are vectors. In this Thesis, the dimension of the vector space will be a finite number $0 < N \in \mathbb{N}$.

A system of generators of the vectors space $\mathbb{V}$ is a subset of vectors $\mathbf{E} = \mathbf{e}_1, \ldots, \mathbf{e}_M$ in $\mathbb{V}$.

A vector space $\mathbb{V}$ is generated by a subset of vectors $\mathbf{E} = \mathbf{e}_1, \ldots, \mathbf{e}_M$ in $\mathbb{V}$. It means that each vector $\mathbf{x} \in \mathbb{V}$ is a linear combination of generators, formally

$$\mathbf{x} = c_1\mathbf{e}_1 + c_2\mathbf{e}_2 + \cdots + c_M\mathbf{e}_M = \mathbf{E}\mathbf{c}, \tag{1.9}$$

where the scalars c_i are called coordinates.

A vector space $\mathbb{V}$ can be generated by more than N systems of generators, which means that a vector $\mathbf{x} \in \mathbb{V}$ can be determined by several representations, and such a system is called *overcomplete*.

The basis of a vector space is the minimal system of its generators. It is the set of linearly independent vectors. Any vector $\mathbf{x}$ in the considered vector space $\mathbb{V}$ can be obtained by a linear combination of these vectors. A basis is any set containing N linearly independent vectors in a finite space of dimension N, which will be marked

in bold with capital letters. If $\mathbf{B} = \mathbf{b}_1, \ldots, \mathbf{b}_N$ is a basis, then each element of $\mathbf{x} \in \mathbb{V}$ can be expressed as

$$\mathbf{x} = \sum_{i=1}^{N} c_i \mathbf{b}_i = \mathbf{B}\mathbf{c}. \tag{1.10}$$

The basis $\mathbf{B} = \mathbf{b}_1, \ldots, \mathbf{b}_N$ is orthogonal if all the basis vectors are perpendicular to each other, in other words, their scalar product is zero:

$$\langle \mathbf{b}_i, \mathbf{b}_j \rangle = 0, \quad \langle \mathbf{b}_i, \mathbf{b}_i \rangle \neq 0. \tag{1.11}$$

The basis $\mathbf{B}$ is normalised if all its vectors have the vector length equal to 1. In other words, the ℓ_2-norm of each vector satisfies the condition $\|\mathbf{b}_i\| = 1$.

A basis is called orthonormal if it is both orthogonal and normalised.

1.2 Sparse representation of signals

In the beginning, the term "sparse" should be clarified. It can be used to describe the properties of a matrix or a vector, which can be called sparse if most of its elements are zero [15].

The term "sparse" can be also used to describe the properties of a signal representation. The representation of a signal is sparse if the signal is modelled as a linear combination of only a few elements from the dictionary $\mathbf{A}$. The dictionary consists of a set of basis elements (called atoms) and it is always formed with respect to a specific task [16]. Sparse representation is important for many applications [15]. The advantages of sparse representations include simplifying the interpretation of data, enabling easy and strong compression or providing numerical stability.

The intention of the sparsity-based methods is to select a few atoms from the dictionary that best represent given signal $\mathbf{y} \in \mathbb{R}^N$. Signal $\mathbf{y}$ is defined by the linear system $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{N \times M}$ is the dictionary and $\mathbf{x} \in \mathbb{R}^N$ is the vector with the transform coefficients [17]. The aim is to find a sparse solution of the given linear system with as few coefficients as possible.

Sparsity k of the solution concerns the number of non-zero coefficients in a vector of length N. The sparsity is usually measured by the ℓ_0-norm, which refers to the number of non-zero entries k. A vector is called k-sparse, when its ℓ_0-norm is equal to k. Usually, it holds that $k < N$ or $k \ll N$ [18].

The basic sparse representation problem is formulated as

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{y}. \tag{1.12}$$

The problem is NP-hard and the solution is difficult to approximate. The ℓ_0-norm is a non-convex function and therefore, it is not possible to use any of the algorithms

of convex optimisation for solving the problem (1.12). To be able to use the convex optimisation, it is necessary to use convex ℓ_p-norms, which are convex for $p \geq 1$. The best choice is to use the ℓ_1-norm, which is the closest convex norm to the ℓ_0-norm and it can be shown that in many cases the solution with ℓ_1-norm is equivalent to the solution obtained by ℓ_0-norm, see [19], for instance. Therefore, the problem (1.12) can be relaxed to

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{y}. \tag{1.13}$$

In the case of noisy signal $\mathbf{y}$, defined as $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$, where $\mathbf{e} \in \mathbb{R}^N$ is noise, the optimisation problem can be recast to

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 \leq \delta, \tag{1.14}$$

where δ is a noise level. It will be shown later that for finding the solution to the optimisation problem (1.14), the convex optimisation can be used.

1.3 Convex optimisation

In mathematics, computer science, and economics, an optimisation problem is determined by finding the best solution among all possible solutions. A general optimisation problem can be defined as follows [20]:

$$\text{minimize} \quad f_0(\mathbf{x}) \quad \text{s.t.} \quad f_i(\mathbf{x}) \leq \delta_i, \quad i = 1, \ldots, m, \tag{1.15}$$

where $\mathbf{x}$ is the optimisation variable of the problem, f_0 is the objective function, f_i are constraint functions, and δ_i are limits of the constraints. The solution to the optimisation problem (1.15) is the vector $\hat{\mathbf{x}}$ producing the smallest objective value among all vectors satisfying the constraints, i.e. for any $\mathbf{z}$ with $f_1(\mathbf{z}) \leq \delta_1, \ldots, f_m(\mathbf{z}) \leq \delta_m$, it holds $f_0(\mathbf{z}) \geq f_0(\hat{\mathbf{x}})$.

Convex optimisation problems form a class of optimisation problems, where both objective and constraint functions are convex, i.e. they satisfy the inequality [20]

$$f_j(\alpha\mathbf{x} + \beta\mathbf{y}) \leq \alpha f_j(\mathbf{x}) + \beta f_j(\mathbf{y}), \quad j = 0, \ldots, m, \tag{1.16}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$.

For finding the exact sparse solution, the ℓ_0-norm is used. However, finding the exact sparse solution of the optimisation problem using the ℓ_0-norm is NP-hard. Such a task is very time-consuming and computationally prohibitive when N is large. Therefore, such an approach is not suitable for practical applications. The computational complexity can be reduced, when the approximation algorithms are

used. In general, such algorithms provide solutions quickly, but at the expense of accuracy.

Approximation algorithms can be divided into three categories [12]: the greedy algorithms (e.g. Matching Pursuit [21], Orthogonal Matching Pursuit [22]), ℓ_1 relaxation algorithms (e.g. Basis Pursuit [23], modified Least Angle Regression [24], Iterative Reweighted Least Squares [25], Dantzig Selector [26] and proximal algorithms [27]) and "hybrid" algorithms, which combine approaches of different algorithms (e.g. A* Orthogonal Matching Pursuit [28]).

While greedy algorithms [17, 29] usually consist of simpler steps and can be faster and more predictable, relaxation methods [30,31] rely on convex optimisation, which can provide better results but can take longer to achieve convergence. The disadvantage of the greedy algorithms is that there is no guarantee of achievement of the global minimum, while ℓ_1-relaxation algorithms rely on the condition that if specific conditions are satisfied, accurate solution is acquired. This doctoral Thesis will focus on solving the convex optimisation problem by proximal algorithms (PAs).

1.3.1 Proximal splitting algorithms

Proximal splitting algorithms (PAs) are in general able to solve unconstrained convex optimisation problems of type

$$\arg\min_{\mathbf{x}} f_1(L_1\mathbf{x}) + \cdots + f_m(L_m\mathbf{x}), \tag{1.17}$$

where functions $f_1, \ldots, f_m$ are convex and $L_1, \ldots, L_m$ are linear operators [31]. Note, however, that some algorithms are restricted regarding the number of minimised functions and linear operators. For instance, the Douglas–Rachford algorithm is able to find the minimum of the sum of only two convex functions, without linear operators (i.e. the linear operators are identities).

The PAs are suitable for finding the minimum of a sum of convex functions during an iterative process. A process of iterations is generally terminated after a pre-defined number of iterations or if a convergence criterion is met, e.g. the change in the solution is small enough: $\|\hat{\mathbf{x}}^{i+1} - \hat{\mathbf{x}}^i\|_2 / \|\hat{\mathbf{x}}^i\|_2 \leq \varepsilon$, where i is the number of the iteration and ε is the parameter of the convergence criterion. Proximal splitting algorithms are proven to provide convergence to the optimal value. However, in practice, convergence and its speed are greatly affected by the nature of the functions and parameters used in the algorithm.

The PAs perform iterations involving the evaluation of proximal operators (see below) and/or gradients related to the individual functions, which is much simpler than minimisation of the composite functional by other means. The advantage of proximal splitting algorithms is that they allow solving the minimisation problems

even with non-smooth functions. Each non-smooth function is involved via its proximity operator.

In the previous chapter, the constrained optimisation problem (1.14) is defined. To be able to use the PAs for solving this problem, it has to be reformulated to its unconstrained form using the indicator function:

$$\arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \iota_{\{\mathbf{x}:\, \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2 \leq \delta\}}, \tag{1.18}$$

or it can be transformed to another unconstrained form:

$$\arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{1.19}$$

which is equivalent to the constrained problem (1.14) since it holds that to each δ, it is possible to find $\lambda > 0$ such that both problems (1.14) and (1.19) lead to the same optimal solution $\hat{\mathbf{x}}$. The problem (1.19) is regularised by the second function called "penalty". The penalisation weight λ influences the result of the optimisation. Higher values of λ penalize non-sparse solutions more than lower values of λ, but at the same time, too large values of λ may lead to deviations from the data. Therefore, the penalisation weight should be tuned carefully.

1.3.2 Proximal operators

Proximal operators are essential elements of proximal algorithms, therefore, their introduction is important. A proximal operator of a function h maps $\mathbf{z} \in \mathbb{R}^N$ to another vector in $\mathbb{R}^N$, such that

$$\operatorname*{prox}_{h}(\mathbf{z}) = \arg\min_{\mathbf{x}\in\mathbb{R}^N} \; h(\mathbf{x}) + \frac{1}{2}\|\mathbf{z}-\mathbf{x}\|_2^2. \tag{1.20}$$

Therefore, $\operatorname{prox}_h(\mathbf{z})$ can be understood as the result of a regularised minimisation of h in the neighbourhood of $\mathbf{z}$.

The most used proximal operators are described in [31–34]. The proximal operators used in this Thesis are presented below.

The proximal operator of function $f = \frac{\zeta}{2}\|\mathbf{P}\cdot-\mathbf{y}\|_2^2$ is defined [31, 32] as

$$\operatorname*{prox}_{\frac{\zeta}{2}\|\mathbf{P}\cdot-\mathbf{y}\|_2^2}(\mathbf{z}) = (\mathbf{I}+\zeta\mathbf{P}^\top\mathbf{P})^{-1}(\mathbf{z}+\zeta\mathbf{P}^\top\mathbf{y}), \tag{1.21}$$

where $\mathbf{I}$ stands for the identity matrix.

Total variation $\mathrm{TV}(\cdot)$ is the relaxed counterpart of $\|\nabla\cdot\|_0$, which is defined as

$$\mathrm{TV}(\mathbf{z}) = \|\nabla\mathbf{z}\|_1 = \sum_{i=1}^{N-1} |z_{i+1}-z_i|. \tag{1.22}$$

The proximal operator of TV$(\cdot)$ has no closed form, but for 1D signals it can be computed fast in finite number of operations using e.g. the Condat algorithm [35].

Soft thresholding is the proximal operator of the ℓ_1-norm and it is defined as

$$\underset{\tau}{\text{soft}}(\mathbf{z}) = \text{sgn}(\mathbf{z}) \odot \max(|\mathbf{z}| - \tau, 0), \tag{1.23}$$

where sgn stands for the signum function, and $\odot$ represents the element-wise product of vectors.

The proximal operator of the ℓ_{21}-norm is

$$\underset{\tau\|[\cdot,\ldots,\cdot]\|_{21}}{\text{prox}}(\mathbf{Z}) = \text{soft}_\tau^{\text{row}}(\mathbf{Z}), \tag{1.24}$$

mapping matrix $\mathbf{Z} = [z_{i,j}]$ to another [33]. It can be shown that it is a soft thresholding with the threshold τ over groups consisting of rows of the matrix; specifically, $\text{soft}_\tau^{\text{row}}$ is a mapping

$$z_{i,j} \mapsto \frac{z_{i,j}}{\|\mathbf{z}_{i,:}\|_2} \max(\|\mathbf{z}_{i,:}\|_2 - \tau, 0). \tag{1.25}$$

The proximal operator of an indicator function of a convex set C, denoted ι_C, is the projection onto that set: $\text{prox}_{\iota_C} = \text{proj}_C$ [32, 36]. Consequently, the proximal operator of $\iota_{\{\mathbf{z}\,:\,\|\mathbf{y}-\mathbf{z}\|_2 \leq \delta\}} = \iota_{B_2(\mathbf{y},\delta)}$ is the projection on the ℓ_2-ball, i.e. the projector $\text{proj}_{B_2(\mathbf{y},\delta)}$ finds the closest point to $\mathbf{z}$ in the ℓ_2-ball $\{\mathbf{z} : \|\mathbf{y} - \mathbf{z}\|_2 \leq \delta\}$:

$$\underset{\iota_{B_2(\mathbf{y},\delta)}}{\text{prox}}(\mathbf{z}) = \underset{B_2(\mathbf{y},\delta)}{\text{proj}}(\mathbf{z}) = \frac{\delta(\mathbf{z} - \mathbf{y})}{\max(\|\mathbf{z} - \mathbf{y}\|, \delta)}. \tag{1.26}$$

Given a convex function f, the proximal operator of its Fenchel–Rockafellar conjugate f^* can be computed at virtually the same cost as prox_f thanks to the Moreau identity [32, 36]:

$$\underset{\alpha f^*}{\text{prox}}(\mathbf{u}) = \mathbf{u} - \alpha \underset{f/\alpha}{\text{prox}}(\mathbf{u}/\alpha) \qquad \text{for } \alpha \in \mathbb{R}^+. \tag{1.27}$$

1.3.3 Algorithms used

Several proximal splitting algorithms exist, and their application depends on the definition of the minimisation problem. There are several minimisation problems presented in this Thesis, and this section discusses the algorithms used to solve the respective problems.

Forward-backward (FB) algorithm [31] is able to solve the optimisation problem of type:

$$\text{minimise } f(\mathbf{x}) + g(\mathbf{x}), \tag{1.28}$$

where both functions are convex. Function f is required to be smooth and differentiable with a β-Lipschitz continuous gradient ∇f [31]. An arbitrary function f is

differentiable with the β-Lipschitz continuous gradient if for a real constant $\beta > 0$ it holds [32]:

$$\|\nabla f(x) - \nabla f(x')\| \leq \beta \|x - x'\|, \quad \forall x, x' \in \mathbb{R}^N. \tag{1.29}$$

Function g can be either smooth or non-smooth.

The FB algorithm is iterative, and each iteration consists of two steps – the forward step and the backward step:

$$\mathbf{x}^{i+1} = \underbrace{\text{prox}_{\gamma^i g}}_{\text{backward step}} \underbrace{(\mathbf{x}^i - \gamma^i \nabla f(\mathbf{x}^i))}_{\text{forward step}}, \tag{1.30}$$

where γ is a step size, prox is a proximal operator and ∇ stands for a gradient. Main computations within each iteration are a forward (explicit) gradient step with respect to f and a backward (implicit) proximal step with respect to g.

In the case of $g = 0$, the equation (1.30) is reduced to the gradient method [31]:

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \gamma^i \nabla f(\mathbf{x}^i). \tag{1.31}$$

On the other hand, when $f = 0$, the equation (1.30) is reduced to the proximal point algorithm [31]:

$$\mathbf{x}^{i+1} = \text{prox}_{\gamma^i g}(\mathbf{x}^i). \tag{1.32}$$

According to the above-mentioned assumptions, the FB algorithm can be considered as a combination of these two basic algorithms (1.31) and (1.32).

The general form of the FB algorithm is introduced in Algorithm 1, where θ^i stands for the relaxation parameter and can vary within each iteration same as γ^i. However, the parameters θ and γ stay fixed for all iterations in practice. Note that the size of the parameter γ depends on β.

Algorithm 1: The Forward-backward algorithm solving (1.28)

Input: Functions f, g

Output: $\mathbf{x}^{i+1}$

Set $\varepsilon \in (0, 1/\beta]$

Set initial variables $\mathbf{x}^0 \in \mathbb{R}^N$

for $i = 0, 1, \ldots$ *until convergence* **do**

  Set $\gamma^i \in [\varepsilon, 2/\beta - \varepsilon]$

  $\mathbf{q}^i = \mathbf{x}^i - \gamma^i \nabla f(\mathbf{x}^i)$

  Set $\theta^i \in [\varepsilon, 1]$

  $\mathbf{x}^{i+1} = \mathbf{x}^i + \theta^i (\text{prox}_{\gamma^i g}\, \mathbf{q}^i - \mathbf{x}^i)$

return $\mathbf{x}^{i+1}$

Douglas–Rachford (DR) algorithm [31] is able to solve the same optimisation problem (1.28) as the FB algorithm, i.e. the sum of two convex functions. In contrast with the FB algorithm, the DR algorithm can be also applied in cases when both functions are generally non-smooth [31].

The DR algorithm is iterative, and each iteration consists of two proximal steps related to functions f and g. The general form of the DR algorithm is introduced in Algorithm 2, where θ^i stands for relaxation parameter, which can vary within each iteration. However, the parameter θ stays fixed for all iterations in practice.

Algorithm 2: The Douglas–Rachford algorithm solving (1.28)

Input: Functions f, g

Output: $\mathbf{x}^{i+1}$

Set $\varepsilon \in (0,1)$, $\gamma > 0$

Set initial variables $\mathbf{q}^0 \in \mathbb{R}^N$

for $i = 0, 1, \ldots$ *until convergence* **do**

  $\mathbf{x}^i = \text{prox}_{\gamma f}\, \mathbf{q}^i$

  Set $\theta^i \in [\varepsilon, 2 - \varepsilon]$

  $\mathbf{q}^{i+1} = \mathbf{q}^i + \theta^i(\text{prox}_{\gamma g}(2\mathbf{x}^i - \mathbf{q}^i) - \mathbf{x}^i)$

return $\mathbf{x}^{i+1}$

Chambolle–Pock (CP) algorithm [37] is tailored to solve optimisation problems of type:

$$\text{minimise} \;\; f(\mathbf{x}) + h(L\mathbf{x}), \tag{1.33}$$

where functions f and h are convex and possibly non-smooth, and L is a linear operator.

The CP algorithm is primal-dual and iterative, and consists of two proximal steps related to functions f and h, combined with the application of L and $L^\top$.

The general form of the CP algorithm is introduced in Algorithm 3, where ζ, σ are positive scalars representing "step sizes", prox_{h^*} can be computed at virtually the same cost as prox_h thanks to the Moreau identity (see Eq. (1.27)) [36]. For $\theta = 1$, the convergence is ensured, when the following condition is satisfied [37]:

$$\zeta\sigma\|L\|^2 < 1, \tag{1.34}$$

where $\|\cdot\|$ denotes the operator/spectral norm. However, a more general algorithm appeared later in [32] that recovers the CP algorithm as its special case, while the step parameter could range to even $\theta \in (0, 2)$, not losing the convergence guarantee. The convergence speed can be significantly improved with a higher θ.

Algorithm 3: The Chambolle–Pock algorithm solving (1.33)

Input: Functions h, f, linear operator $L \in \mathbb{R}^{M \times N}$

Output: $\bar{\mathbf{x}}^{i+1}$

Set parameters $\zeta, \sigma > 0$ and $\theta \in [0, 1]$

Set initial primal variables $\mathbf{x}^0 \in \mathbb{R}^N$ and dual variables $\mathbf{q}^0 \in \mathbb{R}^M$

Set initial output variables $\bar{\mathbf{x}}^0 = \mathbf{x}^0$

for $i = 0, 1, \ldots$ *until convergence* **do**

$\mathbf{q}^{i+1} = \text{prox}_{\sigma h^*}\left(\mathbf{q}^i + \sigma L\, \bar{\mathbf{x}}^i\right)$

$\mathbf{x}^{i+1} = \text{prox}_{\zeta f}\left(\mathbf{x}^i - \zeta L^\top \mathbf{q}^{i+1}\right)$

$\bar{\mathbf{x}}^{i+1} = \mathbf{x}^{i+1} + \theta(\mathbf{x}^{i+1} - \mathbf{x}^i)$

return $\bar{\mathbf{x}}^{i+1}$

Forward-backward based primal-dual (FBB-PD) algorithm [36] is able to solve the optimisation problem of type:

$$\text{minimize} \;\; f(\mathbf{x}) + g(\mathbf{x}) + h(L\mathbf{x}), \tag{1.35}$$

where f is a smooth, convex and differentiable function having a Lipschitzian gradient with a Lipschitz constant $\beta < \infty$ [31, 36], functions g and h are convex and can be either smooth or non-smooth, and L is a linear operator.

The FFB-PD algorithm is primal-dual and iterative, and it is based on the forward-backward approach combining a gradient step (forward step) and a proximal step (backward step). Moreover, it is combined with the application of L and $L^\top$.

The general form of the FBB-PD algorithm is introduced in Algorithm 4, where ∇ stands for gradient and ζ, σ are positive scalars called "step sizes", Id stands for the identity operator and θ^i is the step parameter, which can vary within each iteration. However, θ stays fixed for all iterations in practice.

To ensure convergence, the following condition must be satisfied [36]:

$$1/\zeta \geq \frac{\beta}{2} + \sigma \|L\|^2, \tag{1.36}$$

where $\|\cdot\|$ denotes the operator/spectral norm.

Algorithm 4: The Forward-backward based primal-dual algorithm solving (1.35)

Input: Functions f, g, h, linear operator $L \in \mathbb{R}^{M\times N}$

Output: $\mathbf{x}^{i+1}$

Set $\zeta, \sigma \in (0, \infty)$

Set initial primal variables $\mathbf{x}^0 \in \mathbb{R}^N$ and dual variables $\mathbf{v}^0 \in \mathbb{R}^M$

for $i = 0, 1, \ldots$ *until convergence* **do**

  $\mathbf{r}^i = \text{prox}_{\zeta g}\left(\mathbf{x}^i - \zeta(\nabla f(\mathbf{x}^i) + \sigma L^\top \mathbf{v}^i)\right)$

  $\mathbf{q}^i = (Id - \text{prox}_{h/\sigma})(\mathbf{v}^i + L(2\mathbf{r}^i - \mathbf{x}^i))$

  Choose $\theta^i \in (0, \infty)$

  $\mathbf{x}^{i+1} = \mathbf{x}^i + \theta^i(\mathbf{r}^i - \mathbf{x}^i)$

  $\mathbf{v}^{i+1} = \mathbf{v}^i + \theta^i(\mathbf{q}^i - \mathbf{v}^i)$

return $\mathbf{x}^{i+1}$

Condat algorithm [32, 38] is tailored to solve optimisation problems of a very general form

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}) + \sum_{m=1}^{M} h_m(L_m\mathbf{x}), \tag{1.37}$$

where f is a smooth convex differentiable function and its gradient ∇f is β-Lipschitz continuous, g and h_m are convex functions, which can be either smooth or non-smooth, and the operators L_m are linear and bounded.

The Condat algorithm is primal-dual and iterative. Each iteration consists of proximal steps related to g and h_m, combined with the application of L_m and $L_m^\top$. The general form of the Condat algorithm is introduced in Algorithm 5, where ξ, σ and ρ are parameters of the Condat algorithm.

The convergence of the Condat algorithm is ensured when

$$\xi(\beta/2 + \sigma\|\sum_{m=1}^{M} L_m^\top L_m\|) < 1, \text{and } \rho \in (0, 1], \tag{1.38}$$

where β is the Lipschitz constant of f and $\|\cdot\|$ denotes the operator norm. In the case of $f = 0$, the convergence of the Condat algorithm is ensured when

$$\xi\sigma\|\sum_{m=1}^{M} L_m^\top L_m\| < 1, \text{ and } \rho \in (0, 2). \tag{1.39}$$

Note that it is allowed to set some functions to zero, and a linear operator L_m can be identity Id. Therefore, the Condat algorithm can be used for solving simpler kinds of optimisation problems. It is shown in [32] that the Douglas–Rachford and the Chambolle–Pock algorithms are special cases of the Condat algorithm.

Algorithm 5: The Condat algorithm solving (1.37)

Input: Functions $f, g, h_1, \ldots, h_M$, linear operators $L_1, \ldots, L_M$

Output: $\mathbf{x}^{i+1}$

Set parameters $\xi > 0, \sigma > 0$ and $\rho > 0$

Set initial primal variables $\mathbf{x}^0$ and dual variables $\mathbf{u}_1^0, \ldots, \mathbf{u}_M^0$

for $i = 0, 1, \ldots$ *until convergence* **do**

  $\bar{\mathbf{x}}^{i+1} = \mathrm{prox}_{\xi g}(\mathbf{x}^i - \xi \nabla f(\mathbf{x}^i) - \xi \sum_{m=1}^{M} L_m^\top \mathbf{u}_m^i)$

  $\mathbf{x}^{i+1} = \rho \bar{\mathbf{x}}^{i+1} + (1 - \rho)\mathbf{x}^i$

  for $m = 1, 2, \ldots, M$ **do**

    $\bar{\mathbf{u}}_m^{i+1} = \mathrm{prox}_{\sigma h_m^*}(\mathbf{u}_m^i + \sigma L_m(2\bar{\mathbf{x}}^{i+1} - \mathbf{x}^i)$

    $\mathbf{u}_m^{i+1} = \rho \bar{\mathbf{u}}_m^{i+1} + (1 - \rho)\mathbf{u}_m^i$

  return ${\mathbf{u}_m}^{i+1}$

return $\mathbf{x}^{i+1}$

2 SELECTED SIGNAL PROCESSING TOPICS

This chapter provides a basic overview of the signal-processing topics used in this Thesis. It focuses on the basic presentation of selected parts of the image processing methods, which relate to this Thesis: segmentation, edge detection and enhancement, and denoising/filtering.

2.1 Digital signal processing

Technically, a signal is defined as a function of time, space or another variable. As a signal, it can be considered, for example, a sound, an image, an electric field, etc.

The basic characterisation of signals is based on the properties of the signal (discrete/continuous) in the time domain and in its amplitude:

- analogue signal – continuous in time domain and in its amplitude,
- quantised signal – continuous in time domain and discrete in its amplitude,
- sampled signal – discrete in time domain and continuous in its amplitude,
- digital signal – discrete in time domain and in its amplitude.

The real world is full of analogue signals, so it comes as no surprise that people interact with them on daily basis. Analogue signals carry information about electrical, mechanical, acoustic or physical magnitudes. Often, the real world signals need to be processed. Nowadays, the processing is done via many different sophisticated programs/algorithms. The field dealing with it is called digital signal processing (DSP) and it is done either via computer or digital signal processors, which are used to analyse, modify, optimise, correct, reconstruct, restore or extract information from digital signals.

DSP is used in many fields of our lives: telecommunication (voice transmission, signal compression), commerce (voice and image processing, video effects), industry (process monitoring and control, CAD systems), army (RADAR, SONAR, secured connection), science (monitoring, data analysis, simulation, modelling), medicine (imaging systems (Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound (US)), ECG and EEG analysis), etc.

Digital signal processing methods are capable of processing only finite number of discrete values. To be able to process analogue signals via DSP, it is necessary to convert them into their digital form. A digital signal is usually represented as an array of numbers, e.g. one-dimensional signals are considered as vectors, two-dimensional signals as matrices, etc.

The analogue signal is simply defined as a function of the continuous variable $y(t)$, where y represents the signal and t is the continuous variable representing time, for example. The process of converting the analogue signal to its digital form is

called digitisation. The digitisation process consists of two main steps: spatial/time domain sampling and amplitude quantisation.

Sampling means that the values of the continuous-time signal are sampled in regular time intervals, and the discrete-time signal consists of these sampled values (see Fig. 2.1). The sampled signal is a discrete-time representation of the analogue signal, and it is denoted as $y[n]$, where n is a discrete variable representing the signal samples. The relationship between t and n can be expressed by an equation $t = nT_\mathrm{s}$, where T_s represents the sampling period. The relationship between sampling frequency f_s and sampling period T_s is defined as $f_\mathrm{s} = 1/T_\mathrm{s}$. The analogue signal can be well reconstructed from the sampled signal if the sampling was done under specific condition. The sampling condition is called the Nyquist sampling theorem, which is defined as

$$f_\mathrm{s} > 2f_\mathrm{max}, \tag{2.1}$$

where f_max is the highest frequency contained in the analogue signal. If the sampling theorem is not satisfied, the aliasing appears in the reconstructed signal. Aliasing is an unrecoverable distortion. In the case of images, it is represented by "moiré" pattern in the reconstructed 2D signal. Sometimes the technical equipment can not satisfy the requirements of the sampling frequencies, i.e. the processed signal contains frequencies higher than the equipment is capable of processing due to its limited sampling frequency. In these cases, the low-pass antialiasing filtering is used to remove the frequencies above the Nyquist limit, which, of course, leads to the loss of the signal details. However, this loss caused by low-pass antialiasing filtering is less damaging than aliasing itself. [39–41]

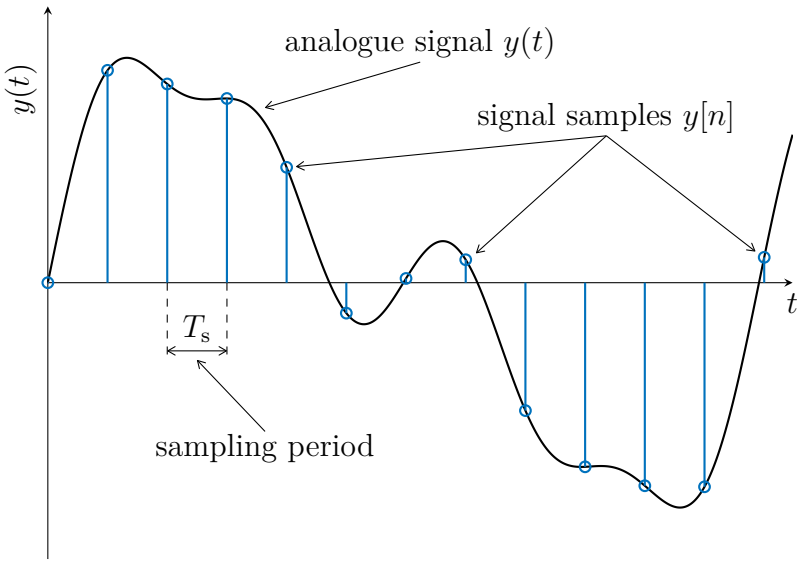


Fig. 2.1: Example of signal sampling.

The next step of digitisation is amplitude quantisation of the sampled signal (see Fig. 2.2). The quantisation is a process, where the continuous-amplitude signal is converted to the discrete-amplitude signal by expressing each sample value to the nearest quantisation level defined by the finite number of the digits, which is usually determined by the number of bits B representing the sample value [40, 41]. This process leads to an unrecoverable non-linear distortion, which is called quantisation error or quantisation noise. Quantisation error $e_{\mathrm{q}}[n]$ is defined as a difference between the sampled $y[n]$ and quantised value $y_{\mathrm{q}}[n]$:

$$e_{\mathrm{q}}[n] = y_{\mathrm{q}}[n] - y[n]. \tag{2.2}$$

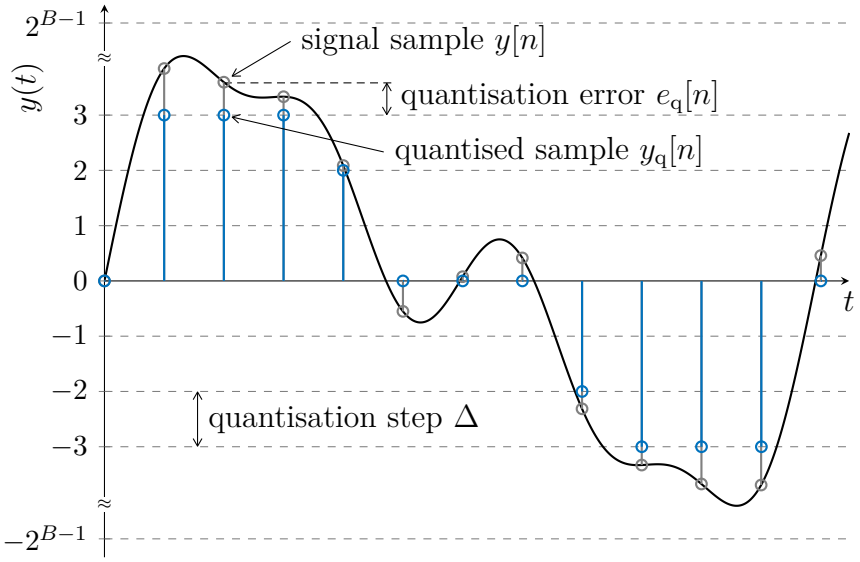


Fig. 2.2: Example of sampled signal quantisation.

Signals can be also divided based on their dimension. One-dimensional (1D) signals $\mathbf{y}[n]$ can represent speech signal, measurement of the temperature, commodity price development, ECG signal, etc. Two-dimensional (2D) signals $\mathbf{Y}[m, n]$ can represent images, like photographs, medical images, etc. Three-dimensional (3D) signals $\mathbf{Y}[m, n, o]$ can represent CT scans or video – "2D image changing in time".

There are several types of images:

- Binary image $\mathbf{Y}[m, n]$ – consists only of pixels of zeros or ones. Often this type of image is used for identifying the object and the background.
- Grayscale image $\mathbf{Y}[m, n]$ – consists of values in a certain range that represent a shade of gray. For example, using 8 bits to represent the shade produces a range from 0 to 255.
- Colour image $\mathbf{Y}[m, n]$ – represented via colour space. The most commonly used colour space is RGB $\mathbf{Y}_{\mathrm{RGB}}[m, n]$, which is composed of three colour channels $\mathbf{Y}_{\mathrm{R}}[m, n]$, $\mathbf{Y}_{\mathrm{G}}[m, n]$, $\mathbf{Y}_{\mathrm{B}}[m, n]$.

2.2 Image segmentation

Image segmentation is a part of the image analysis. Segmentation is used to split the image into objects (segments) based on the characteristics of the pixels, for example. Formally, image segmentation is defined as splitting of the image $\mathbf{Y}$ into segments $\mathbf{Y}_1, \ldots, \mathbf{Y}_S$, where S is the number of segments. These segments are disjoint $\mathbf{Y}_i \cap \mathbf{Y}_j = \varnothing, i \neq j$, and cover the whole image $\mathbf{Y} = \bigcup_{i=1}^{S} \mathbf{Y}_i$ [41].

Therefore, due to segmentation, the representation of the image is viewed as a smaller group of segments instead of the individual image pixels. This helps to analyse the image and its objects by another image analysis techniques. People focus on obtaining different kinds of information from the analysed image, and for each purpose it is necessary to define such a segmentation task that allows them to obtained the needed information. Since high variability of segmentation tasks exist, there is no unitary approach to achieve the best segmentation. Therefore, many segmentation methods and approaches were developed.

Image segmentation techniques can be divided into several classes [41]:

- Homogeneity of areas based segmentation – it is supposed that an image segment is homogeneous with respect to the chosen parameter, which can be described by a scalar or by a vector of high dimensionality (intensity, texture). The simplest segmentation method, thresholding, belongs to this class.
- Region-based segmentation – these methods directly detect the region instead of the edges. This approach is suitable for noisy images. The main segmentation criterion is homogeneity of the region. Homogeneity criterion of the pixels in the region can be tied to the parameters like colour, texture, shape, etc. To this class belong methods like region growing [42], region merging [43], region splitting and merging [44], and watershed [45].
- Edge-based segmentation – it is based on the edge representation of the segmented image and provides borders of the image segments. The edge-based segmentation uses also the idea of homogeneity of segments. However, no strict requirements to the homogeneity are required. The only "strict" requirement is on small changes inside the regions. The ideal detected borders are continuous and closed, reliable, and describe the segments with good accuracy of localisation. Nevertheless, the raw edge representation obtained by basic techniques is far from ideal. Typically, the edges are disconnected, thick and do not correspond to segment borders, which is often caused by noise in the image.
- Segmentation by pattern comparison – this approach searches for the chosen pattern in the image. Since the exact fit of the pattern and the image can not be expected, the similarity criteria are used.

- Segmentation via flexible contour optimisation – typically used in segmentation of medical images, which are noisy and textured, and therefore, the above-mentioned approaches do not work very well. This group includes methods such as parametric flexible contours, geometric flexible contours, and active shape contours [46].

There are some techniques, which can deal with the raw non-ideal edge representation and derive as many "ideal" borders as possible. Among such methods are borders via modified edge representation [47], borders via Hough transform [48], boundary tracking [49], and graph searching methods [50]. Starting point of all these methods is the raw edge representation. Therefore, the success of these approaches is closely related to the quality of the raw edge representation. The better the raw edge representation, the better the edge-based segmentation. [41]

2.3 Edge detection and enhancement

Signal enhancement is viewed as a process of improvement of the signal with respect to some of its properties, where the input is the "imperfect" signal and the output is the "improved" signal. In connection with image edges, the signal enhancement usually consists of sharpening.

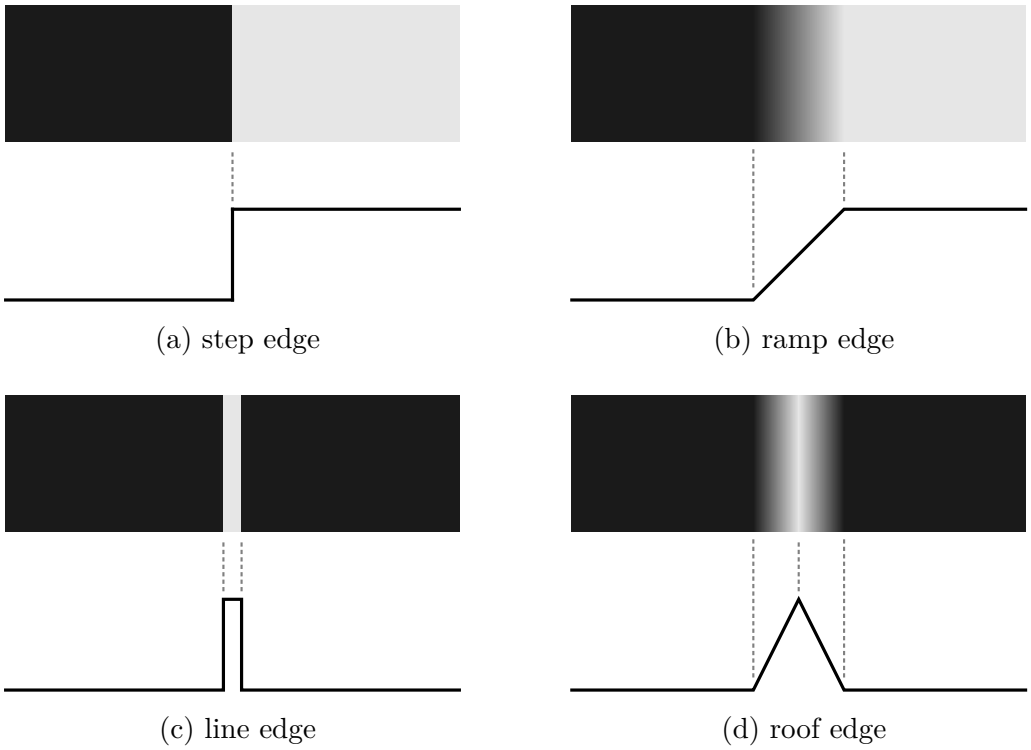
The edge detection, on the other hand, is a part of image analysis. The goal of the image analysis, in general, is to provide a description of the analysed image.

Both edge sharpening and edge detection are based on a relation of the analysed pixel and its neighbourhood.

2.3.1 What is an edge in the image?

An edge can be characterized as a position in the image where is a significant local change in the image intensity. A stronger change of the intensity causes clearer edge, which can be detected more easily and more precisely. Edges provide important visual information, for example each object can be easily described with few key edges and, therefore, edges are important for human sight. [41, 51]

Several types of edges can be recognised: step edge, ramp edge, line edge and roof edge (see Fig. 2.3). The ideal edge, from the signal processing point of view, is the step edge, where the intensity changes from one value to another immediately. A more common in real world is the ramp edge, where the intensity changes from one value to another gradually. Another ideal edge is the line edge, which differs from the step edge in the prompt return to its initial value. However, the roof edges are more typical than line edges for the real images. [52]



(a) step edge (b) ramp edge

(c) line edge (d) roof edge

Fig. 2.3: Examples of different edge types.

A step edge can be detected by the gradient of intensity function of the image. Step edges are localised as positive maxima or negative minima of the first-order derivative or as zero-crossings of the second-order derivative (see Fig. 2.4). The edge can be described by the strength and the direction. Both information are contained in the gradient function. [41, 51]

The first derivative of the edge-profile function results in the highest change of local extrema at the position where the edge is. On the other hand, the second derivative of the edge profile results in zero-crossing at the edge position. Nevertheless, derivatives can not be applied to the digital images. Therefore, the first and the second derivatives are approximated by directional differences. These differences can be expressed by local difference operators [41]. See examples of the operators of the first and second differences for a horizontal and vertical direction below:

$$\begin{aligned} \Delta_x H &= \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \Delta_y H = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \\ \Delta_x^2 H &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \Delta_y^2 H = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{2.3}$$

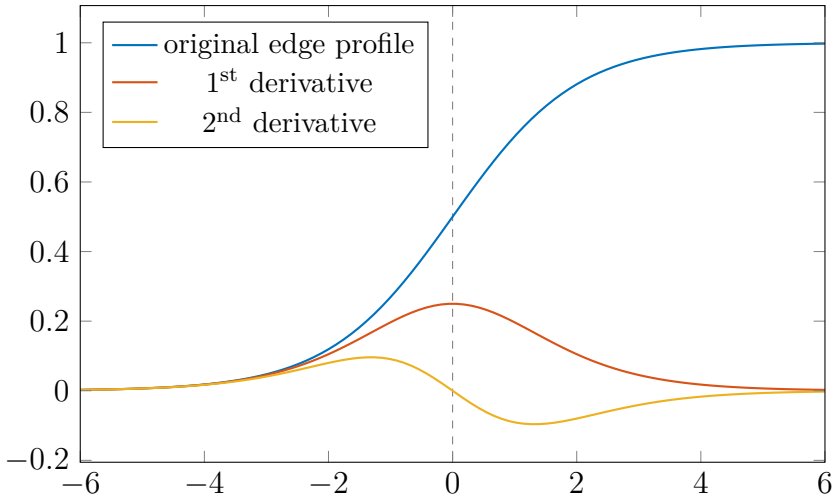


Fig. 2.4: Example of edge localisation via first-order and second-order derivative.

2.3.2 Edge sharpening

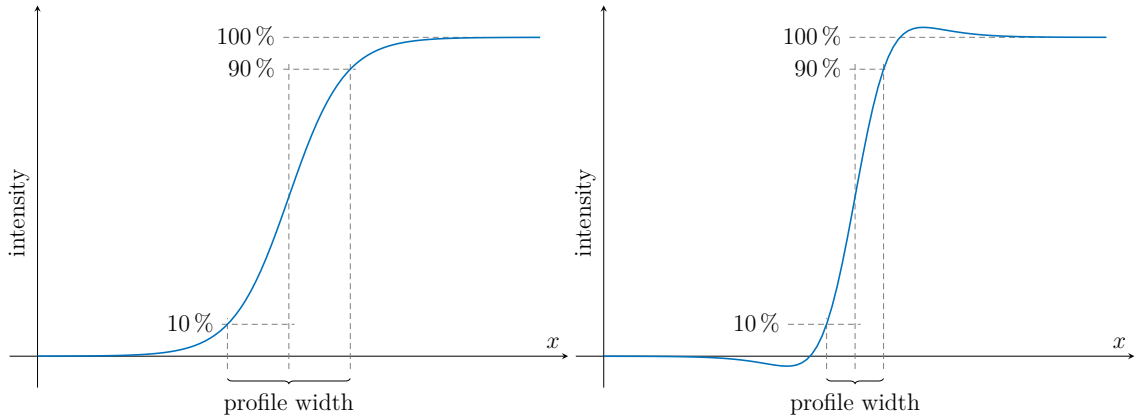
Sometimes, the image edges are too blurry for human sight, making it difficult to perceive finer details in the image. The process of making the edges more visible is called edge sharpening (see Fig. 2.5), which is a frequent task of the image enhancement. Note that sharpening can only enhance the details in the image but cannot improve the resolution of the image. Sharpening generally requires a relative accentuation of the high-frequency components of the image. There is a lot of methods to make edges more visible, for example Sharpening via subtracting of the blurred image, Local sharpening masks, Sharpening via Frequency domain, or Adaptive sharpening. Unsharped/blurred image has a relatively higher content of low-frequencies than sharp image. Therefore, sharpening can be described as augmenting the high-frequency image components. [41, 51]

Sharpening via subtracting

The aim of the sharpening via subtracting of the blurred image is to relatively decrease the low-frequency components of the image. At first, the original image is blurred via an averaging mask. Then, the blurred image is subtracted from the original image to obtain the image with details. Finally, the detailed image is added to the original one leading to a sharpened image. [41]

Local sharpening mask

Using the local sharpening masks is another approach to sharpen an image. There are two ways how this approach can be achieved. The first way is to sharpen the image by adding the Laplacian-filtered image to the original. The Laplacian-filtered



(a) edge profile before sharpening (b) edge profile after sharpening

Fig. 2.5: Example of edge profile before and after edge sharpening.

image is obtained by applying the Laplace mask on the original image. The Laplace operators, which are second-order derivative masks, are defined as follows:

$$H_4^{\mathrm{L}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad H_8^{\mathrm{L}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad H_{12}^{\mathrm{L}} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{2.4}$$

where the upper index L denotes the Laplace mask and the bottom index represents the central value of the mask. Convolution of the image with the local Laplace mask leads to an image with maximum sharpness at the cost of losing the area intensity information. The sharpened image is then obtained by adding this extra sharp image to the analysed one.

The second possibility is to create sharpening masks, which are applying both operations (Laplacian filter and Adding the analysed image) at the same time:

$$H_5^{\mathrm{L}} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{and} \quad H_9^{\mathrm{L}} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{2.5}$$

The output of applying the sharpening filter is the sharpened image itself.

In general, larger sharpening mask leads to an increase of the computational complexity. In this case, it is preferable to use the convolution properties of the Fourier transform. Instead of using the convolution of the image with the mask in the original domain, multiplication of the image spectrum with the frequency characteristic of the mask in the spectral domain is used. [41]

Sharpening via frequency domain

Sharpening via frequency domain is a similar approach to the above-mentioned method. The main operation of this approach is a modification of the image spectrum. Low-frequency image components are forced to zero in the frequency domain, which leads to the detailed image, which is then added to the original image. [41]

Sharpening and noise

Since sharpening relatively increases the high-frequency components of the image, and the image noise has a relatively higher share of higher frequencies, sharpening leads to an increased level of the image noise. The image can be degraded even if the values of the noise in the original image were not high. Note that the noise is more disturbing for the human sight in flat intensity regions than at highly structured areas (edges, small details), which is exploited in a process of adaptive sharpening. [41]

Adaptive sharpening

The principle of the adaptive sharpening is to apply the sharpening only in structured areas, and leave the flat areas untouched. Before applying the sharpening algorithm, it is necessary to distinguish the structured areas from the flat ones. This recognition is done via the application of the local image analysis and subsequent evaluation if the criterion has been met or not. Based on these results, the sharpening is either applied or not. [41]

2.3.3 Edge detection

Edge detection can be interpreted as a transformation of the grayscale or colour image to the binary image, where white pixels represent the edge position and black pixels represent the background or vice versa. Finding the place where the change of the image intensity is strong, is the appropriate way how to find the edges. Many approaches to the edge detection were designed, which will be discussed in more detail in Chapter 3. Nonetheless, an easy and very common approach to the edge detection is application of the edge operators (see Secs. 3.1 and 3.2). [41]

Edge strength and orientation

An edge can be described by its strength and orientation. Edge strength and orientation are computed from the results of the convolution of the original image with the gradient masks for each image position. [41, 51]

Edge detection and noise

The result of the edge detection is significantly affected by noise in the image, which usually causes the detection of false edges. This phenomena is naturally more significant with stronger noise and in the case of simple edge detection methods. These false detected edges should be excluded via post-processing, which takes image context into consideration. [41]

Edge maps

The result of the edge detection approach is full of edge candidates. The decision on which candidate is truly an edge and which is not is a very important part of the post-processing. The simplest method to decide this is to apply thresholding to the edge strength. The threshold value can be fixed or adaptive. The decision process leads to the binary image – so-called edge map. [51]

2.4 Denoising/Filtering of signals

Denoising or noise smoothing can be viewed as a signal enhancement, which focuses on the compensation of the signal imperfections caused by the noise, providing the assumption of the original "clean" signal [41]. The sharpening and denoising are conflicting operations and, therefore, it is important to be familiar with the consequences of using one or the other method. Similarly to sharpening, noise smoothing is realised mostly by local operators (masks) – linear or non-linear and space invariant or adaptive.

An image can be corrupted by different kinds of noise. The origin of the noise plays an important role in choosing the suitable noise-suppressing method.

2.4.1 Types of noise

Noise can be classified into several classes according to different properties [41].

- According to the dependence on the image content, image-dependent and image-independent noise are recognised. Since the image-dependent noise is difficult to deal with effectively, handlig image-independent noise is prevalent.
- According to noise amplitude distribution and spatial distribution, the gray (or white) noise and the impulse noise are identified.
- According to the noise relation to the image content, it is possible to recognise the additive and multiplicative noise. Additive noise is simply added to the original image compared to multiplicative noise, where each pixel is multiplied

by the noise amplitude. From these two types, the additive noise is a more frequently encountered type of noise in most of the image analysis applications.
- According to noise character in the frequency domain, wideband noise and narrowband noise are distinguished. The narrowband noise is characterised by the fact that it occupies only a limited range of frequencies in the frequency domain. Wideband noise, however, is the most common case.

Additive White Gaussian Noise (AWGN)

Additive White Gaussian Noise is one of the most common types of noise. It is a type of random noise that can be added to a signal in an acquisition system, communication/transport channels, data quantisation during the analogue-to-digital conversion, etc.

"Additive" refers to the fact that the noise is added to the signal, rather than being inherent in the signal itself. "White" means that the noise has a flat frequency spectrum, meaning that it has a uniform power across the whole frequency band, i.e. it is equally present at all frequencies. In images, it is also assumed that the white noise corrupts all pixels within the image. Finally, "Gaussian" refers to the statistical distribution of the noise. As a Gaussian noise, it is typically considered noise with zero mean value and intensities that are substantially smaller than the maximum intensity of the image pixels. Since the spectrum of AWGN noise is uniform, which means that all frequencies are corrupted identically, the AWGN noise is considered as wideband noise. [41]

2.4.2 Noise-suppressing methods

Application of the noise-suppressing methods leads to the loss of the sharpness/image details. The reason is that noise occupies the same frequency range as the image details. The noise suppressing methods can be divided into three main classes according to the type of the noise they deal with – narrowband noise suppression, wideband noise suppression and impulse noise suppression [41]. Since the Thesis deals with the wideband noise, for the sake of brevity, only the wideband noise suppression method will be introduced.

Wideband noise suppression

As mentioned earlier, the wideband noise corrupts all of the pixels in the image. Since the values of the noise are not known, it is supposed that they are random; in contrast to the image, where the values are supposed to be fixed (they do not change in time). Therefore, if there are several realisations of the same image with different

realisations of the noise, the basic idea is to average all the image realisations. The result of such an approach improves with the increasing number of the image realisations. Unfortunately, this approach can be used only when there is the possibility to take several realisations of the identical scene, which is not the usual situation. More often it happens that only a single realisation of the image is available.

In the latter case, the realisations of the noise are obtained from the analysed pixel surroundings [41]. The analysed pixel and its surrounding is defined by the local operator and, therefore, this approach is called local averaging. The principle of the local averaging is to replace the analysed pixel with average of the values according to the local operator. This approach can be viewed as the convolution of the image with the mask operator.

Local operator can have different shape, size and weights and is invariant. Since the weights defined in the operator can be different, the resulted average can be weighted. Averaging is the linear operation. Usually, the operators of the size 3×3 are used, examples of some mask operators can be seen below:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2.6}$$

The first mask operator represents a simple average, which leads to the best SNR improvement, however, it causes the highest blurring. The second and third masks represent the weighted average. These two operators emphasise the influence of the central pixel, which is analysed. The third mask causes the smallest amount of blurring of the three.

Local averaging is suitable for the use in flat image areas and unsuitable for the use in areas with edges, where it causes blurring. If high noise suppression is needed, larger masks (up to 9×9) are used. Unfortunately, high suppression also causes strong blurring.

To avoid blurring, the adaptive noise suppression can be used. For human sight, the noise is more disturbing in flat areas than in areas with details. Therefore, the adaptive approach focuses on high noise suppression in flat areas, and lower or no suppression in areas with details. For the adaptive noise suppression, the recognition of the flat area of the image plays a crucial role. The adaptive noise suppression can be considered as the opposite operation to the adaptive sharpening. For more details see [41].

3 HISTORY AND STATE OF THE ART

The edge detection finds applications in many fields such as image segmentation, medical data processing, object tracking, motion detection, pattern recognition, etc. As mentioned earlier, an edge is defined as a position where the intensity of an image changes significantly. However, the estimation of edge significance differs from application to application. Therefore, different approaches are suited for each application. The edges in the image are not easy to locate accurately. The reason is that real images are corrupted by noise, by uneven light intensity, and by defects or imperfections in the imaging technology [53]. To eliminate the above-mentioned issues, the authors of [54] suggest that edge detection methods should include three basic operations: denoising, differentiation, and labelling. According to [55], computer vision depends on correctly identified edges at the right positions, ensuring they are continuous, and maintaining uniform width. In this section, the different edge detection approaches will be described.

The authors of [53] divide edge detection methods into three domains: spatial domain, frequency domain, and wavelet domain. In the spatial domain, gradient operations are performed directly on the pixels. A lot of methods presented in this chapter focus on detection in the spatial domain. For frequency domain detection, it is necessary to convert the image to the frequency domain before applying various operations, such as phase congruency [56].

In the wavelet case, the image is transformed into partial multi-frequency levels. Low frequencies are used to extract the coarse or overall structure of the image, and high frequencies are used to extract contours. Multiresolution analysis [57] is important for the contour detection. The different approaches to contour detection are summarised in [58]. As an example of contour detection approaches, it is possible to mention the active shape model, where the boundary of an object is detected by finding the optimal position of the active shape model points. In cases where very accurate edge detection is required, three different subpixel methods have been developed: curve-fitting (sensitive to noise), partial area effect (computationally expensive), or moment-based methods (computationally expensive).

The authors of [53] divide edge detection methods into two main groups: classical edge detection technology and deep learning-based edge detection technology.

The group of classical edge detection includes pixel-based edge detection technology (which contains gradient-based, wavelet transform-based, fuzzy theory-based, and genetic theory-based edge detection methods) and edge detection methods based on sub-pixel level (which includes matrix-based, interpolation-based, and fitting method-based edge detection methods).

The group of deep learning-based edge detection includes spectral clustering-

based, cross-layer multiscale fusion-based, and encoding and decoding-based edge detection.

The authors of [53] provide references to individual papers that use methods from the categories for edge detection described above, and they also mention the advantages and disadvantages of the presented types of methods. They suppose that what will researchers surely focus on is real-time edge detection.

Categorisation of methods is always a difficult task, and as mentioned above, edge detection methods can be categorised into several different groups based on various factors and points of view. In the rest of the chapter, the methods are coarsely divided into 11 sections according to the main principle of the methods, sorted roughly from the simplest methods to the more complex ones.

3.1 Gradient-based edge detection techniques

Gradient-based (first-order derivative) edge detection is considered a basic and simple edge detection method. First derivative of the edge profile results in the highest change of local extrema at the position where the edge is. The first derivative is approximated by directional differences (see Sec. 2.3.1) for using on the digital images. The basis of this method is a local operator.

The gradient-based edge operators evaluate the value of intensity change in the pixel neighbourhood. Examples of the well known edge operators are described below. Edge operators usually consist of two or more edge masks, where each mask detects only edges of a particular direction. Convolution of the image with each mask provides an image of potential edges – the so-called parametric image. To get the edges of all directions, it is necessary to compose the particular parametric images, which emphasise different edge orientations.

There are several possibilities for how to compose the resulting image of edges – compute the Euclidean distance, choose the maximum value of particular parametric images, or sum the absolute values of parametric images. The resulting edges are relatively thick and should be post-processed by thinning. Also, for each detected edge, its local direction can be computed. [41, 51]

Roberts operator is one of the simplest and oldest edge operators. This operator consist of two diagonal masks H_1^{R}, and H_2^{R} of size 2×2, defined as follows [51]:

$$H_1^{\mathrm{R}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad H_2^{\mathrm{R}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{3.1}$$

Prewitt, Sobel and Kirsch operators approximate the first derivative using differences [51]. All these operators are directional, which means that they detect

edges only in a specific direction. The detection in other directions is very low or none. The basis of each operator consists of two directional masks – one for the horizontal and one for the vertical direction. Prewitt operator uses the masks H_x^{P}, H_y^{P}, which compute the average gradient components across neighbouring lines or columns, defined as follows:

$$H_x^{\mathrm{P}} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^{\mathrm{P}} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{3.2}$$

Sobel operator consists of masks H_x^{S}, H_y^{S}, which are similar to Prewitt masks, with the difference that Sobel masks assign higher weight to the central line or the column, defined as follows:

$$H_x^{\mathrm{S}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^{\mathrm{S}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{3.3}$$

Kirsch operator consists of two masks H_x^{K}, H_y^{K} defined as follows:

$$H_x^{\mathrm{K}} = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix} \quad \text{and} \quad H_y^{\mathrm{K}} = \begin{bmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}. \tag{3.4}$$

Compass detectors consist of the set of eight directional masks. All Prewitt, Sobel and Kirsch operators can be extended to compass detectors, which are less sensitive to the edge orientation [41, 51]. Each detector consists of eight masks with orientations spaced at 45°. See example of four directional masks H_0^{S} to H_3^{S} of the Sobel compass detector:

$$\begin{aligned} H_0^{\mathrm{S}} &= \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_1^{\mathrm{S}} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \\ H_2^{\mathrm{S}} &= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad H_3^{\mathrm{S}} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}. \end{aligned} \tag{3.5}$$

The remaining four masks are the negatives of the first four. Kirsch and Prewitt compass detectors are created in a similar manner.

The advantages of gradient-based methods include, among others, their simplicity, speed, and ease of implementation. Another advantage is that edges are detected together with their orientation. On the other hand, the disadvantages include sensitivity to noise and inaccurate edge detection. [59, 60]

3.2 Laplacian based edge detection techniques

Laplacian based edge detection techniques use the second-order derivative. The second-order derivative of the edge profile results in zero-crossing at the edge position. Therefore, unlike the first-order operators, the edge is not described by a high value but by a zero. As with the first derivative, the second derivative is approximated by directional differences in mask form (see Sec. 2.3.1) for the use on digital images. Laplacian operator approximates the second derivative. Unfortunately, Laplacian itself is very sensitive to noise. Therefore, the Laplacian is not used alone but in combination with a smoothing filter. [41]

3.2.1 Laplacian of Gaussian

The first to combine the Laplacian operator and the Gaussian filter was Marr and Hildreth in 1980 [61], who introduced the so-called Laplacian of Gaussian (LoG). First, the analysed image is smoothed using a Gaussian filter, and then the Laplacian operator is applied to the smoothed image. There is no need to apply the individual steps sequentially. The combination of the Gaussian operator and Laplacian operator leads to the Laplacian of Gaussian (LoG) operator H^{LoG} defined as

$$H^{\mathrm{LoG}} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \tag{3.6}$$

Convolution of the image with the LoG operator produces the image where zeros denote edge positions. To get only the positions of edges, it is necessary to provide some post-processing of the LoG image. The post-processing can be provided by using the 3×3 "cross" mask

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & \times & 1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{3.7}$$

which considers 4 closest neighbours of the analysed pixel. The pixel is marked as edge, if the following conditions are satisfied:

- at least one of the considering neighbours has a different sign than the others,
- the value difference between two extreme differently-signed neighbours is greater than a chosen threshold,
- the value of the analysed pixel lies in between the values of the extreme differently-signed neighbours.

If at least one condition is not satisfied, the analysed pixel is marked as non-edge (background). Edges produced by the LoG operator are thin and usually more precisely localised than edges produced by operators based on the first derivative [41].

The advantage of LoG is an accurate edge detection. The disadvantage is that some edges may not be found [59].

3.3 Canny operator

The Canny detector is one of the most widely used edge detectors in image processing and it is still often considered state of the art. In practice, it is often used as a standard image preprocessing technique [62]. The Canny detector was first presented in 1986 by Canny [63]. The Canny detector was inspired by the use of the Gaussian operator presented in a paper [61] on Marr-Hildreth edge detection [53].

The Canny detector consists of 4 steps: noise suppression/smoothing, gradient computation, non-maximal suppression, and hysteresis thresholding [62]. A Gaussian filter is used for noise suppression. The gradient is computed by convolution with gradient operators in horizontal and vertical directions. Non-maximum suppression consists in suppressing the points that do not have a maximum value in the gradient direction. Hysteresis thresholding requires setting two thresholds – high and low. When gradient magnitude values are greater than the high threshold, they are considered to be edges, values less than the low threshold are suppressed, and if the value lies between the high and low thresholds, the neighbourhood of the analysed point is evaluated to determine whether it is part of an edge or not.

The Canny detector has the advantage of good detection and accurate localisation, robustness to noise, and good edge-tracking capability in various directions. The Canny detector is able to detect long edges, which is positive in subjective evaluation [62, 64]. The disadvantages may be that it is more computationally intensive than using simpler edge operators and that the choice of thresholds affects the accuracy of edge detection [65]. Also, the Canny detector is designed primary for detection of step edges.

3.4 Oriented energy approach

In the late 1980s and 1990s, the development of edge detection dealt with the use of a family of filters of different orientations and scales. Such filtered images provided more information for proper edge detection. This approach is called the oriented energy approach. The authors of [66] and [67] use quadrature pairs of odd and even

symmetric filters. The paper [68] describes and proposes a filter with automatic scale selection.

The paper [66] mentions that methods whose decision phase follows the application of linear filtering introduce systematic errors in the localisation of composite edges (the Canny detector also faces this problem). A composite edge is a combination of several different profiles, e.g. lines, steps, and roofs. If quadratic filtering is used instead of linear filtering, this problem disappears. Perona and Malik use image convolution with pairs of odd and even quadratic filters to detect edges in an image. If a point in the filtered image satisfies a set of rules, it is marked as an edge. In the paper, a comparison of this method and Canny detector is shown, where the example shows that the Canny detector breaks the edge connections.

The paper [67] presents an architecture to synthesise an arbitrarily oriented filter based on a linear combination of basis filters. These filters are denoted as directed filters. The concept of forming steered filters can be extended from 2D to 3D. The steered filters can be formed in quadrature pairs. In this paper, an experiment on edge detection at the circle and the square is performed, comparing the Canny detector with a method using steered filters. Steered filters give good edge detection for both the circle and the square. Unlike the Canny detector, which detects the circle well but detects double edges in the square.

The problem with the edge detectors described earlier is determining how much smoothing to choose in the smoothing stage. Greater smoothing yields higher noise suppression, which makes edge detection easier, but this come at the cost of poorer localisation. Less smoothing will improve edge localisation at the cost of a worse signal-to-noise ratio. In other words, the scale level affects the result of edge detection. In general, different parts of the image need different scale levels. Therefore Lindeberg [68] proposed an edge detection method with automatic selection of the scale level. For this method, it was necessary to introduce a new concept of scale-space edges, for which (i) the magnitude of the gradient reaches a local maximum in the gradient direction and (ii) the normalised strength of the edge response is locally maximum in the whole scale. This definition allows the scale level to vary along the edge. To achieve a trade-off between edge detection and right edge localisation, the detector needs to allow the scale level to vary over the image. By automatically selecting the scale level, the need for an external scale level choice is eliminated. The proposed method first detects scale-space edges and then ranks them according to their significance based on predefined rules. It is then up to the user to decide how many of the most significant edges to display. The proposed method gives better results than extracting edges at a fixed scale level since the analysing system does not know which scale level will be best for a given image.

3.5 Use of texture, brightness and colour features

Another direction in local edge detection is to use combinations of coloured and texture information (features) of the image to learn classifiers. The papers mentioned below point out that methods based only on local features, such as the Canny detector, do not take into account context, mid-level or high-level information, or local features at multiple scales, which is important for good detection and localisation of the edges. This approach is utilised, for example, by [69–71]. For all the papers mentioned, the Berkeley image dataset [72] were used with manually drawn boundaries for learning purposes.

The paper [69] states that an edge in a natural image is characterised by several changes in properties such as colour, brightness, and texture. Therefore, the paper focuses on extracting selected local image features for each image patch. These features are oriented energy, brightness gradient, colour gradient, and texture gradient. The extracted features then serve as an input to a classifier that determines the posterior probability that the analysed pixel is an edge or not. For optimal classifier setup and feature extraction, train data with manually labelled boundaries in the image are used. The proposed method (called Pb) achieves better results than the Canny detector. The Canny detector fails to detect boundaries between texture regions and falsely detects edges within these regions. This is because it only focuses on the brightness change in the image.

The author of [73] focused on creating a classifier that combines features extracted from different scales (different image patch sizes). Using a large scale, the detection is robust but has poor localisation and suppresses details, while using a small scale preserves details but is sensitive to noise and texture. This paper discusses how to properly combine the acquired features to make edge detection as good as possible. The author uses a combination of multi-scale features obtained from the local operators developed in [69]. The features obtained in this way are extended with a localisation feature for each scale (describing the distance of the pixel to the nearest response peak) and a relative contrast feature. The final classifier is then trained on these features. According to the author, the combination of multi-scale features compared to the use of single-scale features yields a 20 % – 50 % improvement in edge detection. The proposed classifier gives better results than the Pb classifier presented in [69].

In paper [71], the authors propose a novel supervised learning algorithm for edge and object boundary detection, which they refer to as Boosted Edge Learning (BEL). The edge decision is made independently of the location in the image and is based on a large number of extracted features. It is a classifier that learns on a large number of features (tens of thousands) extracted across different scales. The features

for each pixel are obtained by analysing its neighbourhood-image patches of large size (50×50), where the analysed pixel is in the patch centre. For example, Haar wavelets are also used to extract the features. In the case of colour images, features are extracted for each colour channel. A Probabilistic Boosting Tree classification algorithm [74] was used for the learning process. The whole learning process is highly adaptive, so there is no need to tune any parameters. The output of the proposed method is the edge probability. In decision-making, the method combines low and mid-level information with context information across different scales. The authors mention that the classification phase is more challenging and that the selection of the discriminative algorithm is more sensitive as a disadvantage. The proposed method gives similar results to the Pb algorithm presented in [69].

The paper [70] presents a multiscale discriminative framework based on learned sparse representations and its application to edge detection and class-specific edge detection. It has been trained and evaluated on the Berkeley dataset. First, the Canny detector was applied, and its output was compared with manually segmented images. Based on this comparison, the pixels were divided into two categories: namely those that are close to the manually segmented edges and those that are not. Subsequently, 14 local classifiers were independently trained using different image patch sizes and two resolutions. The outputs of the classifier were divided into two dictionaries, one for edges and one for non-edges. For both dictionaries, the reconstruction error curves were computed as a function of the sparsity constraint. The resulting linear logistic classifier was then trained based on the curves thus obtained. The proposed method gives similar results to the algorithms presented in [69] and [71].

3.6 Contour grouping methods

Processing the outputs of the local operators to obtain smooth contours is another direction in edge detection methods. Contour grouping methods start with edge detection, often using local operators. They attempt to connect the short edges thus obtained such that the resulting edges are distinct and best describe the objects in the image. Finding such edges is easy in clean images with well separated contours, but it is already a difficult task in natural images where noise and textures are present. The simplest way is to combine edge fragments with a high gradient. Methods that use more complex ways of joining edge fragments into a smooth contour will be described below. A common feature of these methods is that they use local detectors developed in [69]. Methods presented below use the Berkeley image dataset.

The authors of [75] use curvilinear continuity, which has scale-invariant properties, to add contextual information between local edge fragments. The danger in using edge completion is the addition of false boundaries. Therefore, the authors performed an analysis and found that edge linking with the proposed method achieves better detection than the output of local operators in all cases. First, the Pb algorithm is used, whose output is discretised into linear segments. The potential connectivity is then generated using constrained Delaunay triangulation (CDT), which is scale invariant. Two curvilinear continuity models are developed, a simple local model and a global model using conditional random field (CRF) [76] to select the best of the proposed interconnections by Delaunay triangulation. The simple model considers only the local context and is suitable for studying relevant features. The global model is used to build a joint probabilistic model over all edges. The trained models and the Pb algorithm have been tested, among others, to the Berkeley dataset. The global model achieves better results than the local model, and the Pb algorithm gives the worst results.

The authors of [77] present an approach to edge detection by estimating a set of curves in an image. The method is based on a statistical model of a scene with multiple curves and an estimation of the optimal curves (edges). The input of the algorithm is a set of short-oriented segments that connect pixels in the image to their neighbours. The individual short segments are either part of the curve or the background. The authors assume that the curves are drawn from a Markov process that favours smooth curves and a scene description using fewer curves. The estimation of the optimal curves is done using a minimum-cover framework that is simple and powerful. Unfortunately, the min-cover problem is NP-hard and therefore needs to be approximated by a greedy approximation algorithm that sequentially selects objects minimising a "cost per pixel" measure. The presented method shows that curves can be generated by a sequential growth that starts from small segments. Continuous evaluation of the quality of the curve during the algorithm is an important parameter to determine at which curve the growth should continue and at which should be stopped. The output of the presented algorithm is a small set of curves that describes the analysed image. The presented method improves the quality of edge detection relative to Pb [69] and CRF [76].

The authors of [78] propose a contour grouping method based on a 1D topology. They assume that a set of edges that have a well-defined ordering, and the connections between these edges strictly follow this ordering, can be described using a 1D topology. The paper presents a weighted graph formulation with a topological curve grouping score evaluating both the separation from the background and disentanglement within the curve. The weights measure a directional collinearity between adjacent edges. All edges detected by the Pb method are arranged in a

graph and scored. Based on this generated graph, the 1D contour topology has to be selected according to a defined criterion. The edge grouping criterion used is called untangling cycles. It is challenging to find the edge topology by optimising topological criterion, so the authors propose encoding the combinatorial problem as a circular embedding problem. For this, the computation of the dominant complex eigenvectors/eigenvalues of the random walk matrix of the contour grouping graph is required. The authors compared their method with CRF and min-cover, mentioning that their approach yielded improvements in edge detection, and visually their results yielded cleaner edges.

The authors of [72] describe an edge detector that combines several local features into a powerful globalisation framework based on spectral clustering. Inspired by the Pb detector, the authors propose their own multiscale Pb (mPb) detector and a globalisation method they run on top of this detector. The mPb detector computes oriented gradients for brightness, colour, and texture across three scales. The computed local features (multiscale brightness, colour, and texture features) for each position in the image define an affinity matrix that represents the similarity between pixels. For this constructed matrix, they define an eigenproblem and solve it for a fixed number of eigenvectors. The information obtained from the eigenvectors is combined to provide the spectral component of the boundary detector sPb. The output classifier, called the globalised probability of boundary (gPb), is defined as the weighted sum of the outputs of the local (mPb) and spectral (sPb) detector. The authors show that the use of both sPb and mPb detectors alone increases the edge detection success rate compared to the classical Pb method. The best results are achieved using the gPb detector (a combination of sPb and mPb detector).

3.7 Non-derivative edge detector

Another edge detector is SUSAN (Smallest Univalue Segment Assimilating Nucleus), which was introduced in [79]. This method does not require any image derivation or noise reduction. The SUSAN method ignores the noise as long as the noise is small enough.

The basis of the method is to evaluate each point in the image based on the similarity of the central element of the circular mask to its surroundings. The central element of the circular mask is called the Nucleus. The brightness of each pixel of the mask is compared with the brightness of the nucleus. If the difference in brightness of the compared pixel with the nucleus is less than a defined threshold, then it is marked as similar. The region of the mask that has the same or similar brightness value is called the USAN (Univalue Segment Assimilating Nucleus) – the area is defined as a sum of similar pixels.

The USAN region contains information about the structure of the image. The USAN area decreases as the nucleus approaches the edge and reaches a local minimum in the corner. The USAN outputs for each point can be displayed in a 2D graph where edges and two-dimensional features are strongly highlighted. Therefore, the method is called SUSAN. A USAN area is marked as an edge if it is smaller than a fixed threshold (so-called geometric threshold). The direction of the edge is then found using moment calculations. Sometimes it is still necessary to apply post-processing methods such as non-maximum suppression, thinning, or sub-pixel estimation. The authors of [53] state that the method is considered robust, noise-resistant, and reliable compared to gradient-based methods.

3.8 Fuzzy-logic based detection

Another direction for edge detection is the use of fuzzy logic. Fuzzy logic was first introduced by Lofti Zadeh [80] in 1965. Compared to Boolean logic, where values can only take binary values of 0 and 1, in fuzzy logic, values can come from the interval 0 to 1. Edge detection using fuzzy logic consists of several steps [81]. As a first step – pre-processing, features (in context with the surrounding pixels) are extracted for every pixel in the analysed image. A vector of features can be used to describe the pixels. The second step is fuzzy classification, where the image pixels are transformed into predefined classes based on the extracted features. Each class is defined using the same features that describe the image pixel. A fuzzy classifier classifies pixels into individual classes. The third step is to compare the pixel membership thus obtained with predefined rules (the so-called IF-THEN rules), which determine whether the pixel is an edge. Several rules can be defined, depending on the user. Without this step, the output of the classifier would result in thick edges. By applying the rules, the edges are made thinner. The last step can be post-processing, which, for example, removes short edges. The output of the whole algorithm is then an edge map.

The authors of [53] mention that this technique is flexible and gives good results. The advantage of this method is that it can deal with noise, and a user can easily define and change the IF-THEN rules or class membership parameters. Fuzzy logic for edge detection is used by many authors, e.g. [81, 82]. Several other papers are mentioned in [53].

3.9 Evolutionary algorithms

Evolutionary algorithms (EA) belong to the set of general stochastic search algorithms [83]. EA is a metaheuristic optimisation algorithm working with the concept

of population. It is suitable for various optimisation problems with limited computational capability and insufficient or imperfect information. EA is inspired by mechanisms of biological evolution such as reproduction, mutation, recombination, and selection. Although there are many different variations of EA, they all have a common idea, namely natural selection (survival of the fittest) – there is competition for resources among individuals in an environment with limited resources, and only the fittest survive. The basis of EA is established on two factors: variation operators (recombination and mutation), which ensure the necessary diversity in the population, and selection, which provides an increase in the average quality of the solution [84]. The initial steps of EA are:

- initialisation of the population with random characteristics,
- evaluation of the fitness of individual members of the population.

Subsequently, an iterative process is performed until a termination condition is satisfied. The steps of each iteration are as follows:

- selection of parents (i.e. members of the population with the highest fitness),
- recombination of the characteristics of the parents,
- mutation of their offspring,
- evaluation of all the new offspring,
- selection of members for the next generation.

Examples of EA include the following specific algorithms: particle swarm optimisation, bacterial foraging algorithm, ant and bee colony optimisation, genetic algorithm, and so on [53]. EAs can be applied to an image with initial edge estimation to find the optimal threshold for selecting the true edges (e.g. [55], for more references, see [53] and the references therein). Some authors have combined the application of fuzzy logic with the application of EA (e.g. [85, 86]). In these cases, EA is applied to the output of the fuzzy classifier.

3.10 Deep learning

Deep learning methods have become a popular tool in image processing in the last few years. Therefore, many researchers focus their efforts on using deep learning methods to detect edges. Based on the type of learning method, deep learning techniques are divided into three main categories: supervised, semi-supervised, and unsupervised. Supervised learning needs a train dataset that consists of input images and their evaluation (e.g. manually labelled edges). Unsupervised methods need a train set, but it does not contain the evaluation of the input images, e.g. clustering. In semi-supervised training, the train set contains a small number of rated images and a large number of unrated images.

The authors of [87], dedicate their work to making a comprehensive analysis and special research on edge detection algorithms. They divided the methods into two classes – traditional approaches (all methods before deep learning) and algorithms based on deep learning, such as multi-scale feature fusion, codec, network reconstruction, etc. The provided analysis showed that the traditional image edge detection methods had been developed until 2016, and from 2014, the researchers have focused mainly on algorithms based on deep learning. The provided experiments on Berkeley Segmentation Dataset (BSDS500) and other datasets indicated that the performance of the deep learning approaches is very close to the human visual level.

Codec-based edge detection methods use a codec, which accepts input image of any size and produces an image of the same size, such that the final output correspond to every pixel in the original image. This approach contains an encoder and decoder network. Among this approach belongs, for example, CEDN [88], CASENet [89], and RINDNet [90].

Network reconstruction-based methods use a combination of various network modules based on deep learning. Different modules have different advantages for a task, so their combination for specific tasks provides better results. As examples of such methods, it is possible to mention N^4-Fields [91], COB [92], and AMH-Net [93].

Multi-scale feature fusion-based methods improve edge detection performance by fusing features of different scales. The features are extracted by layer-by-layer abstraction, where a higher-layer network has a strong ability to characterise semantic information, while a lower-layer network has a strong ability to characterise geometrical detail information. These methods include, for example, DeepEdge [94], HED [95], BDCN [96], DexiNed [97], and PiDiNet [98].

Other deep-learning approaches for image edge detection are mentioned in [53, 87, 99–101].

3.11 Piecewise-polynomial signals

Some authors take advantage of the fact that a 1D signal can be viewed as a piecewise-polynomial signal that consists of several independent segments. Using this understanding, the image can be considered as a set of independent piecewise-polynomial patches. The authors of [102, 103] showed that images, to a great extent, can be modelled as piecewise smooth 2D functions.

Individual segments are described by several basis polynomials. A signal is then obviously overcomplete in terms of the number of parameters. To find a solution to the signal segmentation, a recovery problem needs to be defined.

The authors of [11] use a greedy approach [29] to solve an over-parametrised

variational problem using sparse representation. The paper shows applications to both 1D and 2D signals. This paper mainly deals with linear over-parametrisations and shows how the proposed method can be used for denoising, segmentation, and geometric inpainting of images. The main output of this approach is a segmented image and as the side product the edge map. This approach suppresses the texture of the objects, and thus the obtained gradient map of the segmented image mainly highlights the object boundaries. This gradient map can be used for edge detection.

Variously defined optimisation recovery problems for signal segmentation can be solved using different approaches, e.g. greedy approach [11] or weighted ℓ_1-minimisation. The paper [11] was the inspiration for this Thesis, which focuses on recovery problems that can be solved with convex relaxation techniques. In our paper [9] about image edge detection, the optimisation problem for image edge detection solved via proximal splitting algorithm is introduced.

3.12 Comparison

The authors of [72] compare the edge detection performance of all the methods described in Secs. 3.1 – 3.6 on the Berkeley Segmentation Dataset (BSDS300) [72] using the precision–recall curves with their maximum F-measure score [69], which is described in more detail in Sec. 6.3. The dataset includes several human edge annotations for each image. This comparison illustrated in Fig. 3.1 shows that their proposed gPb method (see Sec. 3.6) produces the best results. In the graph, the F-measure score obtained for human edge annotations is illustrated as a green dot, which indicates the average agreement between human subjects. The iso-lines represent the positions of equal F-measure score. The graph shows that methods producing only raw edge detection (i.e. methods without post-processing) give significantly worse results than methods that include post-processing.

This comparison allows the method developed within this Thesis to be compared with a wide range of different edge detection methods. The proposed method, however, does not include any post-processing, so its comparison with methods that produce raw edge detection (such as classical gradient-based method (Prewitt, Sobel, Roberts) and Laplacian of Gaussian) will be particularly important.

The authors of [101] compared their RDS (relaxed deep supervision) approach with the gPb method on the Berkeley Segmentation Dataset (BSDS500) [72] and achieved better results with their proposed method RDS (see Fig. 3.2). They also compared their method with other deep learning methods (i.e. DeepEdge, HED, HFL, see the references therein). From the results, it seems that the deep learning methods provide at this time the best results, compared to methods based on different approaches (i.e. gPb, SE, Sketch Tokens, see the references therein).

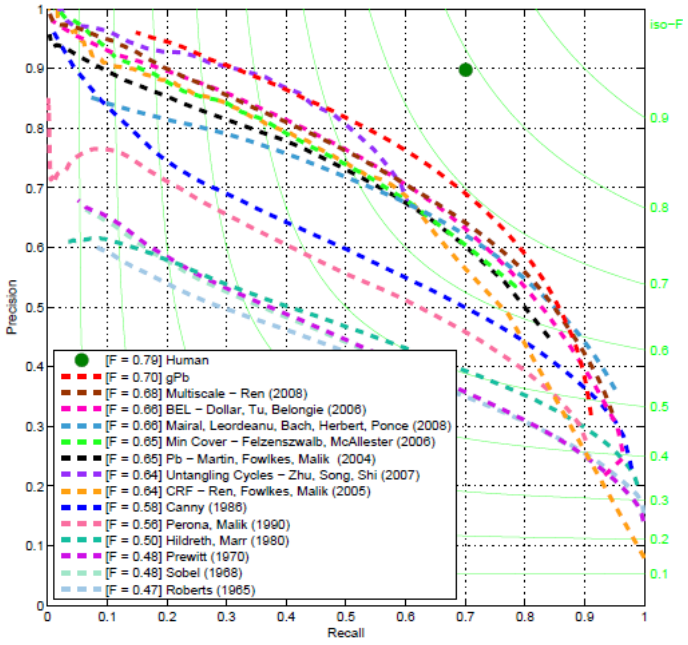


Fig. 3.1: Comparison of edge detection techniques on the Berkeley Segmentation Dataset Benchmark (BSDS300). The techniques in the legend are ranked according to their maximum F-measure. The image is taken from [72].

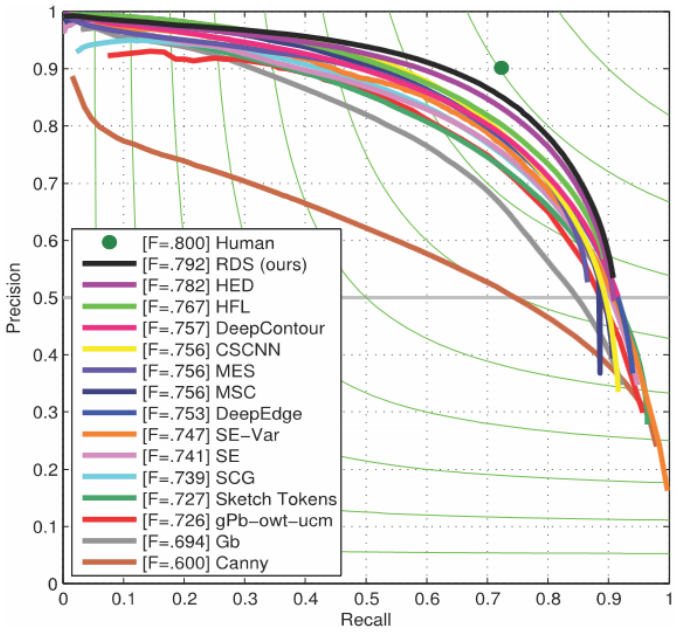


Fig. 3.2: Comparison of edge detection techniques on the Berkeley Segmentation Dataset Benchmark (BSDS500). The techniques in the legend are ranked according to their maximum F-measure. The image is taken from [101].

4 THESIS AIMS AND OBJECTIVES

The main aim of the Thesis is to propose, implement, and evaluate an effective method for image edge detection.

To do this, the first step is to propose a suitable convex optimisation problem that can deal with the problem of edge detection. Since there are many methods for solving convex optimisation problems, appropriate optimisation algorithms will be chosen to solve the problem numerically.

The authors of [11] used greedy approach to solve the optimisation recovery problem for signal segmentation. Inspired by this work, the intent of this Thesis is to explore how the ℓ_1-based convex relaxation methods will deal with the recovery problems for 1D signal segmentation and denoising. To the best of our knowledge, it will be the first time when the ℓ_1-minimisation-based approach will be used for signal segmentation/breakpoint detection.

First, the convex optimisation problems for one-dimensional signal segmentation and denoising will be formulated. It is easier to formulate the recovery problems, implement appropriate algorithms able to solve such formulated problems, and determine their advantages and disadvantages for one-dimensional signals than for 2D signals, i.e. images.

One of the advantages of the the 1D case is that segmentation and denoising are performed simultaneously. Denoising can be considered as a side-product of this solution. Furthermore, each detected segment can be denoised separately once again, to obtain a better denoising effect.

The first step will be to compare these methods on 1D signals, and then the method that gives the most promising results on 1D signals will be extended for the use in the higher dimension. Afterwards, the convex optimisation problems for the detection of edges in images will be formulated, which will then be solved by the most promising algorithm from the 1D segmentation phase.

A necessary part of the Thesis is the evaluation of the obtained results, which will be performed on a generated dataset of 1D synthetic signals and a public image dataset (BSDS500). Several evaluation metrics described in the Thesis will be used to evaluate the results on 1D and 2D signals.

Following the idea of reproducible research, the implementations of the algorithms for 1D signal segmentation and denoising and for image edge detection will be made publicly available.

The following sections treat the Thesis goals in more detail.

4.1 1D signal segmentation and denoising

At first, the whole concept of 1D signal segmentation and denoising needs to be designed. It will be necessary to formulate the signal model, define the specific recovery problems, create a dataset of testing 1D signals, and explore different types of polynomial bases.

Based on the assumptions observed from the signal model formulation, the recovery problem of the signal approximation consisting of convex functions will be formulated. The Thesis will focus on solving the convex optimisation problem via proximal splitting methods. In general, a recovery problem can be solved by more than one proximal splitting algorithm, and all the chosen algorithms for solving the problem will be implemented in Matlab. The aim is to find the appropriate recovery problem and method that converges fast, is robust to noise, and has a low computational cost.

Finally, the evaluation metrics will be proposed and all obtained results will be evaluated according to these metrics to find the most promising approach for the 1D signal segmentation and denoising process. The best-performing approach will be then extended to the two-dimensional case.

4.2 Image edge detection

This part of the Thesis will focus on the problem of image edge detection, which will be performed on both grayscale and colour images from the public image dataset BSDS500. In the case of colour images, image edge detection will be applied to each colour channel separately. A suitable merging approach must then be found to compose the outputs from each channel into a single output.

The results of the proposed edge detection algorithm will be evaluated according to the evaluation metrics and will be compared to other approaches presented in Chapter 3. The comparison will be performed on a test set of images from the public dataset BSDS500, widely used by image processing researchers.

4.3 Algorithm implementation

All chosen algorithms will be implemented in Matlab, some proximal algorithms for convex optimisation will be implemented using the UnLocBox toolbox [34].

A repository of Matlab implementations of the developed 1D signal segmentation and denoising algorithms and image edge detection algorithm will be created, including also the test datasets of 1D signals, images, and bases.

5 SEGMENTATION OF 1D SIGNALS

This section deals with the 1D signal segmentation and denoising process. First, the general description of the proposed method is provided, which includes signal model formulation, definition of processed signals, description of basis generation, and the proposition of individual steps of the 1D signal segmentation and denoising process. Then the used evaluation metrics are defined and signal datasets for experiments are presented. Finally, four different approaches to 1D signal segmentation and denoising are proposed and evaluated.

5.1 General description of the proposed method

In this section, a general concept of the proposed 1D signal segmentation process is introduced with the aim to explain the basic idea of the proposed methodology. This section is divided into several subsections describing individual parts, which are important for the segmentation process. First, the description of the signal model is formulated, which is a very important part since the proposed process is largely based on the assumptions derived from the signal model formulation. Then, types of processed signals and used bases are introduced. Finally, the whole concept of the 1D signal segmentation/denoising process is described.

5.1.1 Signal model description

In continuous time, a polynomial $y(t)$ of degree K can be described as a linear combination of basis polynomials

$$y(t) = x_0 p_0(t) + x_1 p_1(t) + \cdots + x_K p_K(t), \quad t \in \mathbb{R}, \tag{5.1}$$

where scalars x_k, $k = 0, \ldots, K$, are the expansion coefficients in such a basis. The system $\{p_k(t)\}_{k=0}^{K}$ is required to form a basis of the space of continuous-time polynomials of degree up to K.

In a discrete-time setting and with time instants $n = 1, \ldots, N$, the elements of a polynomial signal are represented as

$$y[n] = x_0 p_0[n] + x_1 p_1[n] + \cdots + x_K p_K[n], \quad n = 1, \ldots, N, \tag{5.2}$$

where N can be simply defined as the length of the signal $\mathbf{y}$.

For a given discrete-time polynomial basis $\{p_0[n]\}_{n=1}^{N}, \ldots, \{p_K[n]\}_{n=1}^{N}$, which is assumed to be fixed, every signal given by Eq. (5.2) is determined *uniquely* by the set of coefficients $\{x_k\}_{k=0}^{K}$. This uniqueness is broken when these coefficients are allowed to change in time by using the time index

$$y[n] = x_0[n] p_0[n] + x_1[n] p_1[n] + \cdots + x_K[n] p_K[n], \quad n = 1, \ldots, N. \tag{5.3}$$

This may seem meaningless at this moment, however, such an excess of parameters makes the representation of the signal $\{y[n]\}_{n=1}^{N}$ extremely overcomplete, which will be exploited later on. It is convenient to write this relation in a more compact form, for which a new notation needs to be introduced:

$$\mathbf{y} = \begin{bmatrix} y[1] \\ \vdots \\ y[N] \end{bmatrix}, \ \mathbf{x}_k = \begin{bmatrix} x_k[1] \\ \vdots \\ x_k[N] \end{bmatrix}, \ \mathbf{p}_k = \begin{bmatrix} p_k[1] \\ \vdots \\ p_k[N] \end{bmatrix}, \ k = 0, \ldots, K. \tag{5.4}$$

Then

$$\mathbf{y} = \mathbf{p}_0 \odot \mathbf{x}_0 + \cdots + \mathbf{p}_K \odot \mathbf{x}_K \tag{5.5}$$

with $\odot$ denoting the elementwise (Hadamard) product. To convert Eq. (5.5) to a form involving the standard matrix–vector product, it is necessary to define

$$\mathbf{P}_k = \operatorname{diag}(\mathbf{p}_k) = \begin{bmatrix} p_k[1] & & 0 \\ & \ddots & \\ 0 & & p_k[N] \end{bmatrix}, \ k = 0, \ldots, K, \tag{5.6}$$

allowing to write

$$\mathbf{y} = \mathbf{P}_0\mathbf{x}_0 + \cdots + \mathbf{P}_K\mathbf{x}_K, \tag{5.7}$$

or even more shortly

$$\mathbf{y} = \mathbf{P}\mathbf{x} = \begin{bmatrix} \mathbf{P}_0 \mid \cdots \mid \mathbf{P}_K \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \hline \vdots \\ \hline \mathbf{x}_K \end{bmatrix}, \tag{5.8}$$

where the length of $\mathbf{x}$ is $(K+1)N$ and $\mathbf{P}$ is a fat matrix of size $N \times (K+1)N$. Such a description of a N-dimensional signal is obviously overcomplete in terms of the number of parameters. When the polynomials in $\mathbf{P}$ are fixed, $(K+1)N$ parameters are used to characterise the signal.

Nevertheless, it is assumed that $\mathbf{y}$ is piecewise-polynomial and that it consists of S independent segments. Each segment $s \in \{1, \ldots, S\}$ is then described by $K+1$ basis polynomials. This can be achieved by letting all the vectors $\mathbf{x}_k$ be piecewise-constant within the particular segments, with the change at the segment borders. See Fig. 5.1 for illustration. The positions of the segment borders are unknown and they will be the subject of the search.

Due to the facts described above, if $\mathbf{x}_k$ are piecewise-constant, the finite difference operator ∇ applied to vectors $\mathbf{x}_k$ produces sparse vectors $\nabla\mathbf{x}_k$, where each such a vector $\nabla\mathbf{x}_k$ has $S-1$ non-zeros at maximum. Moreover, the non-zero components of each $\nabla\mathbf{x}_k$ occupy the same positions across $k = 0, \ldots, K$. Operator ∇ computes

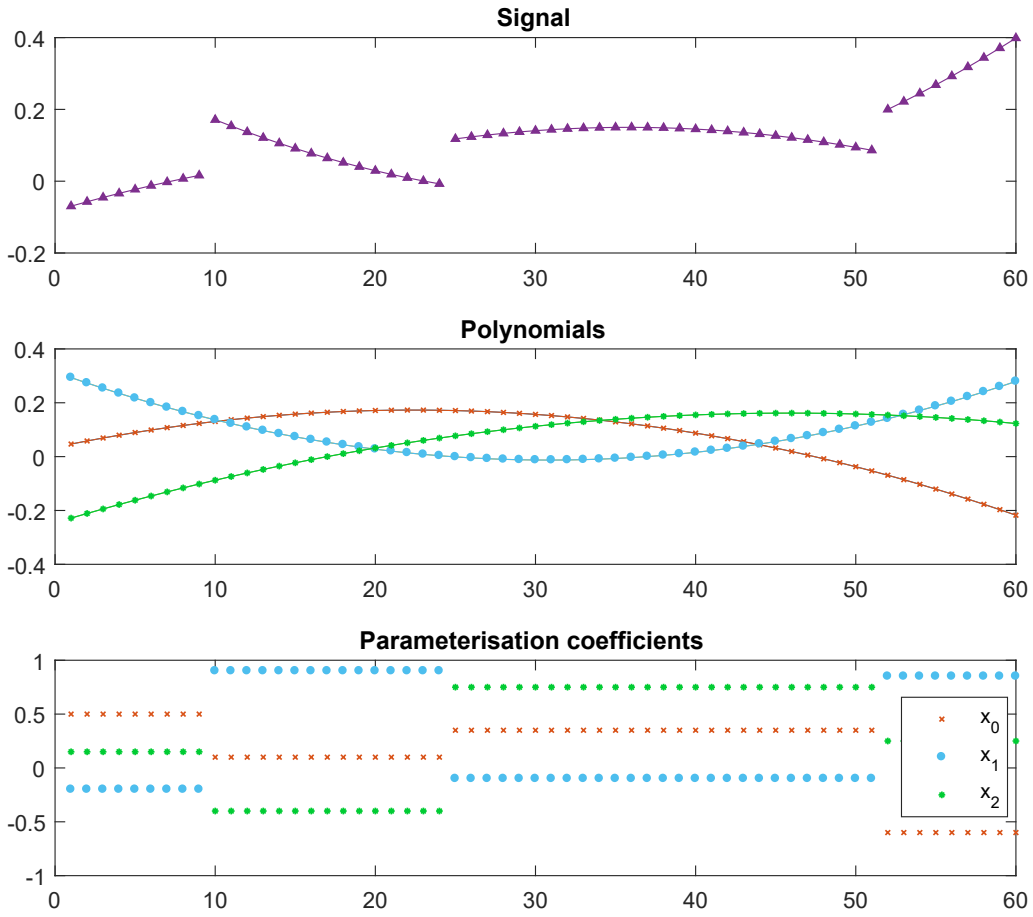


Fig. 5.1: Example of piecewise-quadratic signal parametrisation. The top plot shows the clean piecewise-quadratic signal $\mathbf{y}$ consisting of four segments. The middle plot shows three basis polynomials $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$. The bottom plot shows the parametrisation coefficients $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$. Notice that infinitely many other combinations of values $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ generate (in combination with basis polynomials $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$) the same signal, however, here is shown the piecewise-constant case.

simple differences of each pair of adjacent elements in the vector, i.e. $\nabla\colon \mathbb{R}^N \to \mathbb{R}^{N-1}$ such that $\nabla\mathbf{z} = [z_2 - z_1, \ldots, z_N - z_{N-1}]^\top$.

The above-mentioned formulations describe a clean polynomial signal without noise corruption. In the real world, however, it is usual that signals are corrupted by different types of noise. In this Thesis, it is assumed that the observed signal is corrupted by uncorrelated Gaussian noise with zero mean and non-zero variance. The definition of such a signal is

$$\mathbf{y} = \mathbf{P}\mathbf{x} + \mathbf{e}, \tag{5.9}$$

where $\mathbf{e}$ represents the noise vector of length N.

5.1.2 Types of processed signal

The experiments can be performed on synthetic and real signals. However, the main aim of this chapter is to find the most promising method for 1D signal segmentation, which will be easily done on a large dataset of synthetic signals. Therefore, it is not necessary to extend this chapter with the experiments on real data, however, they can be found in our article [7] and a conference paper [8].

The process of the synthetic signal generation is described in detail in the following section.

Synthetic signal

This section explains how the synthetic signals are generated. For the generation of piecewise-polynomial signals, several parameters need to be set. Such input parameters are: length of the signal N, degree of the polynomial signal K, number of signal segments S, range of jump height between signal segments $\mathbf{h} = [h_1, \ldots, h_L]$, where L is the number of jump levels, and signal-to-noise ratio (SNR). If the signal consists of S segments, it has $S - 1$ segment borders and their positions are set randomly. The minimal length of a signal segment is set to 5 signal samples.

At first, the noiseless piecewise-polynomial signal of degree K and of length N is generated, which is denoted $\mathbf{y}_{\text{clean}}$. With the use of the modified standard basis $\mathbf{S}$ consisting of $K+1$ basis polynomials (see Sec. 5.1.3), S polynomial signal segments of prescribed length are randomly generated. These signal segments are connected to each other, such that no jumps between signal segments are allowed – the jump height is zero. After that, jumps of prescribed height h_l are created between the signal segments in this clean signal. The value h_l is added to all samples of even signal segments of $\mathbf{y}_{\text{clean}}$. This means that the jump height is the same for all jumps in the signal. According to this process, L signal variants of the piecewise-polynomial signal without jumps are created. The variants differ from each other in the height of the jump between the signal segments. Outputs of such a process are $L+1$ clean piecewise-polynomial signals $\mathbf{y}_{\text{clean}}$ (L with and 1 without jumps).

All clean signals are corrupted by the Gaussian i.i.d. noise $\mathbf{e}$, resulting in noisy signals $\mathbf{y}_{\text{noisy}}$, such that

$$\mathbf{y}_{\text{noisy}} = \mathbf{y}_{\text{clean}} + \mathbf{e}, \quad e_n \sim \mathrm{N}(0, \sigma_e^2). \tag{5.10}$$

With these signals, the signal-to-noise ratio (SNR) can be determined as

$$\mathrm{SNR}\left(\mathbf{y}_{\text{noisy}}, \mathbf{y}_{\text{clean}}\right) = 20 \cdot \log_{10} \frac{\|\mathbf{y}_{\text{clean}}\|_2}{\|\mathbf{y}_{\text{noisy}} - \mathbf{y}_{\text{clean}}\|_2}. \tag{5.11}$$

To generate a signal with a defined SNR, the standard deviation σ_e of the respective noise needs to be computed such that

$$\sigma_e = \frac{\|\mathbf{y}_{\text{clean}}\|_2}{\sqrt{N \cdot 10^{\frac{\text{SNR}}{10}}}}. \tag{5.12}$$

It can be noticed that the resulting σ_e is influenced by the energy of the clean signal as well.

Real signal

As a real signal suitable for testing can be selected the OTDR signal. This signal is acquired by an Optical Time Domain Reflectometer (OTDR) instrument.

Optical Time Domain Reflectometry [104] is a basic method used for characterising and evaluating the quality of optical fibres in optical networks. The OTDR is used to measure the total loss, the losses or reflectance of splices and connectors, and to locate fibre breaks and defects. The principle is to send a pulse of light into an optical fibre and measure the travel time and the strength of its reflections – the back-scattered light is measured. The OTDR uses the effects of Rayleigh scattering and Fresnel reflection. Since the reflections follow an exponential model, the acquired data are subject to logarithmisation, which makes the measurement linearised.

Measured OTDR signal shows the events on the fibre, the vertical axis is the power axis, where the events are noticeable (jumps), and the horizontal axis is the distance axis. The event is characterised by loss or reflections, which can be caused by connections or damages of the optical fibre. [104–106]

5.1.3 Types of bases

Several types of discrete-time polynomial bases can be used for signal representation according to Eq. (5.2). The motivation for using several types of bases is to explore whether bases with different properties can influence the quality of signal segmentation. Using certain types of bases can bring advantages in form of no additional need of tuning the parameters. These parameters are used to suppress or benefit some basis vectors of the used basis and they will be described in more detail in the following sections.

In this Thesis, the bases are divided into groups according to their orthogonality and normalisation. All the polynomial bases consist of $K+1$ basis vectors, which are linearly independent discrete-time polynomials $\mathbf{p}_k$. The basis vectors $\mathbf{p}_k$ can be viewed as the columns of the $N \times (K+1)$ matrix and they form the diagonals of the matrix $\mathbf{P}$ in the model (5.8). The newly defined matrix of size $N \times (K+1)$ is

marked with different letters (**S**, **B**, **N**, **O**, **R**) according to the properties of the particular polynomials.

In the following subsections, the terms orthogonality, orthonormality, and normality are used, see the definitions in Sec. 1.1.5.

The discrete version of the orthogonal continuous basis is obtained by its sampling. This discretisation process, however, does not lead to obtaining a discrete orthogonal basis. Such obtained discrete basis is not necessarily orthogonal.

Standard basis

Most of the papers that explicitly model the polynomials utilise directly the standard basis, defined in continuous time as follows:

$$p_0(t) = 1, p_1(t) = t, \ldots, p_K(t) = t^K, \tag{5.13}$$

and in discrete time

$$p_0[n] = n^0, p_1[n] = n^1, \ldots, p_K[n] = n^K, \tag{5.14}$$

which is clearly not orthogonal either in the continuous or discrete setting.

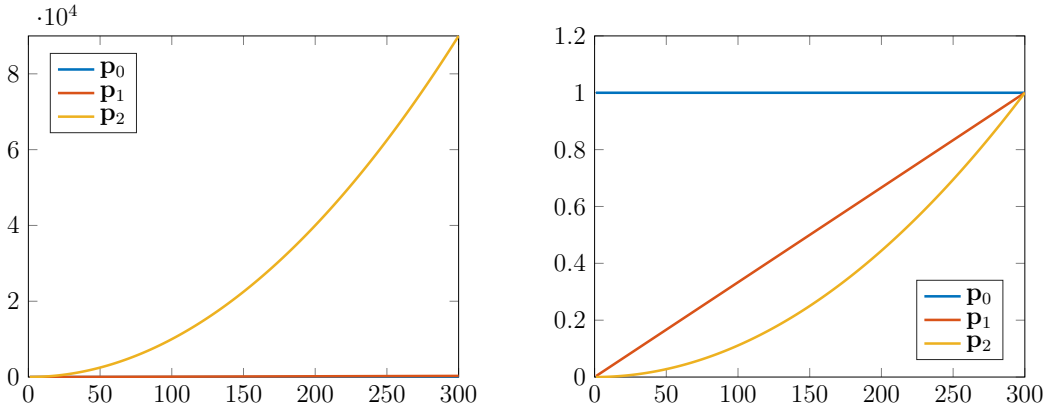
In our previous papers [5, 6] and also in [11], the standard basis is used. It was observed that using the standard basis leads to numerical problems for even moderate N. Personal communication with the authors of [11] confirmed this issue.

Modified standard basis (S)

To avoid the above-mentioned problems, a modified version of the standard basis is used in this Thesis for bases generation and experiments. The modification consists in changing the range of the values in the particular polynomials. The maximum value of standard basis polynomial $\mathbf{p}_k$ is N^k and the range of its values is from 1 to N^k for $k = 0, \ldots, K$. The modification caused that the maximum value of all modified standard basis polynomials $\mathbf{p}_k$ is 1, the range of the values in the modified standard basis polynomial $\mathbf{p}_k$ is from $\frac{1}{N^k}$ to 1 for $k = 0, \ldots, K$. The modified standard basis polynomials are, in contrast to Eq. (5.14), defined as follows:

$$p_k[n] = \left(\frac{n}{N}\right)^k, \text{ for } n = 1, \ldots, N, \text{ and } k = 0, \ldots, K. \tag{5.15}$$

The differences between the standard basis and the modified standard basis **S** are shown in Fig. 5.2.



(a) standard basis (b) modified standard basis

Fig. 5.2: The standard basis and the modified standard basis $\mathbf{S}$ consisting of three basis polynomials of length 300.

Non-orthogonal bases (B)

In this Thesis, the group of the non-orthogonal bases $\mathbf{B}$ includes bases that are non-orthogonal and non-normalised at the same time. The modified standard basis $\mathbf{S}$ belongs to this group. The modified standard basis can be used for the generation of the other B-bases according to formula $\mathbf{B} = \mathbf{S}\mathbf{D}_1\mathbf{A}\mathbf{D}_2$. First, the columns of the modified standard basis $\mathbf{S}$ are normalised using a diagonal matrix $\mathbf{D}_1 \in \mathbb{R}^{(K+1)\times(K+1)}$, then mixed using a random Gaussian matrix $\mathbf{A} \in \mathbb{R}^{(K+1)\times(K+1)}$ and finally dilated to different lengths using matrix $\mathbf{D}_2 \in \mathbb{R}^{(K+1)\times(K+1)}$, which has uniformly random entries at its diagonal. B-bases acquired this way are non-orthogonal and non-normalised. Example of the B-basis is shown in Fig. 5.3a.

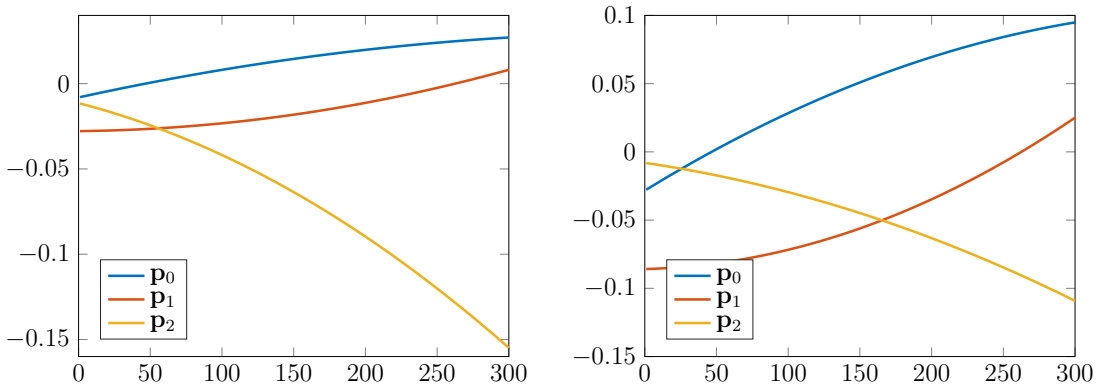
Normalised bases (N)

In this Thesis, the N-bases are non-orthogonal and normalised at the same time. Such N-bases are obtained in a simple way – by normalising the length of the B-basis polynomials – according to formula $\mathbf{N} = \mathbf{B}\mathbf{D}_3$, where the diagonal matrix $\mathbf{D}_3 \in \mathbb{R}^{(K+1)\times(K+1)}$ provides the length normalisation. Example of the N-basis is presented in Fig. 5.3b.

Orthogonal bases (O)

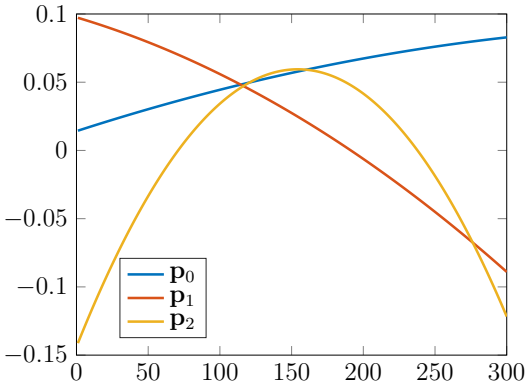
In this Thesis, the O-bases are orthogonal and normalised at the same time. The N-bases are the foundation for O-bases generation. O-bases are obtained by orthogonalisation of N-bases. A matrix $\mathbf{N}$ is decomposed by the singular value decomposition (SVD) [107], i.e.

$$\mathbf{N} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top. \tag{5.16}$$



(a) example of B-basis

(b) example of N-basis



(c) example of O-basis

Fig. 5.3: Examples of B-basis, N-basis and O-basis. B-basis was generated from S-basis, N-basis was obtained from B-basis and O-basis was obtained from N-basis. All bases consist of three basis polynomials of length 300.

Matrix $\mathbf{U}$ consists of $K+1$ orthonormal columns of length N. The new orthonormal system is obtained simply by $\mathbf{O} = \mathbf{U}$. See Fig. 5.3c for the example of O-basis.

The new O-basis spans on the same space as N-basis does because the $\mathbf{N}$ has full rank, $\mathbf{\Sigma}$ contains $K+1$ positive values on its diagonal and $\mathbf{V} \in \mathbb{R}^{(K+1)\times(K+1)}$ is orthogonal. The new O-basis is still consistent with any polynomial basis on $\mathbb{R}$ since both $\mathbf{N}$ and $\mathbf{U}$ can be substituted by their continuous-time counterparts, thus generating the identical polynomial.

Random orthogonal bases (R)

R-bases are randomly generated bases, which are, in this Thesis, orthogonal and normalised at the same time. Also for R-bases, the N-bases are the foundation for their generation. The generation process of R-bases differs from the O-bases generation process as follows: the first step is the same – the SVD is applied to

the $\mathbf{N}$ as in Eq. (5.16), with using new symbolisation, $\mathbf{N} = \mathbf{U_N}\mathbf{\Sigma_N}\mathbf{V_N}^\top$, where $\mathbf{U_N} \in \mathbb{R}^{N\times(K+1)}$, $\mathbf{\Sigma_N} \in \mathbb{R}^{(K+1)\times(K+1)}$, and $\mathbf{V_N} \in \mathbb{R}^{(K+1)\times(K+1)}$.

The second step is the generation of random matrix $\mathbf{A}$ of size $(K+1)\times(K+1)$, where each element independently follows the Gaussian distribution.

The third step is the application of the SVD to the random matrix $\mathbf{A}$, $\mathbf{A} = \mathbf{U_A}\mathbf{\Sigma_A}\mathbf{V_A}^\top$, where $\mathbf{U_A} \in \mathbb{R}^{(K+1)\times(K+1)}$, $\mathbf{\Sigma_A} \in \mathbb{R}^{(K+1)\times(K+1)}$, and $\mathbf{V_A} \in \mathbb{R}^{(K+1)\times(K+1)}$.

Finally, the obtained matrices $\mathbf{U_N}$ and $\mathbf{U_A}$ are used for the new R-basis generation as follows: $\mathbf{R} = \mathbf{U_N}\mathbf{U_A}$. Since both matrices $\mathbf{U_N}$ and $\mathbf{U_A}$ are orthonormal, the columns of $\mathbf{R}$ form an orthonormal basis spanning the desired space. Elements of $\mathbf{U_A}$ determine the linear combinations used in forming $\mathbf{R}$.

A note on other polynomial bases

In this Thesis, the predefined orthogonal polynomial bases as Chebychev or Legendre bases, for example, are not used. Such bases are defined in continuous time and therefore, they are orthogonal with respect to an integral scalar product [108]. For the needs of this Thesis, the discrete polynomial bases are required. Sampling of such systems does not lead to discrete orthogonal bases, orthogonalisation of sampled system via SVD significantly changes the course of the basis vectors. No predefined discrete-time orthogonal polynomial bases were found during working on this Thesis. Because the result of the sampling and follow decomposition is similar to the above-mentioned O-bases, this type of bases was omitted from this Thesis.

5.1.4 Formulation of the whole concept of signal segmentation and denoising

Several methods for 1D signal segmentation and denoising are proposed in this Thesis. Each of the proposed methods can be divided into three main steps, common to all these methods. Therefore, the general method for 1D signal segmentation and denoising can be defined and formally described by these main steps:

- Optimisation step using a proximal splitting algorithm,
- Segmentation and denoising,
  - Detection of segment borders (breakpoints),
  - Smoothing of detected segments.

Individual steps of the processed signal are illustrated in Fig. 5.4.

Proposed methods differ from each other only in the first step (i.e. the optimisation step using a proximal splitting algorithm), specifically in the problem formulation and the proximal splitting algorithm used. A detailed description of particular steps is presented in the following subsections.

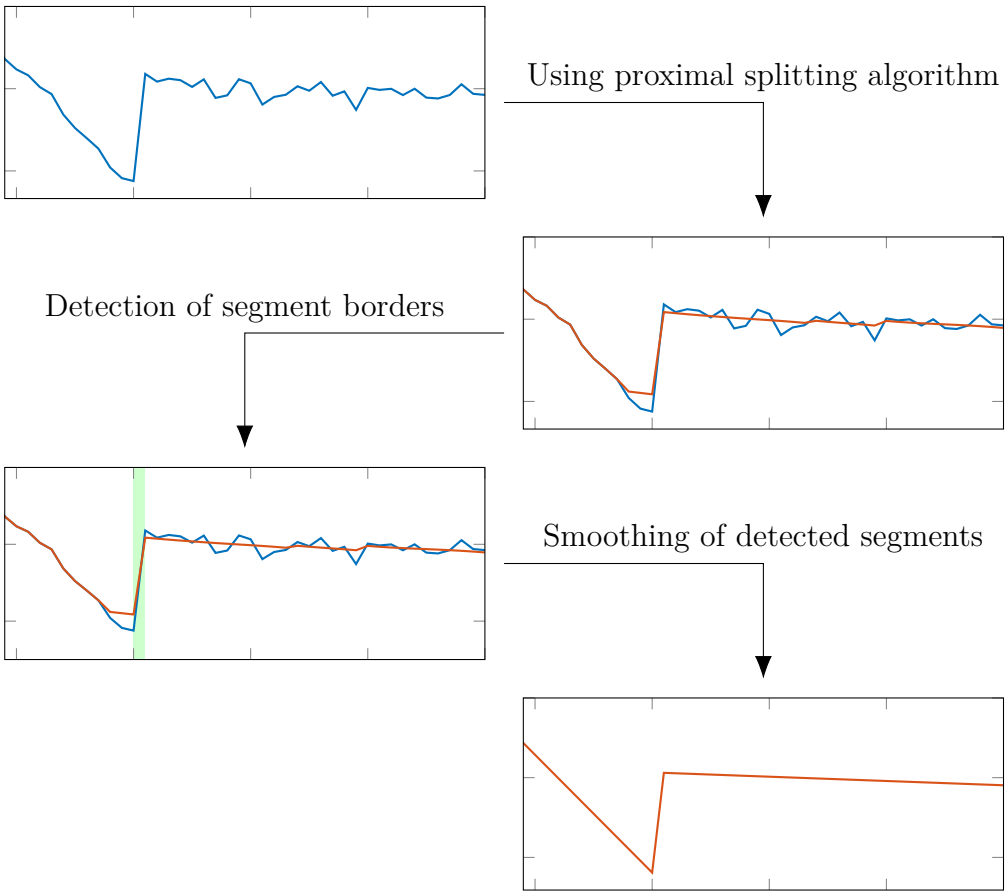


Fig. 5.4: Process of 1D signal segmentation. The diagram shows how the observed signal is processed during the 1D signal segmentation and denoising.

Optimisation step using a proximal splitting algorithm

At first, the general recovery problem suitable for the solution of the 1D signal segmentation and denoising needs to be defined. The below-mentioned assumptions are taken into account to formulate the general recovery problem:

- Observed signal $\mathbf{y}$ can be approximated by a linear combination of basis polynomials $\mathbf{p}_k$.
- The number of the signal segments S is considerably lower than the number of the signal samples N ($S \ll N$), which motivates to utilize sparsity.
- Observed signal $\mathbf{y}$ is corrupted by Gaussian noise.

The above-mentioned assumptions, which are based on the signal model description (see Sec. 5.1.1) imply that parametrisation coefficients $\mathbf{x}_k$ are sparse under the difference operator ∇. According to these assumptions, the general recovery problem consists of two main terms: data fidelity term and penalisation term (penalty).

General formulation of the recovery problem can be written as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \; data\_fidelity\_term(\mathbf{x}) + penalisation\_term(\mathbf{x}), \tag{5.17}$$

where $\hat{\mathbf{x}}$ are the obtained parametrisation coefficients of length $(K+1)N$, and N is the length of the signal $\mathbf{y}$ and K is polynomial degree. The vector $\hat{\mathbf{x}}$ is defined as a set of obtained parametrisation coefficient vectors $\{\hat{\mathbf{x}}_k\}_{k=0}^{K}$.

The proposed recovery problem is an optimisation program consisting of convex functions, which can be solved by using proximal splitting algorithms.

In this Thesis, several recovery problems suitable for the solution of the 1D signal segmentation/denoising process are proposed. Particular solutions to the proposed recovery problems are introduced in the following sections.

Segmentation and denoising

Parametrisation coefficients $\hat{\mathbf{x}}$ obtained as optimisers of the recovery problem (5.17) allow simple estimation of the underlying noiseless signal $\hat{\mathbf{y}}$ (see Fig. 5.6a) according to $\hat{\mathbf{y}} = \mathbf{P}\hat{\mathbf{x}}$. Unfortunately, this approach has two disadvantages. The first disadvantage is that this way, the segment ranges are not obtained. The second one is that the jumps are typically underestimated in size, which comes from the bias inherent to the ℓ_1-norm (see [109, 110] and our conference paper [5]) as part of the optimisation problem.

Obtained parametrisation vectors $\hat{\mathbf{x}}_k$ should be optimally piecewise-constant and after application of the difference operator ∇, the vectors $\nabla\hat{\mathbf{x}}_0, \ldots, \nabla\hat{\mathbf{x}}_K$ should have $S-1$ non-zero values (in each vector) indicating the segment borders (breakpoints). However, it is almost impossible to achieve truly piecewise-constant optimisers $\hat{\mathbf{x}}_k$. Therefore, the vectors $\nabla\hat{\mathbf{x}}_k$ are in practice full of small values. Besides these small values, larger values can be found in the vectors, which indicate possible segment borders. Because the underlying signal is assumed to be piecewise-polynomial, the two-part procedure is applied to obtain the segmented and denoised signal. In the first step "Detection of segment borders," the breakpoints are detected to obtain individual segments. Then each detected segment is denoised individually in the second step called "Smoothing of detected segments."

Detection of segment borders

"Detection of segment borders" is the second main step in the 1D signal segmentation and denoising process. The process of breakpoint detection tries to avoid false detection caused by small values in the difference vectors $\nabla\hat{\mathbf{x}}_k$. The assumption is that the significant values in $\nabla\hat{\mathbf{x}}_k$ are situated at the same positions. This property is very convenient for this detection procedure. As a base for breakpoint detection,

a single vector $\mathbf{d} \in \mathbb{R}^{N-1}$ is computed out of all obtained $\nabla\hat{\mathbf{x}}_k$ using the weighted ℓ_2-norm according to the formula

$$\mathbf{d}[n] = \sqrt[2]{(\alpha_0 \nabla\hat{\mathbf{x}}_0[n])^2 + \cdots + (\alpha_K \nabla\hat{\mathbf{x}}_K[n])^2} \quad \text{for } n = 1, \ldots, N-1, \tag{5.18}$$

where scalars $\alpha_0, \ldots, \alpha_K$ are positive factors computed as $\alpha_k = 1/\max(|\nabla\hat{\mathbf{x}}_k|)$, serving to normalise the range of values in the parametrisation vectors differences.

Then, a moving median filter is applied to $\mathbf{d}$ and the obtained filtered signal is subtracted from $\mathbf{d}$, yielding $\hat{\mathbf{d}}$ (see Fig. 5.5). This approach keeps the significant values and pushes small values to zero in $\hat{\mathbf{d}}$. Values in $\hat{\mathbf{d}}$ satisfying the condition $\left|\hat{\mathbf{d}}\right| > \lambda$ constitute the detected breakpoints. The threshold λ has to be set properly.

Occasionally, it can happen that two or more detected breakpoints are adjacent to each other. Since it does not make sense to keep and process segments of zero length, in such a situation only one of these indexes is chosen – the one with the largest absolute value. Finally, after this step, the positions of possible breakpoints are estimated, which defines the range of each detected segment precisely. The total number of detected segments is $\hat{S}$.

Smoothing of the detected segments

In this final step, each detected signal segment $\hat{s} = 1, \ldots, \hat{S}$ is smoothed/denoised separately, independently of the other segments. This smoothing process of detected segments can be viewed as a sequential process where the number of iterations is equal to the number of detected signal segments $\hat{S}$ – exactly one detected signal segment is smoothed/denoised in each iteration. Smoothing/denoising is done simply by forming a regression matrix $\mathbf{A}$ consisting of $K+1$ basis polynomials as its columns, and performing an ordinary least squares method on the observed noisy signal $\mathbf{y}$ restricted to the segment range. In this step, it does not matter what kind of polynomial basis is used. From the mathematical point of view, it is better to use the orthogonal matrix, since the following procedure requires inversion of the matrix.

For each detected segment $\hat{s}$, a set of parametrisation coefficients $C_{\hat{s}}$ is found according to

$$C_{\hat{s}} = (\mathbf{A}_{\hat{s}}^\top \mathbf{A}_{\hat{s}})^{-1} \mathbf{A}_{\hat{s}}^\top \mathbf{y}_{\hat{s}}, \tag{5.19}$$

where $C_{\hat{s}} = \{c_k\}_{k=0}^K$ is set of constants c_k of detected segment $\hat{s}$. The matrix $\mathbf{A}_{\hat{s}}$ is created by restricting $\mathbf{P}$ to the segment range, and $\mathbf{y}_{\hat{s}}$ is the detected segment in observed signal $\mathbf{y}$. In Matlab, this can be simply written as $C_{\hat{s}} = \mathbf{A}_{\hat{s}} \setminus \mathbf{y}_{\hat{s}}$. Constants c_k in $C_{\hat{s}}$ form the particular piecewise-constant part in parametrisation vectors $\dot{\mathbf{x}}_k$, which are restricted to the segment range and are grouped in matrix $\mathbf{X}_{\hat{s}}$ as follows:

$$\mathbf{X}_{\hat{s}} = [c_0 \cdot \mathbf{1}_{\hat{s}}, \ldots, c_K \cdot \mathbf{1}_{\hat{s}}], \tag{5.20}$$

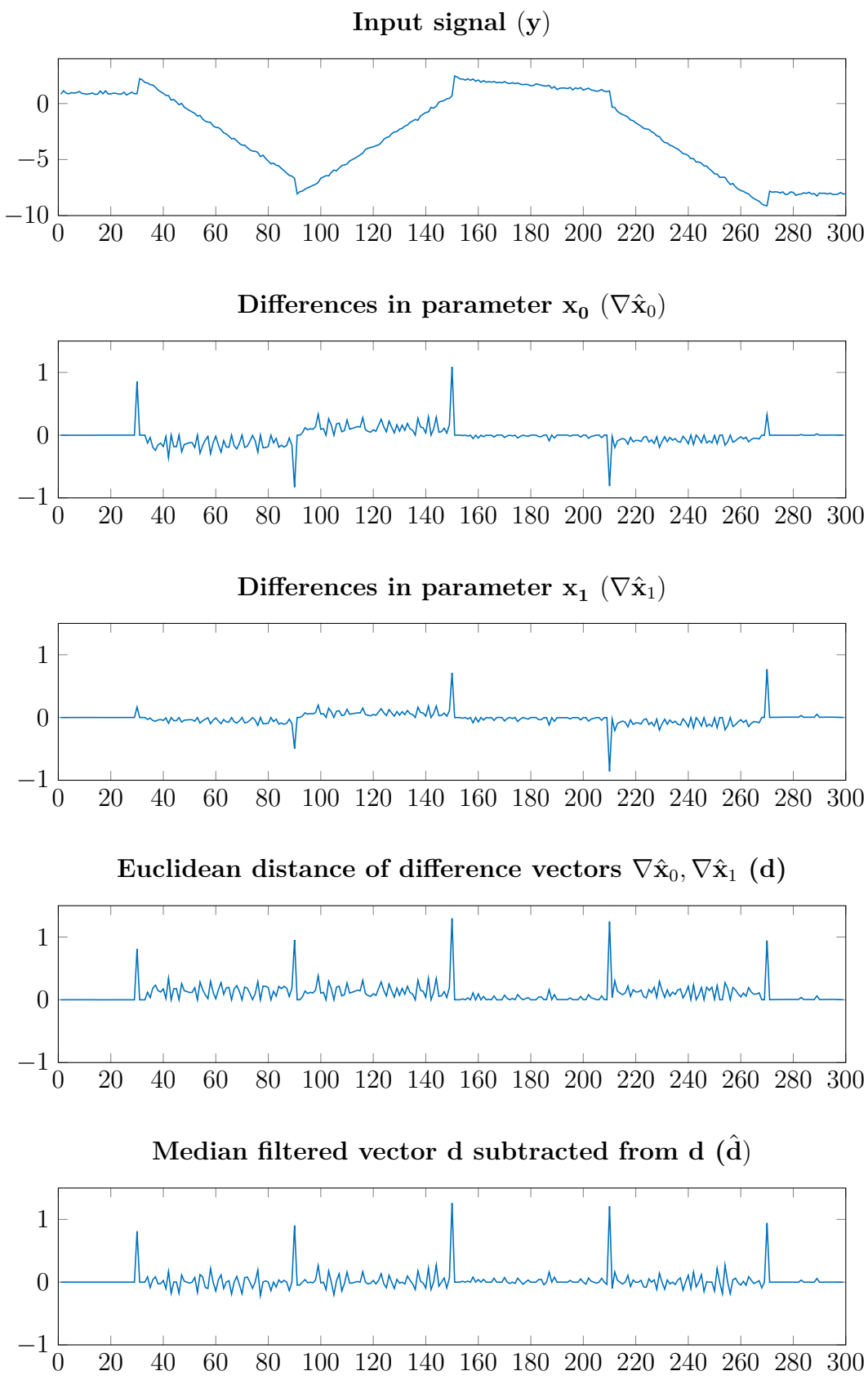


Fig. 5.5: Post-processing of the parametrisation coefficients $\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1$, leading to $\hat{\mathbf{d}}$.

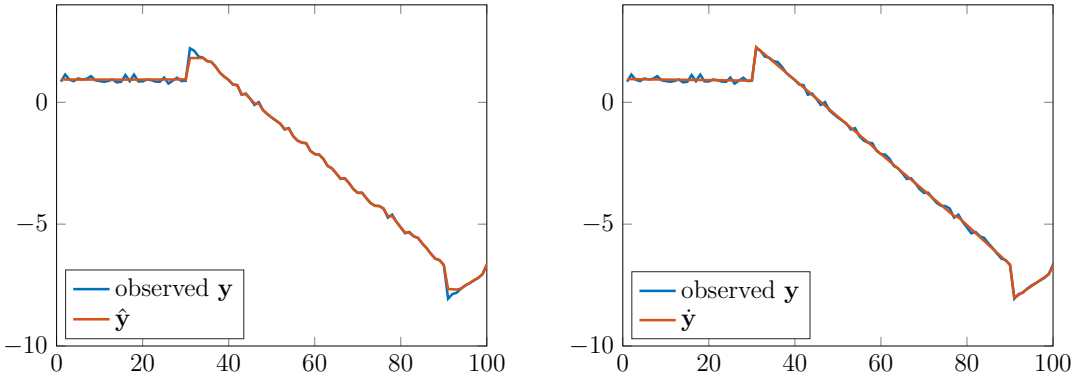
where $\mathbf{1}_{\hat{s}}$ is a vector of ones with length $N_{\hat{s}}$ of the $\hat{s}$-th detected segment, and the matrix $\mathbf{X}_{\hat{s}} \in \mathbb{R}^{N_{\hat{s}} \times (K+1)}$ contains constant parts of the parametrisation vectors $\dot{\mathbf{x}}_k$.

After the determination of all $\mathbf{X}_{\hat{s}}$ for all detected segments $\hat{S}$, the complete parametrisation vectors can be determined as follows:

$$\dot{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_{\hat{S}} \end{bmatrix} = [\dot{\mathbf{x}}_0, \ldots, \dot{\mathbf{x}}_K], \quad \text{for } k = 0, \ldots, K;\ \hat{s} = 1, \ldots, \hat{S}. \tag{5.21}$$

By utilising this process, a new parametrisation coefficient set $\{\dot{\mathbf{x}}_k\}_{k=0}^{K}$ is obtained, which is now constant on the segment-by-segment basis.

According to $\dot{\mathbf{y}} = \mathbf{P}\dot{\mathbf{x}}$, the complete denoised and segmented signal $\dot{\mathbf{y}}$ is then reconstructed (see Fig. 5.6b). The least-squares refit could be seen as a procedure usually termed "debiasing," commonly used in LASSO-type estimation [13, 111], which is inherently biased by the use of the ℓ_1-norm.



(a) result of optimisation step

(b) result of post-processing

Fig. 5.6: Comparison of the signal before and after post-processing. The left-hand side figure shows the noisy signal $\mathbf{y}$ (observed) and the approximation $\hat{\mathbf{y}}$ obtained by the optimisation step (before post-processing). The right-hand side figure shows the observed signal and signal $\dot{\mathbf{y}}$ obtained by post-processing (breakpoint detection and signal denoising).

5.2 Evaluation

Two points of view are used for the evaluation of the results of the 1D signal segmentation and denoising process. The first one is the evaluation of the quality of signal segmentation, i.e. breakpoint detection accuracy. The second one is the evaluation of obtained signal denoising.

5.2.1 Evaluation of breakpoint detection accuracy

For the evaluation of the quality of the breakpoint detection process, it is necessary to know the true positions of the breakpoints. The positions of breakpoints are known for synthetic signals, but in the case of real signals, the positions of the breakpoints need to be labelled manually. For such an evaluation, several metrics are defined and described in the following sections. These markers are based on the vector $\hat{\mathbf{d}}$, which is forced to be non-negative using an absolute value (see Sec. 5.1.4). At first, indexes of each evaluated signal and the respective values have to be divided into disjoint sets – HV_i, OV_i, and HV_v, OV_v, respectively. The process of creating such sets is defined below.

Highest values (HV): The HV is described by two sets HV_i and HV_v. The number of breakpoints in synthetic test signals is known – each tested signal contains $S-1$ breakpoints. The HV_i group thus gathers the indexes of $S-1$ highest values in $\hat{\mathbf{d}}$ that are likely to represent breakpoints, the $S-1$ highest values in $\hat{\mathbf{d}}$ form the set HV_v. These $S-1$ indexes are selected iteratively. The number of iterations is equal to the known number of breakpoints $S-1$. In each iteration, the largest value in $\hat{\mathbf{d}}$ is chosen and assigned to HV_v and its index is assigned to HV_i. It can happen that more high values appear next to each other. To avoid detection of neighbouring values, the two neighbouring indexes (of the chosen highest value) to the left and two to the right are omitted from the largest value detection in the further iteration step. After this iteration process, the HV_i contains $S-1$ indexes of highest values in $\hat{\mathbf{d}}$ and HV_v contains $S-1$ those respective values.

Other values (OV): Also OV is described by two sets OV_i and OV_v. The OV_i group contains the remaining indexes in $\hat{\mathbf{d}}$ (in relation to HV_i). The indexes excluded during the HV_i selection are not considered in OV_i. The OV_v set contains the remaining values in $\hat{\mathbf{d}}$. This way, the number of elements in OV_i and OV_v is N minus $(S-1)$ and minus a particular number of omitted indexes, depending on the particular course of the HV selection process. This means that the number of indexes in OV_i differs with each tested signal.

AAR

AAR is abbreviation for the average of values in HV_v to average of values in OV_v ratio. For each tested signal, the ratio of averages of the values belonging to HV_v versus values belonging to OV_v is computed as follows:

$$\mathrm{AAR} = \frac{\mathrm{avg}(HV_v)}{\mathrm{avg}(OV_v)}, \tag{5.22}$$

where the operator avg represents the arithmetic mean.

This indicator describes if there is a significant difference between the chosen $S-1$ highest values and the rest of the values in $\hat{\mathbf{d}}$. The AAR indicator should be as high as possible to ensure safe recognition of the breakpoints.

MMR

MMR indicator is the abbreviation of minimum-to-maximum ratio. Specifically, it is defined as a ratio of the minimum of HV_v values to the maximum of the OV_v values, formally

$$\text{MMR} = \frac{\min(HV_v)}{\max(OV_v)}. \tag{5.23}$$

This indicator describes if there is a significant difference between the last chosen breakpoint and the highest value in the OV_v. The higher the difference is, the better the detection of the breakpoints.

NoB

The last and most important evaluation parameter is the number of correctly detected breakpoints (NoB). The indexes in HV_i (positions in $\hat{\mathbf{d}}$) are taken as detected breakpoints in the tested signal. The noise in the tested signal can influence the finding of the correct breakpoint positions and therefore, it is considered that the breakpoint is detected correctly if the indicated position lies within an interval of $\pm$ two indexes from the true breakpoint position in the tested signal. The NoB can be the integer value from 0 to $S-1$. If all breakpoints are correctly detected, then $\text{NoB} = S-1$.

5.2.2 Evaluation of signal denoising performance

The quality of signal denoising is evaluated by two measurements – SNR and MSE, which are defined in the following subsections. To measure the denoising success, the clean $\mathbf{y}_{\text{clean}}$ and denoised $\mathbf{y}_{\text{denoised}}$ signals are needed. Two variants of denoised signals are produced during the whole process of signal segmentation. The first one $\hat{\mathbf{y}}$ is the signal obtained after using a proximal splitting algorithm ($\hat{\mathbf{y}} = \mathbf{P}\hat{\mathbf{x}}$). The second one $\dot{\mathbf{y}}$ is obtained after denoising of all detected signal segments ($\dot{\mathbf{y}} = \mathbf{P}\dot{\mathbf{x}}$). The evaluation of signal denoising can be performed, since the experiment dataset consists of noisy synthetic signals for which their clean variant is known.

SNR

With clean and denoised signals, the signal-to-noise ratio (SNR) can be determined, and it is defined as

$$\mathrm{SNR}\left(\mathbf{y}_{\mathrm{denoised}}, \mathbf{y}_{\mathrm{clean}}\right) = 20 \cdot \log_{10} \frac{\|\mathbf{y}_{\mathrm{clean}}\|_2}{\|\mathbf{y}_{\mathrm{denoised}} - \mathbf{y}_{\mathrm{clean}}\|_2}. \tag{5.24}$$

For the above-mentioned variants of denoised signals, the respective variants of SNR ($\mathrm{SNR}_{\hat{\mathbf{y}}}$, $\mathrm{SNR}_{\dot{\mathbf{y}}}$) are obtained.

MSE

The mean square error (MSE) measures the average distance of the denoised signal from the noiseless original, and is defined as

$$\mathrm{MSE}(\mathbf{y}_{\mathrm{denoised}}, \mathbf{y}_{\mathrm{clean}}) = \frac{1}{N}\|\mathbf{y}_{\mathrm{denoised}} - \mathbf{y}_{\mathrm{clean}}\|_2^2. \tag{5.25}$$

For the above-mentioned variants of denoised signals, the respective variants of MSE ($\mathrm{MSE}_{\hat{\mathbf{y}}}$, $\mathrm{MSE}_{\dot{\mathbf{y}}}$) are obtained.

5.3 Datasets for experiments

For the experiments, test datasets of synthetic signals and bases are created. The dataset of synthetic signals is in this Thesis represented by the group of piecewise-linear and piecewise-quadratic signals. The process of generating the synthetic signal is specified in Sec. 5.1.2.

5.3.1 Synthetic piecewise-linear signals

As elementary signals for creating a dataset of linear signals, 10 random piecewise-linear signals $\mathbf{y}_{\mathrm{clean}}$ ($K = 1$) with 6 signal segments ($S = 6$), and without jumps of length $N = 300$ are generated. For each elementary signal, 10 variations of the signal with $L = 10$ levels of the jump height $\mathbf{h} = [h_1, \ldots, h_{10}]$ are generated, examples of such signals are shown in Fig. 5.7. Maximum jump height h_{10} is 1/3 of the signal dynamic range, defined as the difference between the maximum and minimum value of $\mathbf{y}_{\mathrm{clean}}$, formally $\max(\mathbf{y}_{\mathrm{clean}}) - \min(\mathbf{y}_{\mathrm{clean}})$. Minimum jump height h_1 is 1/10 of the maximum jump height. Formally, h_l is defined as follows:

$$h_l = \frac{l \cdot h_L}{L}, \quad l = 1, \ldots, L. \tag{5.26}$$

Five SNR values are prescribed for the experiments: 15, 20, 25, 30, and 35 dB. For each signal and each SNR, 50 realisations of noise are generated, making a set of 27,500 noisy signals $\mathbf{y}$ in total. Examples of clean test signals $\mathbf{y}_{\mathrm{clean}}$ are depicted in Fig. 5.8a.

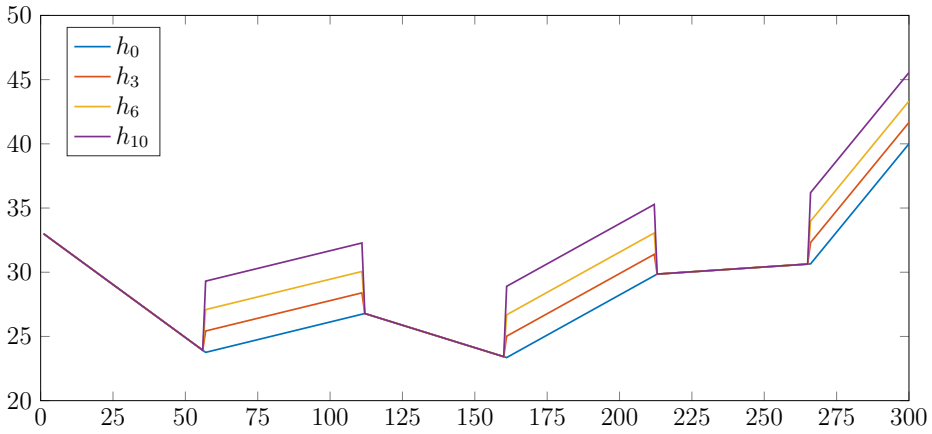
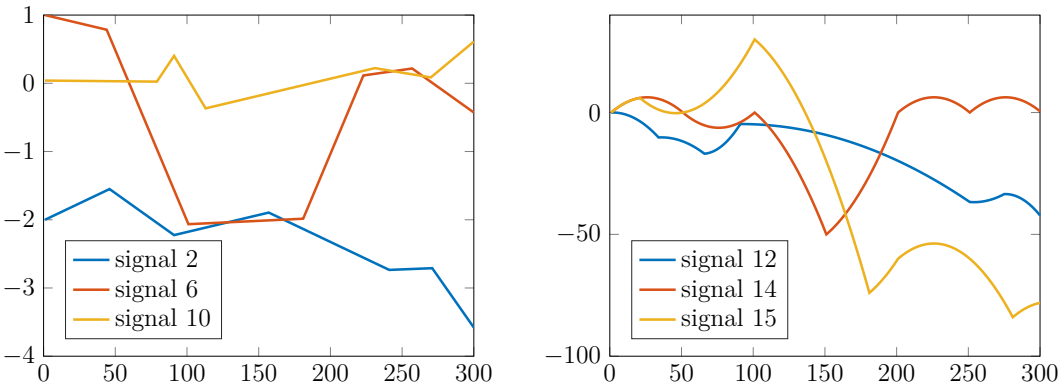


Fig. 5.7: Examples of used clean synthetic piecewise-linear signal with different jump heights h_0, h_3, h_6, h_{10}, where h_0 means zero jump height.



(a) synthetic piecewise-linear signals

(b) synthetic piecewise-quadratic signals

Fig. 5.8: Examples of used synthetic piecewise-linear and piecewise-quadratic signals. The left-hand side figure shows the examples of used clean synthetic piecewise-linear signals – signal 2, 6, and 10. The right-hand side figure shows the examples of used clean synthetic piecewise-quadratic signals – signal 12, 14, and 15.

5.3.2 Synthetic piecewise-quadratic signals

The dataset of piecewise-quadratic signals is created using the same process as the dataset of piecewise-linear signals, with one difference. At the beginning, 10 randomly piecewise-quadratic signals $\mathbf{y}_{\text{clean}}$ (K=2) are generated, instead of the set of 10 randomly generated piecewise-linear signals.

Examples of clean test signals $\mathbf{y}_{\text{clean}}$ are depicted in Fig. 5.8b.

5.3.3 Bases

For the experiments, several types of bases are exploited. For most of the experiments, only the modified standard basis $\mathbf{S}$ (see definition (5.15)) is used.

Out of four types of bases defined in Sec. 5.1.3, only the groups of the O-bases and R-bases are generated and used in this Thesis. This bases selection is formed on the results presented in our article [10], where it was shown that these two types of bases give the best results. Each group consists of 20 particular bases. Both O-bases and R-bases are generated according to the formulas described in Sec. 5.1.3.

5.4 Recovery problem – Total variation

It is supposed that the clean signal consists of several polynomial segments S, which do not have to be connected at the borders, hence jumps are allowed from segment to segment. The number of the segment borders is $S - 1$. The main aim is the detection of positions of segment borders, which are used for signal segmentation. As a secondary aim, denoising of the signal can be considered. According to the signal model description in Sec. 5.1.1, the parametrisation coefficients $\mathbf{x}_k$ are piecewise-constant and positions of the breakpoints should be clearly identifiable in individual $\nabla\mathbf{x}_k$ – the non-zero values indicate the positions of segment borders. Estimation of such parametrisation coefficients $\hat{\mathbf{x}}$ of the observed signal $\mathbf{y}$ is needed. Then the breakpoints are found in the estimated parametrisation coefficients $\hat{\mathbf{x}}$.

According to the above-mentioned assumptions, the following optimisation problem is formulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\nabla\mathbf{x}_0\|_0 + \cdots + \|\nabla\mathbf{x}_K\|_0 \quad \text{s.t.} \quad \|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2 \leq \delta, \tag{5.27}$$

where $\mathbf{x}$ are parametrisation coefficients, $\hat{\mathbf{x}}$ are estimated parametrisation coefficients, ∇ is the difference operator, $\mathbf{y}$ is the observed signal, $\mathbf{P}$ is the matrix of basis polynomials and δ is the parameter reflecting the noise level and model error.

According to the assumptions that the number of signal segments S is considerably lower than the length of the signal N and that $\nabla\mathbf{x}_k$ are sparse, the aim is to minimise the number of non-zero values in these vectors. Therefore, the functional $\|\cdot\|_0$, which counts the non-zero elements of the vector, is used. The $\|\nabla\cdot\|_0$ indicates the number of possible breakpoints in the vector. Using this functional $\|\nabla\cdot\|_0$ on each parametrisation vector $\mathbf{x}_k$, the possible breakpoints in each individual parametrisation coefficients $(\mathbf{x}_0, \ldots, \mathbf{x}_K)$ are estimated. The aim is to find piecewise-constant parametrisation coefficients with a minimum number of possible segment borders, which satisfy the condition $\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2 \leq \delta$.

The model (5.27) is over-parametrised, having $K+1$ times as many unknowns as the signal samples N, and profits from comprehensible prior knowledge about the behaviour of the parameters.

Whereas the solving of the problem (5.27) is NP-hard, this problem is unaffordable in practical applications. In the following subsection, relaxation techniques are used and therefore the problem (5.27) must be reformulated.

5.4.1 Definition of recovery problem

As noted in Sec. 1.3.1, the proximal algorithms are designed to solve the optimisation problems in unconstrained form. Therefore, the recovery problem (5.27) is reformulated to the unconstrained version as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\left\|\mathbf{y}-\mathbf{P}\mathbf{x}\right\|_2^2 + \tau_0\,\mathrm{TV}(\mathbf{x}_0) + \cdots + \tau_K\,\mathrm{TV}(\mathbf{x}_K), \tag{5.28}$$

where $\mathbf{x}$ are parametrisation coefficients, $\hat{\mathbf{x}}$ are estimated parametrisation coefficients, $\mathbf{y}$ is the observed signal, $\mathbf{P}$ is the matrix of polynomials, $\mathrm{TV}(\cdot)$ is the total variation, and τ_k are regularisation weights, which are set carefully according to the noise level and prior experience.

The formulation (5.28) is ℓ_1-based unconstrained approximation of formulation (5.27), where the total variation $\mathrm{TV}(\cdot)$ is the relaxed counterpart of $\|\nabla\cdot\|_0$, which is defined as follows:

$$\mathrm{TV}(\mathbf{z}) = \|\nabla\mathbf{z}\|_1 = \sum_{i=1}^{N-1} |z_{i+1}-z_i|. \tag{5.29}$$

Because the below-described experiments are performed on both linear and quadratic signals, the definition of the general recovery problem (5.28) can be easily simplified for these special cases. For simplification, the recovery problem is introduced only for the linear signal. For the piecewise-linear signal, K is set to $K=1$. In the case that the modified standard basis $\mathbf{S}$ (see definition (5.15)) is used, each linear segment can be individually parametrised by a constant offset and a constant slope. Therefore, the used polynomial matrix $\mathbf{P}$ contains these two standard polynomials – constant offset and constant slope. The basis polynomial $\mathbf{p}_0$ is a vector of ones, which means that $\mathbf{P}_0$ is the identity matrix, and $\mathbf{p}_1$ is a vector with linearly growing entries $\frac{1}{N}, \frac{2}{N}, \frac{3}{N}, \ldots, 1$, where N represents the number of signal elements.

The recovery problem for a linear signal is defined as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\left\|\mathbf{y}-\mathbf{P}\mathbf{x}\right\|_2^2 + \tau_0\,\mathrm{TV}(\mathbf{x}_0) + \tau_1\,\mathrm{TV}(\mathbf{x}_1), \tag{5.30}$$

where the first term is the "data fidelity" term, enforcing the estimate to approximately correspond to the observation $\mathbf{y}$. The Euclidean ℓ_2-norm reflects the fact that gaussianity of the noise is assumed.

The second term $\tau_0 \operatorname{TV}(\mathbf{x}_0) + \tau_1 \operatorname{TV}(\mathbf{x}_1)$ in the problem (5.30) is the "penalisation" term. This functional mathematically transcripts the desired properties of the estimate.

Numerical solution of the recovery problem (5.30) using the Forward-backward and the Douglas–Rachford algorithms was presented in our paper [5].

5.4.2 Algorithms used

The recovery problem (5.28) can be solved via either the Forward-backward or the Douglas–Rachford algorithm. These two mentioned algorithms are able to

$$\text{minimise} \quad f(\mathbf{x}) + g(\mathbf{x}). \tag{5.31}$$

According to the general recovery problem (5.28), the functions are assigned such that $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2$, which is smooth, and function $g(\mathbf{x}) = g(\mathbf{x}_0, \ldots, \mathbf{x}_K) = \tau_0 \operatorname{TV}(\mathbf{x}_0) + \cdots + \tau_K \operatorname{TV}(\mathbf{x}_K)$, which is non-smooth.

According to the recovery problem (5.30) for linear signals, smooth function $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2$ and non-smooth function $g(\mathbf{x}) = g(\mathbf{x}_0, \mathbf{x}_1) = \tau_0 \operatorname{TV}(\mathbf{x}_0) + \tau_1 \operatorname{TV}(\mathbf{x}_1)$ are assigned.

Forward-backward algorithm

The Forward-backward algorithm (FB) solves the problem (5.31) where function f is β-Lipschitz differentiable with constant $\beta < \infty$ [31], and function g is a non-smooth function. The main computational steps within each iteration of the FB algorithm are the gradient step with respect to f and the proximal step with respect to g. The general form of the FB algorithm is presented in Sec. 1.3.3 in Algorithm 1.

For the differentiable function f, its gradient must be found, which is $\mathbf{P}^\top(\mathbf{P} \cdot -\mathbf{y})$. For the FB algorithm, its Lipschitz constant β also needs to be established to guarantee convergence. It can be shown that β equals the square of the maximum singular value of $\mathbf{P}$, which is the operator/spectral norm of $\mathbf{P}$. Since it is known that for spectral norms it holds $\|\mathbf{P}\|^2 = \|\mathbf{P}^\top\mathbf{P}\| = \|\mathbf{P}\mathbf{P}^\top\|$, it is sufficient to find the maximum eigenvalue of $\mathbf{P}\mathbf{P}^\top$. Since the rows of $\mathbf{P}$ are mutually orthogonal, the product $\mathbf{P}\mathbf{P}^\top$ is a diagonal matrix with diagonal elements. And since eigenvalues of the diagonal matrix are exactly the elements on its diagonal, the spectral norm of the matrix is the maximum element on the diagonal.

For the standard basis consisting of $K+1$ polynomial, the largest element is in the lower-right corner and $\beta = K + 1$.

For the standard basis consisting of 2 polynomials, where $K = 1$, the product $\mathbf{P}\mathbf{P}^\top$ is a diagonal matrix with diagonal elements $[1 + \left(\frac{1}{N}\right)^2,\ 1 + \left(\frac{2}{N}\right)^2, \ldots, 1 + 1]$. Therefore, the desired constant is $\beta = 2 = K + 1$.

Function g is a non-smooth function and for the FB algorithm, the proximal operator of g must be found, which will be readily done utilising the fact that $\mathbf{x}_0, \ldots, \mathbf{x}_K$ are separable. Plugging g into Eq. (1.20), the following operator is obtained:

$$\begin{aligned}\underset{g}{\operatorname{prox}}(\mathbf{x}) &= \underset{\mathbf{z}=[\mathbf{z}_0,\ldots,\mathbf{z}_K]\in\mathbb{R}^{N(K+1)}}{\arg\min}\; g\left(\begin{bmatrix}\mathbf{z}_0\\ \vdots \\ \mathbf{z}_K\end{bmatrix}\right) + \frac{1}{2}\|\mathbf{z}-\mathbf{x}\|_2^2 \\ &= \underset{\mathbf{z}_0,\ldots,\mathbf{z}_K}{\arg\min}\; \tau_0\,\mathrm{TV}(\mathbf{z}_0) + \frac{1}{2}\|\mathbf{z}_0-\mathbf{x}_0\|_2^2 + \cdots \\ &\qquad + \tau_K\,\mathrm{TV}(\mathbf{z}_K) + \frac{1}{2}\|\mathbf{z}_K-\mathbf{x}_K\|_2^2 \\ &= \begin{bmatrix}\operatorname{prox}_{\tau_0\mathrm{TV}(\cdot)}(\mathbf{x}_0)\\ \vdots \\ \operatorname{prox}_{\tau_K\mathrm{TV}(\cdot)}(\mathbf{x}_K)\end{bmatrix}.\end{aligned} \tag{5.32}$$

For the linear signal, it applies

$$\underset{g}{\operatorname{prox}}(\mathbf{x}) = \begin{bmatrix}\operatorname{prox}_{\tau_0\mathrm{TV}(\cdot)}(\mathbf{x}_0)\\ \operatorname{prox}_{\tau_1\mathrm{TV}(\cdot)}(\mathbf{x}_1)\end{bmatrix}, \tag{5.33}$$

i.e. stacking two independent half-size vectors, containing (scaled) proximal operators of the TV functional.

In contrast to the general form of the FB algorithm (see Algorithm 1), the parameters θ and γ are set once at the beginning, instead of resetting these parameters in each iteration step.

The FB algorithm for polynomial signals is defined in Algorithm 6. The particular structure of the modified standard basis $\mathbf{P}$ can be exploited to derive the following form of the FB algorithm for the linear signal defined in Algorithm 7. Note that acceleration techniques like FISTA [112] can be applied.

Douglas–Rachford algorithm

The Douglas–Rachford (DR) algorithm is also suitable for solving (5.31) when both functions are generally nonsmooth (nevertheless, the DR algorithm can be applied in the case of smooth function f as well) [31]. Each iteration consists of two proximal steps related to functions f and g. The general form of the Douglas–Rachford algorithm is presented in Sec. 1.3.3 in Algorithm 2.

Functions f and g are defined in the same way as above. The main difference between the FB and the DR algorithm is that in the DR algorithm, no gradient step is exploited but rather the proximal operator of f is used, see Eq. (1.21). For the defined function f, it is clear that $\zeta = 1$ and plugging f into Eq. (1.21) results in

$$\underset{f}{\operatorname{prox}}(\mathbf{x}) = (\mathbf{I} + \mathbf{P}^\top\mathbf{P})^{-1}(\mathbf{x} + \mathbf{P}^\top\mathbf{y}), \tag{5.34}$$

Algorithm 6: The Forward-backward algorithm solving (5.28) – polynomial signal

Input: Functions f, g, input signal $\mathbf{y} \in \mathbb{R}^N$

Output: $\hat{\mathbf{x}} = \mathbf{x}^{i+1}$

Set $\varepsilon = 1/(K+1) - 0.01$, $\gamma = \varepsilon + 0.01 = 1/(K+1)$ and $\theta = 0.99$

Set initial variables $\mathbf{x}^0 \in \mathbb{R}^{N(K+1)}$

for $i = 0, 1, \ldots$ *until convergence* **do**

$$\mathbf{q}^i = \mathbf{x}^i - \gamma(\mathbf{P}^\top(\mathbf{P}\mathbf{x} - \mathbf{y}))$$

$$\mathbf{x}^{i+1} = \begin{bmatrix} \mathbf{x}_0^i \\ \vdots \\ \mathbf{x}_K^i \end{bmatrix} + \theta \left(\begin{bmatrix} \mathrm{prox}_{\gamma\tau_0 \mathrm{TV}(\cdot)}(\mathbf{q}_0^i) \\ \vdots \\ \mathrm{prox}_{\gamma\tau_K \mathrm{TV}(\cdot)}(\mathbf{q}_K^i) \end{bmatrix} - \begin{bmatrix} \mathbf{x}_0^i \\ \vdots \\ \mathbf{x}_K^i \end{bmatrix} \right)$$

return $\mathbf{x}^{i+1}$

Algorithm 7: The Forward-backward algorithm solving (5.30) – linear signal

Input: Functions f, g, input signal $\mathbf{y} \in \mathbb{R}^N$

Output: $\hat{\mathbf{x}} = \mathbf{x}^{i+1}$

Set $\varepsilon = 0.49$, $\gamma = 0.5$ and $\theta = 0.99$

Set initial variables $\mathbf{x}^0 \in \mathbb{R}^{2N}$

for $i = 0, 1, \ldots$ *until convergence* **do**

$$\mathbf{q}^i = \begin{bmatrix} \mathbf{x}_0^i \\ \mathbf{x}_1^i \end{bmatrix} - \gamma \left(\begin{bmatrix} \mathbf{x}_0^i + \mathbf{P}_1 \mathbf{x}_1^i \\ \mathbf{P}_1(\mathbf{x}_0^i + \mathbf{P}_1 \mathbf{x}_1^i) \end{bmatrix} - \begin{bmatrix} \mathbf{y} \\ \mathbf{P}_1 \mathbf{y} \end{bmatrix} \right)$$

$$\mathbf{x}^{i+1} = \begin{bmatrix} \mathbf{x}_0^i \\ \mathbf{x}_1^i \end{bmatrix} + \theta \left(\begin{bmatrix} \mathrm{prox}_{\gamma\tau_0 \mathrm{TV}(\cdot)}(\mathbf{q}_0^i) \\ \mathrm{prox}_{\gamma\tau_1 \mathrm{TV}(\cdot)}(\mathbf{q}_1^i) \end{bmatrix} - \begin{bmatrix} \mathbf{x}_0^i \\ \mathbf{x}_1^i \end{bmatrix} \right)$$

return $\mathbf{x}^{i+1}$

where $\mathbf{I}$ is the identity matrix. Note that $(\mathbf{I} + \mathbf{P}^\top\mathbf{P})^{-1}$ and $\mathbf{P}^\top\mathbf{y}$ can be precomputed. Computation of the inverse matrix is time-consuming and precomputation helps to make the algorithm faster. Theorem 1 in the Appendix A introduces the explicit formula for computing the inversion, which is possible thanks to the multi-diagonal structure of $\mathbf{P}^\top\mathbf{P}$.

The proximal operator of g for a general polynomial signal is defined in (5.32) and for a linear signal in (5.33) – it is the same for both the FB and the DR algorithms.

In contrast to the general form of the DR algorithm (see Algorithm 2), the parameter θ is set once at the beginning – instead of resetting this parameter in each iteration step.

The DR algorithm for the polynomial signal is presented in Algorithm 8, and for the linear signal is defined in Algorithm 9.

Algorithm 8: The Douglas–Rachford algorithm solving (5.28) – polynomial signal

Input: Functions f, g, input signal $\mathbf{y} \in \mathbb{R}^N$

Output: $\hat{\mathbf{x}} = \mathbf{x}^{i+1}$

Set initial variables $\mathbf{q}^0 \in \mathbb{R}^{N(K+1)}$

Set $\gamma = 0.01$, $\varepsilon = 0.01$, $\theta = 0.99$

for $i = 0, 1, \ldots$ *until convergence* **do**

$$\mathbf{x}^i = \left(\mathbf{I} + \mathbf{P}^\top \mathbf{P}\right)^{-1} \left(\mathbf{q}^i - \mathbf{P}^\top \mathbf{y}\right)$$

$$\mathbf{q}^{i+1} = \begin{bmatrix} \mathbf{q}_0^i \\ \vdots \\ \mathbf{q}_K^i \end{bmatrix} + \theta \left(\begin{bmatrix} \mathrm{prox}_{\gamma\tau_0 \mathrm{TV}(\cdot)}(2\mathbf{x}_0^i - \mathbf{q}_0^i) \\ \vdots \\ \mathrm{prox}_{\gamma\tau_K \mathrm{TV}(\cdot)}(2\mathbf{x}_K^i - \mathbf{q}_K^i) \end{bmatrix} - \begin{bmatrix} \mathbf{x}_0^i \\ \vdots \\ \mathbf{x}_K^i \end{bmatrix} \right)$$

return $\mathbf{x}^{i+1}$

Algorithm 9: The Douglas–Rachford algorithm solving (5.30) – linear signal

Input: Functions f, g, input signal $\mathbf{y} \in \mathbb{R}^N$

Output: $\hat{\mathbf{x}} = \mathbf{x}^{i+1}$

Set initial variables $\mathbf{q}^0 \in \mathbb{R}^{2N}$

Set $\gamma = 0.01$, $\varepsilon = 0.01$, $\theta = 0.99$

for $i = 0, 1, \ldots$ *until convergence* **do**

$$\mathbf{x}^i = \left(\mathbf{I} + \mathbf{P}^\top \mathbf{P}\right)^{-1} \left(\mathbf{q}^i - \mathbf{P}^\top \mathbf{y}\right)$$

$$\mathbf{q}^{i+1} = \begin{bmatrix} \mathbf{q}_0^i \\ \mathbf{q}_1^i \end{bmatrix} + \theta \left(\begin{bmatrix} \mathrm{prox}_{\gamma\tau_0 \mathrm{TV}(\cdot)}(2\mathbf{x}_0^i - \mathbf{q}_0^i) \\ \mathrm{prox}_{\gamma\tau_1 \mathrm{TV}(\cdot)}(2\mathbf{x}_1^i - \mathbf{q}_1^i) \end{bmatrix} - \begin{bmatrix} \mathbf{x}_0^i \\ \mathbf{x}_1^i \end{bmatrix} \right)$$

return $\mathbf{x}^{i+1}$

5.4.3 Experiments

In this section, the results obtained from the 1D signal segmentation and denoising process with the recovery problem (5.28) solved by the Forward-backward (FB) and the Douglas–Rachford (DR) algorithms are evaluated.

The experiments are performed on datasets of linear and quadratic synthetic signals, which are introduced in Sec. 5.3. For the purpose of these experiments, the modified standard basis ($\mathbf{P} = \mathbf{S}$) of size $N \times (K+1)$ is used, see Sec. 5.1.3 for more details. For the first experiments with linear signals, the used modified standard basis consists of 2 polynomials ($K = 1$), and for the second experiments, it consists of 3 polynomials ($K = 2$). For experiments on quadratic signals, only the modified standard basis consisting of 3 polynomials ($K = 2$) is used.

The results obtained by the FB and the DR algorithms are evaluated in relation to the signal properties and used polynomial basis, and compared to each other.

Several parameters need to be set for both the "Optimisation step" and the "Detection of segment borders step".

The first group of parameters is set for the "Optimisation step" using proximal splitting algorithm. The list of parameters for the experiments using the FB algorithm is given in the Table 5.1 – for $K = 1$ and $K = 2$. The list of parameters for the DR algorithm is given in the Table 5.2. Parameters of the FB algorithm (β, ϵ, γ, and θ) and parameters of the DR algorithm (ϵ, γ and θ) are explained in Sec. 1.3.3. The FB and the DR algorithms are stopped if either the convergence criterion or the maximum number of iterations ($\mathrm{MAX_{IT}}$) is reached.

Regularisation weights τ_k are set with respect to σ_e, where σ_e is the standard deviation of the noise, which is known from the stage of creating synthetic noisy signals (see Sec. 5.1.2). Tables 5.1 and 5.2 show that the regularisation parameters τ_k most favour the constant offset. The influence of polynomials $\mathbf{p}_k$ with higher k is decreasing. The regularisation parameters τ_k are created by a product of the noise level, which is represented by σ_e, and influence of the particular polynomial $\mathbf{p}_k$, which is described by a constant. These constants are set manually according to the previous experience – they have been tuned to obtain the parametrisation coefficients as close to piecewise-constant as possible – and are the same for processing of all the synthetic signals. Anyway, the regularisation weights τ_k differ with each type and SNR of the analysed signal due to the changing value of σ_e. It is possible that the individually tuned regularisation parameters τ_k for each particular signal can lead to better results, but the individual tuning of the regularisation parameters for thousands of signals would be an incredibly time consuming process. Therefore, the above-described compromise in setting of the regularisation parameters was chosen.

The second group of parameters is set for "Detection of segment borders" step and they are listed in Tables 5.1 and 5.2. The parameter l_{m} represents the window length of the median filter. The parameter λ is a threshold, which decides on the selection of the potential positions of segment borders.

The obtained results are evaluated from two points of view – breakpoint detection accuracy, and signal denoising performance. The metrics used for evaluating the breakpoint detection accuracy are AAR, MMR, and NoB, which are described in Sec. 5.2.1. Furthermore, the metrics for the evaluation of the signal denoising performance are $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$, and they are introduced in Sec. 5.2.2.

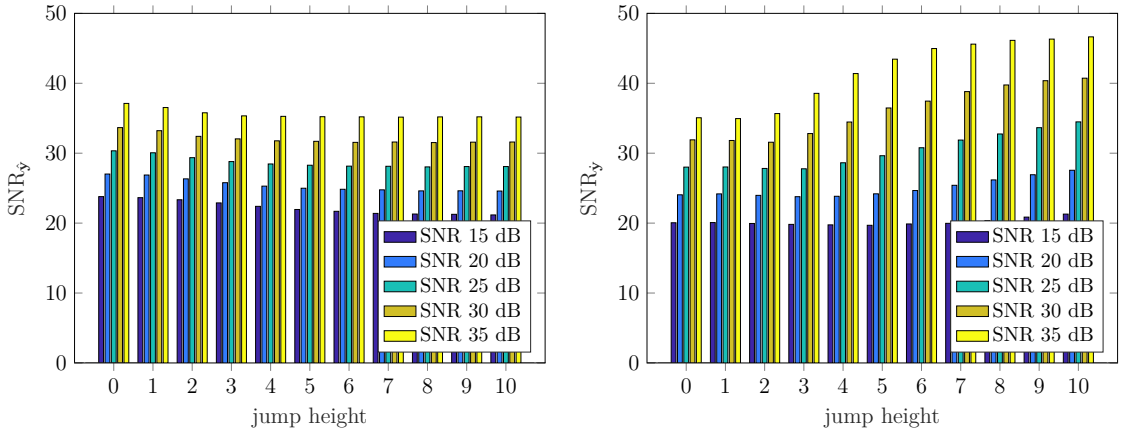
In addition $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$, $\mathrm{SNR}_{\hat{\mathbf{y}}}$ and $\mathrm{MSE}_{\hat{\mathbf{y}}}$ were also introduced. In the following evaluation, only metrics computed on $\dot{\mathbf{y}}$ will be used, i.e., the output signal of the whole "1D signal segmentation and denoising process". The signal $\hat{\mathbf{y}}$ is the output signal of the FB/DR algorithm. Comparisons of the SNR and MSE results between the quadratic signals $\hat{\mathbf{y}}$ and $\dot{\mathbf{y}}$ are shown in Figs. 5.9 and 5.10. The character of the results is the same for all testing cases independently on the optimisation algo-

Tab. 5.1: Parameter settings for the FB algorithm. Table contains the specific setting of parameters needed in 1D signal segmentation and denoising process using the FB algorithm.

Step	**Setting**		
	Parameter	**Value (for $K = 1$)**	**Value (for $K = 2$)**
FB algorithm	$\mathrm{MAX_{IT}}$	3000	3000
	β	2	3
	ϵ	0.49	$1/3 - 0.01$
	γ	0.5	$1/3$
	θ	0.99	0.99
	τ_0	$\sigma_e \cdot 5$	$\sigma_e \cdot 5$
	τ_1	$\sigma_e \cdot 3.5$	$\sigma_e \cdot 3.5$
	τ_2	–	$\sigma_e \cdot 2.5$
Detection of segment borders	l_{m}	5	5
	λ	0.2	0.2

Tab. 5.2: Parameter settings for the DR algorithm. Table contains the specific setting of parameters needed in 1D signal segmentation and denoising process using the DR algorithm.

Step	**Setting**		
	Parameter	**Value (for $K = 1$)**	**Value (for $K = 2$)**
DR algorithm	$\mathrm{MAX_{IT}}$	3000	3000
	ϵ	0.01	0.01
	γ	0.01	0.01
	θ	0.99	0.99
	τ_0	$\sigma_e \cdot 200$	$\sigma_e \cdot 200$
	τ_1	$\sigma_e \cdot 150$	$\sigma_e \cdot 150$
	τ_2	–	$\sigma_e \cdot 130$
Detection of segment borders	l_{m}	5	5
	λ	0.2	0.2

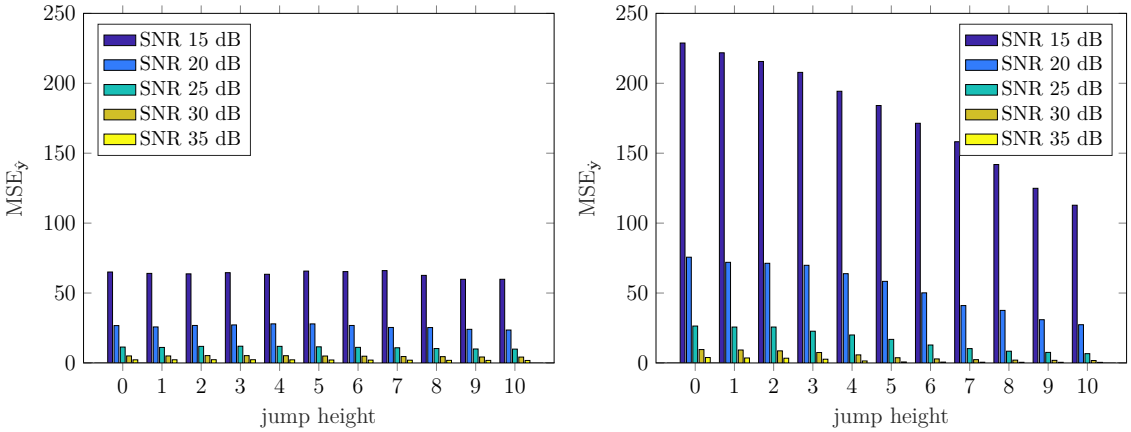


(a) SNR$_{\hat{\mathbf{y}}}$ for FB – quadratic signals (b) SNR$_{\dot{\mathbf{y}}}$ for FB – quadratic signals

Fig. 5.9: Comparison of SNR$_{\hat{\mathbf{y}}}$ and SNR$_{\dot{\mathbf{y}}}$ for the FB algorithm for quadratic signals.

rithm (FB vs DR) and the test signal dataset (linear vs quadratic). From Fig. 5.9a, it follows that SNR$_{\hat{\mathbf{y}}}$ is growing with the signal SNR, and for particular signal SNR, the increase in SNR$_{\hat{\mathbf{y}}}$ is almost independent of the jump height. The increase in SNR$_{\dot{\mathbf{y}}}$ is slightly higher for the smaller jump height. The SNR$_{\dot{\mathbf{y}}}$ is increasing with higher signal SNR and greater jump height (see Fig. 5.9b). For the signals with small jump height, the results are better for SNR$_{\hat{\mathbf{y}}}$. The reason is that when the detected segments are interpolated by a polynomial, the deviation from the original signal can increase because the signal segments could be wrongly detected, which is more often in signals with lower SNR and lower jump height. According to the above-mentioned conclusions, only the SNR$_{\dot{\mathbf{y}}}$ is used for the evaluation. The same applies to the MSE metric (see Fig. 5.10) and, therefore, only MSE$_{\dot{\mathbf{y}}}$ is chosen for the evaluation (MSE$_{\hat{\mathbf{y}}}$ is omitted). Between the SNR$_{\dot{\mathbf{y}}}$ and MSE$_{\dot{\mathbf{y}}}$ there is an indirect proportion – the higher the SNR$_{\dot{\mathbf{y}}}$ the lower the MSE$_{\dot{\mathbf{y}}}$.

All evaluation metrics are calculated for each test signal. To be able to evaluate the success of an algorithm, the average across the signals is computed according to specific criteria – the type of signal, jump height, and SNR. For both linear and quadratic signals, 55 averages are calculated – each across of 500 signals with the same SNR and jump height. Results of particular evaluation metrics are shown in the form of graphs, where each column of the graph represents the average across signals with the same SNR and jump height. Comparisons of the two algorithms are also represented as graphs, where each bar of such a graph represents the difference between the averaged values obtained via the compared algorithms. This comparison is done for each evaluation metric.



(a) $\text{MSE}_{\hat{\mathbf{y}}}$ for DR – quadratic signals (b) $\text{MSE}_{\dot{\mathbf{y}}}$ for DR – quadratic signals

Fig. 5.10: Comparison of $\text{MSE}_{\hat{\mathbf{y}}}$ and $\text{MSE}_{\dot{\mathbf{y}}}$ for the DR algorithm for quadratic signals.

All the experiments in this and all further sections were performed using Matlab, and for some of the proximal algorithms, the UnLocBox toolbox[1] [34] was used.

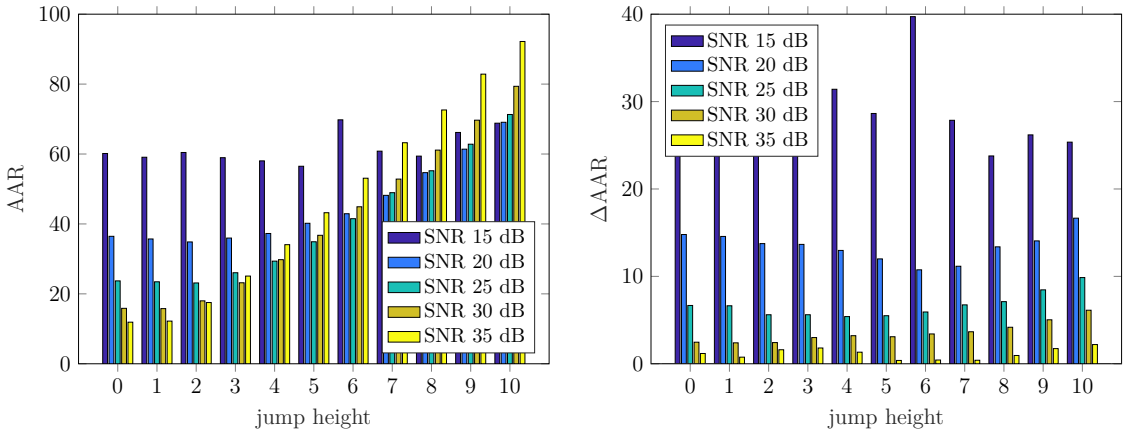
Linear synthetic signals

The first set of experiments is performed on the dataset of linear synthetic signals, which consists of 27,500 noisy signals in total. For the purpose of these experiments, the polynomial basis $\mathbf{P}$ is the modified standard basis $\mathbf{S}$ of size $N \times (K+1)$, where $K = 1$ or $K = 2$ (depending on the type of the experiment). Two items are subject to vary within the experiments, configuring the problem (5.30):

- the input signal $\mathbf{y}$,
- regularisation parameters τ_0, τ_1 (and τ_2 for $K = 2$).

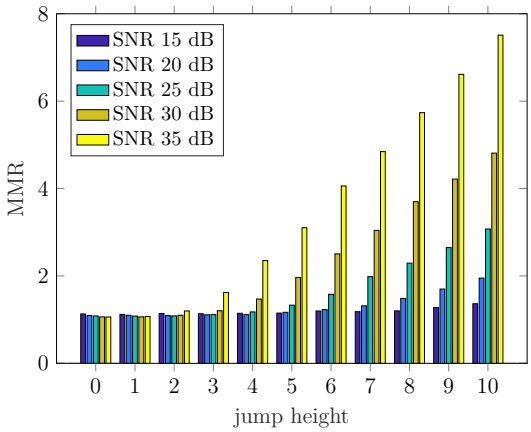
Evaluation of the FB algorithm At first, the results obtained using the 2-polynomial basis $\mathbf{P}$ are evaluated. AAR metric is shown in Fig. 5.11a. For smaller jump height, the AAR is higher for the lower SNR. For greater jump height, the effect is opposite. For both MMR and NoB, it holds that the value is higher with greater jump height and higher SNR, see Figs. 5.11c and 5.11e. From NoB evaluation, it follows that the maximum of correctly detected breakpoints is easier to achieve with higher SNR (lower noise level) and significant jump height between signal segments, as expected. Due to the character of the processed signals, the maximum achievable NoB is 5. The $\text{SNR}_{\dot{\mathbf{y}}}$ is more increasing with greater jump height between the signal segments. $\text{MSE}_{\dot{\mathbf{y}}}$ values are closely related to the resulting $\text{SNR}_{\dot{\mathbf{y}}}$ values – the higher $\text{SNR}_{\dot{\mathbf{y}}}$ the lower $\text{MSE}_{\dot{\mathbf{y}}}$ – see Fig. 5.12c compared with Fig. 5.12a.

[1] https://lts2.epfl.ch/unlocbox

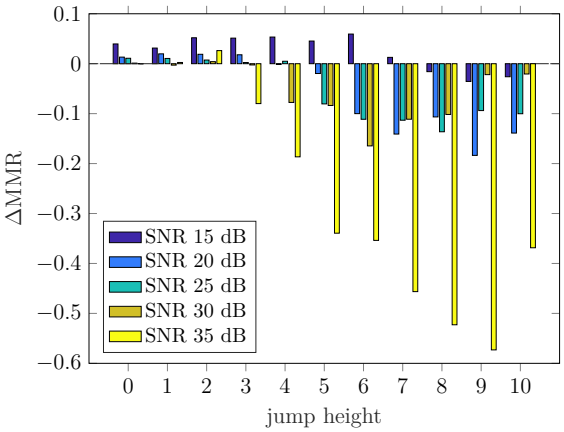


(a) AAR, FB, 2-polynomial basis

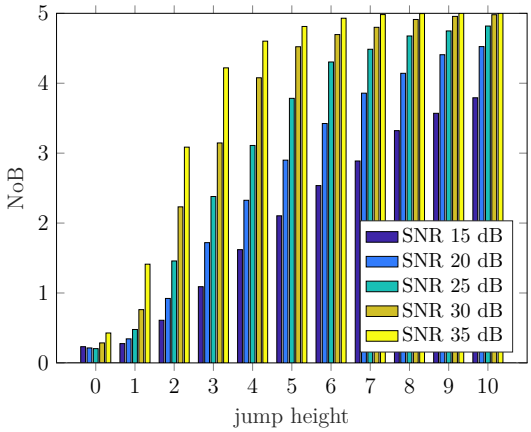
(b) AAR, FB vs DR, 2-polynomial basis



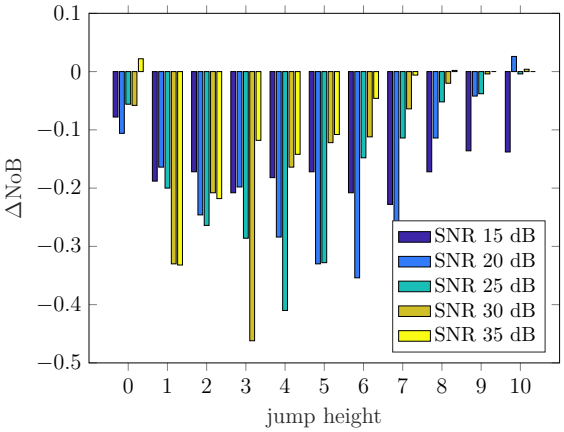
(c) MMR, FB, 2-polynomial basis



(d) MMR, FB vs DR, 2-polynomial basis

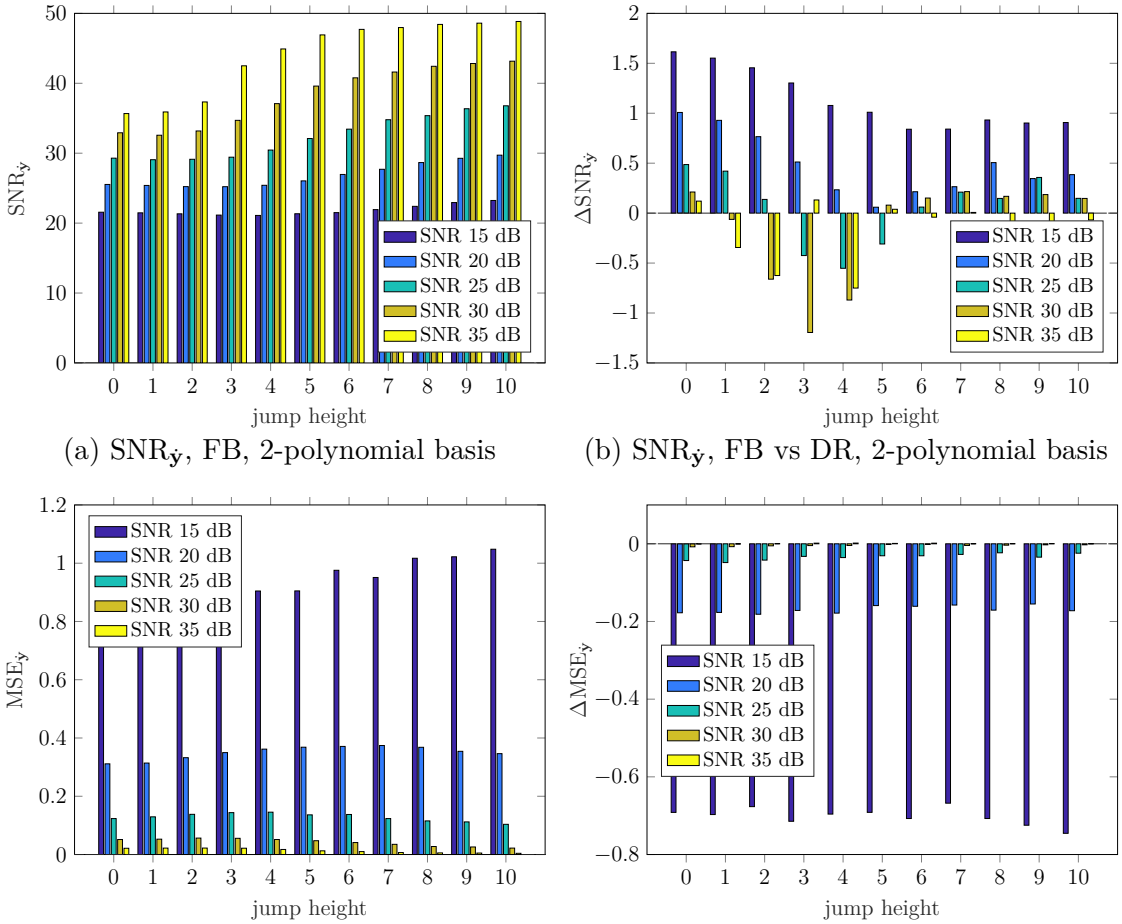


(e) NoB, FB, 2-polynomial basis



(f) NoB, FB vs DR, 2-polynomial basis

Fig. 5.11: AAR, MMR and NoB results for the FB algorithm using 2-polynomial basis $\mathbf{P}$ and their comparison with the DR algorithm. The left-hand side figures show the results for the FB algorithm. The right-hand side figures show the difference between results obtained with the FB and the DR algorithm. Presented results are for linear signals.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, FB, 2-polynomial basis

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, FB vs DR, 2-polynomial basis

(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, FB, 2-polynomial basis

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, FB vs DR, 2-polynomial basis

Fig. 5.12: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the FB algorithm using 2-polynomial basis $\mathbf{P}$ and their comparison with the DR algorithm. The left-hand side figures show the results for the FB algorithm. The right-hand side figures show the difference between results obtained with the FB and the DR algorithm. Presented results are for linear signals.

The synthetic linear signals were also processed using a 3-polynomial basis $\mathbf{P}$. From the results it follows that using 2-polynomial basis for processing of linear signals gives better results in most cases. This result can be explained by the fact that the 2-polynomial basis contains 2 linear polynomials compared to the 3-polynomial basis with 2 linear polynomials and 1 parabolic polynomial. Modelling of the linear signal is easier with only linear polynomials, since the usage of the exponential polynomial can in this case cause some deviation.

Evaluation of the DR algorithm Also, for the DR algorithm, the 2- and 3-polynomial bases $\mathbf{P}$ were used. The character of the results is very similar to the results obtained with the FB algorithm – for both 2- and 3-polynomial bases.

Comparison of the FB and the DR algorithm results The comparison of the FB and the DR algorithms is shown in the form of graphs in Figs. 5.11 and 5.12, where the left-hand side graphs show the results for the FB algorithm using 2-polynomial basis and the graphs on the right-hand side show the difference between results obtained with the FB and the DR algorithms. Values above zero mean that better results are achieved with the FB algorithm and values below zero mean better results for the DR algorithm – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, it applies the opposite.

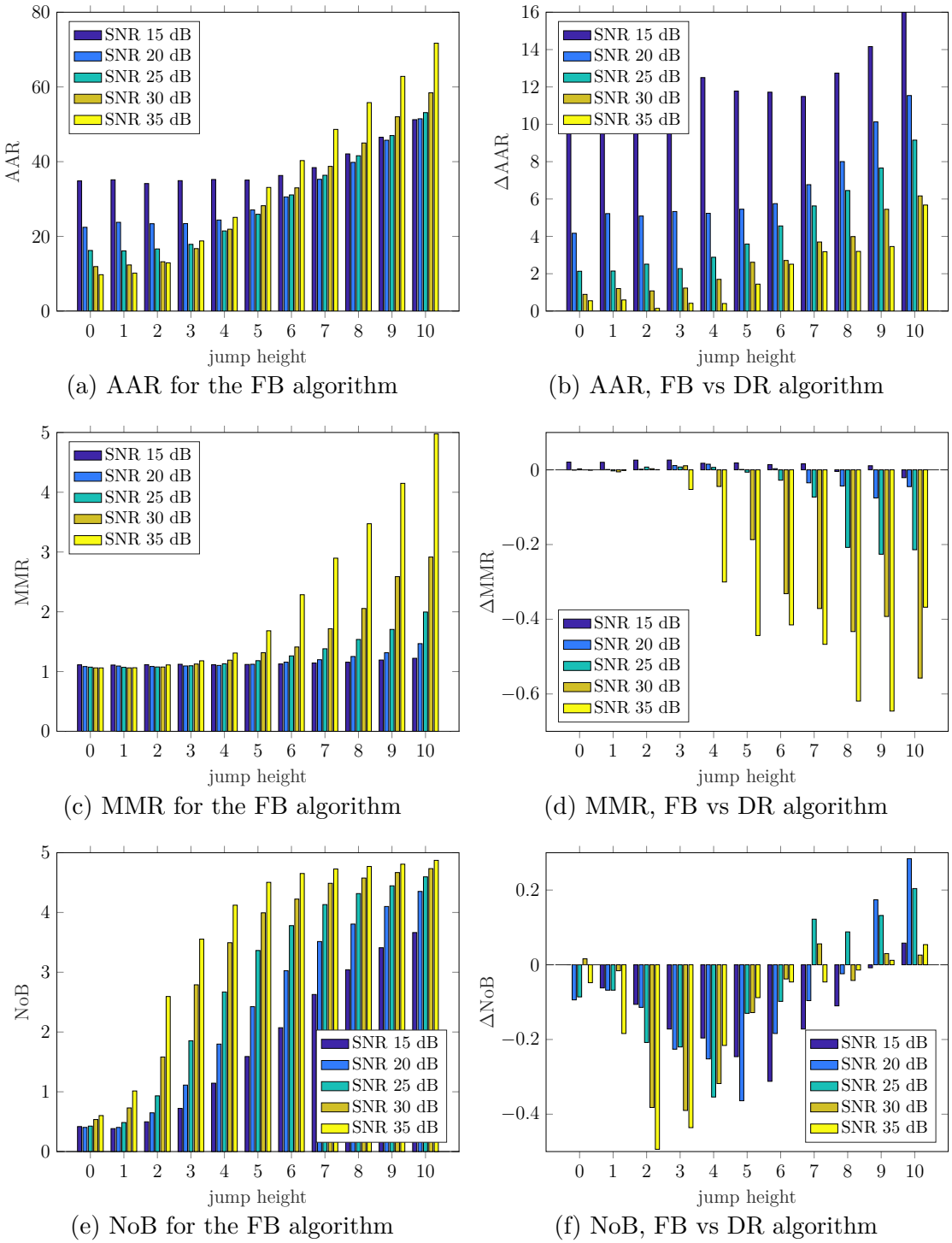
Evaluation of the breakpoint detection reveals that the DR algorithm provides better results than the FB algorithm. Results are a little bit tricky, because the AAR evaluation is undoubtedly better for the FB algorithm. Nevertheless, MMR and NoB values are better for the DR algorithm. The most important evaluation criterion in breakpoint detection is the NoB, i.e., the number of correctly detected breakpoints. From this point of view, better results in breakpoint detection are obtained with the DR algorithm (see Fig. 5.11). Better AAR result for the FB algorithm can be connected with the fact that AAR is computed from the average of the points with the highest value, but these points do not have to be the right detected breakpoints. Evaluation of the denoised signal is in both cases better for the FB algorithm (see Fig. 5.12).

Quadratic signals

The second set of experiments is performed on the dataset of quadratic synthetic signals, which consists of 27,500 noisy signals in total. For the purpose of these experiments, the polynomial basis $\mathbf{P}$ is the modified standard basis $\mathbf{S}$ of size $N \times (K+1)$, where $K = 2$. Two items are subject to vary within the experiments, configuring the problem (5.28):

- the input signal $\mathbf{y}$,
- regularisation parameters τ_0, τ_1 and τ_2.

Evaluation of the FB and the DR algorithm The results are shown in the form of graphs, see Figs. 5.13 and 5.14. In both figures, the graphs on the left-hand side show the results obtained using the FB algorithm, and the graphs on the right-hand side show the comparison of results obtained using the FB and the DR algorithms in form of differences, in the same way as for the linear signals. The character of the results is very similar to the results obtained for linear signals. For AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$, the results are better with higher SNR and greater jump height. For $\mathrm{MSE}_{\dot{\mathbf{y}}}$, it applies the opposite. This can be seen on the left-hand side graphs. The comparisons of the FB and the DR algorithms show that AAR



(a) AAR for the FB algorithm

(b) AAR, FB vs DR algorithm

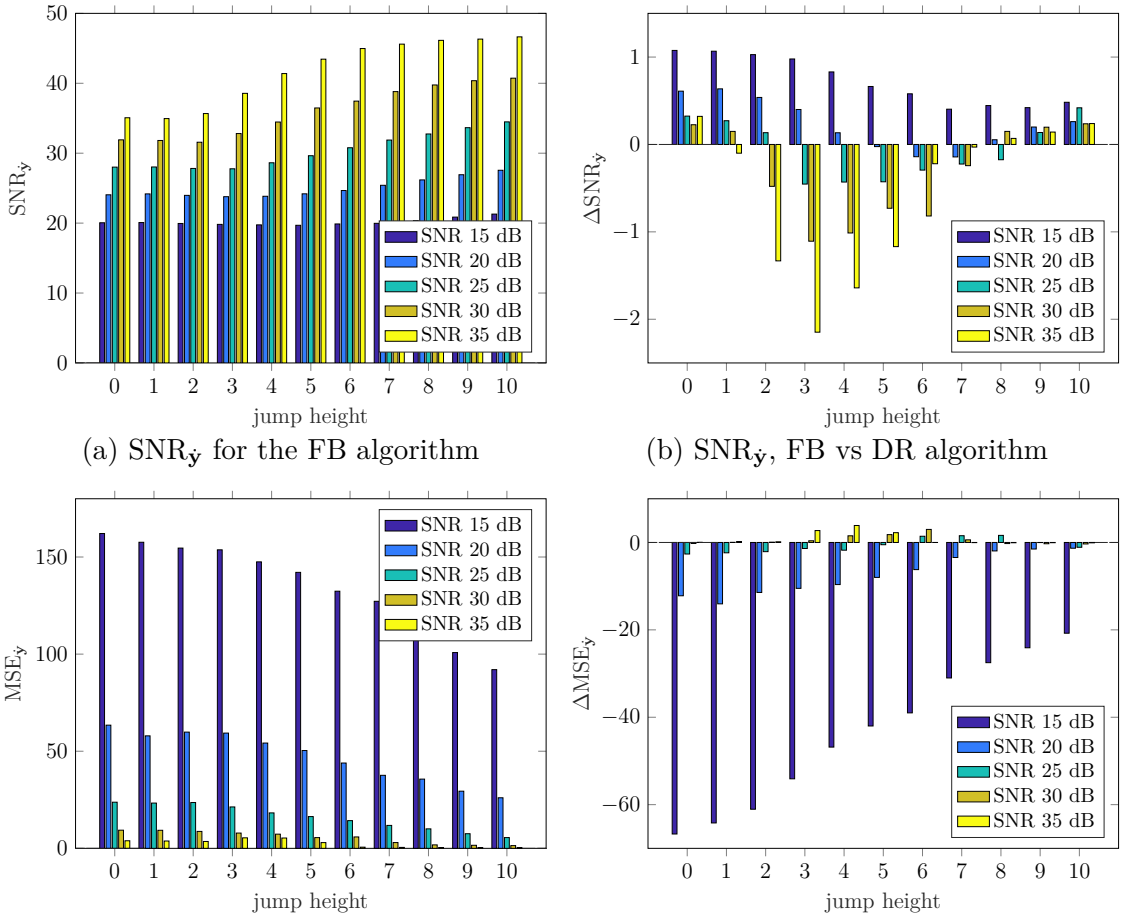
(c) MMR for the FB algorithm

(d) MMR, FB vs DR algorithm

(e) NoB for the FB algorithm

(f) NoB, FB vs DR algorithm

Fig. 5.13: AAR, MMR and NoB results for the FB algorithm and their comparison with the DR algorithm. The left-hand side figures show the results for the FB algorithm. The right-hand side figures show the difference between results obtained with the FB and the DR algorithms. Presented results are for quadratic signals.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$ for the FB algorithm

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, FB vs DR algorithm

(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$ for the FB algorithm

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, FB vs DR algorithm

Fig. 5.14: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the FB algorithm and their comparison with the DR algorithm. The left-hand side figures show the results for the FB algorithm. The right-hand side figures show the difference between results obtained with the FB and the DR algorithms. Presented results are for quadratic signals.

and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better for the FB algorithm, MMR and NoB results are better for the DR algorithm for almost all signals. The $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results do not point out the clear winner – for some signals, it is better to use the FB algorithm, and the DR algorithm for the others.

5.4.4 Partial conclusions

In the experiments comparing the FB and DR algorithms on both the linear and quadratic signals, in general, it holds that better results are achieved with signals of higher SNR and greater jump heights for all metrics, except the AAR.

For processing of the linear signals, it is better to use 2-polynomial basis instead of the 3-polynomial basis for both the FB and the DR algorithms (note that graphs

of this comparison are omitted for brevity).

From the comparison of the FB and the DR algorithms, it follows that the denoising process of both linear and quadratic signals is better when the FB algorithm is used. Nevertheless, the difference in the case of linear signals is only marginal. However, the NoB is the most important metric for evaluation of breakpoint detection. From this point of view, the NoB results show that it is better to use the DR algorithm for the breakpoint detection instead of the FB algorithm for both linear and quadratic signals.

5.5 Recovery problem – ℓ_{21}-norm

The presented recovery problem (5.27) from Sec. 5.4 does not ensure finding the possible breakpoint candidates at the same positions across all difference vectors $\nabla\hat{\mathbf{x}}_k$ because that approach utilises TV to treat each $\mathbf{x}_k$ separately. Therefore, the correct breakpoints may not be detected in all difference vectors $\nabla\hat{\mathbf{x}}_k$, and it might happen that they are eliminated during the "Detection of segment borders" process.

According to the signal model description in Sec. 5.1.1, it can be inferred that that parametrisation coefficients in $\mathbf{x}_k$ are strongly related, and the breakpoints in all difference vectors $\nabla\mathbf{x}_k$ appear at the same positions. This assumption needs to be included in composing the optimisation problem. The so-called mixed ℓ_{21}-norm [113] (see Sec. 1.1.3) is used to enforce *joint* breakpoints across all the difference vectors $\nabla\mathbf{x}_k$ in the following optimisation problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \|[\tau_0\nabla\mathbf{x}_0, \ldots, \tau_K\nabla\mathbf{x}_K]\|_{21} \ \text{s.t.} \ \|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2 \leq \delta, \tag{5.35}$$

where $\mathbf{x}$ are parametrisation coefficients, $\hat{\mathbf{x}}$ are estimated parametrisation coefficients, ∇ is the difference operator, τ_k are $K+1$ positive regularisation weights corresponding to the individual polynomial degrees, $\mathbf{y}$ is the observed signal, $\mathbf{P}$ is the matrix of basis polynomials, and δ is the parameter reflecting the noise level and model error.

The ℓ_{21}-norm is used as a regulariser that promotes sparsity across rows of the matrix. In our case, the columns of the matrix are the difference vectors $\tau_k\nabla\mathbf{x}_k$, and thus, the *joint* sparsity of the differences is enforced. Therefore, the non-zero values of the differences should be concentrated only in a few rows on the same positions, and the remaining positions should be occupied by zeros. The rows with non-zero values correspond to breakpoints.

In the following subsections, the relaxation techniques are used and, therefore, the recovery problem (5.35) can be reformulated.

5.5.1 Definition of recovery problem

Similarly to the previous section, the recovery problem (5.35) is reformulated to the unconstrained form as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2 + \|[\tau_0 \nabla \mathbf{x}_0, \ldots, \tau_K \nabla \mathbf{x}_K]\|_{21}\,, \tag{5.36}$$

and it can be easily rewritten into more compact form:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2 + \|\mathrm{reshape}(L\mathbf{x})\|_{21}\,, \tag{5.37}$$

where $\hat{\mathbf{x}}$ are estimated parametrisation coefficients, $\mathbf{x}$ are parametrisation coefficients, $\mathbf{y}$ is input signal, $\mathbf{P}$ is matrix of polynomials, operator L represents the stacked differences such that

$$L = \begin{bmatrix} \tau_0\nabla & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \tau_K\nabla \end{bmatrix}, \quad L\mathbf{x} = \begin{bmatrix} \tau_0\nabla\mathbf{x}_0 \\ — \\ \vdots \\ — \\ \tau_K\nabla\mathbf{x}_K \end{bmatrix}, \tag{5.38}$$

with the difference operator $\nabla\colon \mathbb{R}^N \to \mathbb{R}^{N-1}$, and $K+1$ regularisation weights τ_k corresponding to the individual polynomial degrees. For the linear operator L, it holds $L\colon \mathbb{R}^{(K+1)N} \to \mathbb{R}^{(K+1)(N-1)}$. The operator reshape()$\colon \mathbb{R}^{(K+1)(N-1)} \to \mathbb{R}^{(N-1)\times(K+1)}$ takes the stacked vector $L\mathbf{x}$ to the form of a matrix with disjoint columns:

$$\mathrm{reshape}(L\mathbf{x}) = \left[\tau_0\nabla\mathbf{x}_0 \mid \cdots \mid \tau_K\nabla\mathbf{x}_K\right]. \tag{5.39}$$

In problem (5.37), the first term is the "data fidelity" term, enforcing the estimate to approximately correspond to the observation $\mathbf{y}$. The Euclidean ℓ_2-norm reflects the fact that gaussianity of the noise is assumed.

The second term in the problem (5.37) is the "penalisation" term. This functional mathematically transcripts the desired properties of the estimate and assigns high values to vectors $\mathbf{x}$ that lack such properties.

Numerical solution to the recovery problem (5.36) using the Forward-backward based primal-dual algorithm and the Chambolle–Pock algorithm was presented in our conference paper [6] and journal article [7].

5.5.2 Algorithms used

The recovery problem (5.37) can be solved via either the Forward-backward based primal-dual algorithm or the Chambolle–Pock algorithm. These two algorithms are able to

$$\text{minimise} \quad f(\mathbf{x}) + h(L\mathbf{x}), \tag{5.40}$$

where both functions f and h are convex and L is a linear operator.

According to the defined recovery problem (5.37), smooth function $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2$ and non-smooth function $h(L\mathbf{x})$ are assigned, such that

$$h(L\mathbf{x}) = \|\text{reshape}(L\mathbf{x})\|_{21} = \|[\tau_0\nabla\mathbf{x}_0 \mid \cdots \mid \tau_K\nabla\mathbf{x}_K]\|_{21}. \tag{5.41}$$

Forward-backward based primal-dual algorithm

The Forward-backward based primal-dual (FBB-PD) algorithm [36] is primal-dual and iterative, and it is able to minimise the sum of tree functions, see problem (1.35). However, when $g = 0$ in (1.35), the FBB-PD algorithm can be used to solve the problem (5.40), where function f is β-Lipschitz differentiable with constant $\beta < \infty$ [31], and function h is a (possibly) non-smooth function. Each iteration of the FBB-PD algorithm consists of the gradient step with respect to function f and the proximal step with respect to function h. The general form of the FBB-PD algorithm is presented in Sec. 1.3.3 in Algorithm 4.

The gradient step requires a gradient of the function f, which is in this case computed as $\mathbf{P}^\top(\mathbf{P}\cdot - \mathbf{y})$. The Lipschitz constant β needs to be established because it plays an important role in ensuring convergence. In Sec. 5.4.2, it was shown that $\beta = K+1$ for the modified standard basis ($\mathbf{P} = \mathbf{S}$) consisting of $K+1$ polynomials, which corresponds to the operator/spectral norm of $\mathbf{P}$.

The function h is a non-smooth function and its proximal operator has been established in definitions (1.24) and (1.25), where $\tau = 1$.

The FBB-PD algorithm utilises the linear operator L and its transpose $L^\top$. The linear operator L is defined in (5.38) and its transpose $L^\top\colon \mathbb{R}^{(K+1)(N-1)} \to \mathbb{R}^{(K+1)N}$ is defined as follows:

$$L^\top(\mathbf{u}) = L^\top\left(\begin{bmatrix}\mathbf{u}_0\\ \hline \vdots \\ \hline \mathbf{u}_K\end{bmatrix}\right) = \begin{bmatrix}\tau_0\nabla^\top\mathbf{u}_0\\ \hline \vdots \\ \hline \tau_K\nabla^\top\mathbf{u}_K\end{bmatrix}, \tag{5.42}$$

with $\nabla^\top$ of size $N \times (N-1)$ and vector $\mathbf{u} \in \mathbb{R}^{(N-1)(K+1)}$. Operator $\nabla^\top$ can be implemented as efficiently as ∇ since it holds

$$\nabla^\top\mathbf{u} = \nabla\left(\begin{bmatrix}0\\ -\mathbf{u}\\ 0\end{bmatrix}\right). \tag{5.43}$$

For the needs of the FBB-PD algorithm, the transpose of the operator reshape() needs to be defined. The operator $\text{reshape}^\top()\colon \mathbb{R}^{(N-1)\times(K+1)} \to \mathbb{R}^{(K+1)(N-1)}$ takes the matrix with disjoint columns to the form of a stacked vector:

$$\text{reshape}^\top\left(\left[\mathbf{u}_0 \mid \cdots \mid \mathbf{u}_K\right]\right) = \begin{bmatrix} \mathbf{u}_0 \\ \hline \vdots \\ \hline \mathbf{u}_K \end{bmatrix}. \tag{5.44}$$

To ensure convergence of the FBB-PD algorithm, the following sufficient condition must be satisfied [36]:

$$1/\zeta \geq \beta/2 + \sigma\|L\|^2, \tag{5.45}$$

where β is the Lipschitz constant, σ and ζ are positive scalars called "step sizes," and the upperbound of $\|L\|$ is derived as follows:

$$\begin{aligned} \|L\|^2 &= \max_{\|\mathbf{x}\|_2=1} \|L\mathbf{x}\|_2^2 \\ &= \max_{\|\mathbf{x}\|_2=1} \|[\tau_0\nabla\mathbf{x}_0, \ldots, \tau_K\nabla\mathbf{x}_K]\|_2^2 \\ &= \max_{\|\mathbf{x}\|_2=1} \left(\|\tau_0\nabla\mathbf{x}_0\|_2^2 + \cdots + \|\tau_K\nabla\mathbf{x}_K\|_2^2\right) \\ &\leq \tau_0^2 \max_{\|\mathbf{x}\|_2=1} \|\nabla\mathbf{x}_0\|_2^2 + \cdots + \tau_K^2 \max_{\|\mathbf{x}\|_2=1} \|\nabla\mathbf{x}_K\|_2^2 \\ &\leq \tau_0^2\|\nabla\|^2 + \cdots + \tau_K^2\|\nabla\|^2 \\ &\leq 4(\tau_0^2 + \cdots + \tau_K^2), \end{aligned} \tag{5.46}$$

since it can be easily shown that $\|\nabla\| \leq 2$. From here, it follows that $\|L\| \leq 2\sqrt{\sum_{k=0}^K \tau_k^2} =: 2\|\boldsymbol{\tau}\|_2$.

The form of the FBB-PD algorithm minimising the problem (5.37) is defined in Algorithm 10, where the *Id* stands for the identity operator.

Algorithm 10: The Forward-backward based primal-dual algorithm solving (5.37)

Input: Functions f, h, input signal $\mathbf{y} \in \mathbb{R}^N$, polynomial matrix $\mathbf{P}$, linear operators $L, L^\top$

Output: $\hat{\mathbf{x}} = \mathbf{x}^{i+1}$

Set $\zeta = 1/\beta, \sigma = \beta/2\|L\|^2$, $\theta = 0.99$

Set initial primal variables $\mathbf{x}^0 \in \mathbb{R}^{N(K+1)}$ and dual variables $\mathbf{v}^0 \in \mathbb{R}^{(N-1)\times(K+1)}$

for $i = 0, 1, \ldots$ *until convergence* **do**

$\mathbf{r}^i = \mathbf{x}^i - \zeta\mathbf{P}^\top(\mathbf{P}\mathbf{x}^i - \mathbf{y}) - \zeta\sigma L^\top\text{reshape}^\top(\mathbf{v}^i)$

$\mathbf{q}^i = (Id - \text{soft}_{1/\sigma}^{\text{row}})(\mathbf{v}^i + \text{reshape}(L(2\mathbf{r}^i - \mathbf{x}^i)))$

$\mathbf{x}^{i+1} = \mathbf{x}^i + \theta(\mathbf{r}^i - \mathbf{x}^i)$

$\mathbf{v}^{i+1} = \mathbf{v}^i + \theta(\mathbf{q}^i - \mathbf{v}^i)$

return $\mathbf{x}^{i+1}$

Chambolle–Pock algorithm

The problem (5.37) can also be solved via the Chambolle–Pock (CP) algorithm [37]. Same as the FBB-PD algorithm, the CP algorithm is primal-dual and iterative. The main difference between the FBB-PD and the CP algorithm is that the CP algorithm uses the proximal operator of f instead of its gradient. The general form of the CP algorithm is presented in Sec. 1.3.3 in Algorithm 3.

Functions f and h are specified in the same way as above. The proximal operator of f has been established in (1.21), and the proximal operator of h has been established in (1.24) and (1.25), where $\tau = 1$. Moreover, the relation (1.27) for the conjugate function should be used. The operators L, its transpose $L^\top$, reshape(), reshape$^\top$(), and the upper bound on $\|L\|$ are defined above. Convergence of the CP algorithm is guaranteed for $\zeta\sigma \leq 1/(4\|\boldsymbol{\tau}\|_2^2)$, where ζ a σ are positive scalars representing "step sizes" and $\boldsymbol{\tau} \in \mathbb{R}^{K+1}$ is vector of τ_k.

In the described setting, the particular CP algorithm minimising the problem (5.37) is presented in Algorithm 11, where the *Id* stands for the identity operator. Note that the matrices $\mathbf{P}^\top\mathbf{y}$ and $\left(\mathbf{I} + \zeta\mathbf{P}^\top\mathbf{P}\right)^{-1}$ can be precomputed, and the explicit formula for computing the inversion is introduced in Theorem 1 in the Appendix A.

Algorithm 11: The Chambolle–Pock algorithm solving (5.37)

Input: Functions f, h, input signal $\mathbf{y} \in \mathbb{R}^N$, polynomial matrix $\mathbf{P}$, linear operators $L, L^\top$

Output: $\hat{\mathbf{x}} = \bar{\mathbf{x}}^{i+1}$

Set parameters $\zeta = 1/\|L\|, \sigma = 1/\|L\|$

Set parameter $\theta = 0.99$

Set initial primal variables $\mathbf{x}^0 \in \mathbb{R}^{N(K+1)}$ and dual variables $\mathbf{q}^0 \in \mathbb{R}^{(N-1)\times(K+1)}$

Set initial output variables $\bar{\mathbf{x}}^0 = \mathbf{x}^0$

for $i = 0, 1, \ldots$ *until convergence* **do**

  $\mathbf{q}^{i+1} = (Id - \text{soft}^{\text{row}}_{1/\sigma})\left(\mathbf{q}^i/\sigma + \text{reshape}(L\,\bar{\mathbf{x}}^i)\right)$

  $\mathbf{x}^{i+1} = \left(\mathbf{I} + \zeta\mathbf{P}^\top\mathbf{P}\right)^{-1}\left(\mathbf{x}^i - \zeta L^\top\text{reshape}^\top(\mathbf{q}^{i+1}) + \zeta\mathbf{P}^\top\mathbf{y}\right)$

  $\bar{\mathbf{x}}^{i+1} = \mathbf{x}^{i+1} + \theta(\mathbf{x}^{i+1} - \mathbf{x}^i)$

return $\bar{\mathbf{x}}^{i+1}$

5.5.3 Experiments

In this section, the results obtained from the 1D signal segmentation and denoising process with the recovery problem (5.37) solved by the FBB-PD and the CP algorithms are evaluated.

As in the previous section, the experiments are performed on datasets of linear and quadratic synthetic signals, which are introduced in Sec. 5.3. For the purpose of these experiments, the modified standard basis ($\mathbf{P} = \mathbf{S}$) is used, see Sec. 5.1.3 for more details. For the first experiments with linear signals, the used modified standard basis consists of 2 polynomials ($K = 1$), and for the second experiments, it consists of 3 polynomials ($K = 2$). For experiments on quadratic signals, the modified standard basis consisting of 3 polynomials ($K = 2$) is used.

The results obtained by the FBB-PD and the CP algorithms are evaluated in relation to the signal properties and used polynomial basis, and also results of both algorithms are compared to each other.

Several parameters need to be set for both the "Optimisation step" and the "Detection of segment borders step". The list of parameters for the experiments using the FBB-PD algorithm is given in Table 5.3 – for $K = 1$ and $K = 2$. The list of parameters for the CP algorithm is given in Table 5.4. Parameters of the FBB-PD algorithm (β, ζ, σ and θ) and parameters of the CP algorithm (ζ, σ and θ) are explained in Sec. 1.3.3. The FBB-PD and CP algorithms are stopped if either the convergence criterion or the maximum number of iterations ($\mathrm{MAX_{IT}}$) is reached. Regularisation weights τ_k are set in the same way as in the previous section. The parameter l_m represents the window length of the median filter. The parameter λ is a threshold, which decides on the selection of the potential positions of segment borders.

Tab. 5.3: Parameter settings for the FBB-PD algorithm. Table contains the specific setting of parameters needed in the 1D signal segmentation and denoising process using the FBB-PD algorithm.

Step	**Setting**		
	Parameter	**Value (for $K = 1$)**	**Value (for $K = 2$)**
FBB-PD algorithm	$\mathrm{MAX_{IT}}$	300	300
	β	2	3
	σ	$1/\|L\|^2$	$3/2/\|L\|^2$
	ζ	0.5	1/3
	θ	0.99	0.99
	τ_0	$\sigma_e \cdot 5$	$\sigma_e \cdot 5$
	τ_1	$\sigma_e \cdot 3.5$	$\sigma_e \cdot 3.5$
	τ_2	–	$\sigma_e \cdot 2.5$
Detection of segment borders	l_m	5	5
	λ	0.2	0.2

Tab. 5.4: Parameter settings for the CP algorithm. Table contains the specific setting of parameters needed in the 1D signal segmentation and denoising process using the CP algorithm.

Step	**Setting**		
	Parameter	**Value (for $K = 1$)**	**Value (for $K = 2$)**
CP algorithm	$\mathrm{MAX_{IT}}$	300	300
	ζ	$1/\|L\|$	$1/\|L\|$
	σ	$1/\|L\|$	$1/\|L\|$
	θ	0.99	0.99
	τ_0	$\sigma_e \cdot 5$	$\sigma_e \cdot 5$
	τ_1	$\sigma_e \cdot 3.5$	$\sigma_e \cdot 3.5$
	τ_2	–	$\sigma_e \cdot 2.5$
Detection of segment borders	l_{m}	5	5
	λ	0.2	0.2

The obtained results are evaluated from two points of view – breakpoint detection accuracy, and signal denoising performance. The used evaluation metrics are identical to the previous section, so as the graphs demonstrating the results of the algorithms.

Linear synthetic signals

The first set of experiments is performed on the dataset of linear synthetic signals, which consists of 27,500 noisy signals in total. For the purpose of these experiments, the polynomial basis $\mathbf{P}$ is the modified standard basis $\mathbf{S}$ of size $N \times (K+1)$, where $K = 1$ or $K = 2$ (depending on the type of experiment). Several items are subject to vary within the experiments, configuring the problem (5.37):

- the input signal $\mathbf{y}$,
- regularisation parameters τ_0, τ_1, (and τ_2 for $K = 2$),
- parameter σ in the FBB-PD algorithm,
- parameters σ and ζ in the CP algorithm.

Evaluation of the FBB-PD algorithm At first, the results obtained by the 2-polynomial basis $\mathbf{P}$ are evaluated. The AAR results are shown in Fig. 5.15a. For all jump heights, the AAR is higher for the lower SNR. The MMR results are shown in Fig. 5.15c. For greater jump heights, it holds that the value increases with increased SNR. For smaller jump heights, the values of MMR are similar for all SNR. For NoB, it holds that the values are higher with greater jump height and higher

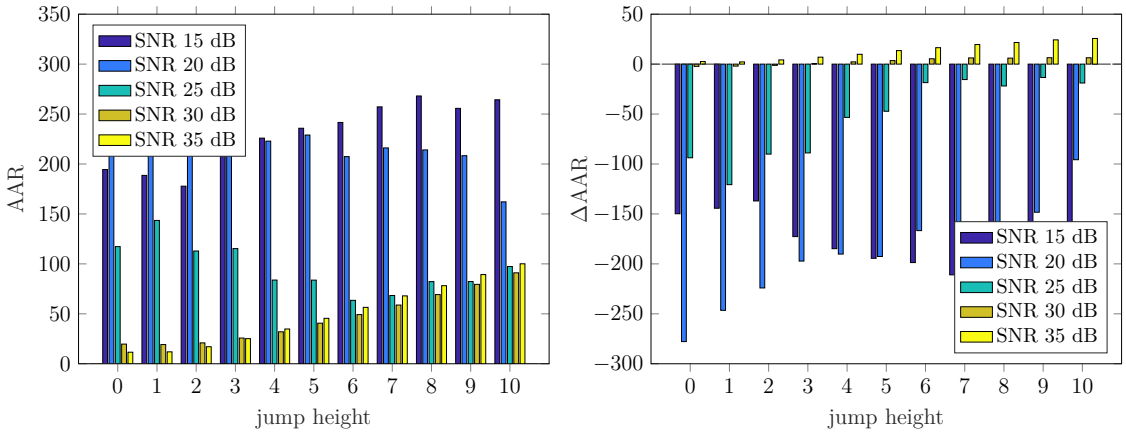
SNR, as expected, see Fig. 5.15e. Due to the character of the processed signals, the maximum achievable NoB is 5. The $\mathrm{SNR}_{\dot{\mathbf{y}}}$ increases with greater jump height between the signal segments. $\mathrm{MSE}_{\dot{\mathbf{y}}}$ values are closely related to the resulting $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values – the higher $\mathrm{SNR}_{\dot{\mathbf{y}}}$ the lower $\mathrm{MSE}_{\dot{\mathbf{y}}}$ – compare Fig. 5.16c with Fig. 5.16a.

The synthetic linear signals were also processed using a 3-polynomial basis $\mathbf{P}$. From the results, it follows that using 2-polynomial basis for processing of linear signals gives better results in AAR, MMR, $\mathrm{SNR}_{\dot{\mathbf{y}}}$. In the case of NoB, the better results are achieved with 3-polynomial basis. For signals with smaller jump heights and lower SNR, it is better to use 3-polynomial basis to get higher $\mathrm{MSE}_{\dot{\mathbf{y}}}$, for the rest of signals, it is better to use 2-polynomial basis.

Evaluation of the CP algorithm Also, for the CP algorithm, the 2- and 3-polynomial bases $\mathbf{P}$ were used. For the 2-polynomial basis, the character of the results is very similar to the results obtained with the FBB-PD algorithm. Significant difference between the FBB-PD and the CP algorithm is in AAR, where for smaller jump heights, the AAR is higher for lower SNR. For greater jump heights, the effect is opposite. The results show that using 2-polynomial basis gives better results for MMR, $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ and using 3-polynomial basis gives better results for AAR and NoB.

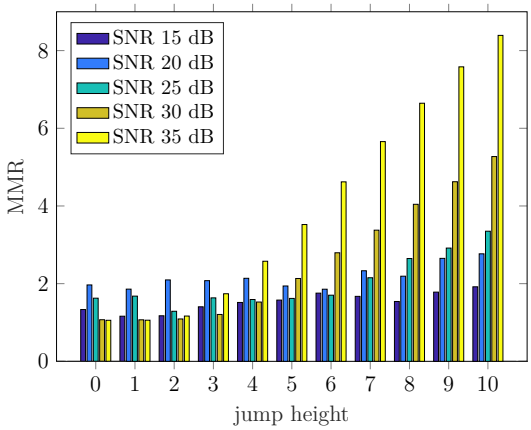
Comparison of the FBB-PD and the CP algorithm results The comparison of the FBB-PD and the CP algorithms is shown in the form of graphs in Figs. 5.15 and 5.16, where the left-hand side graphs show the results for the FBB-PD algorithm using 2-polynomial basis and the graphs on the right-hand side show the difference between results obtained with the CP and the FBB-PD algorithm. Values above zero mean that better results are achieved with the CP algorithm and values below zero mean better results for the FBB-PD algorithm – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, it applies the opposite.

The results of evaluating the breakpoint detection accuracy suggest that using the FBB-PD algorithm is better for lower SNR for almost all the jump heights. Nevertheless, for higher SNR, the results are better for the CP algorithm. From NoB point of view, slightly better results in breakpoint detection are obtained with the FBB-PD algorithm (see Fig. 5.15). Evaluation of the denoised signal is in both cases better for the FBB-PD algorithm (see Fig. 5.16).

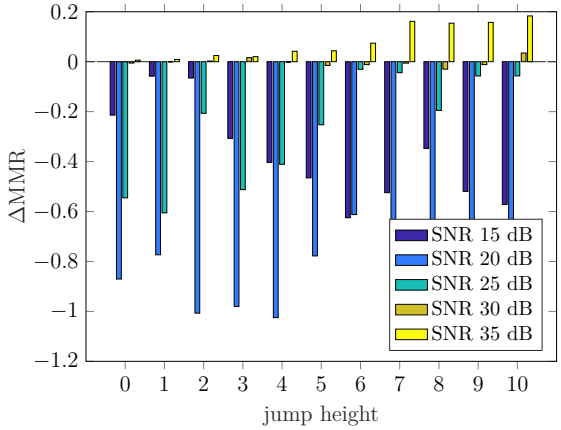


(a) AAR, FBB-PD, 2-polynomial basis

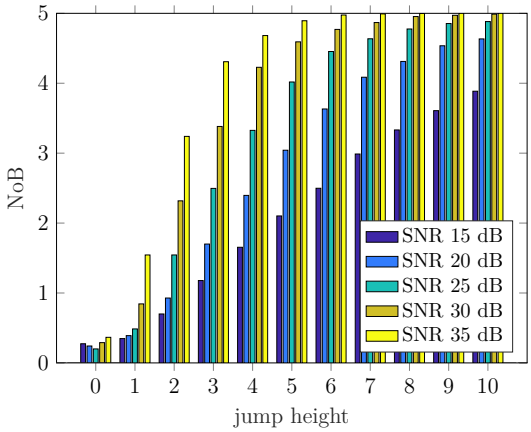
(b) AAR, CP vs FBB-PD, 2-pol. basis



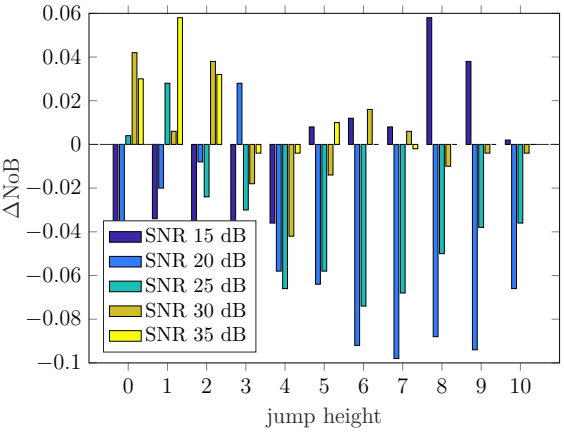
(c) MMR, FBB-PD, 2-polynomial basis



(d) MMR, CP vs FBB-PD, 2-pol. basis

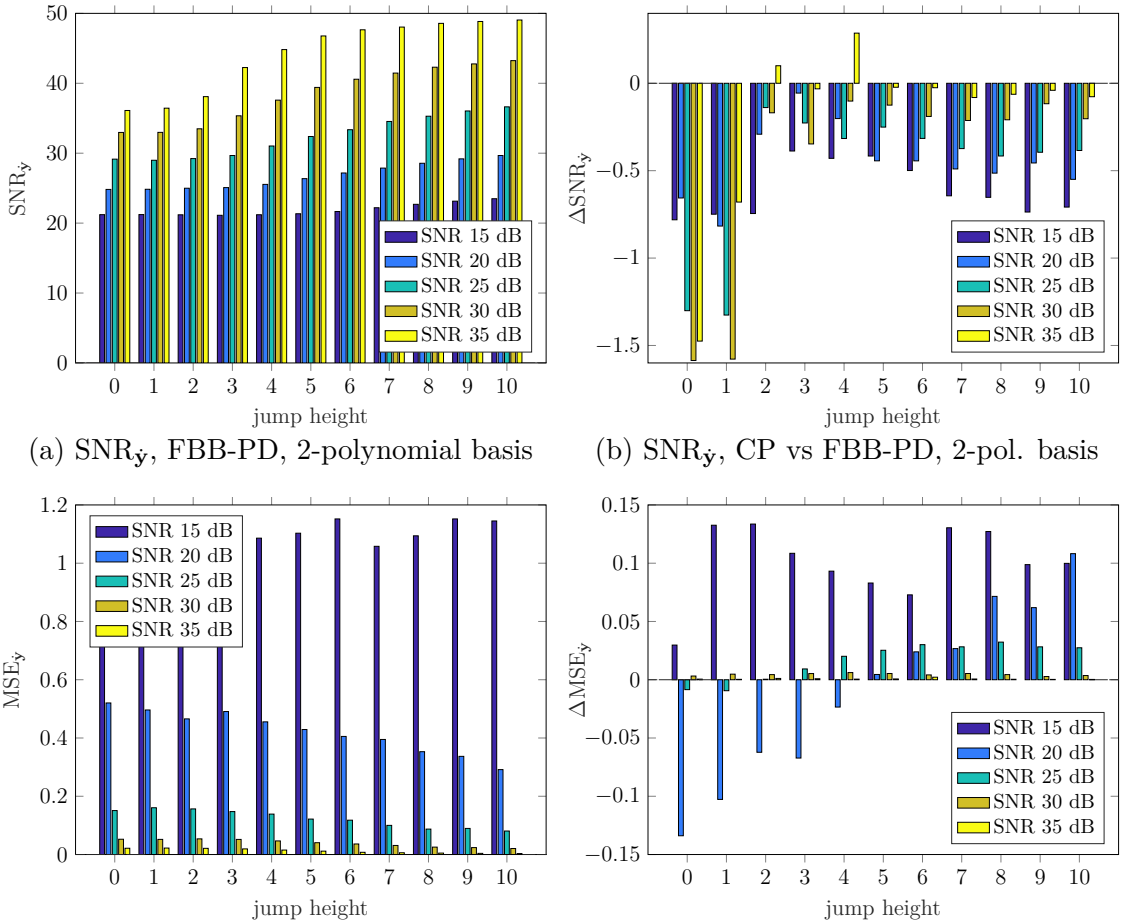


(e) NoB, FBB-PD, 2-polynomial basis



(f) NoB, CP vs FBB-PD, 2-pol. basis

Fig. 5.15: AAR, MMR and NoB results for the FBB-PD algorithm and their comparison with the CP algorithm. The left-hand side figures show the results for the FBB-PD algorithm. The right-hand side figures show the difference between results obtained with the CP and the FBB-PD algorithms. Presented results are for linear signals.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, FBB-PD, 2-polynomial basis

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, CP vs FBB-PD, 2-pol. basis

(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, FBB-PD, 2-polynomial basis

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, CP vs FBB-PD, 2-pol. basis

Fig. 5.16: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the FBB-PD algorithm and their comparison with the CP algorithm. The left-hand side figures show the results for the FBB-PD algorithm. The right-hand side figures show the difference between results obtained with the CP and the FBB-PD algorithms. Presented results are for linear signals.

Quadratic signals

The second set of experiments is performed on the dataset of quadratic synthetic signals, which consists of 27,500 noisy signals in total. For the purpose of these experiments, the polynomial basis $\mathbf{P}$ is the modified standard basis $\mathbf{S}$ of size $N \times (K+1)$, where $K = 2$. Several items are subject to vary within the experiments, configuring the problem (5.37):

- the input signal $\mathbf{y}$,
- regularisation parameters τ_0, τ_1 and τ_2,
- parameter σ in the FBB-PD algorithm,
- parameters σ and ζ in the CP algorithm.

Evaluation of the FBB-PD and the CP algorithm The results are shown in the form of graphs, see Figs. 5.17 and 5.18. In both figures, the graphs on the left-hand side show the results obtained using the FBB-PD algorithm, and the graphs on the right-hand side show the comparison of results obtained using the CP and the FBB-PD algorithms in form of differences, in the same way as for the linear signals.

The character of the results is very similar to the results obtained for linear signals. For AAR, MMR, NoB and $\mathrm{SNR}_{\hat{\mathbf{y}}}$, the results are better with higher SNR and greater jump height. For $\mathrm{MSE}_{\hat{\mathbf{y}}}$, it applies the opposite. This can be seen on the left-hand side graphs.

The comparisons of the FBB-PD and the CP algorithm show that MMR, $\mathrm{SNR}_{\hat{\mathbf{y}}}$ and $\mathrm{MSE}_{\hat{\mathbf{y}}}$ results are better for the FBB-PD algorithm. For AAR metric, it holds that the FBB-PD algorithm gives better results for signals with lower SNR, and the CP algorithm performs better results for signals with higher SNR. For NoB, it holds that the FBB-PD algorithm produces better results for signals with greater jump heights, and the CP algorithm achieves better results for signals with smaller jump heights.

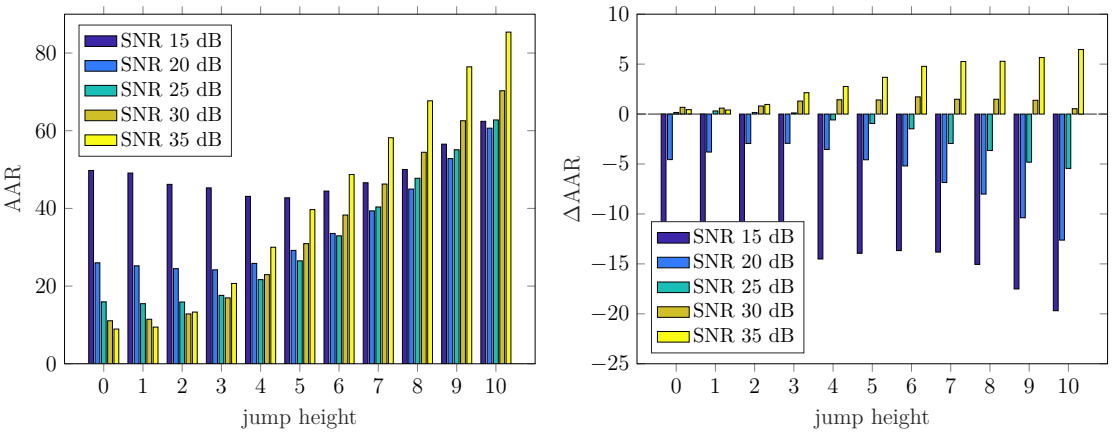
5.5.4 Partial conclusions

In all experiments, it holds that better results are achieved with signals of higher SNR and greater jump heights for all metrics. The only exception is the AAR evaluation of the FBB-PD algorithm with 2-polynomial basis for linear signals.

It is not so evident (like in the case of the FB and the DR algorithm), if it is better to use 2- or 3-polynomial bases for processing of the linear signal (note that graphs of this comparison are omitted for brevity). The most important metric for measuring the breakpoint detection accuracy is the NoB, and from this point of view, it is better to use 3-polynomial basis for both algorithms. Nevertheless, the differences between 2- and 3-polynomial basis are marginal. In the other hand, the most important metric for denoising is $\mathrm{SNR}_{\hat{\mathbf{y}}}$, and from this point of view, it is better to use 2-polynomial basis for both algorithms.

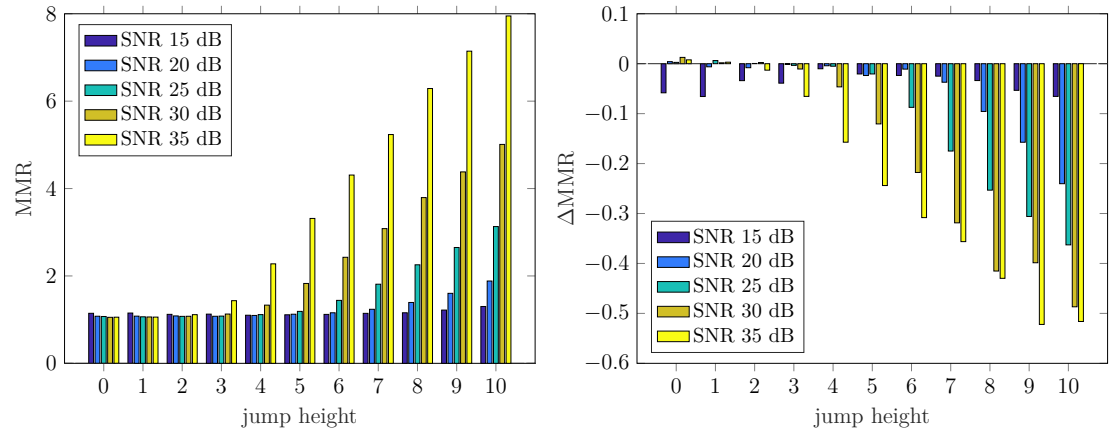
From comparison of the FBB-PD and the CP algorithms, it follows that the denoising process of the linear signals is better when the FBB-PD algorithm is used. In case of breakpoint detection, it seems that two conclusions can be drawn – the signals with higher SNR have better results using the CP algorithm, and the signals with lower SNR have better results with the FBB-PD algorithm.

For quadratic signals, the FBB-PD algorithm gives better results for almost all signals.



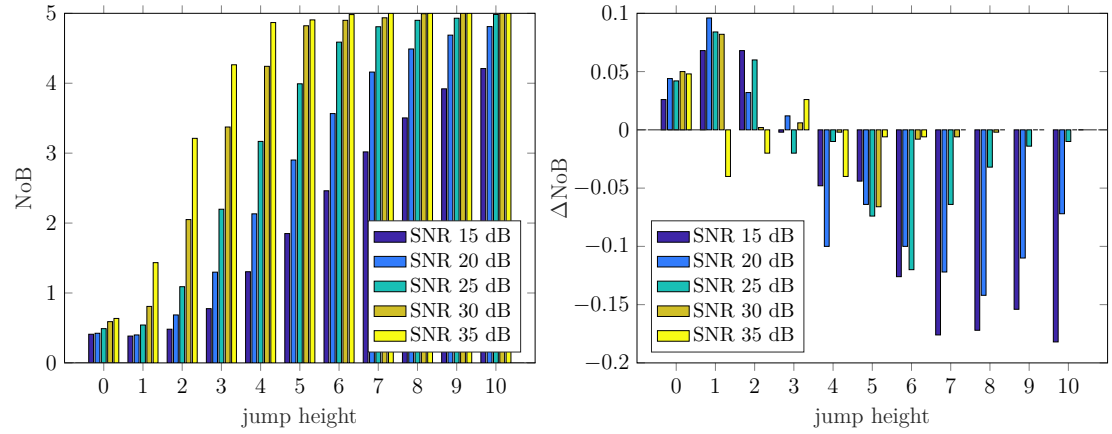
(a) AAR for the FBB-PD algorithm

(b) AAR, CP vs FBB-PD algorithm



(c) MMR for the FBB-PD algorithm

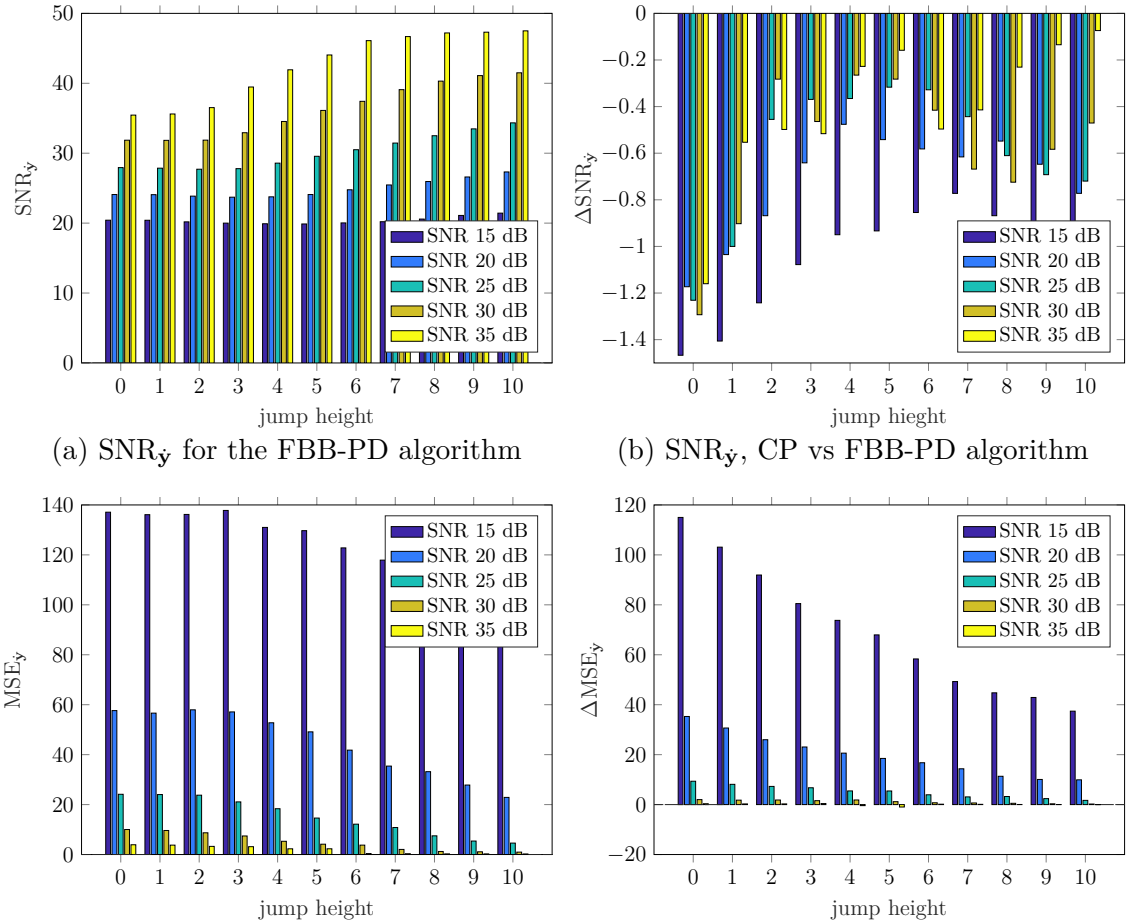
(d) MMR, CP vs FBB-PD algorithm



(e) NoB for the FBB-PD algorithm

(f) NoB, CP vs FBB-PD algorithm

Fig. 5.17: AAR, MMR and NoB results for the FBB-PD algorithm and their comparison with the CP algorithm. The left-hand side figures show the results for the FBB-PD algorithm. The right-hand side figures show the difference between results obtained with the CP and the FBB-PD algorithms. Presented results are for quadratic signals.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$ for the FBB-PD algorithm

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, CP vs FBB-PD algorithm

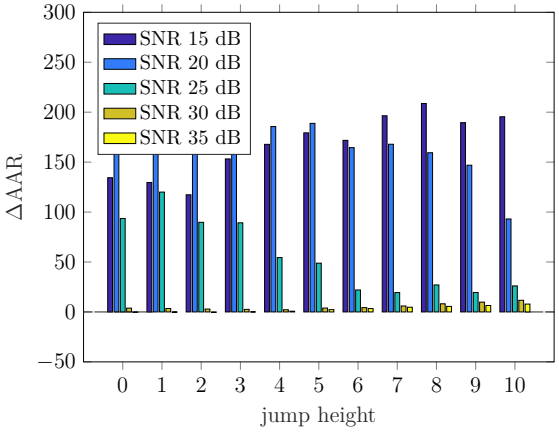
(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$ for the FBB-PD algorithm

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, CP vs FBB-PD algorithm

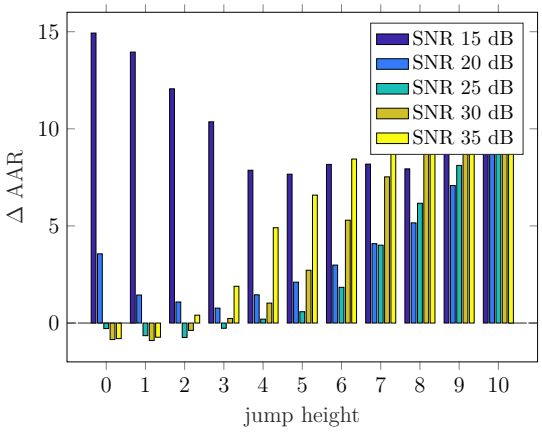
Fig. 5.18: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the FBB-PD algorithm and their comparison with the CP algorithm. The left-hand side figures show the results for the FBB-PD algorithm. The right-hand side figures show the difference between results obtained with the CP and the FBB-PD algorithms. Presented results are for quadratic signals.

The ℓ_{21}-norm was added to the recovery problem (5.37) with the aim of achieving better results than the simple total variation (see Sec. 5.4). The ℓ_{21}-norm guarantees the same position of possible breakpoint candidates through the difference vectors $\nabla\hat{\mathbf{x}}_k$. The comparisons of the results obtained via the FBB-PD algorithm solving the recovery problem (5.37) and the FB algorithm solving the recovery problem (5.28) are shown in the form of graphs in Figs. 5.19 and 5.20. The left-hand side graphs show the comparisons for linear signals and the right-hand side graphs present the results for quadratic signals. Values above zero mean that better results are achieved with the FBB-PD algorithm and values below zero mean better results for the FB algorithm – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, it applies the opposite.

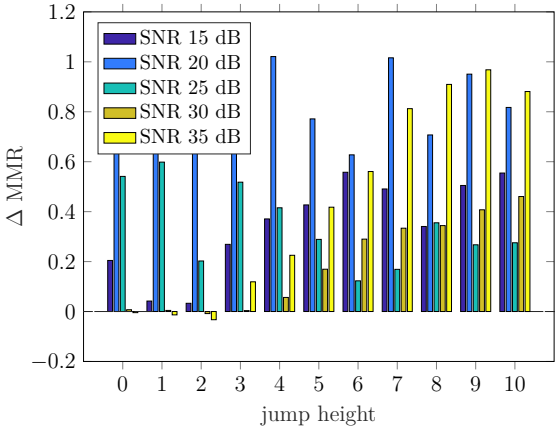
The comparison between the FBB-PD and the FB algorithms shows that for all



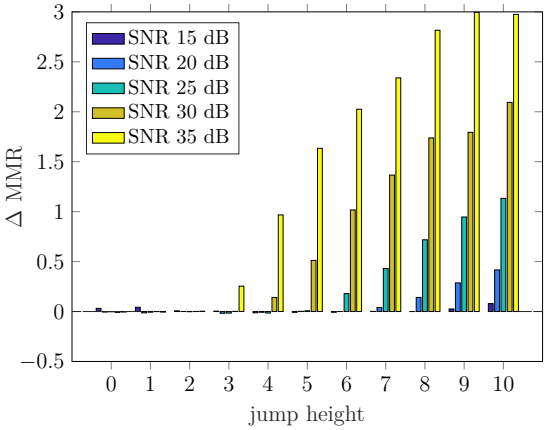
(a) AAR, FBB-PD vs FB – linear



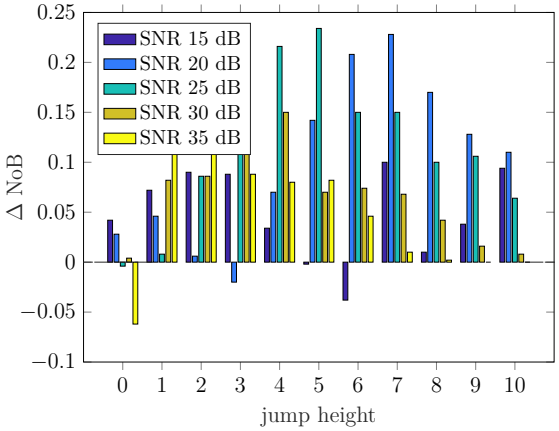
(b) AAR, FBB-PD vs FB – quadratic



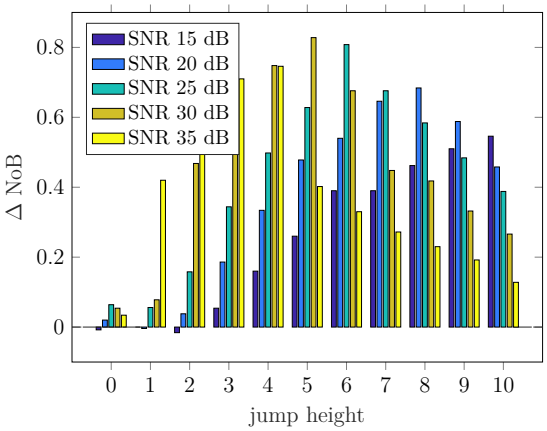
(c) MMR, FBB-PD vs FB – linear



(d) MMR, FBB-PD vs FB – quadratic

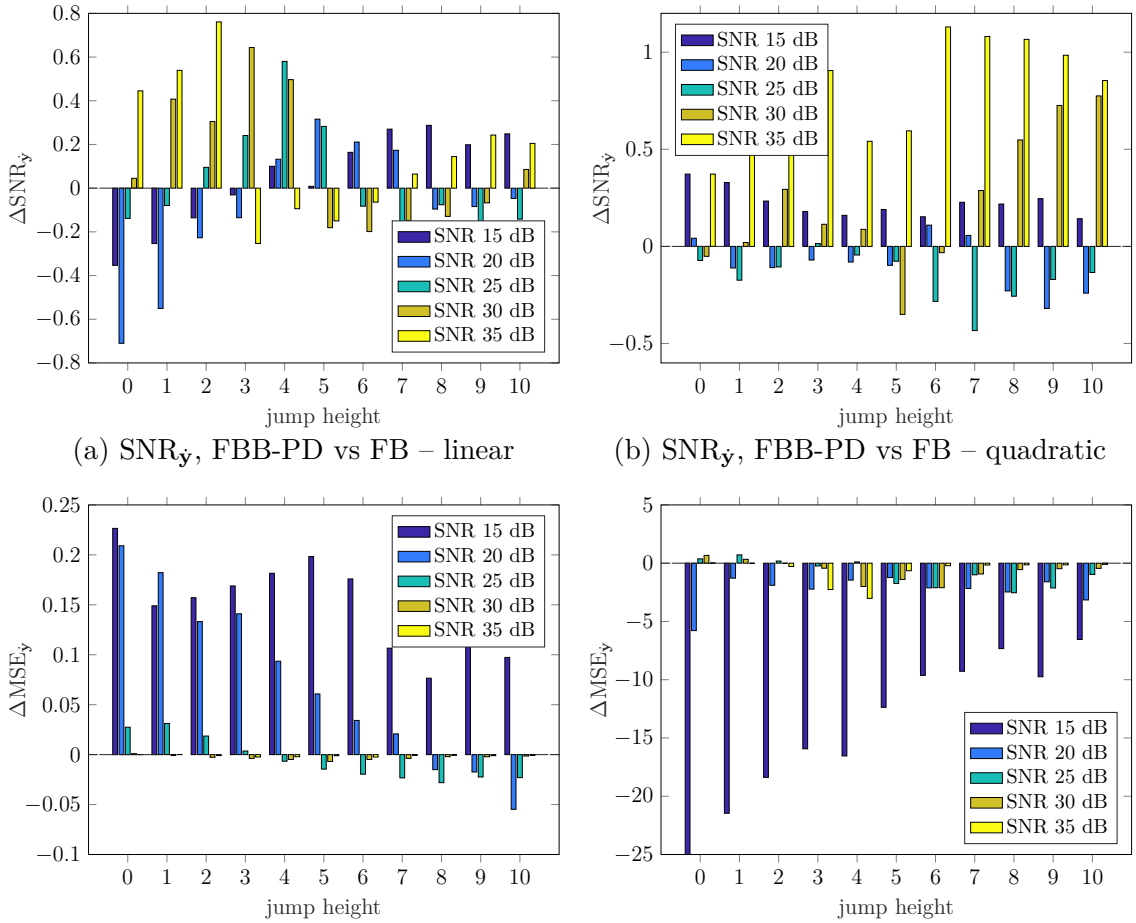


(e) NoB, FBB-PD vs FB – linear



(f) NoB, FBB-PD vs FB – quadratic

Fig. 5.19: Comparison of AAR, MMR and NoB results of the FBD-FB algorithm with the results of the FB algorithm. The left-hand side figures show the difference between results obtained with the FBB-PD and the FB algorithms for linear signals. The right-hand side figures show the difference for quadratic signals.



(a) SNR$_{\dot{\mathbf{y}}}$, FBB-PD vs FB – linear

(b) SNR$_{\dot{\mathbf{y}}}$, FBB-PD vs FB – quadratic

(c) MSE$_{\dot{\mathbf{y}}}$, FBB-PD vs FB – linear

(d) MSE$_{\dot{\mathbf{y}}}$, FBB-PD vs FB – quadratic

Fig. 5.20: Comparison of SNR$_{\dot{\mathbf{y}}}$ and MSE$_{\dot{\mathbf{y}}}$ results of the FBB-PD algorithm with the results of the FB algorithm. The left-hand side figures show the difference between results obtained with the FBB-PD and the FB algorithms for linear signals. The right-hand side figures show the difference for quadratic signals.

metrics, the results of the FBB-PD algorithm are better than the results of the FB algorithm for both linear and quadratic signals. The only exception to this is the MSE$_{\dot{\mathbf{y}}}$ for linear signals, for which the FB algorithm provides slightly better results. The use of the ℓ_{21}-norm has led to a significant improvement in the results compared to the simple total variation-based approach using the FB algorithm. Compared to the results obtained by the DR algorithm, the ℓ_{21} norm-based FBB-PD approach, however, represents an improvement only for quadratic signals.

5.6 Recovery problem – imitation of non-convexity

Another way how the breakpoint detection could be improved is the imitation of non-convexity. As mentioned in Sec. 1.2, ℓ_0-minimisation is often relaxed to ℓ_1-

minimisation for computational reasons. However, the solution of ℓ_1-minimisation does not have to be "the sparsest" in the ℓ_0 sense. To enhance the sparsity of the solution, it is possible to imitate the non-convexity via a series of convex programs. This approach is not new in general, it has been both theoretically and practically justified in [13], for example. Such a series of convex problems is formulated such that the parameters of the currently solved convex problem depend on the solution of the latest problem. The optimisation problem that extends problem (5.35) from previous section by the change of parameters can be formulated as follows:

$$\hat{\mathbf{x}}^{(j)} = \arg\min_{\mathbf{x}} \|\text{reshape}(L^{(j)}\mathbf{x})\|_{21} \text{ s.t. } \|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2 \leq \delta, \tag{5.47}$$

where the linear operator $L^{(j)}$ includes parameters, which change with each problem repetition, where $j = 0, \ldots, J$ represents the counter for problem repetitions, $\hat{\mathbf{x}}^{(j)}$ are estimated parametrisation coefficients, $\mathbf{x}$ are parametrisation coefficients, $\mathbf{y}$ is the input signal, $\mathbf{P}$ is the matrix of polynomials, δ is the parameter reflecting the noise level and model error, and reshape() operator is defined in the same way as in the previous section (see Eq. (5.39)).

After a defined number of inner iteration (IT) of the proximal splitting algorithm, the parameters in the operator $L^{(j)}$ are recomputed. The parameters are usually called weights, and are adaptively changed after each repetition (j). Therefore, the process of re-computation is called re-weighting in this Thesis. The points that are most probably the breakpoints would be gradually assigned lower weight, thus penalised less. The re-computation is described later in this section in more detail.

5.6.1 Definition of recovery problem

The unconstrained version of the problem (5.47) can be formulated as follows:

$$\hat{\mathbf{x}}^{(j)} = \arg\min_{\mathbf{x}} \|\text{reshape}(L^{(j)}\mathbf{x})\|_{21} + \iota_{\{\mathbf{z}\,:\,\|\mathbf{y}-\mathbf{z}\|_2 \leq \delta\}}(\mathbf{P}\mathbf{x}), \tag{5.48}$$

where ι_C denotes the indicator function of a convex set C, the reshape() operator is defined in (5.39), the operator $L^{(j)} = \mathbf{W}^{(j)}\nabla\colon \mathbb{R}^{(K+1)N} \to \mathbb{R}^{(K+1)(N-1)}$ is the linear operator, where $\nabla\colon \mathbb{R}^N \to \mathbb{R}^{N-1}$ is the difference operator and the matrix of weights $\mathbf{W}^{(j)}$ is defined as

$$\mathbf{W}^{(j)} = \begin{bmatrix} \text{diag}(\mathbf{w}_0^{(j)}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \text{diag}(\mathbf{w}_K^{(j)}) \end{bmatrix}, \tag{5.49}$$

where $\mathbf{w}_k^{(j)} = \tau_k \mathbf{w}^{(j)}$ are $K+1$ vectors of weights. Vector $\mathbf{w}^{(j)} \in \mathbb{R}^{N-1}$ is a common vector for $\mathbf{w}_k^{(j)}$, which will be adaptively changed at the end of each repetition.

τ_k are $K+1$ regularisation weights, corresponding to the individual polynomial degrees. They are set and stay fixed according to the noise level and prior experience. Therefore, the linear operator $L^{(j)}$ is defined as follows:

$$L^{(j)} = \begin{bmatrix} \mathrm{diag}(\mathbf{w}_0^{(j)})\nabla & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathrm{diag}(\mathbf{w}_K^{(j)})\nabla \end{bmatrix}, \quad L\mathbf{x} = \begin{bmatrix} \mathrm{diag}(\mathbf{w}_0^{(j)})\nabla\mathbf{x}_0 \\ — \\ \vdots \\ — \\ \mathrm{diag}(\mathbf{w}_K^{(j)})\nabla\mathbf{x}_K \end{bmatrix}. \tag{5.50}$$

The linear operator $L^{(j)}$ is updated after a suitable number of the algorithm iterations by recomputing the weights

$$\left(\mathbf{w}^{(j+1)}\right)_s = \frac{1}{\left\| \left[\left(\nabla\hat{\mathbf{x}}_0^{(j)}\right)_s, \ldots, \left(\nabla\hat{\mathbf{x}}_K^{(j)}\right)_s \right] \right\|_2 + \epsilon}, \tag{5.51}$$

where $(\cdot)_s$ denotes the s-th component of a vector, where $s = 1, \ldots, S$, parameter $\epsilon > 0$ provides numerical stability and prevents division by zero in the case of a zero-valued component in $\nabla\hat{\mathbf{x}}_k^{(j)}$.

It is shown in [13] that ϵ should be set slightly smaller than the expected non-zero magnitudes of $\nabla\hat{\mathbf{x}}_k^{(j)}$. For the non-weighted variant of the Condat algorithm, the weights are not needed. Therefore, the vector of weights $\mathbf{w}$ is set to vector of ones.

The first term of (5.47) is the "penalisation" term. Because the piecewise constant vectors $\mathbf{x}_k$ suggest that these vectors are joint-sparse under the difference operator ∇, the ℓ_{21}-norm (1.7) is used.

The second term in the problem (5.47) is the "data fidelity" term. The Euclidean ℓ_2-norm reflects the fact that gaussianity of the noise is assumed and it should be lower than the noise level δ.

Numerical solution to the recovery problem (5.48) using non-weighted and re-weighted variants of the Condat algorithm was presented in our paper [8].

5.6.2 Algorithm used

The recovery problem (5.48) can be solved via Condat algorithm (see Sec. 1.3.3), which is able to solve general problems in the form of (1.37). In this case, the problem reduces into the form of minimising the sum of two convex functions, formally

$$\text{minimise } h_1(L_1^{(j)}\mathbf{x}) + h_2(L_2\mathbf{x}) \tag{5.52}$$

where both functions h_1 and h_2 are convex and $L_1^{(j)}$ and L_2 are linear operators. With respect to the defined recovery problem, it is assigned $f(\mathbf{x}) = 0$, $g(\mathbf{x}) = 0$, $h_1(L_1^{(j)}\mathbf{x}) = \|\mathrm{reshape}(L^{(j)}\mathbf{x})\|_{21} = \|\mathrm{reshape}(\mathbf{W}^{(j)}\nabla\mathbf{x})\|_{21}$ and $h_2(L_2\mathbf{x}) = \iota_{\{\mathbf{z}\,:\,\|\mathbf{y}-\mathbf{z}\|_2 \leq \delta\}}(\mathbf{P}\mathbf{x})$.

The Condat algorithm

The Condat algorithm is described in more detail in Sec. 1.3.3. The general form of the Condat algorithm is presented in Algorithm 5. Since functions f and g are zero, $\nabla f = 0$ and $\text{prox}_g = Id$. The proximal operator of function h_1 is the group thresholding defined in (1.24) and (1.25), where $\tau = 1$. Finally, the proximal operator of function h_2 is the projection onto $B_2(\mathbf{y}, \delta)$ defined in (1.26).

The Condat algorithm operates with the linear operators $L_1^{(j)}$, L_2 and their transposes $(L_1^{(j)})^\top$, $L_2^\top$, where $L_2 = \mathbf{P}$ and $L_2^\top = \mathbf{P}^\top$. The operator $L_1^{(j)} = L^{(j)}$ is defined in (5.50) and its transpose $(L_1^{(j)})^\top$: $\mathbb{R}^{(K+1)(N-1)} \to \mathbb{R}^{(K+1)N}$ is

$$(L_1^{(j)})^\top(\mathbf{u}) = (L^{(j)})^\top(\mathbf{u}) = (L^{(j)})^\top \left(\begin{bmatrix} \mathbf{u}_0 \\ \hline \vdots \\ \hline \mathbf{u}_K \end{bmatrix} \right) = \begin{bmatrix} \text{diag}(\mathbf{w}_0^{(j)}) \nabla^\top \mathbf{u}_0 \\ \hline \vdots \\ \hline \text{diag}(\mathbf{w}_K^{(j)}) \nabla^\top \mathbf{u}_K \end{bmatrix}, \tag{5.53}$$

with $\nabla^\top$ of size $N \times (N-1)$ and vector $\mathbf{u} \in \mathbb{R}^{(N-1)(K+1)}$. For the needs of the Condat algorithm, the transpose of the reshape() operator is defined in (5.44).

The convergence of the Condat algorithm is guaranteed for $\xi\sigma\|(L_1^{(j)})^\top L_1^{(j)} + L_2^\top L_2\| \leq 1$, where ξ and σ are parameters of the Condat algorithm. The inequality $\|(L_1^{(j)})^\top L_1^{(j)} + L_2^\top L_2\| \leq \|L_1^{(j)}\|^2 + \|L_2\|^2$ is used, therefore the upper bound of $\|L_1^{(j)}\|$ and $\|L_2\|$ is needed. Obviously, $\|L_2\|^2 = \|\mathbf{P}\|^2 = K + 1$. The upper bound on the operator norm $\|\mathbf{L}_1^{(j)}\|$ is derived as follows:

$$\begin{aligned} \|L_1^{(j)}\|^2 &= \max_{\|\mathbf{x}\|_2=1} \|L_1^{(j)}\mathbf{x}\|_2^2 = \max_{\|\mathbf{x}\|_2=1} \left\| \begin{bmatrix} \tau_0 \,\text{diag}(\mathbf{w}^{(j)}) \nabla \mathbf{x}_0 \\ \vdots \\ \tau_K \,\text{diag}(\mathbf{w}^{(j)}) \nabla \mathbf{x}_K \end{bmatrix} \right\|_2^2 \\ &= \max_{\|\mathbf{x}\|_2=1} \left(\sum_{k=0}^{K} \left\| \tau_k \,\text{diag}(\mathbf{w}^{(j)}) \nabla \mathbf{x}_k \right\|_2^2 \right) \\ &\leq \sum_{k=0}^{K} \left(\max_{\|\mathbf{x}\|_2=1} \left\| \tau_k \,\text{diag}(\mathbf{w}^{(j)}) \nabla \mathbf{x}_k \right\|_2^2 \right) \\ &\leq \sum_{k=0}^{K} \left(\tau_i \max(\mathbf{w}^{(j)}) \right)^2 \|\nabla\|^2 \;\leq\; 4(\max(\mathbf{w}^{(j)}))^2 \sum_{k=0}^{K} \tau_k^2, \end{aligned} \tag{5.54}$$

and thus $\xi\sigma(K + 1 + 4(\max(\mathbf{w}^{(j)}))^2\|\boldsymbol{\tau}\|_2^2) \leq 1$. Because of the re-weighting, the weights get changed, and therefore, it is necessary to recompute the operator $L_1^{(j)}$, and its transpose, as well as the norms after a defined number of iterations.

The non-weighted Condat algorithm is presented in Algorithm 12, and the re-weighting variant is presented in Algorithm 13.

Algorithm 12: The Condat Algorithm solving (5.48) for $j = 0$

Input: Functions h_1, h_2, linear operators $L_1 \in \mathbb{R}^{N(K+1)\times(N-1)(K+1)}$, $L_1^\top \in \mathbb{R}^{(N-1)(K+1)\times N(K+1)}$, $L_2 \in \mathbb{R}^{N\times N(K+1)}, L_2^\top \in \mathbb{R}^{N(K+1)\times N}$, parameter δ for proximal operator of function h_2

Output: $\hat{\mathbf{x}} = \mathbf{x}^{(i+1)}$

Set parameters $\xi, \sigma > 0$ and $\rho \in [0, 2]$

Set initial primal variables $\mathbf{x}^{(0)} \in \mathbb{R}^{N(K+1)}$ and dual variables $\mathbf{u}_1^{(0)} \in \mathbb{R}^{(N-1)\times(K+1)}, \mathbf{u}_2^{(0)} \in \mathbb{R}^N$

for $i = 0, 1, \ldots, \mathrm{MAX_{IT}}$ **do**

$\bar{\mathbf{x}}^{(i+1)} = (\mathbf{x}^{(i)} - \xi(L_1^\top \mathrm{reshape}^\top(\mathbf{u}_1^{(i)}) + L_2^\top \mathbf{u}_2^{(i)}))$

$\mathbf{x}^{(i+1)} = \rho\bar{\mathbf{x}}^{(i+1)} + (1-\rho)\mathbf{x}^{(i)}$

$\bar{\mathbf{u}}_1^{(i+1)} = \mathrm{prox}_{\sigma h_1^*}(\mathbf{u}_1^{(i)} + \sigma \mathrm{reshape}(L_1(2\bar{\mathbf{x}}^{(i+1)} - \mathbf{x}^{(i)}))$

$\mathbf{u}_1^{(i+1)} = \rho\bar{\mathbf{u}}_1^{(i+1)} + (1-\rho)\mathbf{u}_1^{(i)}$

$\bar{\mathbf{u}}_2^{(i+1)} = \mathrm{prox}_{\sigma h_2^*}(\mathbf{u}_2^{(i)} + \sigma L_2(2\bar{\mathbf{x}}^{(i+1)} - \mathbf{x}^{(i)})$

$\mathbf{u}_2^{(i+1)} = \rho\bar{\mathbf{u}}_2^{(i+1)} + (1-\rho)\mathbf{u}_2^{(i)}$

return $\mathbf{x}^{(i+1)}$

Algorithm 13: Re-weighting Condat Algorithm solving (5.48)

Input: Functions h_1, h_2, linear operators L_1, $L_1^\top$, $L_2, L_2^\top$, maximal number of iterations $\mathrm{MAX_{IT}}$ of the Condat algorithm, number of re-weighting iterations $\mathrm{MAX_{REW}}$, parameter ϵ for recomputing of weights, τ_ks, parameter δ for proximal operator of function h_2

Output: $\hat{\mathbf{x}} = \hat{\mathbf{x}}^{(j)}$

Set initial vector of weights $\mathbf{w}^{(0)} = \mathbf{1}$, and initial linear operators $L_1^{(0)}, L_1^{\top(0)}$

Compute $\|L_1^{(0)}\|^2$

for $j = 0, 1, \ldots, \mathrm{MAX_{REW}}$ **do**

- Compute the Condat algorithm (Algorithm 12)
  - Input: $L_1^{(j)}, L_1^{\top(j)}, h_1, h_2, \tau_k$s, $\delta, \mathrm{MAX_{IT}}$
  - Return: $\hat{\mathbf{x}}^{(j)}$
- Compute new weights using (5.51):
  - Input: $\mathbf{w}^{(j)}, \epsilon$
  - Return: $\mathbf{w}^{(j+1)}$
- Redefine (5.50), (5.53) and recompute (5.54):
  - Input: $\mathbf{w}^{(j+1)}$, τ_ks
  - Return: $L_1^{(j+1)}, L_1^{\top(j+1)}, \|L_1^{(j+1)}\|^2$

return $\hat{\mathbf{x}}^{(j)}$

5.6.3 Experiments

In this section, the results obtained from the 1D signal segmentation and denoising process with the recovery problem (5.48) solved by the re-weighted and the non-weighted Condat algorithms are evaluated. The main aim is to find out which variant of the algorithm is systematically better.

As in the previous sections, the experiments are performed on datasets of linear and quadratic synthetic signals, which are introduced in Sec. 5.3. For the purpose of these experiments, the modified standard basis ($\mathbf{P} = \mathbf{S}$) is used, see Sec. 5.1.3 for more details. For the first experiments on linear signals, the used modified standard basis consists of 2 polynomials ($K = 1$), and for the second experiments, it consists of 3 polynomials ($K = 2$). For experiments on quadratic signals, a basis consisting of 3 polynomials ($K = 2$) is used.

Several parameters need to be set for both the "Optimisation step" and the "Detection of segment borders step". The list of parameters for the experiments using the Condat algorithm is given in Table 5.5 – for $K = 1$ and $K = 2$. The Condat algorithm is stopped if either the convergence criterion or the maximum number of iterations ($\mathrm{MAX_{IT}}$) is reached.

Tab. 5.5: Parameter settings for the non- and re-weighted Condat algorithm. Table contains the specific setting of parameters needed in the 1D signal segmentation and denoising process using the Condat algorithm.

Step	**Setting**		
	Parameter	**Value (for $K = 1$)**	**Value (for $K = 2$)**
Condat algorithm	σ	$1/\sqrt{(\|L_1^{(j)}\|^2 + \|L_2\|^2)}$	$1/\sqrt{(\|L_1^{(j)}\|^2 + \|L_2\|^2)}$
	ξ	$1/\sqrt{(\|L_1^{(j)}\|^2 + \|L_2\|^2)}$	$1/\sqrt{(\|L_1^{(j)}\|^2 + \|L_2\|^2)}$
	ρ	1.99	1.99
	τ_0	$\sigma_e \cdot 5$	$\sigma_e \cdot 5$
	τ_1	$\sigma_e \cdot 3.5$	$\sigma_e \cdot 3.5$
	τ_2	–	$\sigma_e \cdot 2.5$
	δ	$\|\mathbf{e}\|^2 \cdot 1.05$	$\|\mathbf{e}\|^2 \cdot 1.05$
non-weighted settings	$\mathrm{MAX_{IT}}$	3000	3000
	$\mathrm{MAX_{REW}}$	0	0
re-weighted settings	$\mathrm{MAX_{IT}}$	1000	1000
	$\mathrm{MAX_{REW}}$	3	3
Detection of segment borders	l_{m}	5	5
	λ	0.2	0.2
	ϵ	1	1

The obtained results are evaluated from two points of view – breakpoint detection accuracy, and signal denoising performance. The used evaluation metrics are identical to the previous sections, so as the graphs demonstrating the results of the algorithms.

Linear synthetic signals

The first set of experiments is performed on the dataset of linear synthetic signals, which consists of 27,500 noisy signals in total. For the purpose of these experiments, the polynomial basis $\mathbf{P}$ is the modified standard basis $\mathbf{S}$ of size $N \times (K+1)$, where $K = 1$ or $K = 2$ (depending on the type of experiment). Several items are subject to vary within the experiments, configuring the problem (5.48):

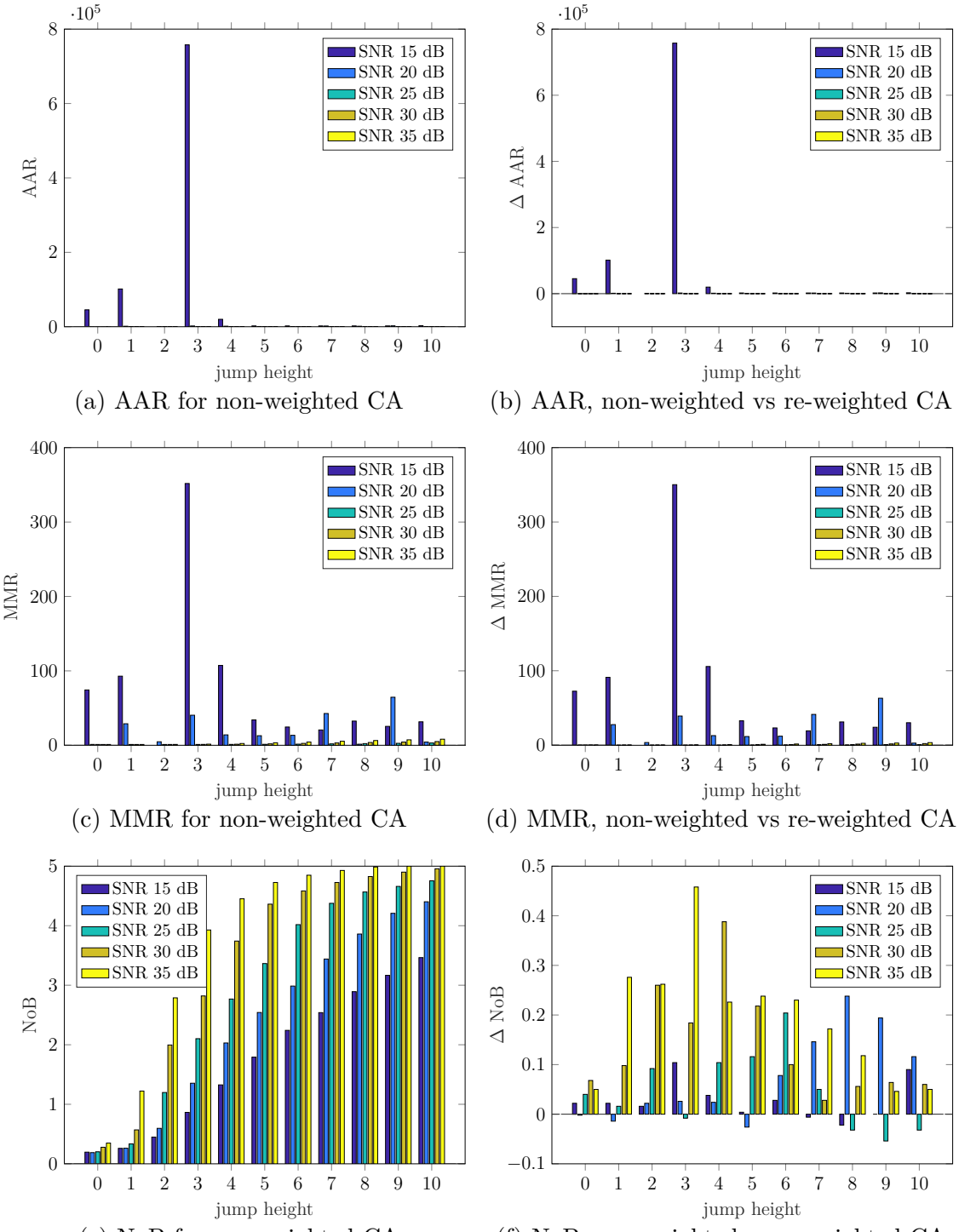
- the input signal $\mathbf{y}$,
- regularisation parameters τ_0, τ_1, (and τ_2 for $K = 2$),
- parameters δ, σ and ξ in the Condat algorithm.

Evaluation of the non-weighted Condat algorithm At first, the results obtained by the 2-polynomial basis $\mathbf{P}$ are evaluated. The AAR, MMR and NoB results are shown in form of graphs on the left-hand side in Fig. 5.21. The AAR results are shown in Fig. 5.21a. For all jump heights, the AAR and MMR is higher for the lower SNR. The MMR results are shown in Fig. 5.21c. According to the assumptions, it holds that NoB values are higher with greater jump height and higher SNR, see Fig. 5.21e. Due to the character of the processed signals, the maximum achievable NoB is 5.

The synthetic linear signals were also processed using a 3-polynomial basis $\mathbf{P}$. Note that the graphs for 3-polynomial bases were omitted for brevity. However, from the results, it follows that using 2-polynomial basis for processing of linear signals gives better results in AAR, MMR, $\mathrm{MSE}_{\dot{\mathbf{y}}}$. In the case of NoB, slightly better results are achieved with 3-polynomial basis. For signals with smaller jump heights, it is better to use 3-polynomial basis to get higher $\mathrm{SNR}_{\dot{\mathbf{y}}}$, for the rest of signals, it is better to use 2-polynomial basis.

The $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are shown on the left-hand side in Fig. 5.22. Fig. 5.22a shows that the $\mathrm{SNR}_{\dot{\mathbf{y}}}$ increases with greater jump height between the signal segments. The $\mathrm{MSE}_{\dot{\mathbf{y}}}$ values decrease with higher SNR see Fig. 5.22c. For the lowest SNR value, it holds that the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ increases with greater jump height.

Comparison of the non-weighted and the re-weighted Condat algorithm The comparison of the non-weighted and the re-weighted Condat algorithms is shown in the form of graphs on the right-hand side in Figs. 5.21 and 5.22. Values above zero mean that better results are achieved with the non-weighted Condat algorithm and



(a) AAR for non-weighted CA

(b) AAR, non-weighted vs re-weighted CA

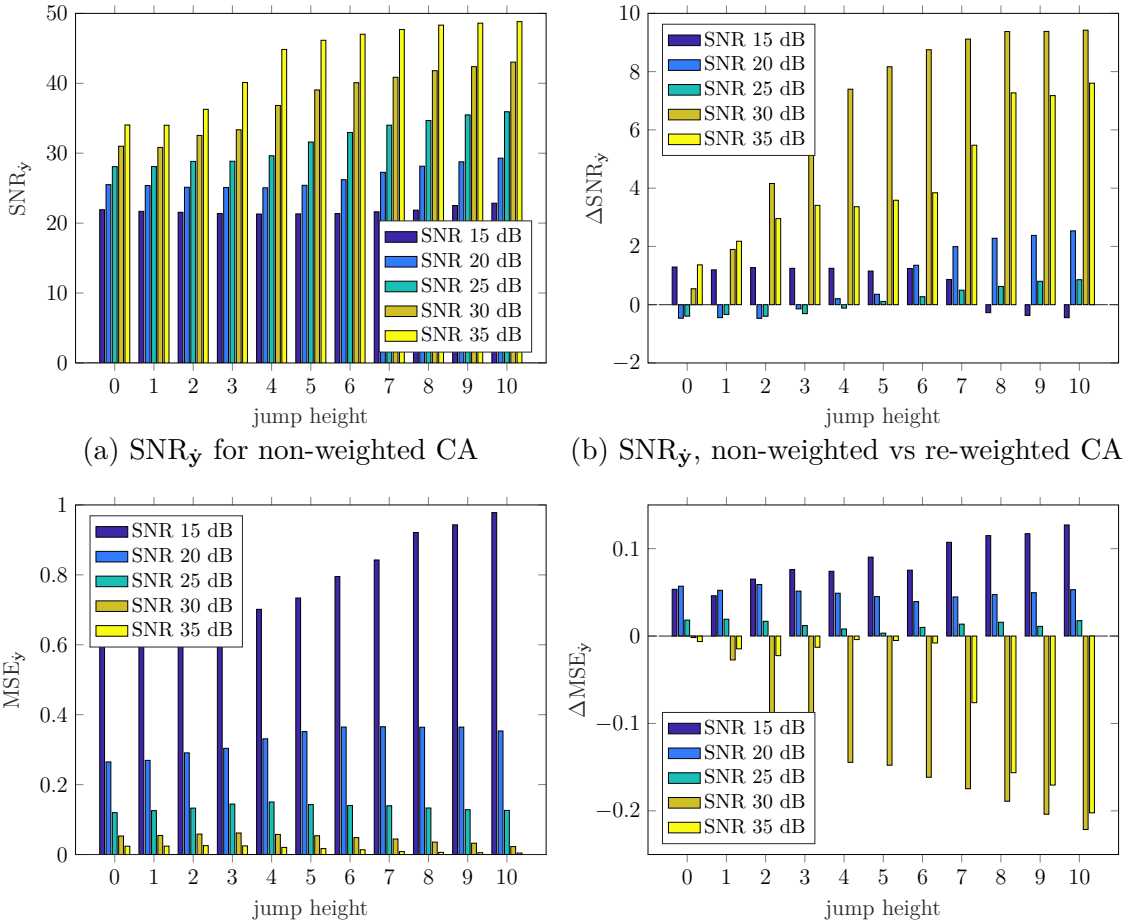
(c) MMR for non-weighted CA

(d) MMR, non-weighted vs re-weighted CA

(e) NoB for non-weighted CA

(f) NoB, non-weighted vs re-weighted CA

Fig. 5.21: AAR, MMR and NoB results for the non-weighted Condat algorithm (CA) and their comparison with the re-weighted Condat algorithm. The left-hand side figures show the results for the non-weighted Condat algorithm. The right-hand side figures show the difference between results obtained with the non-weighted and the re-weighted Condat algorithms. Presented results are for linear signals using 2-polynomial basis.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$ for non-weighted CA

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, non-weighted vs re-weighted CA

(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$ for non-weighted CA

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, non-weighted vs re-weighted CA

Fig. 5.22: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the non-weighted Condat algorithm (CA) and their comparison with the re-weighted Condat algorithm. The left-hand side figures show the results for the non-weighted Condat algorithm. The right-hand side figures show the difference between results obtained with the non-weighted and the re-weighted Condat algorithms. Presented results are for linear signals using 2-polynomial basis.

values below zero mean better results for the re-weighted Condat algorithm – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, it applies the opposite.

At first, the results obtained by the 2-polynomial basis are evaluated. For higher SNR, the AAR results are better using the re-weighted Condat algorithm. For lower SNR, the AAR results are better using the non-weighted Condat algorithm. Nevertheless, for the MMR and NoB, the results are better for the non-weighted variant. For almost all signals, the $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results are better for the non-weighted Condat algorithm. The $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better with the re-weighted Condat algorithm for signals with lower SNR. For higher SNR, the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better for the non-weighted Condat algorithm.

At second, the results obtained by the 3-polynomial basis are evaluated. For higher SNR, the AAR results are better using the re-weighted Condat algorithm. For lower SNR, the AAR results are better using the non-weighted Condat algorithm. Nevertheless, for the MMR and NoB, the results are better for the non-weighted variant. For signals with small jump height and for signals with greater jump height and lower SNR, the $\text{SNR}_{\dot{\mathbf{y}}}$ results are better for the non-weighted Condat algorithm. For signals with greater jump height and higher SNR, it is better to use non-weighted Condat algorithm. The $\text{MSE}_{\dot{\mathbf{y}}}$ results are better with the re-weighted Condat algorithm for almost all signals.

Quadratic signals

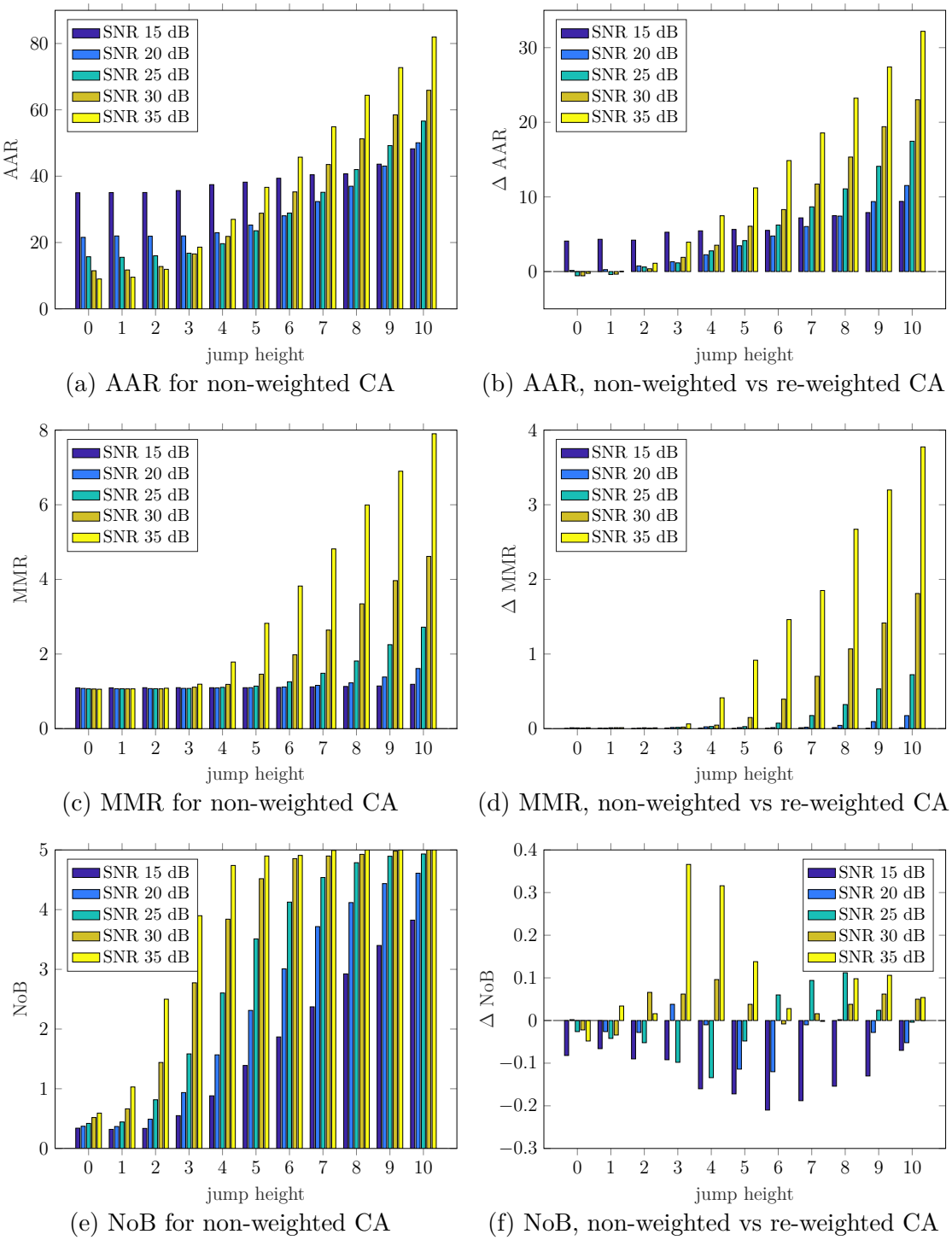
The second set of experiments is performed on the dataset of quadratic synthetic signals, which consists of 27,500 noisy signals in total. For the purpose of these experiments, the polynomial basis $\mathbf{P}$ is the modified standard basis $\mathbf{S}$ of size $N \times (K+1)$, where $K = 2$. Several items are subject to vary within the experiments, configuring the problem (5.48):

- the input signal $\mathbf{y}$,
- regularisation parameters τ_0, τ_1, and τ_2,
- parameters δ, σ and ξ in the Condat algorithm.

Comparison of the non-weighted and the re-weighted Condat algorithm
The results are shown in Figs. 5.23 and 5.24. In both figures, the graphs on the left-hand side show the results obtained using the non-weighted Condat algorithm, and the graphs on the right-hand side show the comparison of results obtained using the non-weighted and the re-weighted Condat algorithms in form of differences, in the same way as for the linear signals.

For greater jump heights, it holds that the AAR and the MMR values increase with increased SNR. For smaller jump heights, the values of MMR are similar for all SNR and the values of AAR increase with lower SNR. Using the non-weighted Condat algorithm, the character of the NoB, $\text{SNR}_{\dot{\mathbf{y}}}$ and $\text{MSE}_{\dot{\mathbf{y}}}$ results is very similar for both the quadratic and linear signals.

The comparisons of the non-weighted and the re-weighted Condat algorithm show that AAR and MMR results are better for the non-weighted Condat algorithm. The NoB results show that the re-weighted Condat algorithm is better for signals with lower SNR, however, the non-weighted Condat algorithm remains better for signals with higher SNR. $\text{SNR}_{\dot{\mathbf{y}}}$ and $\text{MSE}_{\dot{\mathbf{y}}}$ results for almost all types of signals are better for the non-weighted Condat algorithm.



(a) AAR for non-weighted CA

(b) AAR, non-weighted vs re-weighted CA

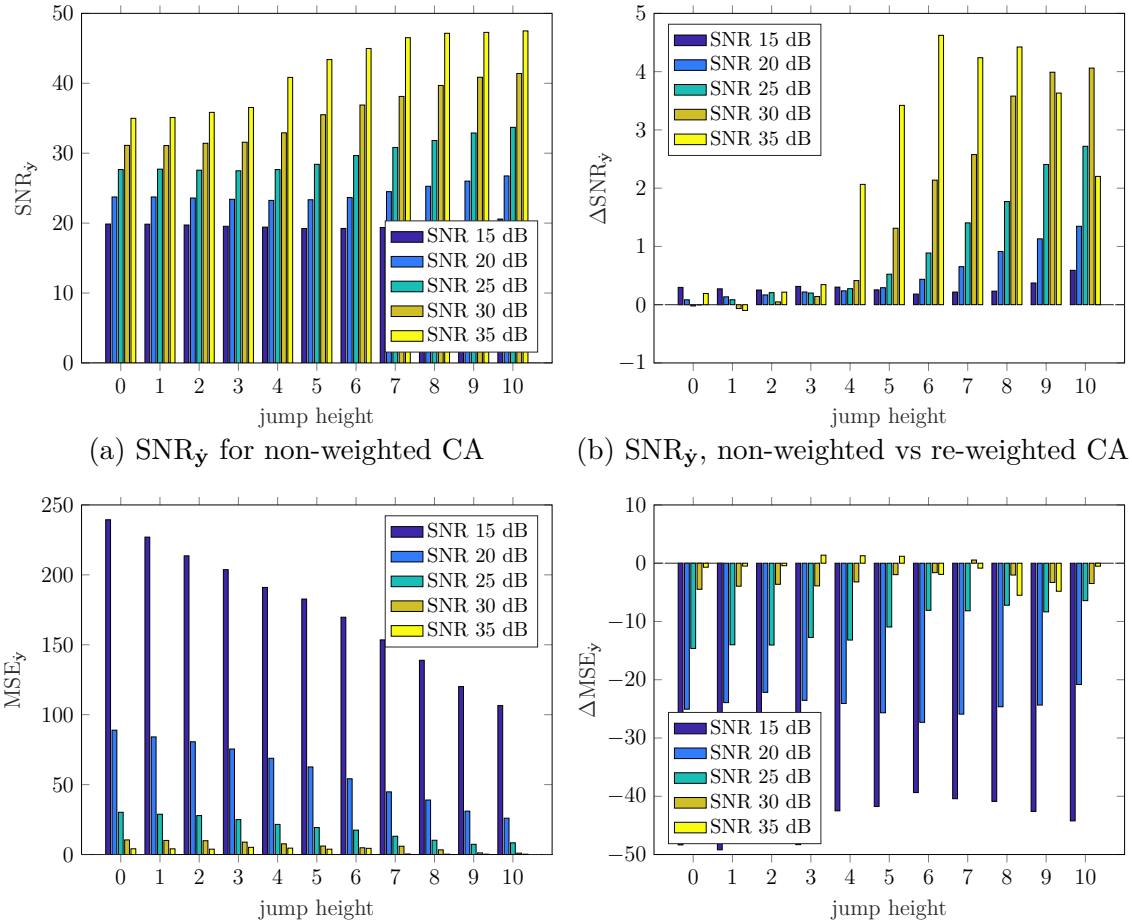
(c) MMR for non-weighted CA

(d) MMR, non-weighted vs re-weighted CA

(e) NoB for non-weighted CA

(f) NoB, non-weighted vs re-weighted CA

Fig. 5.23: AAR, MMR and NoB results for the non-weighted Condat algorithm (CA) and their comparison with the re-weighted Condat algorithm. The left-hand side figures show the results for the non-weighted Condat algorithm. The right-hand side figures show the difference between results obtained with the non-weighted and the re-weighted Condat algorithms. Presented results are for quadratic signals.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$ for non-weighted CA

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, non-weighted vs re-weighted CA

(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$ for non-weighted CA

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, non-weighted vs re-weighted CA

Fig. 5.24: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the non-weighted Condat algorithm (CA) and their comparison with the re-weighted Condat algorithm. The left-hand side figures show the results for the non-weighted Condat algorithm. The right-hand side figures show the difference between results obtained with the non-weighted and the re-weighted Condat algorithms. Presented results are for quadratic signals.

5.6.4 Partial conclusions

In all experiments, it holds that the results improve with a higher SNR and greater jump heights for all metrics, except for the AAR and MMR in the case of linear signals. This holds for both the non-weighted and the re-weighted Condat algorithms.

From comparison of the non-weighted and the re-weighted Condat algorithm, it follows that the denoising process of both linear and quadratic signals is better when the non-weighted Condat algorithm is used. The only exception can be found in the case of the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for linear signals with higher SNR. Generally, in the case of breakpoint detection, it holds that for the both linear and quadratic signals, it is better when the non-weighted Condat algorithm is used. Except for the NoB

results for quadratic signals with lower SNR.

The comparison of the Forward-backward based primal-dual (FBB-PD) algorithm and the non-weighted Condat algorithm is shown in form of graphs – on the left-hand side for linear signals and on the right-hand side for quadratic signals, see Figs. 5.25 and 5.26. Values above zero mean that better results are achieved with the FBB-PD algorithm and values below zero mean better results for the non-weighted Condat algorithm – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, the opposite applies.

The comparison of the FBB-PD and the non-weighted Condat algorithm indicates that the results of the FBB-PD algorithm solving the recovery problem (5.37) are better than results of the non-weighted Condat algorithm solving the recovery problem (5.48) for quadratic signals. It can be shown that the unconstrained versions of the recovery problems (5.37) and (5.48) are the same, when the λ is found (see Sec. 1.3.1).

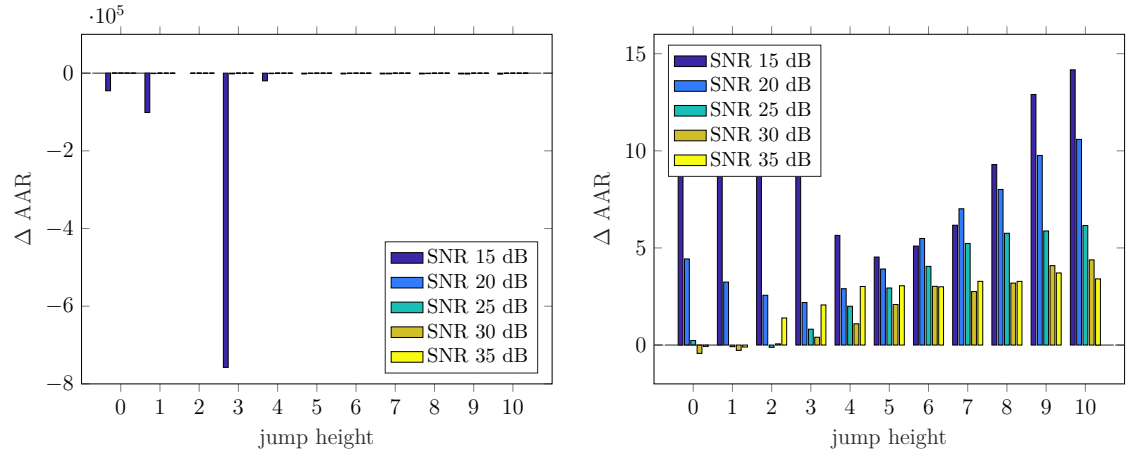
For the linear signals, the results are a little bit tricky. The AAR and MMR results are better using the non-weighted Condat algorithm. However, the NoB results are undoubtedly better using the FBB-PD algorithm. Better $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results are obtained with FBB-PD algorithm, except the signals with small jump height and low SNR. According to the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results, it is better to use the non-weighted Condat algorithm for the signals with lower SNR, and the FBB-PD algorithm for the signals with higher SNR.

In contrast with the FBB-PD algorithm, it was necessary to set one more parameter for the non-weighted Condat algorithm – the parameter δ for the proximal operator of function h_2. It is not easy to set the parameter δ to get the best results for all SNR. The parameter δ can be set in the way to improve the results for higher SNR but for costs of deterioration of the results for the signals with lower SNR. For the signals with big jump heights and high SNR, the differences between FBB-PD and the non-weighted Condat algorithm are not so significant.

5.7 Testing different types of bases

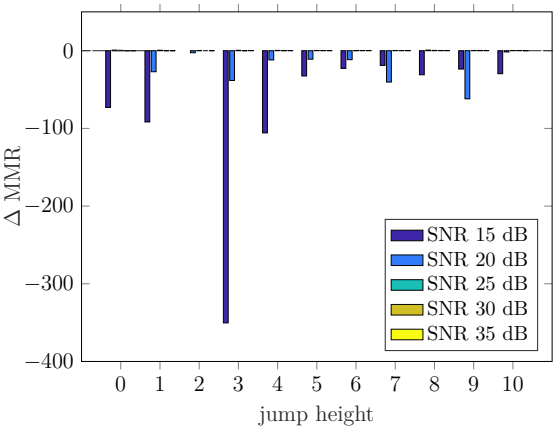
In the previous sections, the modified standard basis was used in all experiments. The disadvantage of using the modified standard basis is that it is necessary to set the regularisation weights for the individual polynomials of the basis, because the influence of the individual polynomials on the segmentation process is not equal and the achieved results are worse without weighting.

The aim of this section is to reduce the number of tunable parameters caused by using the modified standard basis. Therefore, different types of bases introduced in Sec. 5.3.3 will be exploited, which will avoid the weighting of basis polynomials.

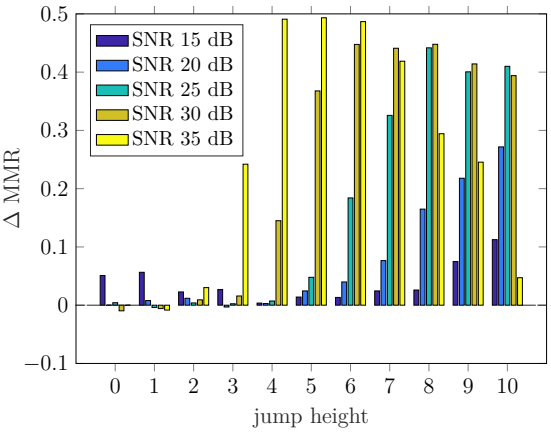


(a) AAR, FBB-PD vs CA – linear

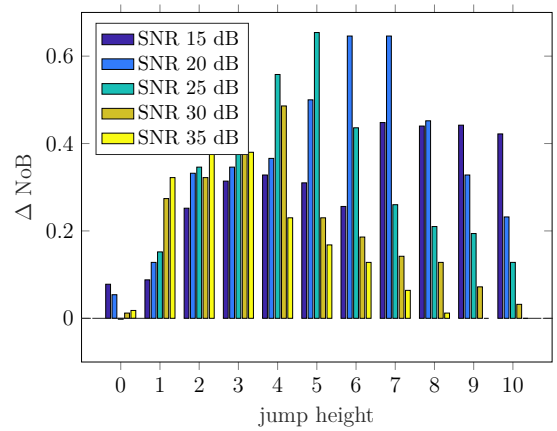
(b) AAR, FBB-PD vs CA – quadratic



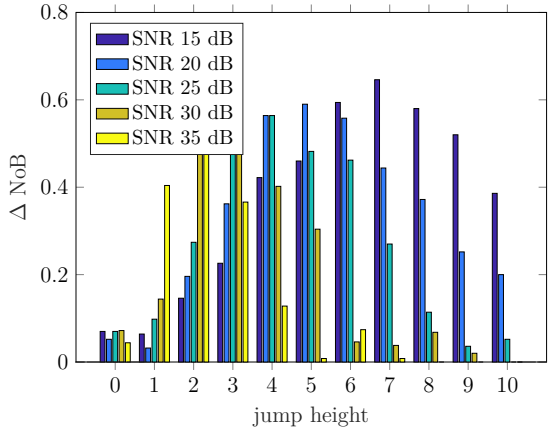
(c) MMR, FBB-PD vs CA – linear



(d) MMR, FBB-PD vs CA – quadratic

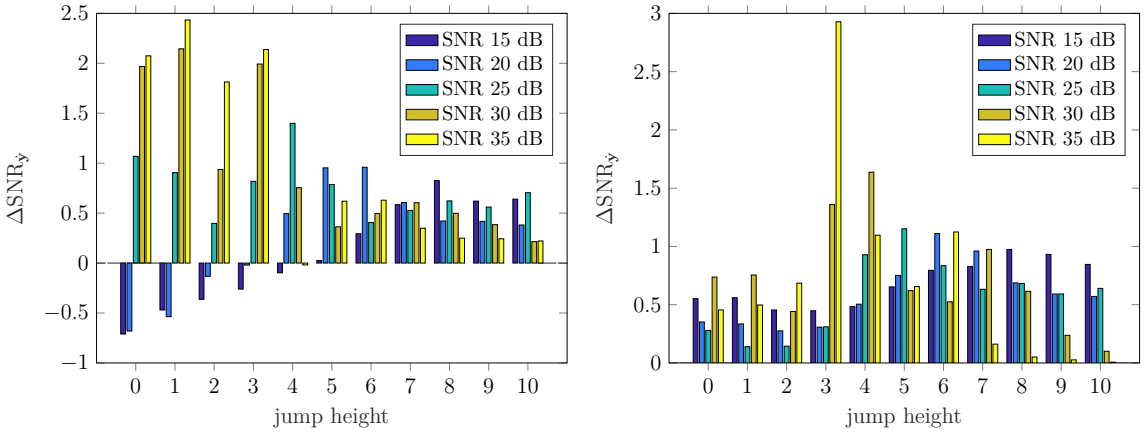


(e) NoB, FBB-PD vs CA – linear



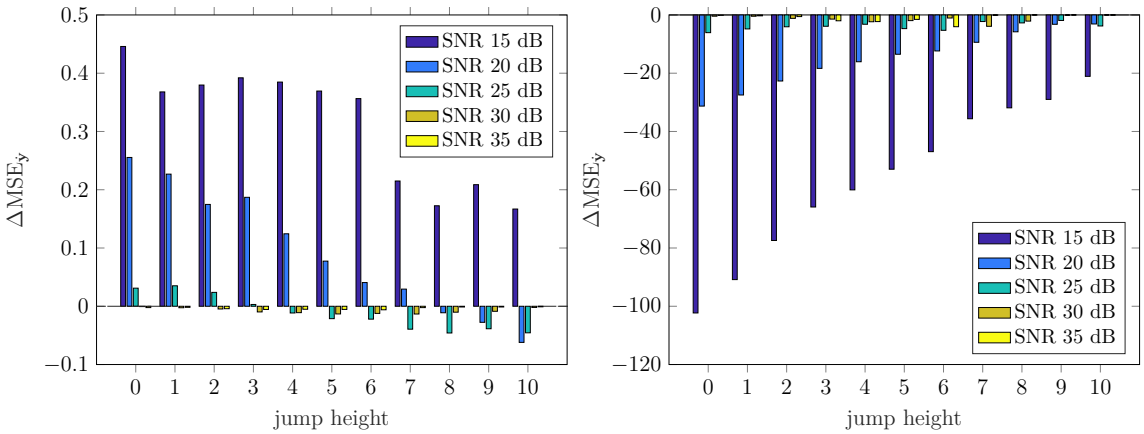
(f) NoB, FBB-PD vs CA – quadratic

Fig. 5.25: Comparison of AAR, MMR and NoB results of the non-weighted Condat algorithm (CA) with the results of the Forward-backward based primal-dual (FBB-PD) algorithm. The left-hand side figures show the difference between results obtained with the non-weighted Condat algorithm and the FBB-PD algorithm for linear signals. The right-hand side figures show the difference for quadratic signals.



(a) SNR$_{\dot{\mathbf{y}}}$, FBB-PD vs CA – linear

(b) SNR$_{\dot{\mathbf{y}}}$, FBB-PD vs CA – quadratic



(c) MSE$_{\dot{\mathbf{y}}}$, FBB-PD vs CA – linear

(d) MSE$_{\dot{\mathbf{y}}}$, FBB-PD vs CA – quadratic

Fig. 5.26: Comparison of SNR$_{\dot{\mathbf{y}}}$ and MSE$_{\dot{\mathbf{y}}}$ results of the non-weighted Condat algorithm (CA) with the results of the Forward-backward based primal-dual (FBB-PD) algorithm. The left-hand side figures show the difference between results obtained with the non-weighted Condat algorithm and the FBB-PD algorithm for linear signals. The right-hand side figures show the difference for quadratic signals.

This approach was already presented in our article [10], where it was demonstrated that using another types of bases is a promising way how to improve the segmentation results of the non-weighted Condat algorithm. The paper shows that the best results are achieved with R-bases and O-bases. Therefore, the experiments in this section will be performed only for these two types of bases.

From the comparison between the Forward-backward based primal-dual (FBB-PD) algorithm and the non-weighted Condat algorithm in Sec. 5.6, it seems that it is advantageous to use the FBB-PD algorithm. However, the non-weighted Condat algorithm offers a better opportunity for solving the task in more dimensions. The question is, whether using other types of the polynomial bases can help to improve the results of the non-weighted Condat algorithm. The aim is to find an algorithm

producing the best possible results with a minimal number of tunable parameters.

The optimisation problem for this section is formulated as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\text{reshape}(L\mathbf{x})\|_{21} \text{ s.t. } \|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2 \leq \delta, \tag{5.55}$$

where L is a linear operator, $\hat{\mathbf{x}}$ are estimated parametrisation coefficients, $\mathbf{x}$ are parametrisation coefficients, $\mathbf{y}$ is input signal, $\mathbf{P}$ is the matrix of polynomials, δ is the parameter reflecting the noise level and model error, and reshape() operator is defined in the same way as in previous sections (see definition (5.39)).

The linear operator L represents the stacked differences as defined in Eq. (5.38) for the FBB-PD algorithm. Since the non-weighted Condat algorithm is used, the definition of the linear operator L is identical as for the FBB-PD algorithm.

The optimisation problem is identical as in Secs. 5.5 and 5.6.

5.7.1 Definition of the recovery problem

For the purposes of this section, the FBB-PD algorithm and the non-weighted Condat algorithm are used. Therefore, two unconstrained versions of the recovery problem (5.55) have to be defined.

The unconstrained version of the recovery problem (5.55) for the needs of the FBB-PD algorithm is formulated as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2 + \|\text{reshape}(L\mathbf{x})\|_{21}\,. \tag{5.56}$$

On the other hand, the unconstrained version of the problem (5.55) for the needs of the non-weighted Condat algorithm is formulated as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\text{reshape}(L\mathbf{x})\|_{21} + \iota_{\{\mathbf{z}\,:\,\|\mathbf{y}-\mathbf{z}\|_2 \leq \delta\}}(\mathbf{P}\mathbf{x}), \tag{5.57}$$

where ι_C denotes the indicator function of a convex set C.

5.7.2 Algorithm used

The recovery problem (5.56) is solved via the FBB-PD algorithm (see Sec. 1.3.3). The FBB-PD algorithm is able to

$$\text{minimise} \quad f(\mathbf{x}) + h(L\mathbf{x}), \tag{5.58}$$

where both functions f and h are convex and L is a linear operator. According to the defined recovery problem (5.56), smooth function $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2^2$ and non-smooth function $h(L\mathbf{x}) = \|\text{reshape}(L\mathbf{x})\|_{21}$ are assigned.

The FBB-PD algorithm is defined in the same way as in Sec. 5.5.2. Individual steps of the algorithm are described in Algorithm 10.

The recovery problem (5.57) is solved via the non-weighted Condat algorithm (see Sec. 1.3.3). The non-weighted Condat algorithm is able to

$$\text{minimise } h_1(L_1\mathbf{x}) + h_2(L_2\mathbf{x}), \tag{5.59}$$

where both functions h_1 and h_2 are convex and L_1 and L_2 are linear operators. With respect to the general ability of the Condat algorithm to solve (1.37), according to the defined recovery problem (5.57), it is assigned: $f(\mathbf{x}) = 0$, $g(\mathbf{x}) = 0$, $h_1(L_1\mathbf{x}) = \|\text{reshape}(L\mathbf{x})\|_{21}$ and $h_2(L_2\mathbf{x}) = \iota_{\{\mathbf{z}\,:\,\|\mathbf{y}-\mathbf{z}\|_2 \leq \delta\}}(\mathbf{P}\mathbf{x})$.

The non-weighted variant of the Condat algorithm is defined in the same way as in Sec. 5.6.2, and individual steps of the algorithm are described in Algorithm 12.

5.7.3 Experiments

The experiments are performed, as in the previous sections, on datasets of linear and quadratic synthetic signals, which are introduced in Sec. 5.3. However, in contrast to the previous experiments, the number of signals in the datasets is decreased for these experiments. For the purpose of these experiments, set of 20 orthogonal bases ($\mathbf{P} = \mathbf{O}$) and set of 20 random orthogonal bases ($\mathbf{P} = \mathbf{R}$) are used, see Sec. 5.1.3 and Sec. 5.3.3 for more details. Since the number of testing bases is increased, the number of signals in the datasets is decreased to maintain an acceptable cost of the computational time. For each combination of signal, jump height and SNR, only 20 realisations of noise are generated instead of 50. These are the main differences compared to the experiments of the previous sections. For all experiments on linear and quadratic signals, the used bases consist of 3 polynomials ($K = 2$).

The results obtained by the non-weighted Condat algorithm and the FBB-PD algorithm are evaluated in relation to the signal properties and the results of both algorithms are compared to each other.

Several parameters need to be set for both the "Optimisation step" and the "Detection of segment borders step". The list of parameters for the experiments using the non-weighted Condat algorithm is given in Table 5.5 – for $K = 2$ with the difference in τ parameters settings, where $\tau_0 = \tau_1 = \tau_2 = 1$. The list of parameters for the experiments using the FBB-PD algorithm is given in Table 5.3 – for $K = 2$ with the difference in τ_k parameters settings, where $\tau_0 = \tau_1 = \tau_2 = \sigma_e$ and $\text{MAX}_{\text{IT}} = 500$. Both algorithms are stopped if either the convergence criterion or the maximum number of iterations (MAX_{IT}) is reached.

The obtained results are evaluated from two points of view – breakpoint detection accuracy, and signal denoising performance. The evaluation metrics are identical as in the previous sections. Also, the graphs demonstrating the results of the algorithms are created in a similar way as the graphs in the previous sections. The individual

bar values in the presented graphs are averaged across the results of all 20 O-bases and 20 R-bases, respectively.

Linear synthetic signals

The first set of experiments is performed on the dataset of linear synthetic signals, which consists of 11,000 noisy signals in total. For the purpose of these experiments, the polynomial bases $\mathbf{P}$ are the orthogonal bases $\mathbf{O}$ or random orthogonal bases $\mathbf{R}$ of size $N \times (K+1)$, where $K = 2$. Several items are subject to vary within the experiments, configuring problem (5.55):

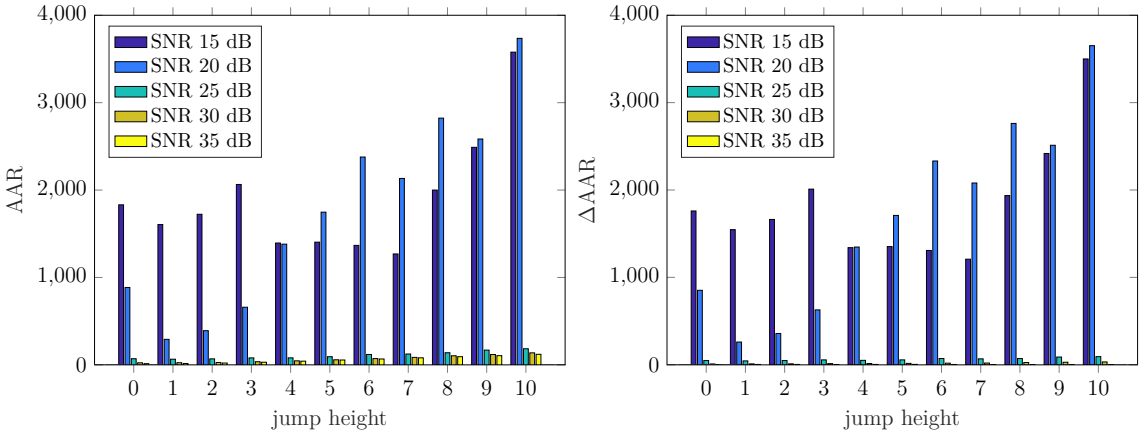
- the input signal $\mathbf{y}$,
- the polynomial basis $\mathbf{P}$,
- parameter δ in the Condat algorithm,
- regularisation parameters τ_0, τ_1, and τ_2 in the FBB-PD algorithm,
- parameter σ in the FBB-PD algorithm.

Evaluation of the FBB-PD algorithm using O-bases and R-bases At first, the results obtained using O-bases are evaluated. The AAR, MMR and NoB results are shown in form of graphs on the left-hand side in Fig. 5.27. The AAR results are shown in Fig. 5.27a; for all jump heights, the AAR is higher for the lower SNR. The MMR results are shown in Fig. 5.27c; for almost all jump heights, the MMR is higher for the lower SNR. For NoB, it holds that the values are higher with greater jump height and higher SNR, see Fig. 5.27f. From the NoB evaluation, it follows that the maximum of correctly detected breakpoints is achieved with higher SNR (lower noise level) and significant jump height between signal segments, as expected. Due to the nature of processed signals, the maximum achievable NoB is 5. Note that the maximum of detected breakpoints is not achieved.

The $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are shown on the left-hand side of Fig. 5.28. Fig. 5.28a indicates that the $\mathrm{SNR}_{\dot{\mathbf{y}}}$ increases with increasing jump height. The $\mathrm{MSE}_{\dot{\mathbf{y}}}$ values decrease with higher SNR see Fig. 5.28c.

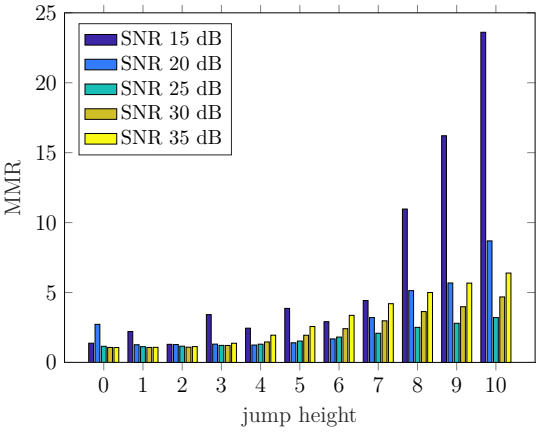
The synthetic linear signals were also processed using R-bases. The character of the results using R-bases is very similar to the results obtained using O-bases.

Comparison of the FBB-PD algorithm and the non-weighted Condat algorithm using O-bases and R-bases At first, the results obtained using O-bases are evaluated. The comparison of the FBB-PD algorithm and the non-weighted Condat algorithm is shown in the form of graphs on the right-hand side of Figs. 5.27 and 5.28. Values above zero indicate that better results are achieved with the FBB-PD algorithm and values below zero mean better results for the non-weighted Condat

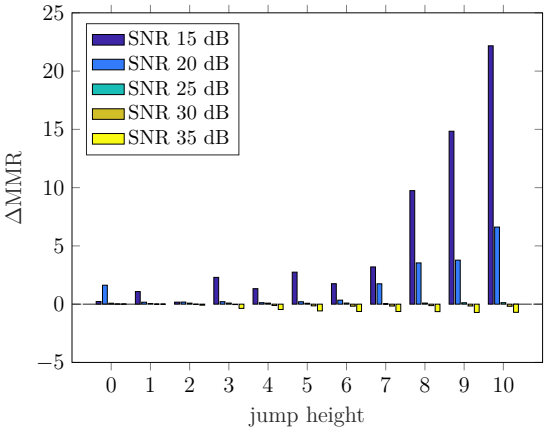


(a) AAR for FBB-PD – O-bases

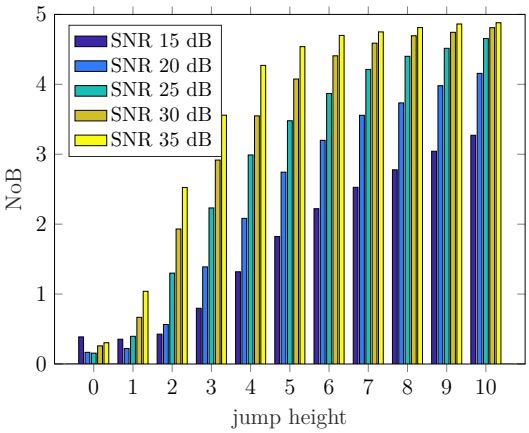
(b) AAR, FBB-PD vs CA – O-bases



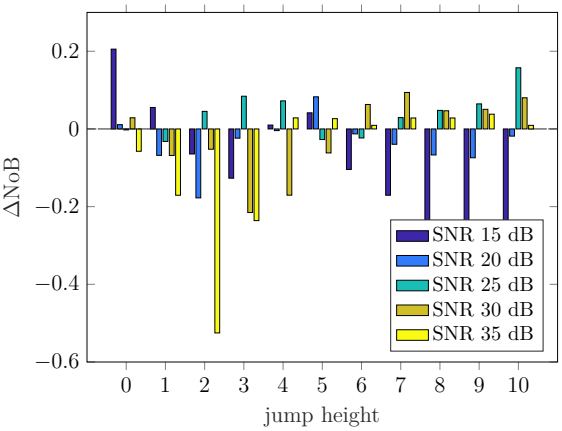
(c) MMR for FBB-PD – O-bases



(d) MMR, FBB-PD vs CA – O-bases

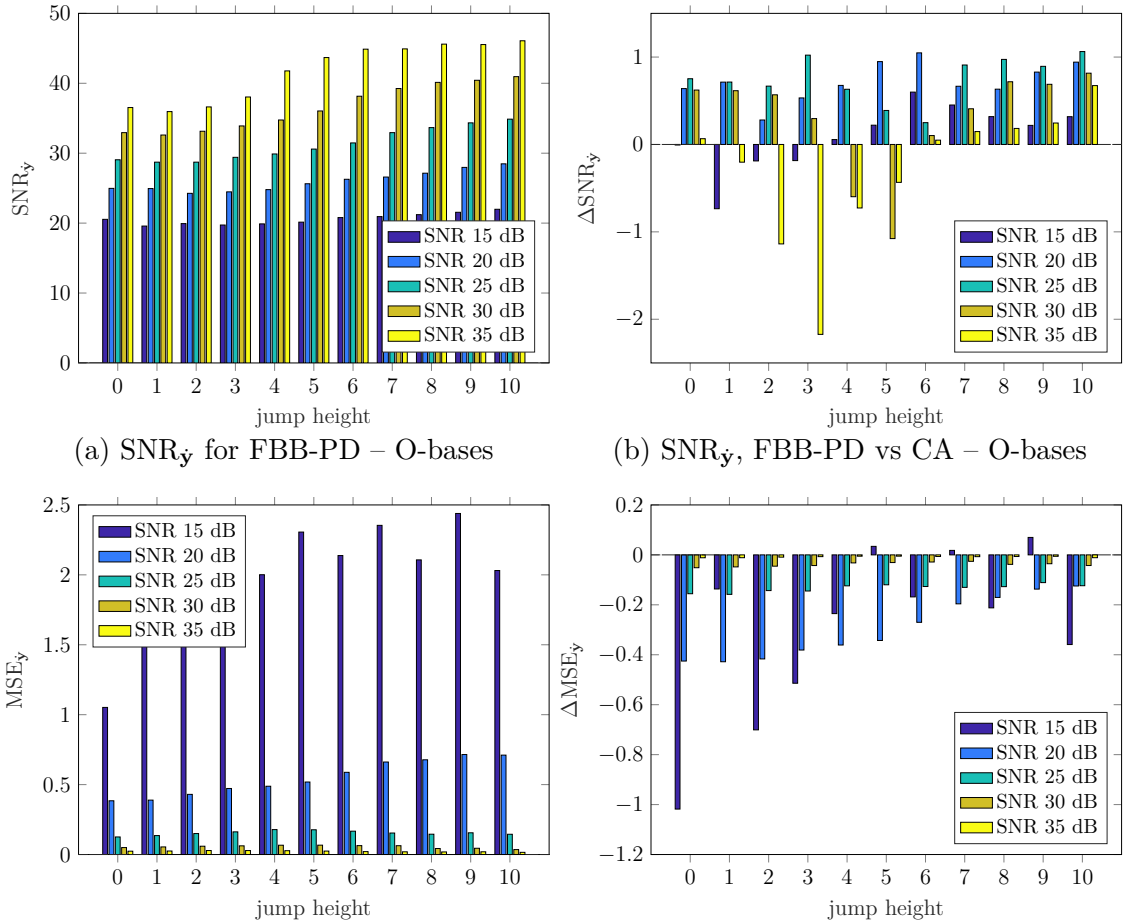


(e) NoB for FBB-PD – O-bases



(f) NoB, FBB-PD vs CA – O-bases

Fig. 5.27: AAR, MMR and NoB results for the FBB-PD algorithm using O-bases and their comparison with the non-weighted Condat algorithm using O-bases. The left-hand side figures show the results for the FBB-PD algorithm. The right-hand side figures show the difference between results obtained with the FBB-PD algorithm and the non-weighted Condat algorithms. Presented results are for linear signals.



(a) $\mathrm{SNR}_{\dot{\mathbf{y}}}$ for FBB-PD – O-bases

(b) $\mathrm{SNR}_{\dot{\mathbf{y}}}$, FBB-PD vs CA – O-bases

(c) $\mathrm{MSE}_{\dot{\mathbf{y}}}$ for FBB-PD – O-bases

(d) $\mathrm{MSE}_{\dot{\mathbf{y}}}$, FBB-PD vs CA – O-bases

Fig. 5.28: $\mathrm{SNR}_{\dot{\mathbf{y}}}$ and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results for the FBB-PD algorithm using O-bases and their comparison with the non-weighted Condat algorithm using O-bases. The left-hand side figures show the results for the FBB-PD algorithm. The right-hand side figures show the difference between results obtained with the FBB-PD algorithm and the non-weighted Condat algorithms. Presented results are for linear signals.

algorithm – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, the opposite applies.

The AAR and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better using FBB-PD algorithm. For higher SNR, the MMR results are better using the non-weighted Condat algorithm. For lower SNR, the MMR results are better using the FBB-PD algorithm. It is not evident, if the NoB results are better with using the FBB-PD algorithm or the non-weighted Condat algorithm. For almost all signals, the $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results are better for the FBB-PD algorithm.

The character of the results using R-bases is almost similar with the results using O-bases.

Quadratic signals

The second set of experiments is performed on the dataset of quadratic synthetic signals, which consists of 11,000 noisy signals in total. For the purpose of these experiments, the polynomial bases $\mathbf{P}$ are the orthogonal bases $\mathbf{O}$ or random orthogonal bases $\mathbf{R}$ of size $N \times (K+1)$, where $K = 2$. Several items are subject to vary within the experiments, configuring the problem (5.55):

- the input signal $\mathbf{y}$,
- the polynomial basis $\mathbf{P}$,
- parameter δ in the Condat algorithm,
- regularisation parameters τ_0, τ_1, and τ_2 in the FBB-PD algorithm,
- parameter σ in the FBB-PD algorithm.

Evaluation of the FBB-PD algorithm using O-bases and R-bases At first, results obtained using O-bases are evaluated. The character of the AAR, $\text{SNR}_{\dot{\mathbf{y}}}$, and $\text{MSE}_{\dot{\mathbf{y}}}$ results for quadratic signals is very similar to the results obtained for the linear signals. For MMR results, it holds that for lower jump heights, the MMR is higher for the lower SNR. For the greater jump height, the opposite holds. For NoB results, it holds that for the three biggest jump heights and the two highest SNR, the maximum NoB of 5 is achieved.

The synthetic quadratic signals were also processed using R-bases. The character of the results using R-bases is very similar to the results obtained using O-basis.

Comparison of the FBB-PD algorithm and the non-weighted Condat algorithm using O-bases and R-bases At first, the results obtained using O-bases are evaluated. The character of the AAR, MMR, $\text{SNR}_{\dot{\mathbf{y}}}$, and $\text{MSE}_{\dot{\mathbf{y}}}$ results for quadratic signals using the O-bases is similar to the results for linear signals using the O-bases. The NoB results have, however, different character, for almost all signals, better results are obtained with the non-weighted Condat algorithm.

Comparing the results using the O-bases and R-bases on quadratic signals leads to a similar conclusion as in the case of linear signals, i.e. the character of the results remains the same independently on the type of the basis used.

Comparison of the non-weighted Condat algorithm using O-bases and R-bases Because the non-weighted Condat algorithm is assumed to be the preferred option for signal segmentation, the comparison of R-bases and O-bases is done only with the non-weighted Condat algorithm, which is shown in Figs. 5.29 and 5.30. The left-hand side graphs show the comparison for linear signals and the right-hand side graphs present the results for quadratic signals. Values above zero mean that better results are achieved with the R-bases and values below zero mean better results for

the O-bases – this applies for AAR, MMR, NoB and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ values; for $\mathrm{MSE}_{\dot{\mathbf{y}}}$, the opposite applies .

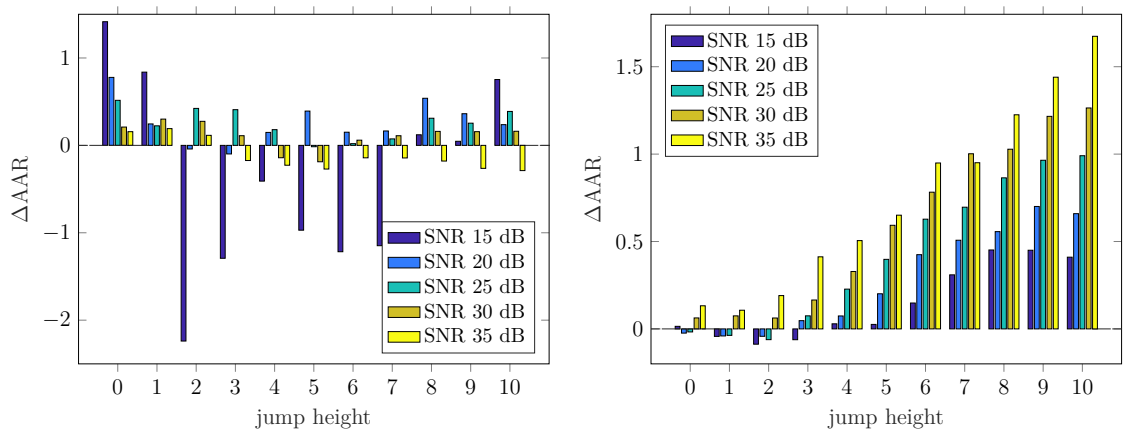
At first, the linear signals were evaluated. For almost all signals, the AAR results are better using R-bases, see Fig. 5.29a. For MMR, it holds that the results are better using O-bases, see Fig. 5.29c. From the NoB point of view, the results for half of the signals are better with O-bases and for the second half of the signals with the R-bases, see Fig. 5.29e. For the signals with lower SNR, it holds that the $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results are better with the R-bases. For the signals with higher SNR, it holds that the $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results are better with the O-bases, see Fig. 5.30a. For the signals with lower SNR, it holds that the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better with the O-bases. For the signals with higher SNR, it holds that the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better with the R-bases, see Fig. 5.30c.

The results for the quadratic signals suggest that for almost all signals, the AAR, MMR, NoB, and $\mathrm{SNR}_{\dot{\mathbf{y}}}$ results are better using R-bases, see Figs. 5.29b, 5.29d, 5.29f, 5.30b. The only exception is the $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results, where better results were obtained using O-bases, see Fig. 5.30d

5.7.4 Partial conclusions

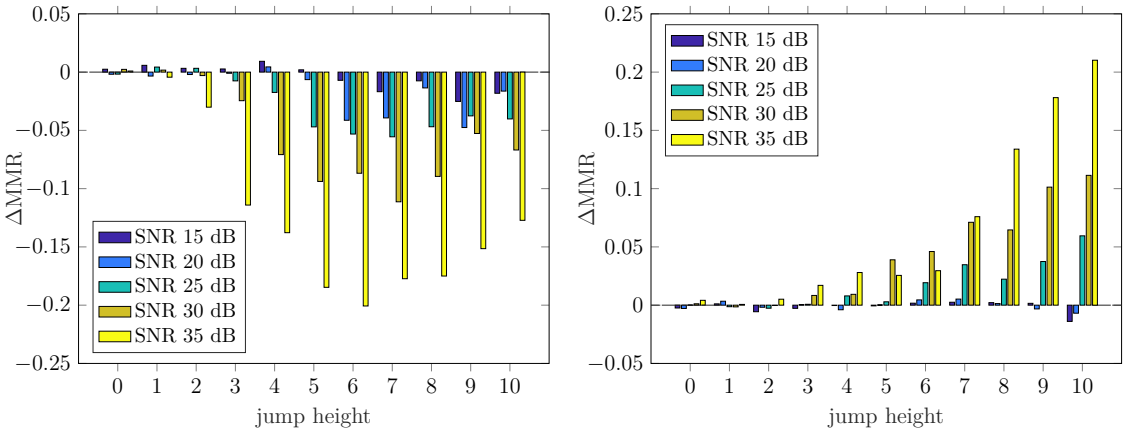
In this section, two types of bases (O-bases and R-bases) were used. Based on the results from Sec. 5.6, the experiments were performed for the FBB-PD algorithm and also for the non-weighted Condat algorithm. For both linear and quadratic signals, it applies that the AAR, MMR, $\mathrm{SNR}_{\dot{\mathbf{y}}}$, and $\mathrm{MSE}_{\dot{\mathbf{y}}}$ results are better for the FBB-PD algorithm. Nevertheless, the NoB results give better results with using the non-weighted Condat algorithm. From the segmentation process point of view, the NoB metric is the most important. Therefore, the non-weighted Condat algorithm seems to be a preferred option.

As a consequence, the comparison of R-bases and O-bases was done only for the non-weighted Condat algorithm, see Figs. 5.29 and 5.30. The results for the quadratic signals suggest that for almost all signals it is better to use R-bases. However, for the linear signals, the maximum of NoB was not achieved using either algorithm (the FBB-PD or the non-weighted Condat algorithm) and type of the basis (O-bases and R-bases). From the results in Sections 5.5 and 5.6, it follows that it is recommended to use the modified standard basis $\mathbf{S}$ for the linear signals instead of the 3-polynomial R-bases and O-bases.



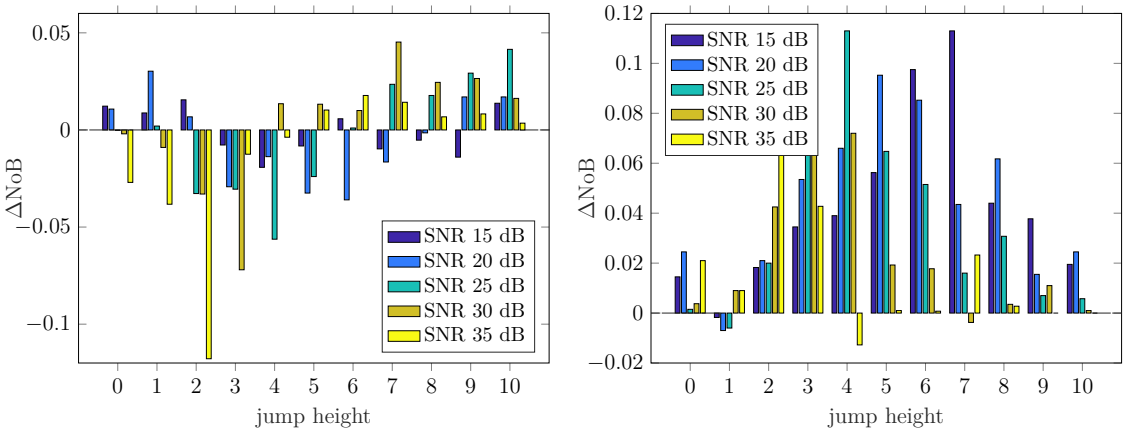
(a) AAR, R-bases vs O-bases – linear

(b) AAR, R-bases vs O-bases – quadratic



(c) MMR, R-bases vs O-bases – linear

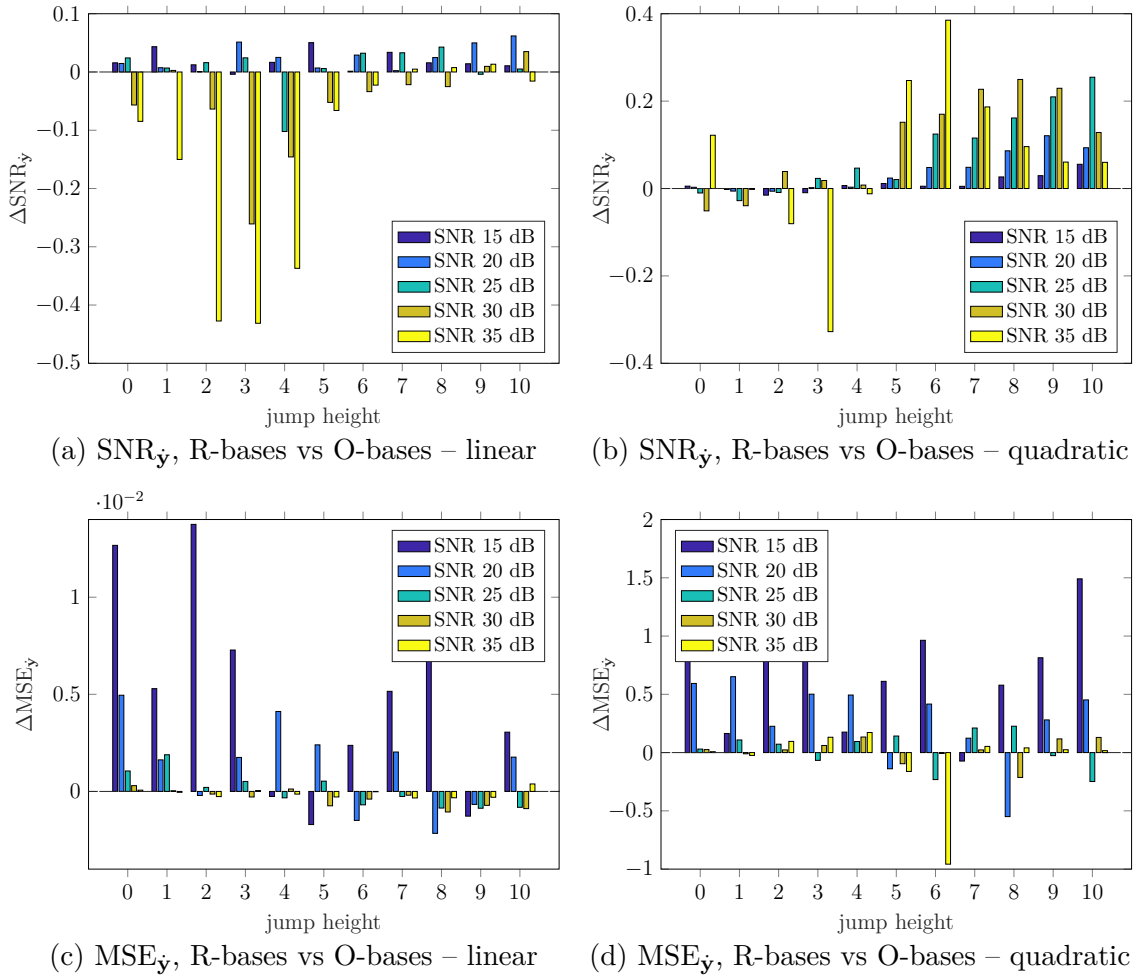
(d) MMR, R-bases vs O-bases – quadratic



(e) NoB, R-bases vs O-bases – linear

(f) NoB, CA R-bases vs O-bases – quadratic

Fig. 5.29: Comparison of AAR, MMR and NoB results of the non-weighted Condat algorithm using R-bases with the results of the non-weighted Condat algorithm using O-bases. The left-hand side figures show the difference between results obtained for linear signals and the right-hand side figures show the results for quadratic signals. Values above zero mean better results for R-bases and vice versa.



(a) SNR$_{\dot{\mathbf{y}}}$, R-bases vs O-bases – linear

(b) SNR$_{\dot{\mathbf{y}}}$, R-bases vs O-bases – quadratic

(c) MSE$_{\dot{\mathbf{y}}}$, R-bases vs O-bases – linear

(d) MSE$_{\dot{\mathbf{y}}}$, R-bases vs O-bases – quadratic

Fig. 5.30: Comparison of SNR$_{\dot{\mathbf{y}}}$ and MSE$_{\dot{\mathbf{y}}}$ results of the non-weighted Condat algorithm using R-bases with the results of the non-weighted Condat algorithm using O-bases. The left-hand side figures show the difference between results obtained for linear signals and the right-hand side figures show the results for quadratic signals. SNR$_{\dot{\mathbf{y}}}$ values above zero mean better results for R-bases and vice versa, however, for MSE$_{\dot{\mathbf{y}}}$ results, it holds the opposite.

6 EDGE DETECTION IN IMAGES

From the previous chapter, it is evident that the presented 1D signal segmentation process provides promising results in breakpoint detection. This process can be easily extended for the use in 2D as well. The process of breakpoint detection on images is called edge detection and it is described in more details in the following sections.

6.1 General description of the proposed method

In this section, a general concept of the proposed image edge detection is introduced. This section is divided into several subsections describing individual components, which are needed for the edge detection process. First, the description of the 2D signal model is formulated. Then, the types of processed signals and the bases used are introduced. Finally, the entire concept of the image edge detection process is presented.

6.1.1 2D signal model description

The 2D signal model used in this Thesis is a natural extension of the 1D signal model described in Sec. 5.1.1. Therefore, the images are assumed to consist of non-overlapping piecewise-polynomial patches. The edges in the image represent positions where the adjacent patches (segments) change their polynomial characterisation. Within the patches (segments), the respective representation stays steady.

Recall that a piecewise-polynomial 1D signal is described as follows:

$$\mathbf{y} = \mathbf{P}\mathbf{x} = \left[\mathbf{P}_0 \mid \cdots \mid \mathbf{P}_K\right] \begin{bmatrix} \mathbf{x}_0 \\ \hline \vdots \\ \hline \mathbf{x}_K \end{bmatrix} = \mathbf{p}_0 \odot \mathbf{x}_0 + \cdots + \mathbf{p}_K \odot \mathbf{x}_K, \tag{6.1}$$

where $\mathbf{y} \in \mathbb{R}^N$ is the 1D signal, $\mathbf{P}_k \in \mathbb{R}^{N \times N}$ is the polynomial matrix with the polynomial $\mathbf{p}_k$ on its diagonal, and $\mathbf{x}_k \in \mathbb{R}^N$ are the parametrisation vectors.

Let the 2D signal (image) $\mathbf{Y} \in \mathbb{R}^{M \times N}$ contain entries $Y[m, n]$, $m = 1, \ldots, M$, $n = 1, \ldots, N$. The image is modeled in the similar way as the 1D signal (see Eq. (6.1)):

$$\mathbf{Y} = \bar{\mathbf{P}}_{00} \odot \mathbf{X}_{00} + \bar{\mathbf{P}}_{01} \odot \mathbf{X}_{01} + \cdots + \bar{\mathbf{P}}_{KK} \odot \mathbf{X}_{KK}, \tag{6.2}$$

where $\mathbf{X}_{kl}$ are matrices of parametrisation coefficients of the size $M \times N$, and $\bar{\mathbf{P}}_{kl}$ are basis images of size $M \times N$ as defined below in Eq. (6.3).

The image $\mathbf{Y}$ is assumed to be piecewise-polynomial and consists of S independent patches (segments). Each patch $s \in \{1, \ldots, S\}$ is described by $(K+1)^2$ basis polynomial images $\mathbf{P}_{kl}$. Matrices $\mathbf{X}_{kl}$ are assumed to be piecewise-constant within the particular patches, with the change at the patch borders.

For the images, the polynomial basis images $\bar{\mathbf{P}}_{kl} \in \mathbb{R}^{M \times N}$ are needed instead of the polynomial vectors $\mathbf{p}_k$. The extension from the polynomial vectors to the polynomial basis images is done through the Kronecker product of 1D polynomial vectors $\mathbf{p}_{k\mathrm{v}} \in \mathbb{R}^N$ and $\mathbf{p}_{l\mathrm{h}} \in \mathbb{R}^M$, where the subscripts "v" and "h" represent the vertical and the horizontal directions of the 1D polynomial vectors, respectively. And the subscripts "k" and "l" represent the degree of the vertical and horizontal polynomials, respectively. Note that for the definition of the 2D signal model, the degrees of the polynomials in the vertical and the horizontal direction are supposed to be identical, i.e. K.

The set of basis 2D polynomials are formed by all the combinations of $\mathbf{p}_{k\mathrm{v}}$ and $\mathbf{p}_{l\mathrm{h}}$, where $k = 0, \ldots, K$ and $l = 0, \ldots, K$ such that

$$\begin{aligned} \bar{\mathbf{P}}_{00} &= \mathbf{p}_{0\mathrm{v}} \cdot \mathbf{p}_{0\mathrm{h}}^{\top} \\ \bar{\mathbf{P}}_{01} &= \mathbf{p}_{0\mathrm{v}} \cdot \mathbf{p}_{1\mathrm{h}}^{\top} \\ &\vdots \\ \bar{\mathbf{P}}_{KK} &= \mathbf{p}_{K\mathrm{v}} \cdot \mathbf{p}_{K\mathrm{h}}^{\top} \end{aligned} \tag{6.3}$$

making altogether $(K+1)^2$ polynomial basis images. When both the vertical and horizontal polynomial vectors form the respective 1D orthonormal bases, the new generated polynomial basis images form the 2D orthonormal basis, the proof of this statement is given in the Appendix A (see Proof A.2).

Also, the 2D signal model can be expressed as a matrix-vector multiplication

$$\mathbf{y} = \mathbf{P}\mathbf{x}. \tag{6.4}$$

To show this, the vectorised form $\mathbf{y}$ and $\mathbf{x}$ of the image $\mathbf{Y}$ and parametrisation matrices $\mathbf{X}_{kl}$, respectively, need to be defined. Also, the polynomial matrix $\mathbf{P}$ is defined in a different way than in Eq. (6.1). The below-introduced definitions of $\mathbf{y}, \mathbf{P}$, and $\mathbf{x}$ are used in the entire Chapter 6. Vectorisation can be easily done with the vectorisation operator $\mathrm{vec}(\cdot)$, which stacks the image columns, one after another, into a single column.

The model (6.2) reads

$$\mathbf{y} = \mathbf{P}_{00} \cdot \mathbf{x}_{00} + \cdots + \mathbf{P}_{KK} \cdot \mathbf{x}_{KK}, \tag{6.5}$$

where

$$\begin{aligned} \mathbf{y} &= \mathrm{vec}(\mathbf{Y}), \\ \mathbf{P}_{kl} &= \mathrm{diag}(\mathrm{vec}(\bar{\mathbf{P}}_{kl})), \\ \mathbf{x}_{kl} &= \mathrm{vec}(\mathbf{X}_{kl}). \end{aligned} \tag{6.6}$$

In a short and more convenient form,

$$\begin{aligned} \mathbf{y} = \mathbf{P}\mathbf{x} &= [\mathbf{P}_{00} \mid \cdots \mid \mathbf{P}_{KK}] \begin{bmatrix} \mathbf{x}_{00} \\ \hline \vdots \\ \hline \mathbf{x}_{KK} \end{bmatrix} \\ &= \left[\mathrm{diag}(\mathrm{vec}(\bar{\mathbf{P}}_{00})) \mid \cdots \mid \mathrm{diag}(\mathrm{vec}(\bar{\mathbf{P}}_{KK}))\right] \begin{bmatrix} \mathrm{vec}(\mathbf{X}_{00}) \\ \hline \vdots \\ \hline \mathrm{vec}(\mathbf{X}_{KK}) \end{bmatrix}, \end{aligned} \tag{6.7}$$

where $\mathbf{y} \in \mathbb{R}^{MN}$, $\mathbf{P} \in \mathbb{R}^{MN \times (K+1)^2 MN}$ and $\mathbf{x} \in \mathbb{R}^{(K+1)^2 MN}$.

For the parametrisation vectors $\mathbf{x}_{kl}$, it holds the same as for the parametrisation vectors $\mathbf{x}_k$ in 1D signal model description. Vectors $\mathbf{x}_{kl}$ are assumed to be piecewise-constant. The finite difference operator ∇ (defined in Sec. 5.1.1) applied to them produces sparse vectors $\nabla\mathbf{x}_{kl}$ with non-zero components occupying the same positions across $k = 0, \ldots, K$ and $l = 0, \ldots, K$.

The above-mentioned formulations describe a clean polynomial image without noise corruption. Unfortunately, in the real world, it is common that the images are corrupted by noise. For the purpose of this Thesis, uncorrelated Gaussian noise with zero mean and non-zero variance is used. The definition of such an image is

$$\mathbf{y} = \mathbf{P}\mathbf{x} + \mathrm{vec}(\mathbf{E}), \tag{6.8}$$

where $\mathbf{E}$ represents the noise matrix of size $M \times N$.

6.1.2 Formulation of the whole concept of image edge detection

Concept of the image edge detection is divided into two main steps:

- Optimisation using a proximal splitting algorithm,
- Edge identification.

These steps are similar to the 1D signal segmentation and denoising process the steps, except for the last step "Smoothing of detected segments," which is not utilised in proposed image edge detection method. Since vectorised images are used for the process of the image edge detection, the illustration in Fig. 5.4 demonstrates the

individual steps of both 1D and 2D process (except the last step "Smoothing of detected segments"). The definition of the first step is the same as for 1D approach, presented in the Sec. 5.1.4.

Edge identification

Parametrisation coefficients $\hat{\mathbf{x}}$ obtained as the optimisers of the recovery problem (5.17) allow simple estimation of the underlying noiseless signal $\hat{\mathbf{y}}$ according to $\hat{\mathbf{y}} = \mathbf{P}\hat{\mathbf{x}}$. There are two approaches to edge detection.

The first approach is simple. First the denoised reconstructed image $\hat{\mathbf{Y}}$ is obtained by the matrisation of the signal $\hat{\mathbf{y}}$. Since the denoising is often the first step in edge detection, one can think about the application of the Sobel operator (see Sec. 3.1) to the reconstructed image to obtain an image of differences $\mathbf{D}$ (raw edge detection).

The second approach starts with the matrisation of parametrisation vectors $\hat{\mathbf{x}}_{kl}$. Obtained parametrisation matrices $\hat{\mathbf{X}}_{kl}$ should be optimally piecewise-constant within the particular patches. After the application of the edge operator, the particular images of differences $\mathbf{D}_{kl}$ should have non-zero values (in each matrix) indicating the edges at the same positions. It was decided that the multiple information contained in $\hat{\mathbf{X}}_{kl}$ will be exploited. Therefore, the Sobel operator is computed on all individual parametrisation matrices resulting in images of differences $\mathbf{D}_{kl}$. Afterwards, a single image of differences (raw edge detection) $\mathbf{D}$ is computed out of all obtained $\mathbf{D}_{kl}$ using the ℓ_2-norm according to the formula

$$\mathbf{D}[m,n] = \sqrt[2]{(\mathbf{D}_{00}[m,n])^2 + \cdots + (\mathbf{D}_{KK}[m,n])^2}, \text{ for } m = 1,\ldots,M, n = 1,\ldots,N. \tag{6.9}$$

6.2 Datasets for experiments

For the experiments, datasets of images and bases are needed. The image dataset is divided into three groups: train, validation, and test group.

6.2.1 Natural images

For the experiments, the Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500) [72] is used. This dataset consists of 500 natural RGB images $\mathbf{Y}_{\mathrm{RGB}}$. Each image of the BSDS500 dataset was manually annotated on average by five different subjects (see some examples of the natural images and their annotations in Fig. 6.1). Further information on how the annotations were taken can be found in [69]. The BSDS500 is divided into 3 groups: 200 train images, 100 images for validation, and 200 test images. BSDS500 with its human annotations serves as the ground truth for comparing different segmentation and edge detection methods.

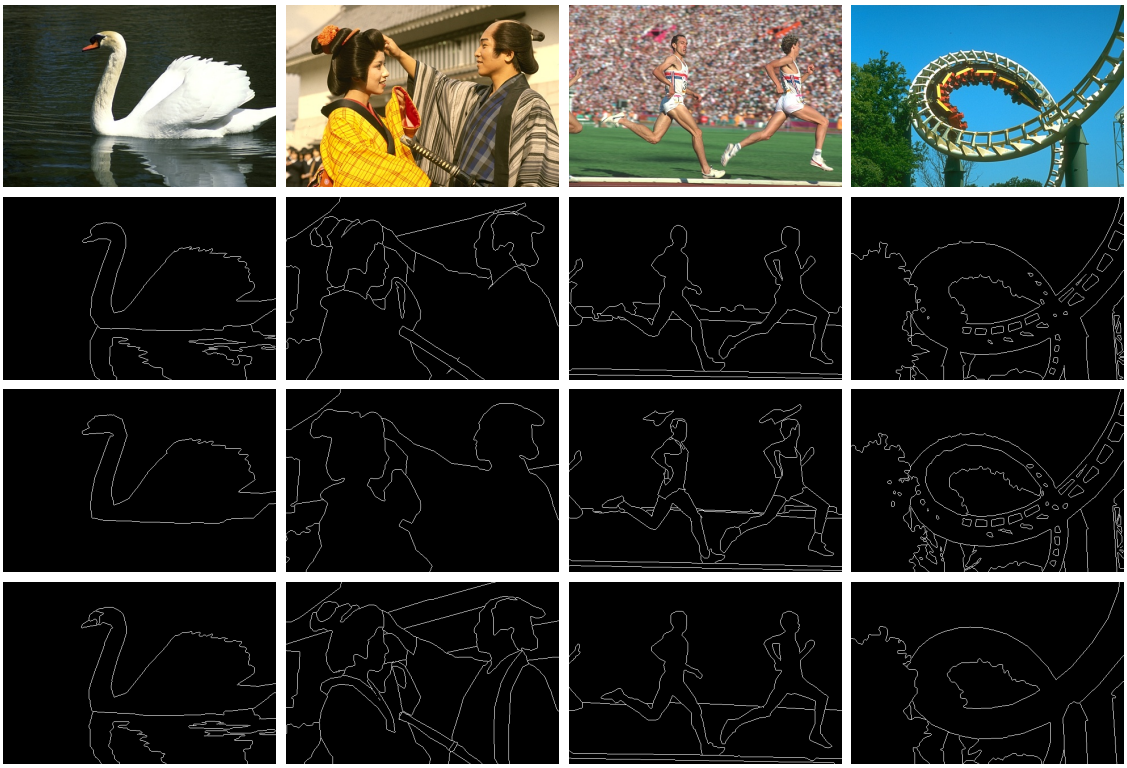


Fig. 6.1: Several examples of RGB natural images of BSDS500 dataset with their annotations by humans.

For the testing, also the Berkeley Segmentation Data Set and Benchmarks 300 (BSDS300) [72] is used. This dataset consists of 300 natural RGB images $\mathbf{Y}_{\mathrm{RGB}}$, and is divided into 2 groups: 200 train images, 100 test images. The train subset of BSDS300 is equal to the train subset of BSDS500 and the test subset of BSDS300 is same as the validation subset of BSDS500.

The algorithm presented in this Thesis was tuned only on the train and validation groups of images. The final evaluation is performed on the test group of images.

6.2.2 Types of processed images

For the purpose of this Thesis, three groups of real images $\mathbf{Y} \in \mathbb{R}^{M \times N}$ are used.

The first group of images consists of the set of RGB images denoted as $\mathbf{Y}_{\mathrm{RGB}}$. The second group of images consists of the grayscale images $\mathbf{Y}_{\mathrm{gray}}$, which were obtained from the first group of RGB images by converting the RGB image to the grayscale image. For the conversion, the Matlab function "rgb2gray" was used. By converting, the hue and saturation information are eliminated while the luminance is retained.

The third group is formed with noisy grayscale images $\mathbf{Y}_{\mathrm{noisy}}$. The Gaussian i.i.d. noise $\mathbf{E}$ is added to the grayscale images $\mathbf{Y}_{\mathrm{gray}}$, resulting in grayscale noisy

images $\mathbf{Y}_{\text{noisy}}$, such that

$$\mathbf{Y}_{\text{noisy}} = \mathbf{Y}_{\text{gray}} + \mathbf{E}, \quad e_{m,n} \sim \mathrm{N}(0, \sigma_{\mathrm{E}}^2). \tag{6.10}$$

With these images, the signal-to-noise ratio (SNR) can be determined, defined as

$$\mathrm{SNR}\left(\mathbf{Y}_{\text{noisy}}, \mathbf{Y}_{\text{gray}}\right) = 20 \cdot \log_{10} \frac{\|\mathbf{Y}_{\text{gray}}\|_2}{\|\mathbf{Y}_{\text{noisy}} - \mathbf{Y}_{\text{gray}}\|_2}. \tag{6.11}$$

To generate an image with a defined SNR, the standard deviation σ_{E} of the respective noise needs to be computed such that

$$\sigma_{\mathrm{E}} = \frac{\|\mathbf{Y}_{\text{gray}}\|_2}{\sqrt{MN \cdot 10^{\frac{\mathrm{SNR}}{10}}}}. \tag{6.12}$$

It can be noticed that the resulting σ_{E} is influenced by the energy of the "clean" image as well.

6.2.3 Bases

For the experiments, two types of bases consisting of the $(K+1)^2$ basis images, where $K = 2$, are used. The generation process of polynomial basis images is defined in Eq. (6.3) and it is independent of the particular choice of the basis type.

The first one is a standard basis $\bar{\mathbf{S}}$, which was generated using the polynomials of the modified standard basis $\mathbf{S}$ (see Sec. 5.1.3) using Eq. (6.3). See Fig. 6.2 for examples of standard basis images of $\bar{\mathbf{S}}$.

The second one is an random orthonormal basis $\bar{\mathbf{R}}$, which was generated using the polynomials of the orthonormal basis $\mathbf{R}$ (see Sec. 5.1.3) using Eq. (6.3). See Fig. 6.3 for examples of orthonormal basis images of $\bar{\mathbf{R}}$.

6.3 Evaluation

The evaluation of the edge detection accuracy in natural images is not an easy task. There is a lot of edges with different significance in natural images. The assessment of significance of edges is a very subjective task. However, there exist datasets of natural images with human annotations of image edges/segments. These are used for the evaluation of the edge detection accuracy and for the comparison of the results of different algorithms.

For evaluation of the edge detection accuracy, the precision–recall curve and F-measure score [69] are used.

The parametric curve reflects the relation between the precision and the recall for changing threshold value of the edge detector.

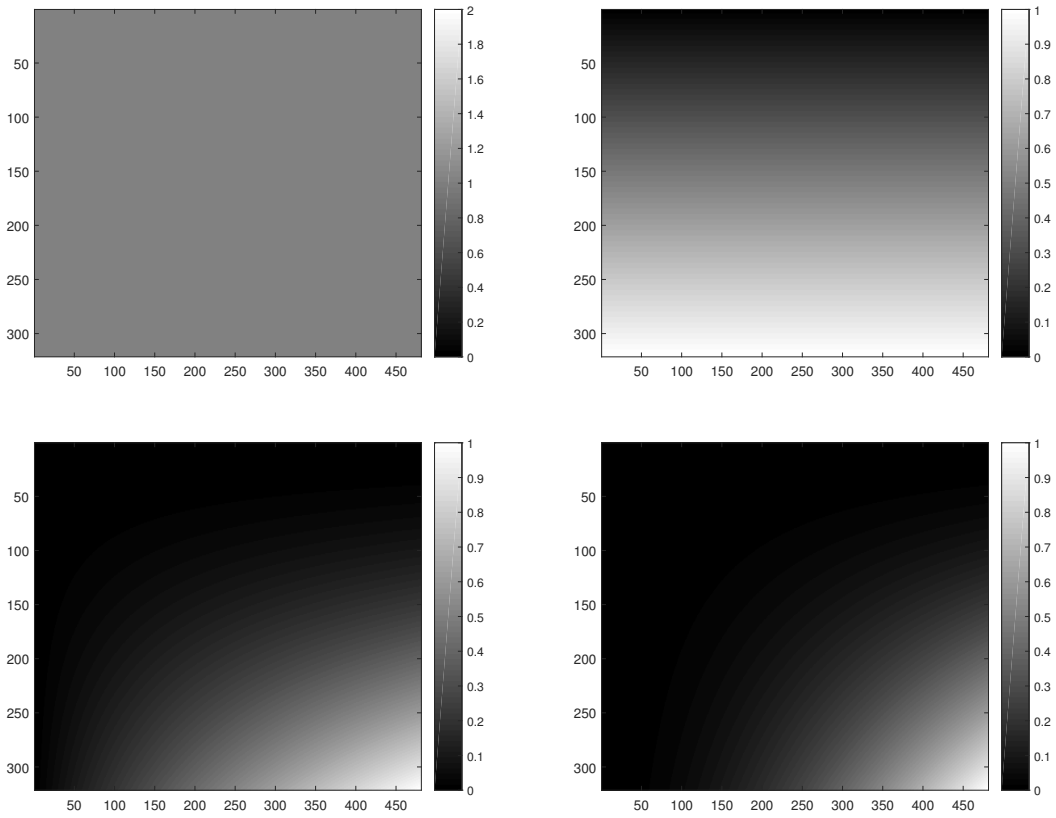


Fig. 6.2: Examples of polynomial images $\bar{\mathbf{S}}_{00}$, $\bar{\mathbf{S}}_{01}$, $\bar{\mathbf{S}}_{12}$, $\bar{\mathbf{S}}_{22}$ of size 321×481, generated from the 1D modified standard basis $\mathbf{S}$ of degree $K = 2$.

Precision (P) is defined as a ratio of true positive (TP) detections to the sum of false positive (FP) and true positive (TP) detections:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{FP} + \mathrm{TP}}. \tag{6.13}$$

Recall (R) is defined as a ratio of true positive (TP) detections to the sum of false negative (FN) and true positive (TP) detections:

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{FN} + \mathrm{TP}}. \tag{6.14}$$

F-measure score (F) is defined as a harmonic mean of precision and recall:

$$\mathrm{F} = 2\frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}}. \tag{6.15}$$

The optimal threshold value of the detector gives the maximal F-measure score along the curve.

Recall to the positions of the edges in natural images are subjective, the output of the edge detector has to be compared with several manually annotated edges

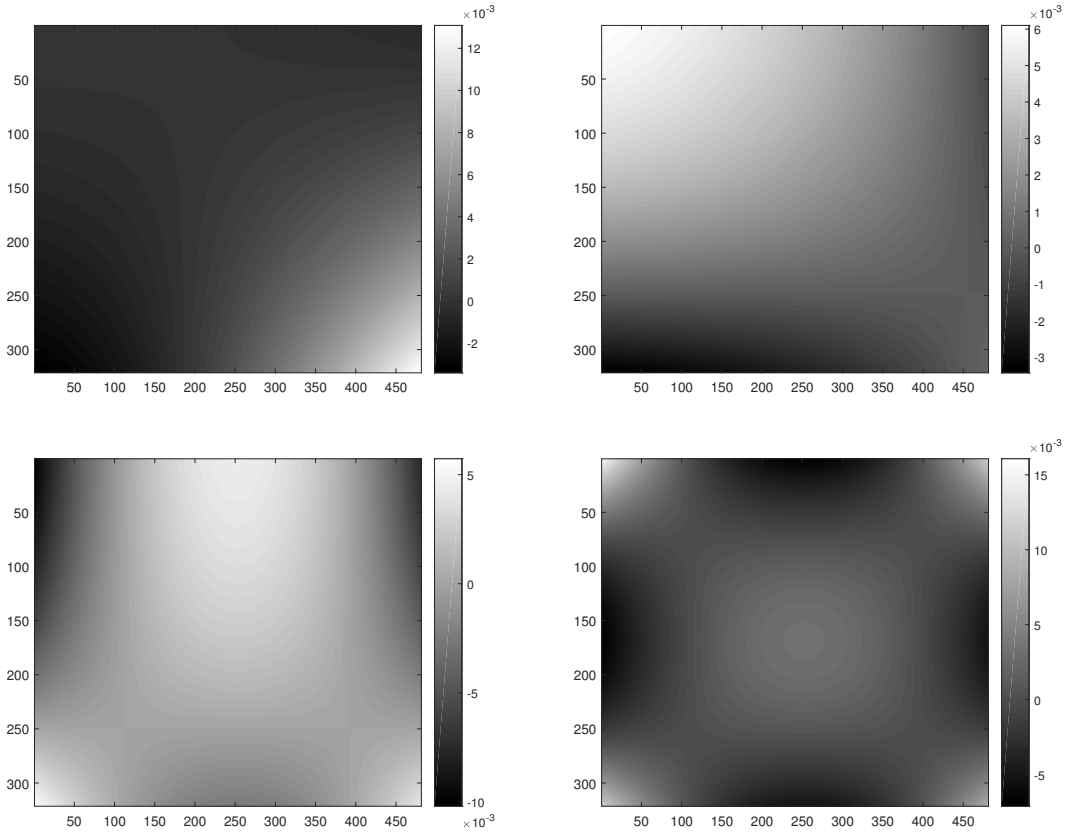


Fig. 6.3: Examples of polynomial images $\bar{\mathbf{R}}_{00}$, $\bar{\mathbf{R}}_{11}$, $\bar{\mathbf{R}}_{21}$, $\bar{\mathbf{R}}_{22}$ of size 321×481, generated from the 1D random orthonormal basis $\mathbf{R}$ of degree $K = 2$.

separately. The edge pixel is counted as false positive (FP) only if there is no match with any human manual annotations of the edges. Since the manual annotations contain edge localisation errors, the correspondence procedure has to tolerate small localisation errors at the cost of permitting multiple detection. If no localisation errors were permitted, it would lead to overpenalisation of algorithms that generate usable, though slightly mislocalised edges. The exact procedure of determining true positive (TP), false positive (FP) and false negative (FN) detection in relation to manual annotations for each pixel is presented in [69].

A simplified description of the correspondence procedure is:

- Create ground truth binary boundary maps from the manual annotations of edges/segments.
- Set thresholds for the output of the edge detector.
- For each threshold:
  - create binary boundary map of the thresholded edge detector's output,
  - apply thinning to the boundary map,
  - compare the new binary boundary map with the ground truth maps.

- Compute the R, P and F-measure.
- Create the precision–recall curve.

A similar metric to precision–recall curve is the receiver operating characteristic (ROC) curve. ROC graph is created with *fallout* and recall. Recall is the same as above, and fallout is a probability that a true negative is labelled as a false positive. ROC curves are not suitable for quantifying boundary detection, since the fallout depends on the size of the pixels and therefore, it is not a meaningful quantity for a boundary detector [69].

6.4 Recovery problem – verticals and horizontals

Since it is assumed that the clean image consists of several non-overlapping polynomial patches, the piecewise-constant parametrisation coefficients $\{\mathbf{x}_{k\ell}\}_{k,\ell}$, which are sparse under the difference operator ∇, can be found. The image consists of edges with different orientations. To successfully detect the edges in the image, it is necessary to seek the edges in at least two perpendicular directions (usually the vertical and horizontal direction). Therefore, two forms of the difference operator are used – the horizontal and the vertical one. As mentioned before, the analysed image is supposed to be corrupted by an i.i.d. Gaussian noise.

These assumptions lead to the following formulation of the constrained optimisation problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \| \operatorname*{reshape}_{\mathrm{v}}(L_{\mathrm{v}}\mathbf{x})\|_{21} + \lambda \| \operatorname*{reshape}_{\mathrm{h}}(L_{\mathrm{h}}\mathcal{M}(\mathbf{x}))\|_{21} \right\} \quad \text{s.t. } \|\mathbf{y} - \mathbf{P}\mathbf{x}\|_2 \leq \delta, \quad (6.16)$$

where the vector $\hat{\mathbf{x}}$ represents the obtained optimal parametrisation coefficients, $\mathbf{y}$ is the vectorised image $\mathbf{Y}$, $\mathbf{x}$ are the vectorised parametrisation coefficients $\mathbf{X}$, and $\mathbf{P}$ is a matrix of polynomial bases images. The parameter δ reflects the noise level and the model imperfections. The operator L_{v} represents the stacked vertical differences such that

$$L_{\mathrm{v}} = \begin{bmatrix} \nabla & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \nabla \end{bmatrix}, \quad L_{\mathrm{v}}\mathbf{x} = \begin{bmatrix} \nabla\mathbf{x}_{00} \\ \hline \vdots \\ \hline \nabla\mathbf{x}_{KK} \end{bmatrix}, \quad (6.17)$$

with the difference operator $\nabla\colon \mathbb{R}^{MN} \to \mathbb{R}^{(M-1)N}$. For the linear operator L_{v}, it holds $L_{\mathrm{v}}\colon \mathbb{R}^{(K+1)^2MN} \to \mathbb{R}^{(K+1)^2(M-1)N}$. The operator L_{h} represents the stacked horizontal differences and it is defined similar as the operator L_{v} with the only

difference in ∇, where now $\nabla\colon \mathbb{R}^{MN} \to \mathbb{R}^{M(N-1)}$. Therefore, it is necessary to tweak the entries of $\mathbf{x}$ such the horizontal differences can be computed. For this rearrangement, the operator $\mathcal{M}$ is used. The operators L_h and $\mathcal{M}$ are defined as follows:

$$L_\mathrm{h} = \begin{bmatrix} \nabla & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \nabla \end{bmatrix}, \quad L_\mathrm{h}\mathcal{M}(\mathbf{x}) = \begin{bmatrix} \nabla \operatorname{vec}(\mathbf{X}_{00}^\top) \\ \hline \vdots \\ \hline \nabla \operatorname{vec}(\mathbf{X}_{KK}^\top) \end{bmatrix}, \tag{6.18}$$

where the matrix $\mathbf{X}_{kl}$ is obtained by matrisation of vector $\mathbf{x}_{kl}$.

For the linear operator L_h, it holds $L_\mathrm{h}\colon \mathbb{R}^{(K+1)^2MN} \to \mathbb{R}^{(K+1)^2M(N-1)}$. λ is a non-negative weighting scalar parameter, which can be set to emphasise the vertical over the horizontal edges and vice versa. If $\lambda = 1$, the importance of vertical and horizontal edges is equal. If the size of the image grows proportionally in both directions, the parameter λ should not change. The operator $\operatorname{reshape}_\mathrm{v}()\colon \mathbb{R}^{(K+1)^2(M-1)} \to \mathbb{R}^{(M-1)\times(K+1)^2}$ takes the stacked vector $L_\mathrm{v}\mathbf{x}$ to the form of a matrix with disjoint columns:

$$\operatorname{reshape}_\mathrm{v}(L_\mathrm{v}\mathbf{x}) = \left[\nabla\mathbf{x}_{00} \mid \cdots \mid \nabla\mathbf{x}_{KK}\right]. \tag{6.19}$$

The operator $\operatorname{reshape}_\mathrm{h}()\colon \mathbb{R}^{(K+1)^2(N-1)} \to \mathbb{R}^{(N-1)\times(K+1)^2}$ takes the stacked vector $L_\mathrm{h}\mathcal{M}(\mathbf{x})$ to the form of a matrix with disjoint columns:

$$\operatorname{reshape}_\mathrm{h}(L_\mathrm{h}\mathcal{M}(\mathbf{x})) = \left[\nabla \operatorname{vec}(\mathbf{X}_{00}^\top) \mid \cdots \mid \nabla \operatorname{vec}(\mathbf{X}_{KK}^\top)\right]. \tag{6.20}$$

6.4.1 Recovery problem

The unconstrained version of the problem (6.16) can be formulated as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{ \|\operatorname*{reshape}_\mathrm{v}(L_\mathrm{v}\mathbf{x})\|_{21} + \lambda \|\operatorname*{reshape}_\mathrm{h}(L_\mathrm{h}\mathcal{M}(\mathbf{x}))\|_{21} \right\} + \iota_{\{\mathbf{z}\colon \|\mathbf{y}-\mathbf{z}\|_2 \le \delta\}}(\mathbf{P}\mathbf{x}), \tag{6.21}$$

where ι_C denotes the indicator function of a convex set C.

The first term of the problem (6.21) is the "penalisation" term. Parametrisation vectors $\mathbf{x}_{kl}$ are supposed to be piecewise-constant and joint-sparse under the difference operators ∇. Therefore, the ℓ_{21}-norm (see Eq. (1.7)) is used.

The second term in the problem (6.21) is the "data fidelity" term. The Euclidean ℓ_2-norm reflects the fact that gaussianity of the noise and the image imperfections are assumed and they should be lower than the noise level δ. The parameter δ should be tuned carefully.

Numerical solution of the recovery problem (6.21) using the Condat algorithm was presented in our paper [9].

6.4.2 Algorithm used

The recovery problem (6.21) can be solved via Condat algorithm (see Sec. 1.3.3), which is able to solve general problems in the form of (1.37). In our case, the problem reduces into the form of minimising the sum of three convex functions:

$$\text{minimise } h_1(L_1\mathbf{x}) + h_2(L_2\mathbf{x}) + h_3(L_3\mathbf{x}) \tag{6.22}$$

where functions h_m are convex and L_m are linear operators. With respect to the defined recovery problem (6.21), it is assigned

$$h_1(L_1\mathbf{x}) = \| \underset{\text{v}}{\text{reshape}}(L_{\text{v}}\mathbf{x})\|_{21}, \tag{6.23a}$$

$$h_2(L_2\mathbf{x}) = \lambda\| \underset{\text{h}}{\text{reshape}}(L_{\text{h}}\mathcal{M}(\mathbf{x}))\|_{21}, \tag{6.23b}$$

$$h_3(L_3\mathbf{x}) = \iota_{\{\mathbf{z}\,:\,\|\mathbf{y}-\mathbf{z}\|_2\leq\delta\}}(\mathbf{P}\mathbf{x}). \tag{6.23c}$$

From the above-mentioned equations, it can be derived that $L_1 = L_{\text{v}}, L_2 = L_{\text{h}}$, and $L_3 = \mathbf{P}$.

The Condat algorithm

The general form of the Condat algorithm is presented in Algorithm 5. Since the functions f and g are zero, $\nabla f = 0$ and $\text{prox}_g = Id$. The proximal operator of function h_1 and h_2 is the group thresholding defined in (1.24) and (1.25), where $\tau = 1$. Finally, the proximal operator of function h_3 is the projection onto $B_2(\mathbf{y}, \delta)$ defined in (1.26), which finds the closest point in the ℓ_2-ball $\{\mathbf{z} : \|\mathbf{y} - \mathbf{z}\|_2 \leq \delta\}$ with respect to the input point.

The Condat algorithm exploits linear operators L_m and their transposes $L_m^\top$, for $m = 1, \ldots, 3$. The operator $L_1 = L_{\text{v}}$ and its transpose $L_1^\top \colon \mathbb{R}^{(K+1)^2(M-1)N} \to \mathbb{R}^{(K+1)^2MN}$ is defined as follows:

$$L_1^\top(\mathbf{u}) = L_{\text{v}}^\top(\mathbf{u}) = L_{\text{v}}^\top\left(\begin{bmatrix}\mathbf{u}_{00}\\ \text{—}\\ \vdots\\ \text{—}\\ \mathbf{u}_{KK}\end{bmatrix}\right) = \begin{bmatrix}\nabla^\top\mathbf{u}_{00}\\ \text{—}\\ \vdots\\ \text{—}\\ \nabla^\top\mathbf{u}_{KK}\end{bmatrix}, \tag{6.24}$$

with $\nabla^\top$ of size $MN \times (M-1)N$ and vector $\mathbf{u} \in \mathbb{R}^{(M-1)N(K+1)}$. The transpose of the $L_2 = L_{\text{h}}$ is defined similarly as $L_1^\top$, with the difference in sizes – $\nabla^\top$ has size $MN \times M(N-1)$ and vector $\mathbf{u} \in \mathbb{R}^{M(N-1)(K+1)}$. Transpose of the $L_3 = \mathbf{P}$ is defined as $L_3^\top = \mathbf{P}^\top$. For the needs of the Condat algorithm, the transpose of the $\text{reshape}_{\text{v}}()$ and $\text{reshape}_{\text{h}}()$ operators needs to be defined. The operator $\text{reshape}_{\text{v}}^\top() \colon \mathbb{R}^{(M-1)N\times(K+1)^2} \to \mathbb{R}^{(M-1)N(K+1)^2}$ takes the matrix with disjoint

columns back to the form of a stacked vector:

$$\operatorname{reshape}_{\mathrm{v}}^{\top}\left(\left[\mathbf{u}_0 \mid \cdots \mid \mathbf{u}_K\right]\right) = \begin{bmatrix} \mathbf{u}_0 \\ \hline \vdots \\ \hline \mathbf{u}_K \end{bmatrix}. \tag{6.25}$$

The operator $\operatorname{reshape}_{\mathrm{h}}^{\top}() \colon \mathbb{R}^{M(N-1)\times(K+1)^2} \to \mathbb{R}^{M(N-1)(K+1)^2}$ is defined similar to the $\operatorname{reshape}_{\mathrm{v}}^{\top}()$ with the only difference in size. Based on the properties of $\mathcal{M}$, it holds that $\mathcal{M}^{\top} = \mathcal{M}$.

To ensure convergence of the Condat algorithm the following sufficient condition must be satisfied:

$$\xi\sigma\|L_1^{\top}L_1 + L_2^{\top}L_2 + L_3^{\top}L_3\| \leq 1, \tag{6.26}$$

where $\|\cdot\|$ denotes the operator norm and ξ and σ are internal parameters of the Condat algorithm. The inequality $\|L_1^{\top}L_1 + L_2^{\top}L_2 + L_3^{\top}L_3\| \leq \|L_1\|^2 + \|L_2\|^2 + \|L_3\|^2$ is used, therefore the upper bounds of $\|L_1\|, \|L_2\|$ and $\|L_3\|$ are needed. The operator norm $\|L_1\|$ is derived as follows:

$$\begin{aligned} \|L_1\|^2 = \|L_{\mathrm{v}}\|^2 &= \max_{\|\mathbf{x}\|_2=1} \|L_{\mathrm{v}}\mathbf{x}\|_2^2 \\ &= \max_{\|\mathbf{x}\|_2=1} \left(\sum_{k=0}^{K}\sum_{\ell=0}^{K} \|\nabla \operatorname{vec}(\mathbf{x}_{k\ell})\|_2^2 \right) \\ &\leq \sum_{k=0}^{K}\sum_{\ell=0}^{K} \left(\max_{\|\mathbf{x}\|_2=1} \|\nabla \operatorname{vec}(\mathbf{x}_{k\ell})\|_2^2 \right) \\ &\leq \sum_{k=0}^{K}\sum_{\ell=0}^{K} \|\nabla\|^2 = \|\nabla\|^2(K+1)^2 \;\leq\; 4(K+1)^2, \end{aligned} \tag{6.27}$$

and the same holds for $\|L_2\|$. Finally, the upper bound of $\|L_3\| = \|\mathbf{P}\|$ has to be found, $\|\mathbf{P}\|^2 = \|\mathbf{P}\mathbf{P}^{\top}\|$ and thus it is sufficient to find the maximum eigenvalue of $\mathbf{P}\mathbf{P}^{\top}$. Since $\mathbf{P}$ has a multi-diagonal structure, the product $\mathbf{P}\mathbf{P}^{\top}$ is a diagonal matrix and thus it suffices to find the maximum element on the diagonal. Therefore, the convergence of the Condat algorithm is guaranteed in the case

$$\xi\sigma\left(\max(\operatorname{diag}(\mathbf{P}\mathbf{P}^{\top})) + 8(K+1)^2\right) \leq 1. \tag{6.28}$$

The Condat algorithm solving the recovery problem (6.21) is presented in Algorithm 14.

6.5 Experiment – image edge detection

In this section, the results obtained from the image edge detection process defined in the recovery problem (6.21) solved by the Condat algorithm (see Alg. 14) are

Algorithm 14: The Condat Algorithm solving (6.21)

Input: Functions h_1, h_2, h_3, linear operators $L_1, L_1^\top, L_2, L_2^\top, L_3, L_3^\top, \mathcal{M}, \mathcal{M}^\top$, parameter λ for proximal operator of h_2, parameter δ for proximal operator of h_3

Output: $\hat{\mathbf{x}} = \mathbf{x}^{(i+1)}$

Set the parameters $\xi, \sigma > 0$ and $\rho \in [0, 2]$

Set the initial primal variable $\mathbf{x}^{(0)}$and dual variables $\mathbf{u}_1^{(0)}, \mathbf{u}_2^{(0)}, \mathbf{u}_3^{(0)}$

for $i = 0, 1, \ldots, \mathrm{MAX_{IT}}$ **do**

$\bar{\mathbf{x}}^{(i+1)} = \mathbf{x}^{(i)} - \xi \left(L_{\mathrm{v}}^\top \operatorname{reshape}_{\mathrm{v}}^\top(\mathbf{u}_1^{(i)}) + \mathcal{M}^\top L_{\mathrm{h}}^\top \operatorname{reshape}_{\mathrm{h}}^\top(\mathbf{u}_2^{(i)}) + \mathbf{P}^\top \mathbf{u}_3^{(i)} \right)$

$\mathbf{x}^{(i+1)} = \rho\, \bar{\mathbf{x}}^{(i+1)} + (1-\rho)\mathbf{x}^{(i)}$

$\mathbf{p}_1 = \mathbf{u}_1^{(i)} + \sigma \operatorname{reshape}_{\mathrm{v}}(L_{\mathrm{v}}(2\bar{\mathbf{x}}^{(i+1)} - \mathbf{x}^{(i)}))$

$\bar{\mathbf{u}}_1^{(i+1)} = \mathbf{p}_1 - \operatorname{soft}_1^{\mathrm{group}}(\mathbf{p}_1)$

$\mathbf{u}_1^{(i+1)} = \rho\, \bar{\mathbf{u}}_1^{(i+1)} + (1-\rho)\mathbf{u}_1^{(i)}$

$\mathbf{p}_2 = \mathbf{u}_2^{(i)} + \sigma \operatorname{reshape}_{\mathrm{h}}(L_{\mathrm{h}}\mathcal{M}(2\bar{\mathbf{x}}^{(i+1)} - \mathbf{x}^{(i)}))$

$\bar{\mathbf{u}}_2^{(i+1)} = \mathbf{p}_2 - \operatorname{soft}_\lambda^{\mathrm{group}}(\mathbf{p}_2)$

$\mathbf{u}_2^{(i+1)} = \rho\, \bar{\mathbf{u}}_2^{(i+1)} + (1-\rho)\mathbf{u}_2^{(i)}$

$\mathbf{p}_3 = \mathbf{u}_3^{(i)} + \sigma\, \mathbf{P}(2\bar{\mathbf{x}}^{(i+1)} - \mathbf{x}^{(i)})$

$\bar{\mathbf{u}}_3^{(i+1)} = \mathbf{p}_3 - \sigma \operatorname{proj}_{B_2(\mathbf{y},\delta)}(\mathbf{p}_3/\sigma)$

$\mathbf{u}_3^{(i+1)} = \rho\, \bar{\mathbf{u}}_3^{(i+1)} + (1-\rho)\mathbf{u}_3^{(i)}$

return $\mathbf{x}^{(i+1)}$

evaluated. The aim of the experiments is to set the optimal parameter δ for image edge detection for both grayscale and RGB images.

For the purpose of the experiments, 200 train + 100 validation RGB images $\mathbf{Y}_{\mathrm{RGB}}$, and 200 train + 100 validation grayscale images $\mathbf{Y}_{\mathrm{gray}}$ are used. The process of generating grayscale images is described in Sec. 6.2.2.

In Sec. 6.2.3, two types of bases ($\bar{\mathbf{S}}$, $\bar{\mathbf{R}}$) are introduced. Based on the promising results obtained using the random orthonormal bases $\mathbf{R}$ in Sec. 5.7, the aim was to use random orthonormal basis $\bar{\mathbf{R}}$ for the experiments in 2D. Unfortunately, the use of random orthonormal basis $\bar{\mathbf{R}}$ requires almost eight times as many iterations for convergence than the use of modified standard basis $\bar{\mathbf{S}}$, and the time for detecting edges in the image is thus beyond an acceptable time. Therefore for the purpose of these experiments, only modified standard basis $\bar{\mathbf{S}}$ consisting of the $(K+1)^2$ basis images, where $K = 2$, is used for the experiments. It is necessary to create a new basis $\bar{\mathbf{S}}$ (see Sec. 6.2.3) for each processed image, since the image dimensions (M, N) change with each image.

The "Optimisation step" requires setting of several parameters. The list of parameters for the experiments using the Condat algorithm is given in Table 6.1.

Tab. 6.1: Parameter settings for the Condat algorithm. Table contains the specific setting of parameters needed in the image edge detection process using the Condat algorithm.

Step	**Setting**	
	Parameter	**Value (for $K = 2$)**
Condat algorithm	σ	$1/\sqrt{(\|L_1\|^2 + \|L_2\|^2) + \|L_3\|^2)}$
	ξ	$1/\sqrt{(\|L_1\|^2 + \|L_2\|^2) + \|L_3\|^2)}$
	ρ	1.99
	$\tau_0 - \tau_8$	1
	δ	$\in \{1000, 2000, \ldots, 11000\}$
	λ	1
	$\mathrm{MAX}_{\mathrm{IT}}$	1000

The Condat algorithm is stopped if either the convergence criterion or the maximum number of iterations ($\mathrm{MAX}_{\mathrm{IT}}$) is reached. The parameter δ changes within the experiments.

The obtained results are evaluated via precision–recall curves and their maximum F-measure score (see Sec. 6.3). The output of the method for each value of the parameter δ is an image of differences $\mathbf{D}$, based on which the precision–recall curve is generated. Since the aim is to find the optimal value of the parameter δ, all obtained precision–recall curves (for the defined set of δ values) are plotted into a common graph. For each curve, the highest F-measure score is found. Based on the highest F-measure score, the optimal value of δ is determined.

6.5.1 Training phase

For the purpose of the first experiments, the train dataset of BSDS500 consisting of 200 natural images is used. Parameter δ is changes within the experiments, since the aim is to find its optimal value for edge detection in the images.

Experiments with grayscale images

The first set of experiments is done for the grayscale images $\mathbf{Y}_{\mathrm{gray}}$. The parameter δ takes values from the set $\delta \in \{1000, 2000, \ldots, 10000\}$.

Results for grayscale images The precision–recall curves for 10 parameters $\delta \in \{1000, 2000, \ldots, 10000\}$ are plotted in Fig. 6.4. The green dotted circle (in all figures shown in the training phase) represents the typical human F-measure score.

It is only an approximation from the validation and test subsets, as the exact value of the human F-measure scores for the train subset is not known. The highest F-measure score obtained across all δ settings is F = 0.539 for $\delta = 6000$. Refinement of the parameter δ will be done during the validation phase in Sec. 6.5.2. Examples of edge detection results are presented in Fig. 6.12.

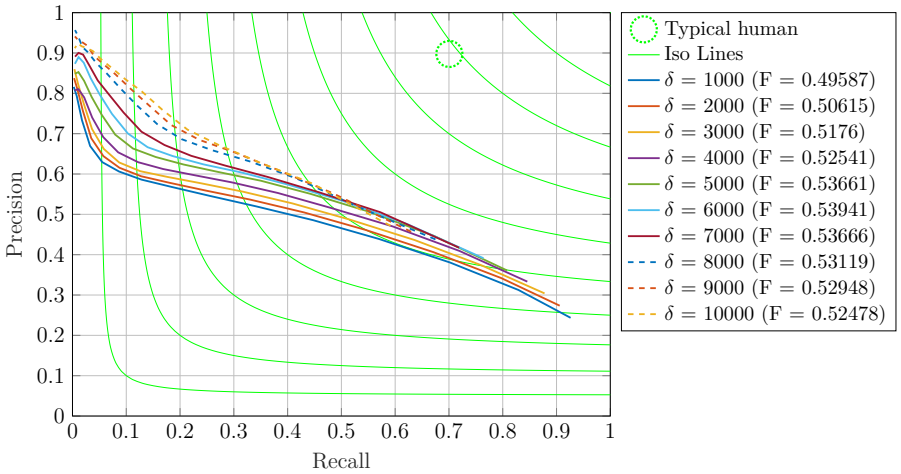


Fig. 6.4: Precision–recall curves with their maximum F-measure scores for δ settings. Presented results are for train subset of BSDS500 consisting of 200 grayscale natural images.

Experiments with RGB images

The second set of experiments is performed on the RGB images. The parameter δ takes values from the set $\delta \in \{1000, 2000, \ldots, 11000\}$. Since the RGB image consists of tree colour channels – R,G,B, the image edge detection is computed for each channel separately. Detected images of differences of the individual channels need to be merged into a single output, which can be done by several approaches. In these experiments, four merging approaches are proposed:

- approach V1 – the maximum value of the differences detected across all the channels, computed as

$$\mathbf{D}_{\mathrm{V1}}[m,n] = \max(\mathbf{D}_{\mathrm{R}}[m,n], \mathbf{D}_{\mathrm{G}}[m,n], \mathbf{D}_{\mathrm{B}}[m,n]), \tag{6.29}$$

- approach V2 – mean value of the differences detected across all channels, computed as

$$\mathbf{D}_{\mathrm{V2}}[m,n] = \mathrm{avg}(\mathbf{D}_{\mathrm{R}}[m,n], \mathbf{D}_{\mathrm{G}}[m,n], \mathbf{D}_{\mathrm{B}}[m,n]), \tag{6.30}$$

- approach V3 – mean value of the two highest differences detected across all channels, computed as

$$\mathbf{D}_{\mathrm{V3}}[m,n] = \mathrm{avg}(\mathrm{sum}(\mathbf{D}_{\mathrm{R}}[m,n], \mathbf{D}_{\mathrm{G}}[m,n], \mathbf{D}_{\mathrm{B}}[m,n]) - \min(\mathbf{D}_{\mathrm{R}}[m,n], \mathbf{D}_{\mathrm{G}}[m,n], \mathbf{D}_{\mathrm{B}}[m,n])), \quad (6.31)$$

- approach V4 – RGB to the Y component of the YCbCr model, computed as

$$\mathbf{D}_{\mathrm{V4}}[m,n] = 16 + \frac{65.738}{256}\mathbf{D}_{\mathrm{R}}[m,n] + \frac{129.057}{256}\mathbf{D}_{\mathrm{G}}[m,n] + \frac{25.064}{256}\mathbf{D}_{\mathrm{B}}[m,n]. \quad (6.32)$$

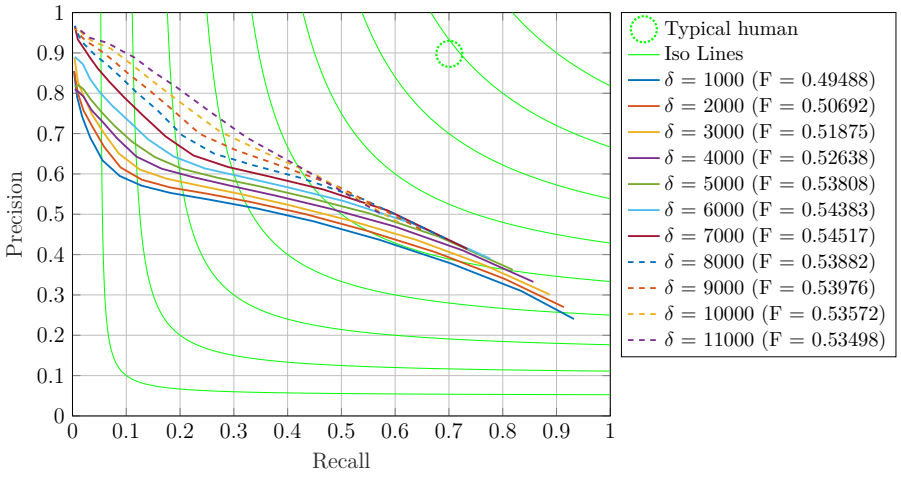
Experimenting with RGB images has two aims – determine the colour channel that contains the greatest amount of edge information, and determine the best merging approach with its optimal δ.

Results for RGB images – R,G,B channels The precision–recall curves for 10 parameters $\delta \in \{1000, 2000, \ldots, 11000\}$ are plotted in Fig. 6.5. The highest F-measure score obtained across all δ settings for R channel is F = 0.545 for $\delta = 7000$, for G channel it is F = 0.539 for $\delta = 6000$, and for B channel it is F = 0.536 for $\delta = 8000$. According to the results, there are only small differences between the colour channels. However, the results indicate that most edge information is contained in the R channel. The results also show that F-measure score obtained from each of the RGB channels is very similar to the best F-measure score obtained for grayscale images. Examples of the edge detection results for each colour channel are presented in Fig. 6.6.

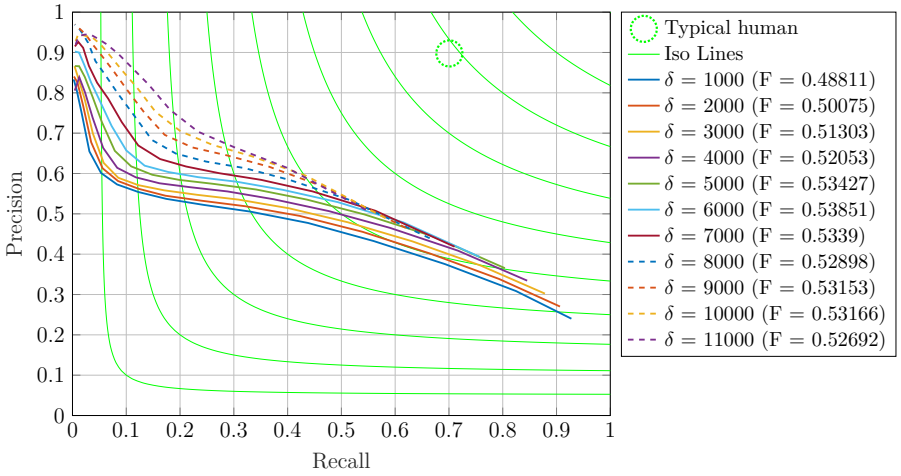
Results for RGB images – merging approaches The precision–recall curves for 10 parameters $\delta \in \{1000, 2000, \ldots, 11000\}$ for each merging approach are plotted in Fig. 6.7, and the maximum obtained F-measure scores are shown in Fig. 6.8. The highest F-measure obtained across all δ settings for merging approach V1 is F = 0.576 with $\delta = 7000$, for V2 it is F = 0.559 with $\delta = 10000$, for V3 it is F = 0.564 with $\delta = 7000$, and for V4 it is F = 0.550 with $\delta = 10000$. The obtained F-score results suggest that the preferred approach of merging the edge information from the RGB channels is the V1 approach, i.e. the maximum value of the detected differences. Examples of edge detection results for each merging approach are presented in Fig. 6.9.

6.5.2 Validation phase

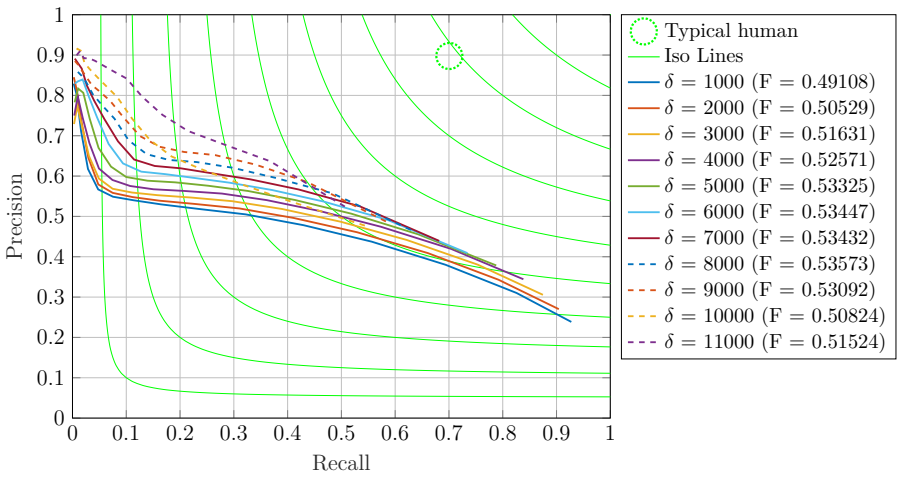
Since the steps sizes between the δ parameters were quite large, additional experiments are performed to refine the optimal δ value. Refinement of the parameter δ is the only aim of the experiments on validation dataset. For the purpose of these experiments, the validation set of BSDS500 consisting of 100 natural images is used.



(a) red channel



(b) green channel



(c) blue channel

Fig. 6.5: Precision–recall curves with their maximum F-measure scores for δ settings for each colour channel (red, green, blue). Presented results are for train subset of BSDS500 consisting of 200 RGB natural images.

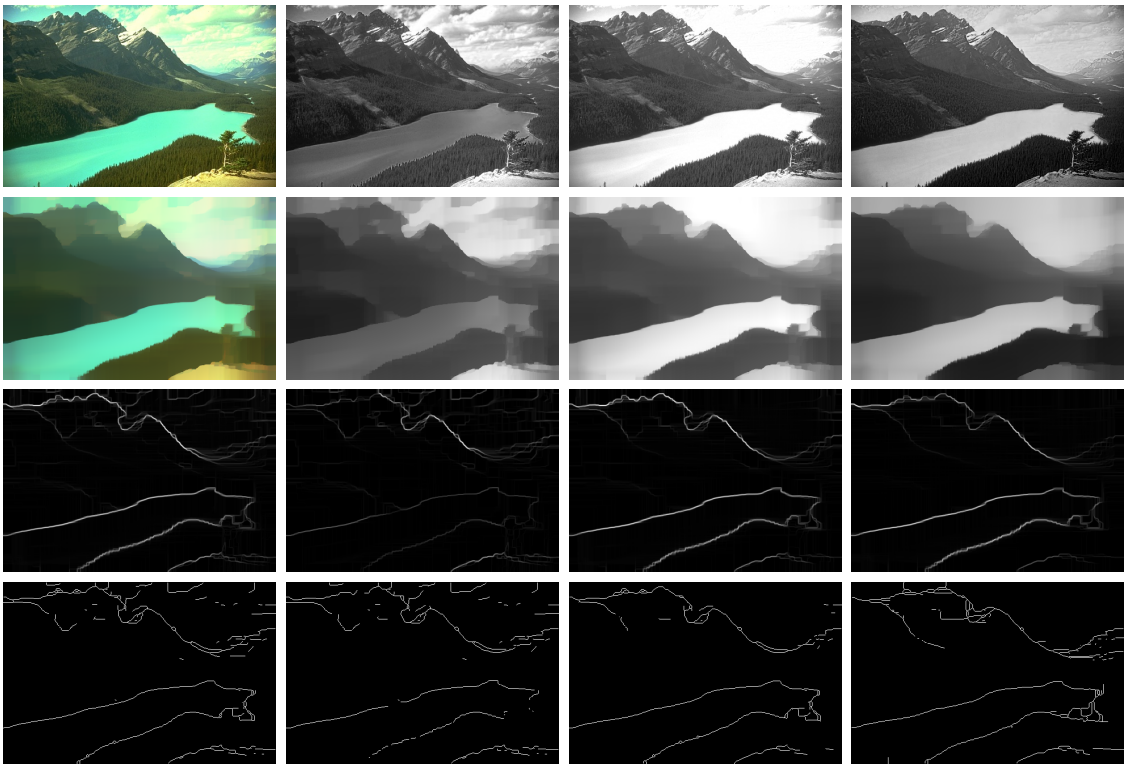


Fig. 6.6: Examples of edge detection results for individual colour channels. From the left: RGB channels, R channel, G channel and B channel. From the top: original image $\mathbf{Y}$, reconstructed image $\hat{\mathbf{Y}}$, image of differences $\mathbf{D}$, thresholded and thinned image of differences. Presented results are for RGB train image subset of BSDS500.

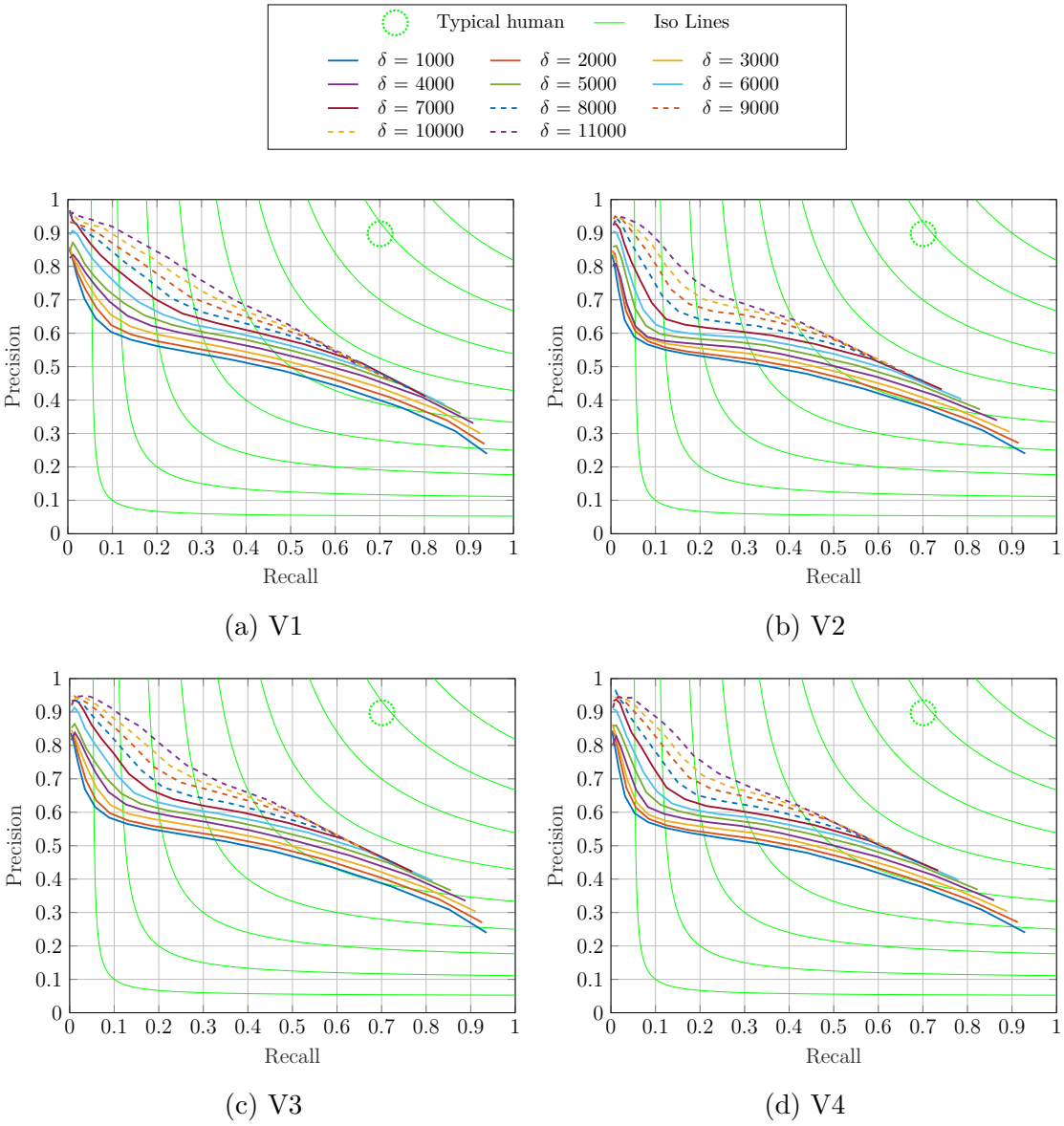
Experiments with grayscale images

The first set of experiments is done for the grayscale images with the aim to refine the optimal δ. The parameter δ takes values from the set $\delta \in \{5000, 5250, \ldots, 7000\}$.

Results for grayscale images The precision–recall curves for 9 parameters $\delta \in \{5000, 5250, \ldots, 7000\}$ are plotted in Fig. 6.10. The highest F-measure score obtained across all δ settings is $\mathrm{F} = 0.528$ for $\delta = 6250$. The optimal value of δ for edge detection in grayscale images is, therefore, set to $\delta = 6250$, and it will be used for experiments on the test dataset in Sec. 6.6.

Experiments with RGB images

The second set of experiments is performed on the RGB images. Since the aim is to refine the parameter δ, only the best merging approach V1 (the maximum value approach) from the training phase is used. The parameter δ takes values from the set $\delta \in \{6000, 6250, \ldots, 8000\}$.



(a) V1

(b) V2

(c) V3

(d) V4

Fig. 6.7: Precision–recall curves for δ settings for each merging approach V1–V4, which were obtained for train subset of BSDS500 consisting of 200 RGB natural images.

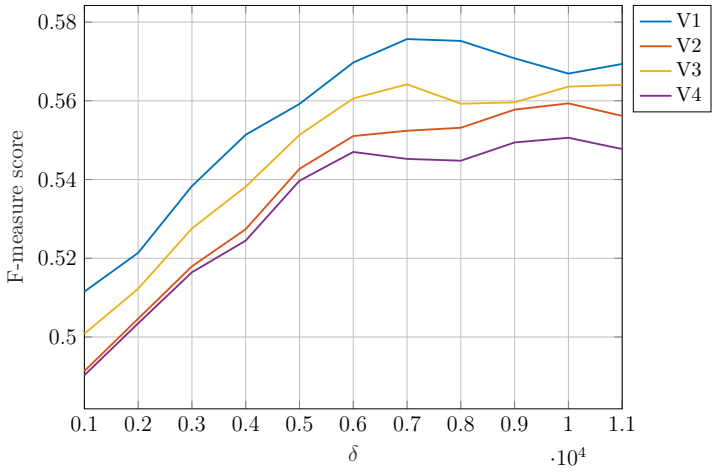


Fig. 6.8: Maximum F-measure scores in dependency of parameter δ for each merging approach V1–V4.

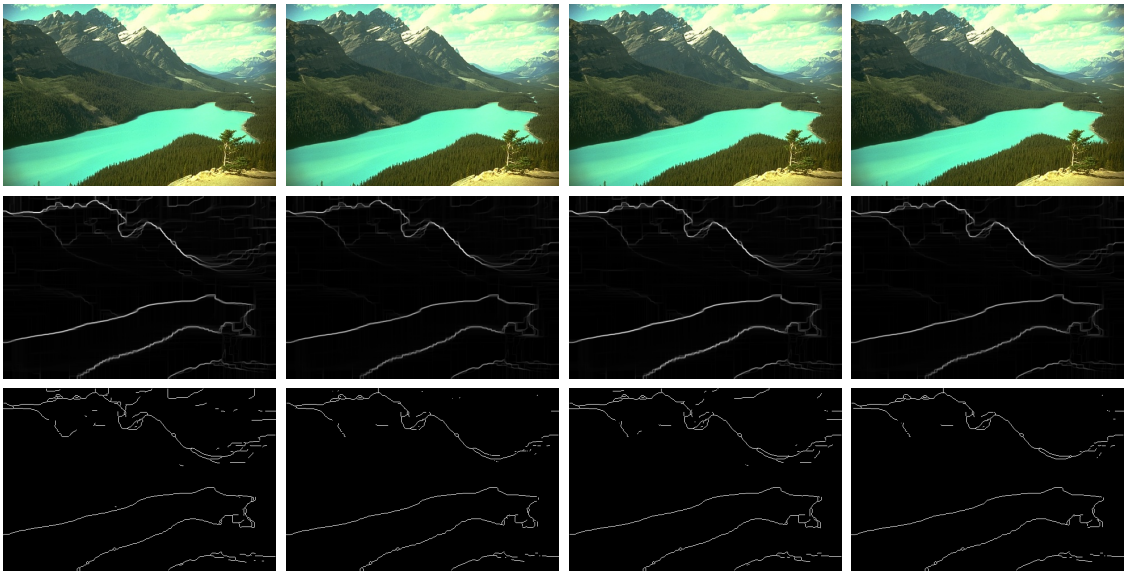


Fig. 6.9: Examples edge detection results for individual merging approaches. From the left: V1 approach, V2 approach, V3 approach and V4 approach. From the top: original colour image, differences of the reconstructed image, thresholded and thinned differences. Experiments were performed on RGB train image subset of BSDS500.

Results for RGB images Similarly to the case of grayscale images, the precision–recall curves for $\delta \in \{6000, 6250, \ldots, 8000\}$ are displayed in Fig. 6.11. According to these results, the highest F-measure score obtained across all δ settings is $F = 0.557$ for $\delta = 7500$. Therefore, the value 7500 for the parameter δ is assumed to be the optimum for RGB images and will be used for the experiments on the test dataset in Sec. 6.6.

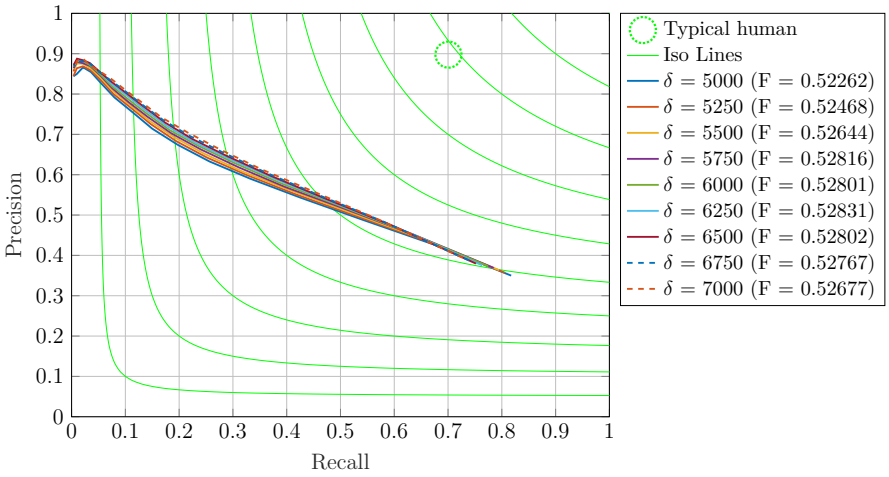


Fig. 6.10: Precision–recall curves with their maximum F-measure scores for δ settings. Presented results are for validation subset of BSDS500 consisting of 100 grayscale natural images.

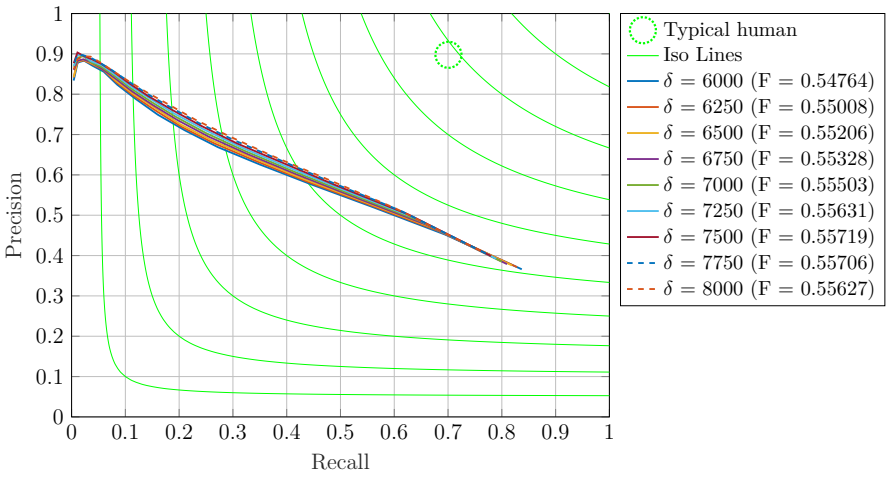


Fig. 6.11: Precision–recall curves with their maximum F-measure scores for δ settings. Presented experiments are for validation subset of BSDS500 consisting of 100 RGB natural images.

6.6 Comparison with other methods

Since the results of some presented edge detection methods (see Chapter 3) are computed on the test subset of BSDS300 dataset and others on BSDS500, the proposed image edge detection method with optimal δ is run on both mentioned datasets.

For testing on BSDS500 dataset, the optimal parameters δ obtained from the validation phase are used.

The test subset of BSDS300 consists of 100 natural images. This test subset is equal to the validation subset of BSDS500. Since the train subsets of both datasets

BSDS300 and BSDS500 are the same, it is not allowed to use the optimal value of δ obtained from the validation phase, and thus the optimal δ for testing phase on BSDS300 was determined based on the results of the training phase.

Examples of edge detection results for both RGB and grayscale images are presented in Fig. 6.12.

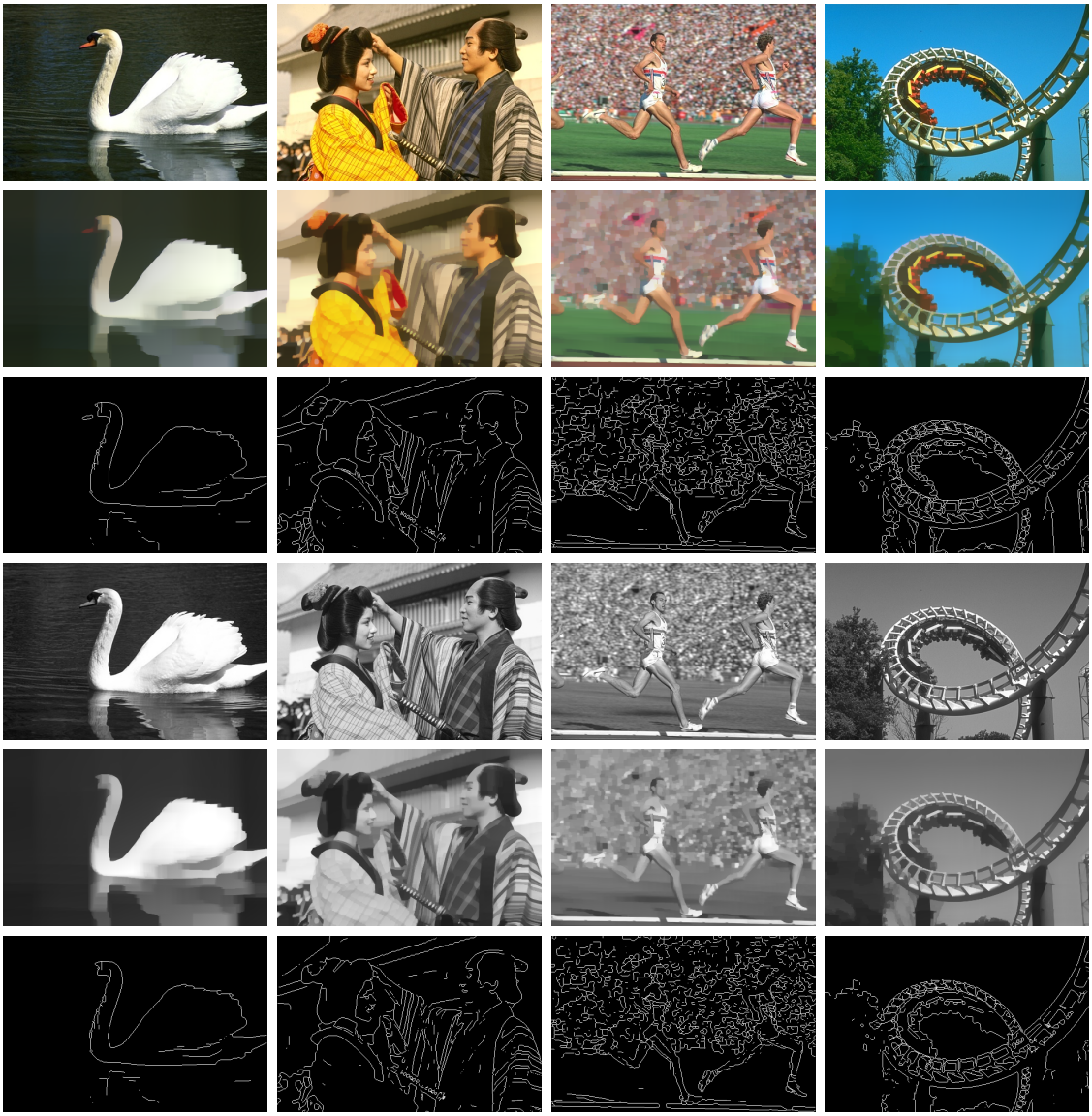


Fig. 6.12: Examples of edge detection results for RGB and grayscale images of BSDS500 test subset. From the top: original colour image, reconstructed RGB image, thresholded and thinned differences of the reconstructed RGB image, grayscale image, reconstructed grayscale image, thresholded and thinned differences of the reconstructed grayscale image.

6.6.1 Results on BSDS300

For grayscale images, the parameter δ is set to $\delta = 6000$. For RGB images, the parameter δ is set to $\delta = 7000$. The precision–recall curves for grayscale and RGB images are plotted in Fig. 6.13 on the right-hand side. The precision–recall curves of other approaches described in Sec. 3 and presented in [72] are displayed in Fig. 6.13 on the left-hand side.

The highest obtained F-measure score for grayscale images is F = 0.528, and for RGB images it is F = 0.555. The proposed image edge detection method for both grayscale and RGB images gives better results in terms of F-measure score than the classical gradient-based edge detection techniques (Prewitt, Sobel, and Roberts) and Laplacian of Gaussian. One of the most important results is that the proposed approach achieves better results than other "standard" methods without special post-processing. However, it turns out that the post-processing plays a critical role in successful image edge detection and the "standard" methods are not competitive with the more sophisticated methods with post-processing or deep learning-based methods.

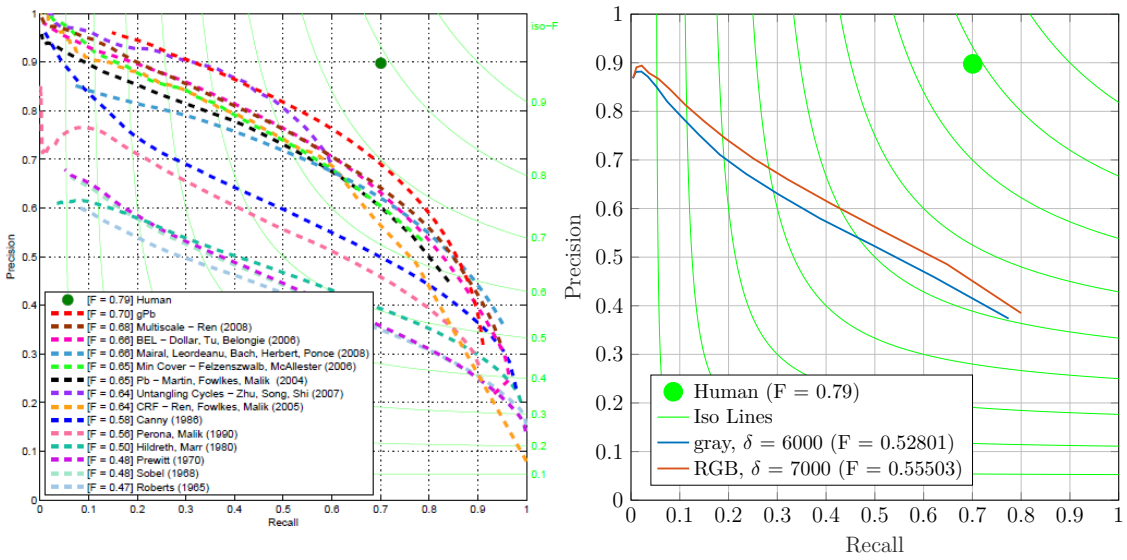


Fig. 6.13: Precision–recall curves with their maximum F-measure scores for δ settings. The precision–recall curves of other approaches described in Sec. 3 are displayed on the left-hand side. The precision–recall curves of the proposed approach for RGB and grayscale images are plotted on the right-hand side. Presented results are for test subset of BSDS300 consisting of 100 natural images.

6.6.2 Results on BSDS500

For the final evaluation, 200 test RGB images $\mathbf{Y}_{\text{RGB}}$, and 200 test grayscale images $\mathbf{Y}_{\text{gray}}$ are used. For grayscale images, the parameter δ is set to $\delta = 6250$. For RGB images, the parameter δ is set to $\delta = 7500$. The precision–recall curves for grayscale and RGB images are plotted in Fig. 6.14 on the right-hand side. The precision–recall curves of other approaches described in Sec. 3 and presented in [101] are displayed in Fig. 6.14 on the left-hand side. The precision–recall curves of the Canny detector and other approaches presented in [101] are displayed in Fig. 6.14 on the left side.

The highest obtained F-measure score for grayscale images is F = 0.560, and for RGB images it is F = 0.603, which is a little bit higher than the F-measure score of the Canny detector (F = 0.600).

Compared to the results on the BSDS300 dataset, the edge detection results on the BSDS500 dataset are available only for the Canny detector and more sophisticated methods. However, they are more relevant for the proposed method since the parameter δ could be fine-tuned on the validation subset. The proposed method achieved comparable and even slightly better results than the Canny detector, which is considered as the standared in image edge detection and contains some additional post-processing. Nonetheless, also the results on the BSDS500 points out the superior performance of the methods that are either based on deep learning or employ sophisticated post-processing techniques.

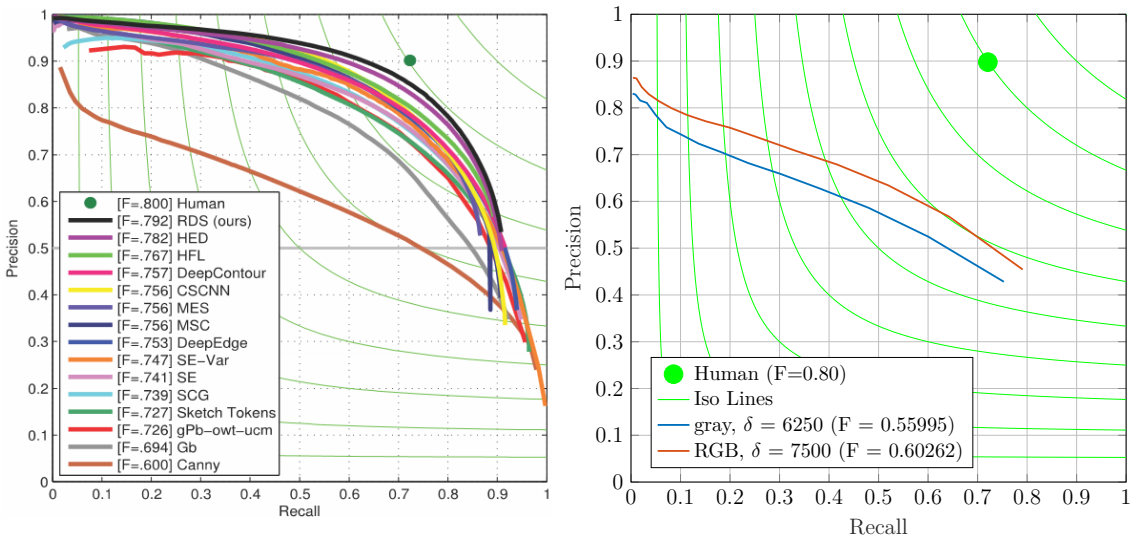


Fig. 6.14: Precision–recall curves with their maximum F-measure scores for δ settings. The precision–recall curves of other approaches described in Sec. 3 are displayed on the left-hand side. The precision–recall curves of proposed approach for RGB and grayscale images are plotted on the right-hand side. Presented results are for test subset of BSDS500 consisting of 200 natural images.

6.7 Edge detection in noisy images

Apart from the experiments on the clean images, experiments on noisy images to examine the performance of the proposed method under noise are designed.

For the purpose of these experiments, 200 test grayscale images from BSDS500 dataset are used. Grayscale images of BSDS500 are assumed to be clean. All 200 grayscale images were corrupted by Gaussian i.i.d. noise in line with the process described in Sec. 6.2.2, resulting in SNR $\in \{15, 20, 25, 30, 35\}$ dB.

Note that the parameter δ should be adjusted according the noise level for optimal performance. However, attempting to fine-tune the parameter δ for each image or noise level can be impractical and time consuming. Moreover, compared to this testing scenario, the noise level in real-world images is often unknown. Therefore, this experiment focuses purely on evaluating the general noise-robustness of the proposed method using the best value of parameter δ obtained from the validation phase on images without additional noise ($\delta = 6250$).

A comparison of edge detection results for images with different noise levels is shown in Fig. 6.15, and the resulting precision–recall curves for images with different noise level are displayed in Fig. 6.16.

6.8 Partial conclusions

The experiments performed on grayscale and RGB images were divided into three phases – training phase, validation phase, and testing phase. The BSDS500 dataset of natural images was used for these experiments.

In the training phase on grayscale images, the optimal parameter $\delta = 6000$ was chosen for the proposed image edge detection method, which resulted in the highest F-measure score for the dataset of train images. For RGB images, multiple criteria were evaluated in the training phase. The main reason is that the proposed image edge detection method was applied to each colour channel separately, and the resulting edge estimates need to be merged into one common output. To do so, four merging approaches have been proposed. For the train dataset, merging approach denoted V1 (the maximum value approach) gave the best F-measure score (for $\delta = 7000$). It was further evaluated that the R channel carries most of the edge information among all channels. The edges extracted from each colour channel carry approximately the same amount of edge information as the edges extracted from the grayscale images. Merging the outputs from the R, G, and B channels increases the final F-measure score by approximately 0.03 (i.e. 5.5 %).

The validation phase was used to refine the optimal δ value. In the training phase, the steps between the values of δ were quite large. Thus, in the validation

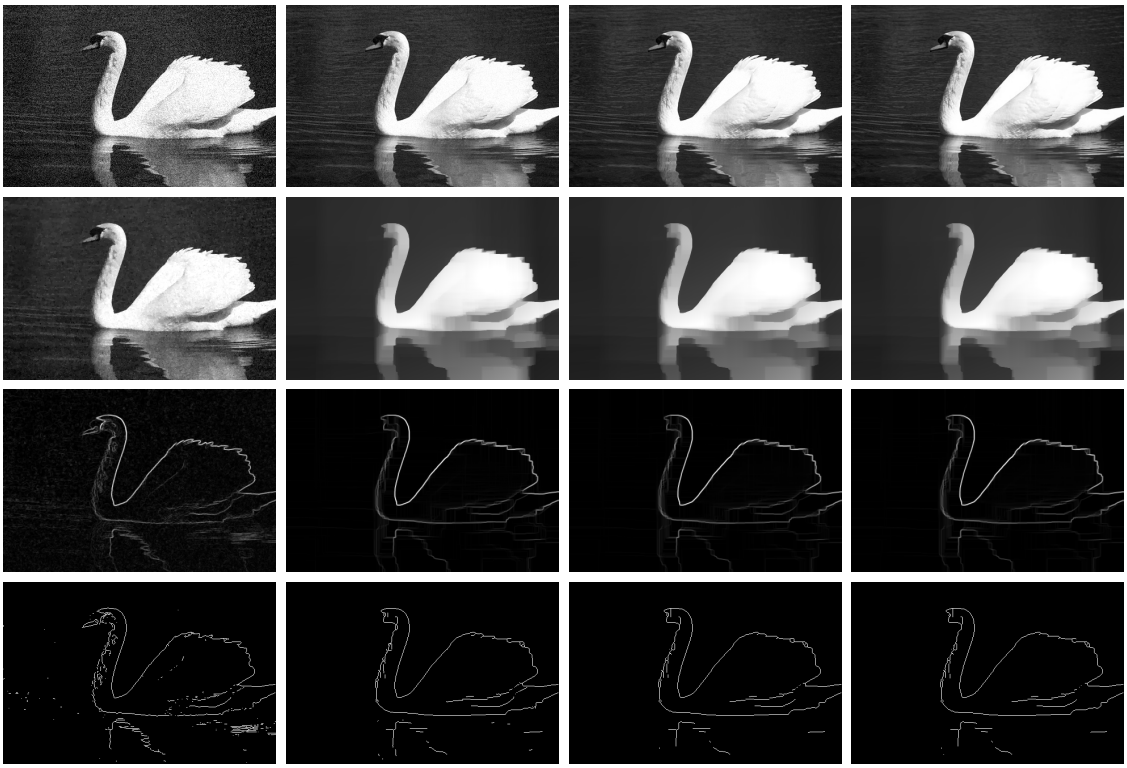


Fig. 6.15: Examples edge detection results for noisy grayscale images from BSDS500 test subset. From the left: SNR = 15 dB, SNR = 25 dB, SNR = 35 dB and clean grayscale image. From the top: noisy grayscale image, reconstructed image, differences of the reconstructed image, thresholded and thinned differences.

phase, the steps were reduced to a quarter. The obtained optimal δ values were then used in the testing phase. For grayscale images, the optimal δ was set to $\delta = 6250$. For RGB images, only the V1 merging approach was used, and the optimal δ value for the image edge detection method was set to $\delta = 7500$.

In the testing phase, the experiments were performed on two test subsets of the BSDS300 and BSDS500 datasets. The maximum F-measure score achieved was compared with other edge detection methods presented in Sec. 3. For the test subset from the BSDS300 dataset, the values obtained from the training phase, i.e. δ for RGB images and δ for grayscale images, were determined as the optimal δ. The reason is that the train subsets of both datasets are identical and the validation subset of the BSDS500 dataset matches the test subset of the BSDS300 dataset. For the BSDS500 test dataset, the optimal δ parameter for RGB and grayscale images was set to values obtained from the validation phase.

In the case of the BSDS300 dataset, the proposed image edge detection method for both grayscale and RGB images gives better results than gradient-based edge detection techniques (Prewitt, Sobel, and Roberts) and Laplacian of Gaussian.

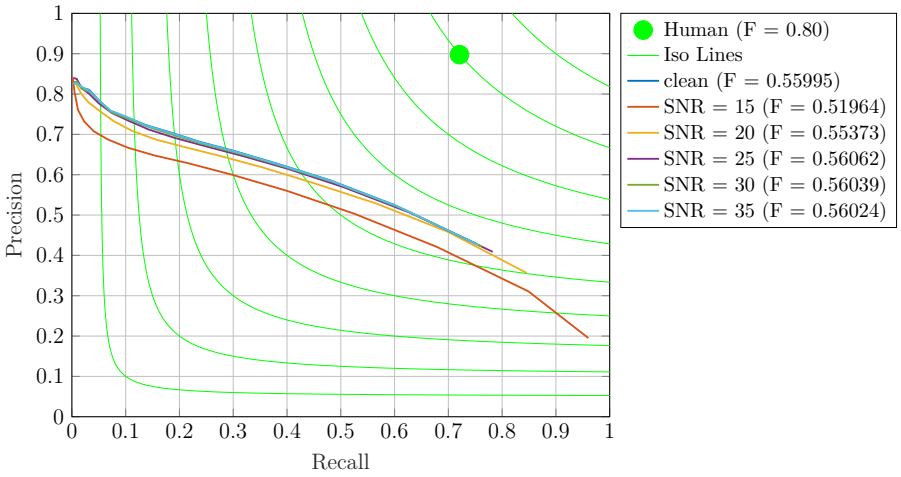


Fig. 6.16: Precision–recall curves with their maximum F-measure scores for optimal δ setting. Presented results are for test subset of BSDS500 consisting of 200 noisy grayscale natural images with SNR $\in \{15, 20, 25, 30, 35\}$ dB.

In the case of the BSDS500 dataset, the proposed image edge detection method for grayscale images gives worse results than the Canny detector. In case of RGB images, the results are slightly better than results of the Canny detector.

Compared to the results on the BSDS300 dataset, the edge detection results on the BSDS500 dataset are available only for the Canny detector and more sophisticated methods. However, they are more relevant for the proposed method since the parameter δ could be fine-tuned on the validation subset.

The proposed image edge detection method cannot be compared with more sophisticated edge detection methods because the output of our method is only raw edges. Better results would certainly be obtained by applying post-processing to the output. Further work could be done in this direction.

The next part of the experiments was performed on grayscale noisy images from the BSDS500 test dataset with different noise levels (SNR $\in \{15, 20, 25, 30, 25\}$ dB). The aim was to determine how the proposed method would behave when processing noisy images. From the outputs, it can be seen that at a lower noise level (SNR $\in \{25, 30, 25\}$ dB), the resulting F-measure score is similar to the F-measure score obtained on the clean grayscale images. Unfortunately, it is not possible to evaluate the level of denoising using e.g. SNR or MSE metrics because the δ parameter was set to the value at which the best edge detection is achieved.

CONCLUSION

This Thesis dealt with breakpoint/edge detection in 1D and 2D signals using sparse signal representation and convex optimisation. The main objective of this Thesis was to provide a novel method for detecting edges in an image. Several suitable approaches were selected and tested on 1D signals and the best of them was subsequently extended to 2D.

Chapter 1 included a basic overview, the notation used and explained sparse signal representation and convex optimisation, including proximal splitting algorithms, which were implemented in this Thesis.

Chapter 2 presented a discussion of a selection of topics from the field of image processing that were relevant to this Thesis. This included image segmentation, edge detection and enhancement, and denoising.

Chapter 3 gave an overview of the edge detection methods used. The methods were divided into sections according to their main principle, from the simplest to the more complex. The simplest techniques included, for example, the Sobel's edge operator and the Laplacian of Gaussian. The more complex algorithms included methods using texture, brightness, and colour features, and the Canny detector, which is considered to be the standard in image detection. Detection methods based on fuzzy logic, evolutionary algorithms or deep learning, and methods using optimisation recovery problems were presented as the most complex methods.

In Chapter 5, the input data and the general concept for the 1D signal segmentation and denoising were introduced, several possible solutions were designed, and the experiments and their results were evaluated. The main aim of Chapter 5 was to find the most promising algorithm for breakpoint detection in a 1D signal, which was then extended for the use in 2D in Chapter 6.

Sec. 5.1 introduced a general concept of the proposed 1D signal segmentation process intending to explain the basic idea of the proposed methodology. For this purpose, the description of the signal model was formulated first. This was a very important part since the proposed procedure was largely based on the assumptions derived from the signal model formulation. Then, the processed type of 1D signal and the polynomial bases used for the experiments in Chapter 5 were introduced, including the process of their generation. Finally, a general framework common to all the proposed methods used in Chapter 5 for 1D signal segmentation and denoising was presented. The general method was divided into two main steps – "Optimisation step using a proximal splitting algorithm" and "Segmentation and denoising step". The methods proposed in Chapter 5 differed from each other only in the first step, namely in the particular formulation of the problem and in the proximal splitting algorithm that was used.

The definition of the evaluation metrics was discussed in Sec. 5.2. Two perspectives – the quality evaluation of segmentation and denoising – were used to evaluate the results of the experiments. For the experiments in Chapter 5, a dataset of synthetic 1D piecewise-linear signals, synthetic 1D piecewise-quadratic signals, and a number of polynomial bases was created, and described in more detail in Sec. 5.3.

In Sec. 5.4, an ℓ_1-based unconstrained formulation of the recovery problem using the total variation was proposed, and two proximal algorithms (the Forward-backward (FB) and the Douglas–Rachford (DR) algorithm) were used to solve this recovery problem. The carried experiments revealed that the denoising process is better when the FB algorithm is used. However, the DR algorithm obtained a better breakpoint detection than the FB algorithm. Considering that noise reduction was only a by-product of the proposed algorithm, performance in breakpoint detection were more important.

Since the total variation treated each parametrisation vector separately, it did not ensure that possible breakpoint candidates were found at the same positions across all difference vectors. Therefore, the correct breakpoints could be discarded by this approach. To avoid this phenomenon, the ℓ_{21}-norm was used to enforce joint breakpoints over the difference vectors. The recovery problem presented in Sec. 5.5 was solved using the Forward-backward based primal-dual (FBB-PD) and the Chambolle–Pock (CP) algorithm. For both denoising and breakpoint detection, the FBB-PD algorithm gave better results than the CP algorithm. The comparison of the FBB-PD algorithm with the FB algorithm revealed that the FBB-PD algorithm provided better results in all areas. Based on these results, a conclusion can be made that the use of the ℓ_{21}-norm has led to a significant improvement in the results, compared to the simple total variation-based approach.

Another idea to improve the breakpoint detection was to imitate non-convexity via a series of convex programs, where the parameter of the currently solved convex problem depended on the latest solution. In this sense, the extended unconstrained recovery problem from Sec. 5.5 was presented in Sec. 5.6, and it was solved using the re-weighted and non-weighted variants of the Condat algorithm. The comparison of the non-weighted and re-weighted Condat algorithms showed that the breakpoint detection and denoising was better when the non-weighted Condat algorithm was used. Thus, there was no confirmation that the imitation of non-convexity would bring an improvement in the detection of breakpoints. Compared to the FBB-PD algorithm, the non-weighted Condat algorithm achieved worse results in the breakpoint detection and denoising. With the non-weighted Condat algorithm, it was necessary to adjust the δ parameter according to the noise level. The differences between FBB-PD and the non-weighted Condat algorithm were not so significant for the signals with big jump heights and high SNR.

Sec. 5.7 focused on the use of different types of bases, specifically O- and R- bases, in contrast to Sections 5.4 – 5.6 where a modified standard S-basis was used. The recovery problem was the same as in Sec. 5.6 and the FBB-PD and the non-weighted Condat algorithms were used to solve it. From the comparison between the FBB-PD algorithm and the non-weighted Condat algorithm in Sec. 5.6, it appeared that it was advantageous to use the FBB-PD algorithm. However, for solving the task in more dimensions, the non-weighted Condat algorithm offers a better possibility. From the breakpoint detection point of view, it was evaluated that the non-weighted Condat algorithm gave better results than the FBB-PD algorithm for both O- and R-bases. The results showed that the standard modified basis performed the best for linear signals and R-bases were the best for quadratic signals.

Chapter 6 dealt with edge detection in images. First, the general concept of the proposed image edge detection is introduced, followed by the presentation of the image dataset and the evaluation metric used, and finally, by the description of the specific image edge detection solution, including experiments and their evaluation. A comparison with other edge detection approaches was also included in this chapter.

Sec. 6.1 concentrated on the formulation of the 2D signal model description, which is an extension of the 1D signal model, and the general concept of image edge detection, which was divided into two main steps – "Optimisation step using a proximal splitting algorithm" and "Edge identification".

Sec. 6.2 described the 2D polynomial bases and the natural image datasets BSDS500 and BSDS300 (Berkeley Segmentation Data Set) that were used in the experiments. These datasets are often used for the comparison of segmentation and edge detection methods containing manually annotated image edges/segments used as the ground truth. The BSDS500 dataset is divided into three classes: train, validation, and test subset, and the BSDS300 is only a subset of the BSDS500 containing only images from the train and validation subsets. To evaluate the accuracy of the edge detection, the precision–recall curve and the F-measure score, defined in Sec. 6.3, were used.

The extended recovery problem from Sec. 5.7 for the use with 2D signals is presented in detail in Sec. 6.4. A suitable algorithm for solving the defined recovery problem seemed to be the non-weighted Condat algorithm, which obtained the most promising results when used on 1D signals.

Sec. 6.5 focused on experiments on grayscale and colour images from the train and validation subsets of the BSDS500 dataset. A standard modified basis was chosen for the experiments, since our experiments showed that the computational time requirement using R-basis is unacceptable for image processing. The experiments were divided into two phases – a training phase and a validation phase. The output of the proposed edge detection was compared with the image annotations from the

BSDS500 dataset.

In the training phase, several values of parameter δ were tested. The output of the training phase was the δ with the highest F-measure score (for both grayscale and RGB images). In the case of RGB images, the image edge detection algorithm was applied to each colour channel separately. It was necessary to compose the outputs of the channels into one output. Therefore, four merging approaches were proposed, and the best merging method was selected for the other experiments.

The purpose of the validation phase was to refine the parameter δ settings, where the steps between the tested values of parameter δ were smaller than in the training phase. The δ parameter with the highest F-measure scores was identified as the optimal setting for the testing phase.

In Sec. 6.6, the proposed edge detection method was compared with several other algorithms presented in Chapter 3. The comparison was made on the test subset of the public datasets BSDS300 and BSDS500. The edge detection method obtained better results for RGB images than for its grayscale variants. Our proposed method (for both RGB and grayscale images) obtained better results on the BSDS300 dataset than the classical gradient-based edge detection techniques (Prewitt, Sobel, and Roberts) and the Laplacian of Gaussian, whose output is a raw edge detection. Considering that our proposed approach did not involve any post-processing, such as merging parts of edges into a continuous whole, the results obtained were very good in terms of the raw edge detection. All methods that performed better than the proposed approach included some post-processing, or were much more sophisticated. Considering that the more sophisticated methods require some initial estimation of edges (raw edge detection) their results could be even better when combined with our proposed method. The method was also tested on the BSDS500 dataset, where it gave slightly better results on RGB images than the Canny detector, which includes post-processing.

Sec. 6.7 was devoted to experiments on noisy grayscale images from the BSDS500 test subset. The aim was to find out how our proposed method copes with edge detection on noisy images with varying levels of noise. The results showed that on less noisy images (SNR $\in \{25, 30, 35\}$), the method gave almost the same results, according to the F-measure score, as when used on "clean" images.

The experiments were performed using Matlab, and for the proximal algorithms, the UnLocBox toolbox was used. Software package for 1D signal segmentation and image edge detection containing all implemented methods was made publicly available on GitHub at:

`https://github.com/xnovos11/breakpoint_edge_detection`.

Other ideas and possible directions for future research can be considered. Decomposing the image into LAB channels instead of RGB channels could be a possible approach to improve edge detection in colour images. Furthermore, the input data for the "Optimisation using proximal splitting algorithm" step could be not only intensity images but also parametric maps, which could be obtained, for example, by texture analysis of the processed image. The resulting outputs could then be combined with the outputs from the individual colour channels. Moreover, the coupling of the colour channels can be already incorporated into the recovery problem [114, Chapter 6], instead of applying the proposed method to each colour channel separately. As a final direction for future development, the extension of the proposed method by post-processing to produce compact, thin edges can be considered.

AUTHOR'S BIBLIOGRAPHY

[1] M. Novosadová. Segmentace 3D obrazových dat s využitím pokročilých texturních a tvarových příznaků. In *Proceedings of the 20th Conference STUDENT EEICT 2014 Volume 2*, pages 42–44. Brno University of Technology, Faculty of Electrical Engineering and Communication, Apr. 2014.

[2] J. Jan, M. Novosadová, J. Demel, P. Ouředníček, J. Chmelík, and R. Jakubíček. Combined bone lesion analysis in 3D CT data of vertebrae. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6374–6377, Milan, Italy, Nov. 2015.

[3] M. Novosadová. Sparse signal recovery via convex optimization. In *Proceedings of the 22nd Conference STUDENT EEICT 2016*, pages 354–358. Brno University of Technology, Faculty of Electrical Engineering and Communication, Apr. 2016.

[4] M. Novosadová and P. Rajmic. Piecewise-polynomial signal segmentation using proximal splitting convex optimization methods. In *Applied mathematical programming and Modelling APMOD*, pages 1–14, Brno, Czech Republic, June 2016.

[5] P. Rajmic and M. Novosadová. On the limitation of convex optimization for sparse signal segmentation. In *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pages 550–554, Vienna, Austria, June 2016.

[6] M. Novosadová and P. Rajmic. Piecewise-polynomial curve fitting using group sparsity. In *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 320–325, Lisbon, Portugal, Oct. 2016.

[7] P. Rajmic, M. Novosadová, and M. Daňková. Piecewise-polynomial signal segmentation using convex optimization. *Kybernetika*, 53(6):1131–1149, Dec. 2017.

[8] M. Novosadová and P. Rajmic. Piecewise-polynomial signal segmentation using reweighted convex optimization. In *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, pages 769–774, Barcelona, Spain, 2017.

[9] M. Novosadova and P. Rajmic. Image edges resolved well when using an overcomplete piecewise-polynomial model. In *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–10, Cairns, QLD, Australia, Dec. 2018.

[10] M. Novosadová, P. Rajmic, and M. Šorel. Orthogonality is superiority in piecewise-polynomial signal segmentation and denoising. *EURASIP Journal on Advances in Signal Processing*, 2019(6):1–15, Jan. 2019.

REFERENCES

[11] R. Giryes, M. Elad, and A. M. Bruckstein. Sparsity based methods for overparameterized variational problems. *SIAM Journal on Imaging Sciences*, 8(3):2133–2159, 2015.

[12] P. Rajmic. *Řídké a nízkohodnostní reprezentace signálů s aplikacemi*. Habilitační práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014.

[13] E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14:877–905, 2008.

[14] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of The National Academy of Sciences*, 100(5):2197–2202, 2003.

[15] J. M. Giron-Sierra. *Sparse Representations*, pages 151–261. Springer Singapore, Singapore, 2017.

[16] P. Rajmic and M. Daňková. *Úvod do řídkých reprezentací signálů a komprimovaného snímání*. Vysoké učení technické v Brně, 2014.

[17] M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 1st edition, 2010.

[18] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: Algorithms and applications. *IEEE Access*, 3:490–530, 2015.

[19] R. Gribonval and M. Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49:3320–3325, 2003.

[20] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[21] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[22] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[23] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[25] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, 1997.

[26] E. Candes and T. Tao. The dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35(6):2313–2351, 2007.

[27] N. Parikh and S. Boyd. Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

[28] N. B. Karahanoglu and H. Erdogan. A* orthogonal matching pursuit: Best-first search for compressed sensing signal recovery. *Digital Signal Processing*, 22(4):555–568, 2012.

[29] J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[30] M. Fornasier, editor. *Theoretical Foundations and Numerical Methods for Sparse Recovery*. Berlin, De Gruyter, 2010.

[31] P. Combettes and J. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, 49:185–212, 2011.

[32] L. Condat. A generic proximal algorithm for convex optimization—application to total variation minimization. *IEEE Signal Processing Letters*, 21(8):985–989, 2014.

[33] M. Kowalski. Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324, 2009.

[34] N. Perraudin, D. I. Shuman, G. Puy, and P. Vandergheynst. UNLocBoX: A MATLAB convex optimization toolbox using proximal splitting methods. 2014, arXiv: 1402.0779.

[35] L. Condat. A direct algorithm for 1-D total variation denoising. *IEEE Signal Processing Letters*, 20(11):1054–1057, 2013.

[36] N. Komodakis and J. Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015.

[37] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[38] L. Condat. A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. *Journal of Optimization Theory and Applications*, 158(2):460–479, 2013.

[39] Michael Weeks. *Digital Signal Processing Using MATLAB and Wavelets*. George Washington University, Jones and Bartlett Publishers, 2011.

[40] J. G. Proakis and D. G. Manolakis. *Digital signal processing*. Prentice Hall, 1996.

[41] J. Jan. *Medical Image Processing, Reconstruction and Restoration: Concepts and Methods*. CRC Press, 1st edition, 2005.

[42] S. W. Zucker. Region growing: Childhood and adolescence. *Computer Graphics and Image Processing*, 5(3):382–399, 1976.

[43] J. L. Muerle and D. C. Allen. Experimental evaluation of techniques for automatic segmentation of objects in a complex scene. In *Pictorial Pattern Recognition*, pages 3–13. Thompson, Washington, 1968.

[44] S. L. Horowitz and T. Pavlidis. Picture segmentation by a direct split and merge procedure. In *2nd International Joint Conference on Pattern Recognition*, pages 424–433, 1974.

[45] B. P. Dobrin, T. Viero, and M. Gabbouj. Fast watershed algorithms: analysis and extensions. In *Nonlinear Image Processing V (SPIE 1994 International Symposium on Electronic Imaging: Science and Technology)*, pages 209–220, San Jose, CA, 1994.

[46] P. Brigger, J. Hoeg, and M. Unser. B-spline snakes: a flexible tool for parametric contour detection. *IEEE Transactions on Image Processing*, 9(9):1484–1496, 2000.

[47] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, 1992.

[48] P. Mukhopadhyay and B. B. Chaudhuri. A survey of Hough Transform. *Pattern Recognition*, 48(3):993–1010, 2015.

[49] C. Lopez-Molina, C. Marco-Detchart, H. Bustince, and B. De Baets. A survey on matching strategies for boundary image comparison and evaluation. *Pattern Recognition*, 115:107883, 2021.

[50] N. Narappanawar, B. Madhusudan Rao, and M. Joshi. Graph theory based segmentation of traced boundary into open and closed sub-sections. *Computer Vision and Image Understanding*, 115(11):1552–1558, 2011.

[51] W. Burger and M. Burge. *Principles of Digital Image Processing: Fundamental Techniques*. Springer London, 2010.

[52] N. Senthilkumaran and R. Rajesh. Edge detection techniques for image segmentation – A survey of soft computing approaches. *International Journal of Recent Trends in Engineering*, 1(2):250–254, 2009.

[53] K. Muntarina, S. B. Shorif, and M. S. Uddin. Notes on edge detection approaches. *Evolving Systems*, 13:169–182, 2022.

[54] G. Papari and N. Petkov. An improved model for surround suppression by steerable filters and multilevel inhibition with application to contour detection. *Pattern Recognition*, 44(9):1999–2007, 2011.

[55] O. P. Verma, N. Agrawal, and S. Sharma. An optimal edge detection using modified artificial bee colony algorithm. In *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences 86*, pages 157–168, 2016.

[56] P. Kovesi. Phase congruency detects corners and edges. *Digital Image Computing: Techniques and Applications 2003*, 1:309–318, CSIRO Publishing, 2003.

[57] F. Han, B. Liu, J. Zhu, and B. Zhang. Algorithm design for edge detection of high-speed moving target image under noisy environment. *Sensors*, 19(2):1–27, 2019.

[58] G. Papari and N. Petkov. Edge and line oriented contour detection: State of the art. *Image and Vision Computing*, 29(2):79–103, 2011.

[59] K. B. Krishnan, S. P. Ranga, and N. Guptha. A survey on different edge detection techniques for image segmentation. *Indian Journal of Science and Technology*, 10(4):1–8, 2017.

[60] Z. Hussain and D. Agarwal. A comparative analysis of edge detection techniques used in flame image processing. *International Journal of Advance Research In Science And Engineering (IJARSE)*, (4):3703–3711, 2015.

[61] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[62] S. Wang, F. Ge, and T. Liu. Evaluating edge detection through boundary detection. *EURASIP Journal on Advances in Signal Processing*, 2006(1):1–15, 2006.

[63] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[64] S. Kaur and I. Singh. Comparison between edge detection techniques. *International Journal of Computer Applications*, 145(15):15–18, 2016.

[65] B. K. Balabantaray, O. P. Sahu, N. Mishra, B. B. Biswal. A quantitative performance analysis of edge detectors with hybrid edge detector. *Journal of Computers*, 12(2):165–173, 2017.

[66] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *[1990] Proceedings Third International Conference on Computer Vision*, pages 52–57, 1990.

[67] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[68] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International journal of computer vision*, 30(2):117–156, 1998.

[69] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[70] J. Mairal, M. Leordeanu, F. R. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *European Conference on Computer Vision*, pages 43–56, 2008.

[71] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 1964–1971, 2006.

[72] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[73] X. Ren. Multi-scale improves boundary detection in natural images. In *European Conference on Computer Vision (ECCV)*, pages 533–545, 2008.

[74] Z. Tu. Probabilistic boosting-tree: learning discriminative models for classification, recognition, and clustering. In *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pages 1589–1596, 2005.

[75] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pages 1214–1221, 2005.

[76] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*, pages 282–289, 2001.

[77] P. Felzenszwalb and D. McAllester. A min-cover approach for finding salient curves. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 185–185, 2006.

[78] Q. Zhu, G. Song, and J. Shi. Untangling cycles for contour grouping. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[79] S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[80] L. Zadeh, G. Klir, and B. Yuan. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers*. Advances in fuzzy systems : applications and theory. World Scientific, 1996.

[81] L. R. Liang and C. G. Looney. Competitive fuzzy edge detection. *Applied Soft Computing*, 3(2):123–137, 2003.

[82] L. Hu, H. Cheng, and M. Zhang. A high performance edge detector based on fuzzy inference rules. *Information Sciences*, 177(21):4768–4784, 2007.

[83] P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265, 2016.

[84] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, 2007.

[85] O. P. Verma and A. S. Parihar. An optimal fuzzy system for edge detection in color images using bacterial foraging algorithm. *IEEE Transactions on Fuzzy Systems*, 25(1):114–127, 2017.

[86] M. Zhao, A. Fu, and H. Yan. A technique of three-level thresholding based on probability partition and fuzzy 3-partition. *IEEE Transactions on Fuzzy Systems*, 9(3):469–479, 2001.

[87] R. Sun, T. Lei, Q. Chen, Z. Wang, X. Du, W. Zhao, and A. K. Nandi. Survey of image edge detection. *Frontiers in Signal Processing*, 2:1–13, 2022.

[88] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 193–202, 2016.

[89] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam. CASENet: Deep category-aware semantic edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5964–5973, 2017.

[90] M. Pu, Y. Huang, Q. Guan, and H. Ling. RINDNet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6859–6868, 2021.

[91] Y. Ganin and V. Lempitsky. N^4-fields: Neural network nearest neighbor fields for image transforms. In *Asian Conference on Computer Vision (ACCV)*, pages 536–551, 2014.

[92] K.-K. Maninis, J. Pont-Tuset, P. Arbelaez, and L. Van Gool. Convolutional oriented boundaries. In *European Conference on Computer Vision (ECCV)*, pages 580–596, 2016.

[93] D. Xu, W. Ouyang, X. Alameda-Pineda, E. Ricci, X. Wang, N. M. Sebe. Learning deep structured multi-scale features using attention-gated CRFs for contour prediction. In 31st Conference on Neural Information Processing Systems (NIPS 2017), pages 3964–3973, 2017.

[94] G. Bertasius, J. Shi, and L. Torresani. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4380–4389, 2015.

[95] S. Xie and Z. Tu. Holistically-nested edge detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015.

[96] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang. Bi-directional cascade network for perceptual edge detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3823–3832, 2019.

[97] X. Soria, E. Riba, and A. Sappa. Dense extreme inception network: Towards a robust CNN model for edge detection. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1912–1921, 2020.

[98] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikäinen, and L. Liu. Pixel difference networks for efficient edge detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5097–5107, 2021.

[99] S.-M. Hou, C.-L. Jia, Y.-B. Wanga, and M. Brown. A review of the edge detection technology. *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC)*, 1(2):26–37, 2021.

[100] E. Elizar, M. A. Zulkifley, R. Muharar, M. H. M. Zaman, and S. M. Mustaza. A review on multiscale-deep-learning applications. *Sensors*, 22(19), 2022.

[101] Y. Liu and M. S. Lew. Learning relaxed deep supervision for better edge detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 231–240, 2016.

[102] J.-L. Starck, E. J. Candes, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):670–684, 2002.

[103] G. Kutyniok and W.-Q. Lim. Compactly supported shearlets are optimally sparse. *Journal of Approximation Theory*, 163(11):1564–1589, 2011.

[104] GN Nettest. *Understanding OTDR*. GN Nettest, 2000. Rev. A.

[105] Optical time domain reflectometers – pocket guide. Technical report, Agilent Technologies, Germany, 2001.

[106] M. Filka, T. Horváth, P. Münster, M. Čučka, R. Šifta, F. Urban, D. Grenar, and M. Kyselák. *Optoelektronika pro telekomunikace a informatiku*. 2017.

[107] J. Bisgard. *Analysis and Linear Algebra: The Singular Value Decomposition and Applications*. American Mathematical Society, 2019.

[108] G. Walter and X. Shen. *Wavelets and Other Orthogonal Systems, Second Edition*. Studies in Advanced Mathematics. Taylor & Francis, 2000.

[109] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the Lasso and Generalizations*. Monographs on Statistics and Applied Probability 143. CRC Press, Boca Raton, 2015.

[110] P. Rajmic. Exact risk analysis of wavelet spectrum thresholding rules. In *Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 455–458, 2003.

[111] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[112] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[113] M. Kowalski and B. Torrésani. Structured Sparsity: from Mixed Norms to Structured Shrinkage. In *Signal Processing with Adaptive Sparse Structured Representations (SPARS'09)*, pages 1–6, 2009.

[114] D. L. Kristian Bredies. *Mathematical Image Processing*. Birkhäuser Cham, 2019.

[115] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[116] M. Šorel and F. Šroubek. Fast convolutional sparse coding using matrix inversion lemma. *Digital Signal Processing*, 55:44–51, 2016.

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols

∇	the finite difference operator
α_k	normalized positive factors
δ	the noise level
λ	threshold for breakpoint detection
σ	noise standard deviation
τ_k	regularisation weights
$\mathbf{B}$	non-orthogonal basis
$\mathbf{d}$	ℓ_2-norm of differences of parametrisation coefficients
$\hat{\mathbf{d}}$	post-processed $\mathbf{d}$
$\mathbf{D}$	image of differences
$\mathbf{e}$	Gaussian i.i.d. noise vector
$\mathbf{E}$	Gaussian i.i.d. noise matrix
F	F-measure score
h_l	jump height
$\mathbf{I}$	identity matrix
Id	identity operator
K	degree of the polynomial signal
l_{m}	window length of the median filter
L	linear operator
M	number of rows in the matrix
N	number of the signal samples / number of columns in the matrix
$\mathbf{N}$	normalised basis
$\mathbf{O}$	orthogonal basis
$\mathbf{p}_k$	basis polynomial
P	precision

$\mathbf{P}$	matrix of polynomials / polynomial basis
$\bar{\mathbf{P}}_{kl}$	polynomial basis image
R	recall
$\mathbf{R}$	random orthogonal basis
$\bar{\mathbf{R}}$	random orthogonal basis consisting of basis images
s	segment of the signal
$\hat{s}$	detected segment of the signal
S	number of the signal segments
$\hat{S}$	number of the detected signal segments
$\mathbf{S}$	modified standard basis
$\bar{\mathbf{S}}$	modified standard basis consisting of basis images
$\mathbf{w}_k$	vector of weights
$\mathbf{W}$	matrix of weights
$\mathbf{x}$	parametrisation coefficients
$\hat{\mathbf{x}}$	obtained parametrisation coefficients
$\dot{\mathbf{x}}$	parametrisation coefficients of denoised signal $\dot{\mathbf{y}}$
$\mathbf{X}$	matrix of parametrisation coefficients
$\mathbf{y}$	observed polynomial signal
$\hat{\mathbf{y}}$	obtained polynomial signal
$\dot{\mathbf{y}}$	denoised polynomial signal
$\mathbf{y}_{\text{clean}}$	clean piecewise-polynomial signal
$\mathbf{y}_{\text{denoised}}$	denoised piecewise-polynomial signal
$\mathbf{y}_{\text{noisy}}$	noisy piecewise-polynomial signal
$\mathbf{Y}$	2D signal / image
$\mathbf{Y}_{\text{gray}}$	grayscale image
$\mathbf{Y}_{\text{noisy}}$	noisy image
$\mathbf{Y}_{\text{RGB}}$	RGB image
$\hat{\mathbf{Y}}$	obtained polynomial image

Abbreviations

AAR	Average of values in HV to average of values in OV Ratio
AWGN	Additive White Gaussian Noise
BDCN	Bi-directional cascade Network
BEL	Boosted Edge Learning
BSDS	Berkeley Segmentation Dataset
CAD	Computer-Aided Design
CA	Condat Algorithm
CDT	Constrained Delaunay Triangulation
CEDN	Convolutional Encoder-Decoder Network
CNN	Convolutional Neural Network
COB	Convolutional Oriented Boundaries
CP	Chambolle–Pock (algorithm)
CRF	Conditional Random Field
CT	Computer Tomography
DR	Douglas–Rachford (algorithm)
DSP	Digital Signal Processing
EA	Evolutionary Algorithm
ECG	Electrocardiogram
EEG	Electroencephalogram
FBB-PD	Forward-Backward Based Primal-Dual (algorithm)
FB	Forward-Backward (algorithm)
FISTA	Fast Iterative Shrinkage/Thresholding Algorithm
FN	False Negative
FP	False Positive
gPb	Globalized Probability of boundary
HED	Holistically-Nested Edge Detection
HFL	High-for-Low

HV	Highest values
LASSO	Least Absolute Shrinkage and Selection Operator
LoG	Laplacian of Gaussian
MMR	Minimum-to-Maximum Ratio
mPb	Multiscale Probability of boundary
MR(I)	Magnetic Resonance (Imaging)
MSE	Mean Square Error
N^4-Fields	Neural Network Nearest Neighbor Fields
NoB	Number of Breakpoints
NP-hard	Nondeterministic Polynomial-time Hard
OTDR	Optical Time-Domain Reflectometry
OV	Other values
PA	Proximal algorithm
Pb	Probability of boundary
PiDiNet	Pixel Difference Networks
RADAR	Radio Detection and Ranging
RDS	Relaxed Deep Supervision
RGB	Red Green Blue
ROC	Receiver operating characteristic
sPb	Spectral Probability of boundary
SE	Structured Edges
SNR	Signal-to-Noise Ratio
SONAR	Sound Navigation and Ranging
(S)USAN	(Smallest) Univalue Segment Assimilating Nucleus
SVD	Singular Value Decomposition
TP	True Positive
TV	Total Variation
US	Ultrasound

A APPENDIX

A.1 Inversion matrix theorem

Theorem 1 *Let* $\mathbf{P} = \left[\mathbf{I}\ \mathbf{D}\ \mathbf{D}^2 \dots \mathbf{D}^k\right]$ *be a matrix with* $\mathbf{D} = \operatorname{diag}(d_1, \dots, d_N)$ *square and diagonal. Then for the inversion of the* $N(k+1) \times N(k+1)$ *matrix* $\mathbf{I} + \lambda \mathbf{P}^\top \mathbf{P}$ *with* $\lambda \in \mathbb{R}^+$ *it holds*

$$\left(\mathbf{I} + \lambda \mathbf{P}^\top \mathbf{P}\right)^{-1} = \mathbf{I} - \lambda \left(\mathbf{I} \otimes \mathbf{C}^{-1}\right) \mathbf{P}^\top \mathbf{P},$$

where the identity matrices $\mathbf{I}$ *are of appropriate sizes, symbol* $\otimes$ *denotes the Kronecker product and the* $N \times N$ *matrix* $\mathbf{C}$ *is defined by*

$$\mathbf{C} = \mathbf{I} + \lambda \sum_{i=0}^{k} \mathbf{D}^{2i} = \mathbf{I} + \lambda \mathbf{P}\mathbf{P}^\top.$$

Proof: In the definition of matrix $\mathbf{C}$, we utilize the fact that $\mathbf{P}\mathbf{P}^\top = \left[\mathbf{I} \dots \mathbf{D}^k\right] \cdot \left[\mathbf{I} \dots \mathbf{D}^k\right]^\top = \sum_{i=0}^{k} \mathbf{D}^{2i}$. We have $(\mathbf{I} \otimes \mathbf{C})^{-1} = (\mathbf{I} \otimes \mathbf{C}^{-1})$ and we use this fact to find the product:

$$\begin{aligned}
&\left(\mathbf{I} + \lambda \mathbf{P}^\top \mathbf{P}\right)^{-1} \cdot \left(\mathbf{I} + \lambda \mathbf{P}^\top \mathbf{P}\right) \\
&= \left[\mathbf{I} - \lambda \left(\mathbf{I} \otimes \mathbf{C}^{-1}\right) \mathbf{P}^\top \mathbf{P}\right] \cdot \left(\mathbf{I} + \lambda \mathbf{P}^\top \mathbf{P}\right) \\
&= \mathbf{I} - \lambda \left(\mathbf{I} \otimes \mathbf{C}^{-1}\right) \mathbf{P}^\top \mathbf{P} + \lambda \mathbf{P}^\top \mathbf{P} - \lambda^2 \left(\mathbf{I} \otimes \mathbf{C}^{-1}\right) \mathbf{P}^\top \mathbf{P} \mathbf{P}^\top \mathbf{P} \\
&= \mathbf{I} + \left(\mathbf{I} \otimes \mathbf{C}^{-1}\right) \left[-\mathbf{I} + (\mathbf{I} \otimes \mathbf{C}) - \lambda \mathbf{P}^\top \mathbf{P}\right] \lambda \mathbf{P}^\top \mathbf{P}.
\end{aligned}$$

Now, since

$$\mathbf{I} \otimes \mathbf{C} = \begin{bmatrix} \mathbf{I} + \lambda \sum_{i=0}^{k} \mathbf{D}^{2i} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{I} + \lambda \sum_{i=0}^{k} \mathbf{D}^{2i} \end{bmatrix}$$

and

$$\mathbf{P}^\top \mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{D} & \mathbf{D}^2 & \dots & \mathbf{D}^k \\ \mathbf{D} & \mathbf{D}^2 & \mathbf{D}^3 & \dots & \mathbf{D}^{k+1} \\ \mathbf{D}^2 & \mathbf{D}^3 & \mathbf{D}^4 & \dots & \mathbf{D}^{k+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}^k & \mathbf{D}^{k+1} & \mathbf{D}^{k+2} & \dots & \mathbf{D}^{2k} \end{bmatrix},$$

we see that

$$-\mathbf{I} + (\mathbf{I} \otimes \mathbf{C}) - \lambda \mathbf{P}^\top \mathbf{P} = \begin{bmatrix} \lambda \sum_{i \neq 0} \mathbf{D}^{2i} & -\lambda \mathbf{D} & -\lambda \mathbf{D}^2 & \cdots & -\lambda \mathbf{D}^k \\ -\lambda \mathbf{D} & \lambda \sum_{i \neq 1} \mathbf{D}^{2i} & -\lambda \mathbf{D}^3 & \cdots & -\lambda \mathbf{D}^{k+1} \\ -\lambda \mathbf{D}^2 & -\lambda \mathbf{D}^3 & \lambda \sum_{i \neq 2} \mathbf{D}^{2i} & \cdots & -\lambda \mathbf{D}^{k+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\lambda \mathbf{D}^k & -\lambda \mathbf{D}^{k+1} & -\lambda \mathbf{D}^{k+2} & \cdots & \lambda \sum_{i \neq k} \mathbf{D}^{2i} \end{bmatrix}.$$

Multiplying this matrix from right by $\lambda \mathbf{P}^\top \mathbf{P}$ results in the zero matrix.

Since $\mathbf{I} + \lambda \mathbf{P}^\top \mathbf{P}$ is symmetric, so it is its inverse, and it is not necessary to prove the multiplication in the opposite order. □

We note that the theorem could be proven with the help of the (Sherman–Morrison–Woodbury) inversion lemma [115, 116], but we prefer the presented approach for clarity.

A.2 Orthonormal basis theorem

Theorem 2 *Let* $\{\mathbf{p}_{k\mathrm{v}}\}_{k=0,\dots,K}$ *and* $\{\mathbf{p}_{l\mathrm{h}}\}_{l=0,\dots,K}$ *be two orthonormal bases of a* K*-dimensional space. Then, the set* $\{\bar{\mathbf{P}}_{ij}\}$*,* $i = 0, \dots, K$*,* $j = 0, \dots, K$*, defined as*

$$\bar{\mathbf{P}}_{ij} = \mathbf{p}_{i\mathrm{v}} \cdot \mathbf{p}_{j\mathrm{h}}^\top,$$

forms an orthonormal basis of the space of matrices of order $(K+1)$*, equipped with the usual Frobenius inner product* $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{trace}\left(\mathbf{A}^\top \mathbf{B}\right)$*.*

Proof: For $\{\bar{\mathbf{P}}_{ij}\}$ to form an orthonormal basis, we first need to show that it is an orthonormal system, i.e.

$$\langle \bar{\mathbf{P}}_{ij}, \bar{\mathbf{P}}_{kl} \rangle = \begin{cases} 1 & \text{if} \quad i = k, j = l, \\ 0 & \text{otherwise,} \end{cases}$$

or, using the Kronecker delta notation, $\langle \bar{\mathbf{P}}_{ij}, \bar{\mathbf{P}}_{kl} \rangle = \delta_{ik}\delta_{jl}$. We can compute

$$\begin{aligned} \langle \bar{\mathbf{P}}_{ij}, \bar{\mathbf{P}}_{kl} \rangle &= \mathrm{trace}\left(\bar{\mathbf{P}}_{ij}^\top \bar{\mathbf{P}}_{kl}\right) \\ &= \mathrm{trace}\left(\left(\mathbf{p}_{i\mathrm{v}}\mathbf{p}_{j\mathrm{h}}^\top\right)^\top \mathbf{p}_{k\mathrm{v}}\mathbf{p}_{l\mathrm{h}}^\top\right) \\ &= \mathrm{trace}\left(\mathbf{p}_{j\mathrm{h}} \underbrace{\mathbf{p}_{i\mathrm{v}}^\top \mathbf{p}_{k\mathrm{v}}}_{\delta_{ik}} \mathbf{p}_{l\mathrm{h}}^\top\right) \\ &= \delta_{ik}\,\mathrm{trace}\left(\mathbf{p}_{j\mathrm{h}}\mathbf{p}_{l\mathrm{h}}^\top\right) \\ &= \delta_{ik}\mathbf{p}_{j\mathrm{h}}^\top \mathbf{p}_{l\mathrm{h}} \\ &= \delta_{ik}\delta_{jl}. \end{aligned}$$

Both δ_{ik} and δ_{jl} were obtained using the fact that $\{\mathbf{p}_{k\mathrm{v}}\}_{k=0,\dots,K}$ and $\{\mathbf{p}_{l\mathrm{h}}\}_{l=0,\dots,K}$ are orthonormal bases. The relation $\mathrm{trace}\left(\mathbf{p}_{j\mathrm{h}}\mathbf{p}_{l\mathrm{h}}^\top\right) = \mathbf{p}_{j\mathrm{h}}^\top \mathbf{p}_{l\mathrm{h}}$ could be proven directly by computing the diagonal entries of the matrix $\mathbf{p}_{j\mathrm{h}}\mathbf{p}_{l\mathrm{h}}^\top$.

It remains to show that the system $\{\bar{\mathbf{P}}_{ij}\}$ spans the whole space. This is trivial, since we have an orthonormal system of $(K+1)^2$ elements in a $(K+1)^2$-dimensional space.