Universität Regensburg
Fakultät für Mathematik

Solving Convex and Non-Convex Functionals in Image Processing using a Primal-Dual Algorithm

MASTERARBEIT

zur Erlangung des akademischen Grades
Master of Science
im Studiengang Computational Science

Eingereicht bei Prof. Dr. Harald Garcke
am 28.02.2016

Vorgelegt von:
Michael Bauer
Banater Str. 1
84061 Ergolsbach
Geboren am 23.08.1987 in Gräfelfing

Abstract

We present a comparison of different convex total variation based image processing algorithms and the non-convex Mumford-Shah Functional. We will revisit and make use of the fast primal-dual algorithm of [**?**]. We consider all steps, projections, operators and computations and will give a range of proofs. We also present a various range of applications and compare the presented methods to each other. Providing run-times, used memory and results will lead us to a conclusion which problem fits best to several situations.

Contents

List of Tables

List of Figures

1 Introduction

In the second decade of the twenty-first century autonomous driving seems to be the big thing for the Car, Computer and Software industry. The expectation is nothing less than having fewer, or even no, accidents. With these cars, industry is trying to change the world to a better. As this may comfort many people, it poses a lot of problems which need to be solved, for instance a stable hardware or even more important viable software. One field of research - of many others - is called Machine Learning, where the computer is trained to make the right decisions in the right situations. The car should be able to accommodate to all possible circumstances which could appear during a drive. To make it even more complicated, the car would need to take a decision in real-time. As one can imagine, it is extremely difficult to develop fast, stable and tractable methods.
Learning from a certain situation, the car and the computer respectively needs data from the environment. This can be the shape of another car, traffic lights, differences in the lighting conditions or the distance to other objects. What all these information have in common, they can be collected via images. Small cameras are tracking the traffic and surrounding and providing data. Using the images, the car's computer can be trained when to stop or where to turn right.
But it's not only the autonomously driving cars which make use of images to learn. Doctors use X-ray view or MRI scans in patient treatment, semiconductor companies use pictures of wafers seeking for damages on it, e.g. scratches or dark spots and unfortunately images are also used in modern warfare. For this reason image processing has become more important during the last decades. Researchers dealt with many problems like edge detection, deblurring, denoising, inpainting or image segmentation. In the field of edge detection the probably most famous researcher is John F. Canny. He provided the first tractable and stable edge detection algorithm, presented in the year 1986 (reference). One step in Canny's edge detector is to denoise and smooth the image. Therefore a Gaussian-Filter is used. The idea behind this filter is to convolute an image with a Gaussian kernel, or also called Gaussian curve. This filter provides reasonable results, but no more no less.
Only three years after Canny published his work, two researchers, namely David Mumford and Jayant Shah, published a paper whose impact continous to this day - to date it was cited over 4.500 times (scholar). They proposed to minimize the energy of the so called Mumford-Shah Functional in order to approximate an image optimally. Solving this optimization problem leads to applications like image denoising, inpainting and even segmentation could be handled. Unfortunately, this functional is by definition non-convex. For this reason finding the minimal energy of it is a NP-hard problem. This fact makes it so difficult to deal with but also so interesting.
One approach to compute the minimizer is by convex relaxation. It makes the non-

convex problem convex and one can apply a fast and tractable algorithm. The idea of convex relaxation goes back to Bouchite, Alberti, Dal Maso and was then used and further developed by Thomas Pock et.al. to make use of a fast primal-dual algorithm. It leads us to almost exact solutions and is highly parallelizable. Possibly inspired by the Mumford-Shah Functional another idea for image processing took place in 1992. Rudin, Osher and Fatemi focused on total variation based imaging. The total variation is a concept developed in the 19th century. Today there is a large community which associates total variation almost all with image processing. What Rudin et. al. proposed was also a minimization problem, namely the $TVL2$-Model (or ROF-Model). As the name suggests it is based on the total variation and the $L2$-Norm. The formulation of it, as we will see, looks quite similar to the Mumford-Shah Functional, but has one term left and is by definition convex. This makes it much easier to handle all computations but the output images are not even as good as they are applying Mumford-Shah. Applications which arise from the ROF-Model are rare. Removing noise from a given input image is the most common use. But as one later sees: Solutions are better than using a Gaussian-Filter.
By replacing the $L2$-Norm with the $L1$-Norm in the ROF-Model one derives the $TVL1$-Model, hence the name. A convex model and minimization its energy again is quite easy to compute. In this work we will see, that it is possibly the best model of the presented. It can deal with - so called - salt and pepper noise, handles inpainting, output images are sharp and close to the original image and most important: it runs in parallel on a GPU in real-time.
This work will present all of these methods, clarify which properties they have and make all computations clear. We also want to present the primal-dual algorithm which can be used to solve all models. We provide a large range of applications and the underlying run-times. At the end of this work we want to compare the models and conclude which one is the exactest, fastest and most tractable one.

2 Basic Concepts

In the first chapter we give an introduction to the most important concepts we meet in this thesis. We start by defining images in a mathematical manner, then introduce basic concepts of convex analysis and the total variation.

2.1 Images in Mathematics

Images can be viewed as mathematical objects. We can distinguish between discrete and continuous images. Let us first introduce what images are in the mathematical sense and then give some examples. The definition and some examples can also be found in [**?**].

Definition 2.1 (Image) *Let $\Omega \subset \mathbb{R}^n$ be an image domain and C be a color space. A n-dimensional image u is a mapping $u : \Omega \longrightarrow C$, where each point in Ω corresponds to a color value in C.*

Remark 2.2 (Continuous and discrete images) *Yet, we did not tell anything about the difference between continuous and discrete images. We distinguish these two cases by the image domain Ω.*

1. *Let $\Omega = \{1, ..., N\}^n$, then u is a discrete image, since Ω is a discrete set.*
2. *Let $\Omega \subseteq \mathbb{R}^n$, then u is a continuous image. For instance, we could set $\Omega = [a, b]^n$ with $a, b \in \mathbb{R}$ and $a < b$.*

There are several image types like binary images, or colored images. The property to which class an image u belongs is determined by the color space C:

- *Let $C = \{0, 1\}$, then we call the image binary.*
- *In the case of grayscaled images we set $C = \{0, ..., 2^k - 1\}$, where the value k determines the bit depth. Usually we find in computers $k = 8$, i.e. grayscaled images take values in between 0 and 255, where $0 =$ black and $255 =$ white.*
- *n-dimensional color images have $C = \{0, ..., 2^k - 1\}^n$, or*
- *as a last example an image u could also map to continuous color spaces, e.g. $C = [a, b]^n$ or $C = \mathbb{R}^n$. Most common are RGB (red-green-blue) images with $n = 3$, $a = 0$ and $b = 1$.*

In this thesis we consider two cases for the domain Ω. One where Ω is two-dimensional, and the other where Ω is a three-dimensional space. In the first case, we call a point (i, j) in Ω pixel, in the second case a point (i, j, k) is called voxel.

Computationally, it is a convenient method to store a two-dimensional image $u \in \mathbb{R}^{N \times M}$ not as a grid (like a matrix), but as a vector $u \in \mathbb{R}^{N \cdot M}$. To derive such a vector representation one needs to linearize the arguments of the function u. In other words we transform (i, j) to $(j + i \cdot N)$. The particular pixels are stored row wise, starting at the top left corner of the image and stopping at the bottom right corner. The access of one element becomes

$$u(i, j) \rightsquigarrow u(j + i \cdot N).$$

There are several programming languages where the indices of a vector start with 0. An example would be C++, which we used for our implementations. We have $u(0, 0) = u(0 + 0 \cdot N) = u(0)$ and $u(N - 1, M - 1) = u(M - 1 + (N - 1) \cdot N)$. Even though we will treat u as a vector we still denote the discrete pixel positions with the notation

$$u(i, j) = u_{i,j} \ \ \forall i = 1, ..., N, j = 1, ..., M.$$

Further, note that the notation $\langle \cdot, \cdot \rangle$ has two different meanings in this thesis. In the sense of infinite spaces, e.g. $u \in L^1(\Omega)$, the inner product of functions is defined by

$$\langle u, v \rangle = \int_\Omega u(x)v(x)dx.$$

Whereas, in a finite space, e.g. the euclidean space, this expression stands for the standard inner product (see also equation **??**).

2.2 Convex Optimization and Convex Analysis

In this section we cover a few topics which are related to convex analysis. Since we are considering (convex) optimization problems, we first define a convex program where we mainly follow [**?**].

An optimization problem is of the form

$$\begin{array}{lll} \min\limits_{u \in \mathbb{R}^n} & F(u) & \\ \text{subject to} & G_i(u) \leq 0, & i = 1, ..., m \\ & H_j(u) = 0 & j = 1, ..., p, \end{array} \tag{2.1}$$

where

- $u \in \mathbb{R}^n$ is called the *optimization variable*,
- $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ *objective function* or *cost function*,
- $G_i(u) \leq 0$ *inequality constraints* for all $i = 1, ..., m$,

- $H_j(u) = 0$ *equality constraints* for all $j = 1, ..., p$,
- $G_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ for all $i = 1, ..., m$ the *inequality constraint functions* and
- $H_j(u) = 0$ for all $j = 1, ..., p$ the *equality constraint functions.*

If we have no constraints, meaning $m = p = 0$, we say the system **??** is *unconstrained.* It is called convex if all, the objectiv, equality and inequality functions, are convex. Further, we define the domain of the optimization problem **??** by

$$\mathcal{D} = \operatorname{dom}(F) \cap \bigcap_{i=1}^{m} \operatorname{dom}(G_i) \cap \bigcap_{j=1}^{p} \operatorname{dom}(H_j). \tag{2.2}$$

This set is the set of points for which the objective and all constraint functions are defined. We call a point $u \in \mathcal{D}$ *feasible* if it satisfies all constraints. The optimization problem **??** is said to be feasible if there exists at least one feasible point, and *infeasible* otherwise. We call the set of all feasible points the *feasible set* or alternatively the *constraint set.*
We define the optimal value u^* of the system **??** by

$$u^* = \inf \big\{ F(u) : G_i(u) \leq 0, i = 1, ..., m, H_j(u) = 0, j = 1, ..., p \big\},$$

where we allow u^* to take on the extended values $\pm\infty$. We set $u^* = \infty$ if the problem is infeasible, since $\inf(\emptyset) = \infty$. If there are feasible points u_k with $F(u_k) \longrightarrow -\infty$ as $k \longrightarrow \infty$, then $u^* = -\infty$, and we say that problem **??** is *unbounded below.*

Since, convex optimization is based on convex functions we need to define these class of functions and its properties. We set $X \subseteq \mathbb{R}^n$ and its dual space $X^* = Y$. According to [**?**] and [**?**] we have the following basic definitions, propositions and theorems.

Definition 2.3 (Convex Set) *A subset $C \subseteq \mathbb{R}^n$ is said to be convex if and only if for any $u_1, u_2 \in C$, the line segment $[u_1, u_2] \subseteq C$, that is, for any $t \in [0, 1]$,*

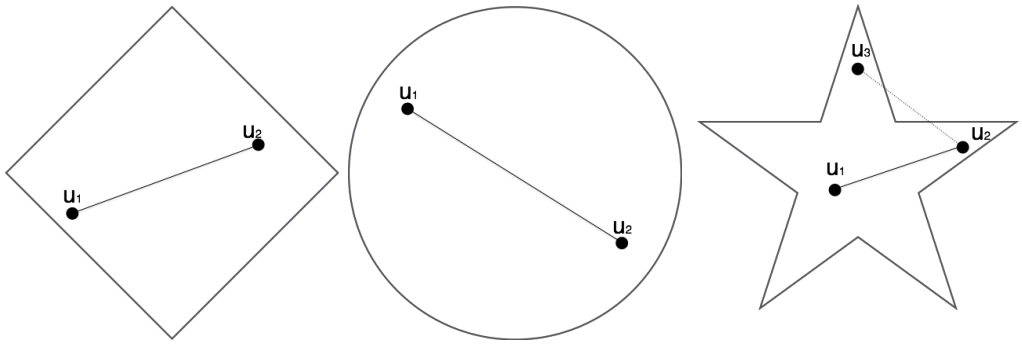
$$u_1 t + u_2(1 - t) \in C. \tag{2.3}$$

We call C strictly convex, if it is closed and

$$(1 - t)u_1 + u_2 t \in \operatorname{int} C, \ \ \forall u_1, u_2 \in C, \ \ u_1 \neq u_2, \ \ t \in [0, 1], \tag{2.4}$$

where int *stands for interior.*

This definition ensures that if C is convex, we can always find two arbitrary points in C, such that the line segment $[u_1, u_2]$ with end points u_1, u_2 lies fully in C (see figure **??**).



(a) l_1 unit sphere (b) l_2 unit sphere (c) Star: a non-convex set.

Figure 2.1: We see three different sets, where (a) and (b) are convex, but (c) is not. (a) refers to the unit sphere of the l_1 norm where (b) refers to the unit sphere of the l_2 norm. In (c) we see that the line segment of $[u_2, u_3]$ lies not in the set itself.

Definition 2.4 (Inidcator Function, Support Function) *For any subset $C \subseteq X$ of a vector space, the indicator function $\delta_C : X \longrightarrow \mathbb{R}_\infty$ is defined as*

$$\delta_C(u) = \begin{cases} 0 & \text{if } u \in C, \\ \infty & \text{if } u \notin C. \end{cases} \tag{2.5}$$

The support function of the set C is defined as

$$S_C(u) = \sup_{p \in C} \langle p, u \rangle, \tag{2.6}$$

where we allow $S_C(u)$ to be $+\infty$.

Definition 2.5 (Convex Function, Proper, Lower-Semicontinuouity) *Let $C \in \mathbb{R}^n$ be a convex set. A function $F : C \longrightarrow \mathbb{R}_\infty$ is said to be*

- *convex if for all $u_1, u_2 \in C$ and any $t \in [0, 1]$ the inequality*

$$F(u_1 t + u_2(1-t)) \leq F(u_1)t + F(u_2)(1-t) \tag{2.7}$$

  *is satisfied. F is called strictly convex, if the inequality **??** holds strictly, whenever u_1, u_2 are distinct points and $t \in (0, 1)$.*

- *proper if F is not identically $-\infty$ or $+\infty$.*

- *lower-semicontinuous (l.s.c) if for any $u \in C$ and a sequence (u_n) converging to u,*

$$F(u) \leq \liminf_{n \to \infty} F(u_n). \tag{2.8}$$

We define $\Gamma_0(C)$ as the set of all convex, proper, l.s.c. functions on C.

A common example for a strictly convex function would be a quadratic function, c.f. example **??** and figure **??** (a). It is illustrated, together with the plot in (b) of the function

$$F(u) = \begin{cases} u^2 & x \in [-1, 1] \\ -\frac{1}{2}u(u-4) & x \in (1, 3]. \end{cases} \tag{2.9}$$

being lower-semicontinuous.

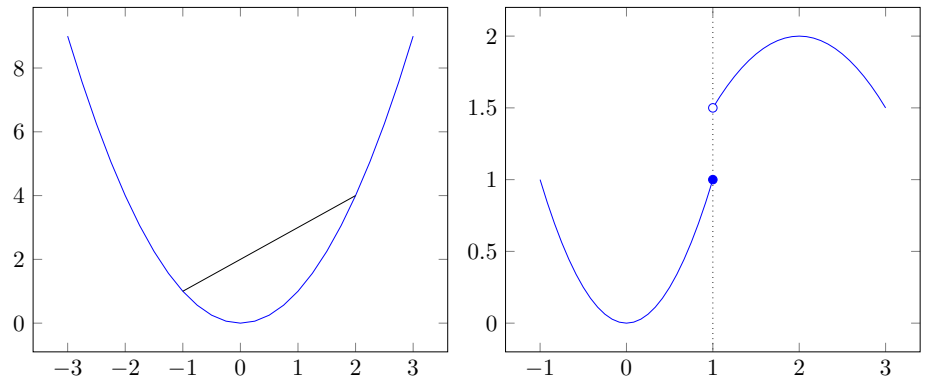
Proposition 2.6 *Let $F, G : X \longrightarrow \mathbb{R}_\infty$ be two convex functions. Then $F + G$ is also convex.*

Proof Define $H(u) = F(u)+G(u)$ with F, G convex and choose $u_1, u_2 \in X$ and $t \in [0, 1]$, then we obtain

$$\begin{aligned} H(u_1 t + u_2(1-t)) &= F(u_1 t + u_2(1-t)) + G(u_1 t + u_2(1-t)) \\ &\leq F(u_1)t + F(u_2)(1-t) + G(u_1)t + G(u_2)(1-t) \\ &= \underbrace{F(u_1)t + G(u_1)t}_{=H(u_1)t} + \underbrace{F(u_2)(1-t) + G(u_2)(1-t)}_{H(u_2)(1-t)} \\ &= H(u_1)t + H(u_2)(1-t), \end{aligned}$$

where we used the convexity of F and G. This shows that H is convex. ■

Remark 2.7 (Concave Function) *If $-F$ is (strictly) convex, then we say that F is (strictly) concave. If F is both convex and concave we say that F is affine, i.e. equality holds in Equation* **??**.



(a) A quadratic function. (b) A l.s.c. function.

Figure 2.2: (a) shows one line segment above a convex (quadratic) function which lies completely in epi(F). (b) The l.s.c. function of Equation **??**. The left part of the function accesses the value at $(1, 1)$, but the right one does not.

Example 2.8 *1. Let $A \in \mathbb{R}^{m\times n}$ be a real matrix and $F : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ a linear function with $F(u) = Au$. Then F is convex and concave, hence linear functions are affine.*

Proof *Choose $u_1, u_2 \in \mathbb{R}, t \in [0,1]$. We get*

$$F(u_1t + u_2(1-t)) = F(u_1t) + F(u_2(1-t)) = F(u_1)t + F(u_2)(1-t)$$

by definition of linearity. ■

2. Let $A \in \mathbb{R}^{n\times n}$ be a real, symmetric, positiv definite matrix and $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ a linear function with $F(u) = \frac{1}{2}u^TAu + b^Tu + c$. Then F is strictly convex. So, for $t \in [0,1]$ we get

$$\begin{aligned}
F(u_1t + u_2(t-1)) &= \frac{1}{2}(u_1t + u_2(t-1))^TA(u_1t + u_2(t-1)) + b^T(u_1t + u_2(t-1)) + c \\
&= \frac{1}{2}\left(t^2u_1^TAu_1 + (t-1)^2u_2^TAu_2 + 2t(t-1)u_1^TAu_2\right) \\
&+ tb^Tu_1 + (t-1)b^Tu_2 + c + c - c \\
&= \underbrace{\frac{1}{2}t^2u_1^TAu_1}_{\leq t\frac{1}{2}u_1^TAu_1} + tb^Tu_1 + c \\
&+ \underbrace{\frac{1}{2}(t-1)^2u_2^TAu_2}_{\leq (t-1)\frac{1}{2}u_2^TAu_2} + (t-1)b^Tu_2 + c \\
&+ \underbrace{\underbrace{t(t-1)}_{<0}u_1^TAu_2 - c}_{<0} \\
&< F(u_1)t + F(u_2)(t-1). \qquad (2.10)
\end{aligned}$$

*3. Each norm $||\cdot|| : X \longrightarrow \mathbb{R}$ on a normed vector space X is convex (see also figure **??** for the l_1 and l_2 norms).*

Proof *Choose $u, v \in X, t \in [0,1]$, then*

$$||ut + v(1-t)||_X \overbrace{\leq}^{(*)} ||ut||_X + ||v(1-t)||_X \overbrace{=}^{(**)} ||u||_Xt + ||v||_X(1-t).$$

In $(*)$ we used the triangle inequality and in $(**)$ absolute homogeneity with the fact that $t \in [0,1]$. ■

Sometimes in literature you find another definition of convex functions. Therefore let us first introduce the epigraph and the domain of a function.

Definition 2.9 (Domain, Epigraph) *For any function $F : X \longrightarrow \mathbb{R}_\infty$, we define the domain*

$$\operatorname{dom}(F) = \{u \in X : F(u) < +\infty\}, \tag{2.11}$$

and the epigraph

$$\operatorname{epi}(F) = \{(u,t) \in X \times \mathbb{R} : t \geq F(u)\} \in \mathbb{R}^{n+1}. \tag{2.12}$$

Then we have the following definition for a convex function.

Definition 2.10 (Convex Function) *A function $F : X \longrightarrow \mathbb{R}_\infty$ is convex if* $\operatorname{dom}(F)$ *and* $\operatorname{epi}(F)$ *are convex sets.*

Definition 2.11 (Closed Function) *Let $F : X \longrightarrow \mathbb{R}_\infty$ be a function. Then F is closed if* $\operatorname{epi}(F)$ *is a closed set.*

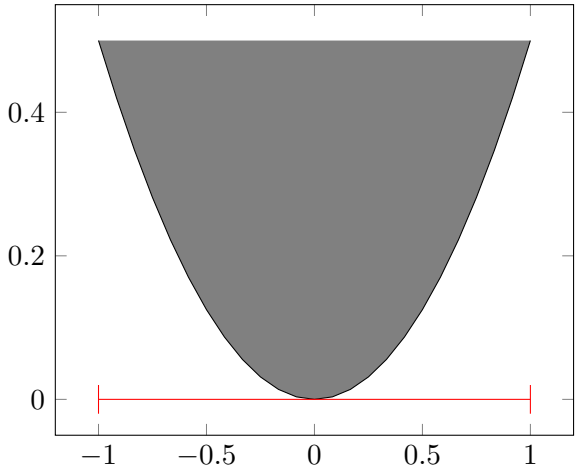


Figure 2.3: The domain (red) denoted as $\operatorname{dom}(F) = [-1,1]$ and epigraph $\operatorname{epi}(F) = \{(u,t) \in [-1,1] \times \mathbb{R} : t \geq \frac{1}{2}u^2\}$ (gray) of a function $F : [-1,1] \longrightarrow \mathbb{R}$ with $F(u) = \frac{1}{2}u^2$.

Now, that we are set up with convexity, we want to introduce an important concepts, which is used in this thesis.

Definition 2.12 (Legendre-Fenchel conjugate) *Let $F : X \longrightarrow \mathbb{R}_\infty$ be a convex function. We define the Legendre-Fenchel conjugate F^* of F for any $p \in X^*$ by*

$$F^*(p) = \sup_{u \in X} \big(\langle p, u\rangle - F(u)\big). \tag{2.13}$$

Remark 2.13

- *Without a proof we state that F^* - as the supremum of linear, continuous functions - is convex and lower-semicontinous, even F is not. In addition F^* is proper if F is convex and proper.*

- *In some literature the Legendre-Fenchel conjugate can also be found as convex conjugate. We use these two expressions equivalently.*

Theorem 2.14 *Let $F \in \Gamma_0(X)$, then $F^{**} = F$.*

The last theorem assures that we can rewrite Equation **??** to derive

$$F(u) = \big(F^*(p)\big)^*(u) = \sup_{p \in X^*} \big(\langle u, p\rangle - F^*(p)\big).$$

Example 2.15 *Let us view some examples on the Legendre-Fenchel conjugate.*

1. *The Legendre-Fenchel conjugate of the indicator function of a set $C \subseteq X$ is given by*
$$\delta_C^*(p) = \sup_{u \in C} \langle p, u\rangle - \delta_C(u) = \sup_{u \in C} \langle p, u\rangle,$$
which is the support function.

2. *Let $F : X \longleftarrow \mathbb{R}_\infty$ be an arbitrary function and $\alpha > 0$. Then the convex conjugate of $\alpha F(u)$ is given by*
$$F^*(p) = \alpha F^*(\frac{p}{\alpha}).$$
We compute
$$F^*(p) = \sup_{u \in X} \langle p, u\rangle - \alpha F(u) = \alpha\big(\underbrace{\sup_{u \in X} \langle \frac{p}{\alpha}, u\rangle - F(u)}_{= F^*(\frac{p}{\alpha})} \big) = \alpha F^*(\frac{p}{\alpha}),$$
which shows the desired equation.

3. *Now, let $||\cdot||$ be a norm on X, with dual norm $||\cdot||_*$ on X^*. We show that*
$$F^*(p) = \begin{cases} 0 & \text{if } ||p||_* \leq 1, \\ \infty & \text{else,} \end{cases} = \delta_{||p||_* \leq 1}(p),$$
i.e. the convex conjugate of a norm is the indicator function of the unit ball of its dual norm. We have
$$F^*(p) = \sup_{u \in X} (\langle p, u\rangle - ||u||).$$
First let $||p||_* > 1$. By definition of the dual norm ($||p||_* = \sup_{||x|| \leq 1} |\langle p, x\rangle|$) there is a $x \in \mathbb{R}^n$ for which we observe that $\langle p, x\rangle > 1$.
We take $u = tx$ and let $t \longrightarrow \infty$, then we have
$$\langle p, u\rangle - ||u|| = t(\langle p, x\rangle - ||x||) \longrightarrow \infty.$$
This shows that $F^*(p) = \infty$. On the other hand if $||p||_* \leq 1$ Cauchy-Schwarz-inequality assures that
$$\langle p, u\rangle \leq ||u||\,||p||_*$$

for all $u \in X$. But this implies

$$\langle p, u\rangle - ||u|| \leq ||u||\,||p||_* - ||u|| = ||u||\,(||p||_* - 1) \leq 0.$$

Because $||p||_* \leq 1$ holds, to get the supremum we need to choose $u = 0$. This shows that $F^* = \sup_{u\in X}\langle p, 0\rangle - ||0|| = 0$.

4. *As a last example we show that for a function $F(u) = \frac{1}{2}||u||^2$ its conjugate is $F^*(p) = \frac{1}{2}||p||_*^2$. With Cauchy-Schwarz-inequality we have $\langle p, u\rangle \leq ||p||_*\,||u||$. We observe*

$$\langle p, u\rangle - \frac{||u||^2}{2} \leq ||p||_*\,||u|| - \frac{||u||^2}{2},$$

where the righthand side is a quadratic function of $||u||$. Define $H(x) := -\frac{x^2}{2} + y\,x$, then the maximal value of H is $x = y$, since $H'(x) = -x + y$ and $H'(x) = 0$ implies $x = y$. Plugging $||u||$ into H we see, that the maximum is attained at $||p||_*$. It follows

$$\langle p, u\rangle - \frac{||u||^2}{2} \leq ||p||_*||p||_* - \frac{||p||_*}{2} = \frac{||p||_*^2}{2}.$$

This shows that $F^*(p) = \sup_{u\in X}\langle u, p\rangle - \frac{||u||_2^2}{2} \leq \frac{||p||_*^2}{2}$. Let us now proof the inequality in the other direction. We assume that we find an arbitrary u which satisfies $\langle u, p\rangle = ||u||\,||p||_*$. Further, we assume that $||u|| = ||p||_*$ where u is scaled so that the equation holds. Then we have

$$\langle p, u\rangle - \frac{||u||^2}{2} = ||u||\,||p||_* - \frac{||u||^2}{2} = \frac{||p||_*^2}{2},$$

but for this particular u it holds that

$$\langle p, u\rangle - \frac{||u||^2}{2} \leq \sup_{u\in X}\langle p, u\rangle - \frac{||u||^2}{2} = F^*(p).$$

From these two inequalities it follows that $F^*(p) = \frac{||p||_*^2}{2}$. ■

Another important property of functions is differentiability. Unfortunatelly, in some cases a function F is not differentiable everywhere. For this, we want to define the so called subdifferential and therefore the subgradient.

Definition 2.16 (Subgradient, Subdifferential) *Let X be an open, convex set and $F : X \longrightarrow \mathbb{R}_\infty$ a (convex) function. A vector y is called subgradient of F in $u_0 \in X$, if*

$$F(u) \geq F(u_0) + \langle y, u - u_0\rangle \;\; \forall u \in X. \tag{2.14}$$

The set

$$\partial F(u_0) = \{y \in X :\, F(u) \geq F(u_0) + \langle y, u - u_0\rangle \;\; \forall u \in \mathrm{dom}(F)\}$$

is called subdifferential of F in $u_0 \in X$ and $\mathrm{dom}(\partial F) = \{u : \partial F(u) \neq \emptyset\} \subset F$.

If F and G are differentiable we have $\partial(F+G) = \partial F + \partial G$. For the subdifferential this holds under some conditions. We want to give the corresponding Proposition without giving a proof. But, we state that in our computations this equality can always be used.

Proposition 2.17 *Let F, G be convex and assume $\mathrm{int}(\mathrm{dom}G) \cap \mathrm{dom}F \neq \emptyset$: then*

$$\partial(F+G) = \partial F + \partial G.$$

To illustrate the subgradient and subdifferential, respectively, we want to give some examples.

Example 2.18 *1. Take the absolute value function $F(u) = |u|$ in $\mathbb{R}$ which is defined by*

$$F(u) = \begin{cases} u & \text{if } u \geq 0, \\ -u & \text{else.} \end{cases}$$

Since $F(u)$ is not differentiable in 0, but on $\mathbb{R}/\{0\}$ we compute the subgradient y by

- *$\partial F(u) = 1$ if $u > 0$,*
- *$\partial F(u) = -1$ if $u < 0$,*
- *and finally*

$$F(0) + \langle y, (u-0)\rangle \leq F(u) \Longleftrightarrow \langle y, u\rangle \leq |u|.$$

If $u \geq 0$ this is equivalent to $y \geq 1$. If $u < 0$ we get

$$\langle y, u\rangle \leq |u| \Longleftrightarrow \langle y, u\rangle \leq -u \Longleftrightarrow y \geq -1.$$

Finally, we summarize

$$y = \begin{cases} 1 & \text{if } u > 0, \\ -1 & \text{if } u < 0, \\ [-1, 1] & \text{if } u = 0. \end{cases}$$

2. Let $F(u) = ||u||_1$. As a non-differentiable convex function we seek for the subdifferential. For that we can express the l_1-norm as

$$||u||_1 = |u_1| + ... + |u_n| = \max\big\{\langle y, u\rangle : p_i \in \{-1, 1\}\big\},$$

*for all $u \in X$. One gets $||p||_\infty \leq 1$. If we find a p such that $\langle p, u\rangle = ||u||_1$ then immediately inequality **??** would be satisfied, because*

$$F(u) \geq F(u_0) + \langle y, u\rangle \Longleftrightarrow ||u||_1 \geq 0 + \langle y, u\rangle = ||u||_1.$$

On the other hand, if we choose $p_i = -1$ if $u_i < 0$ and $p_i = 1$ if $u_i > 0$ (this was also what we got by calculating the subgradient of the absolute value function) we only have the case left where $u_i = 0$. In this case, we can choose both $p_i = -1$

or $p_i = 1$ *or equivalently looking at the non-differentiable point* $u_0 = 0$ *and if* $||y||_\infty \leq 1$*, we observe*

$$F(u) \geq F(0) + \langle y, (u-0)\rangle \Longleftrightarrow ||u||_1 \geq \langle y, u\rangle.$$

This means we have

$$y = \begin{cases} 1 & \text{if } u > 0, \\ -1 & \text{if } u < 0, \\ -1 \text{ or } 1 & \text{if } u = 0. \end{cases}$$

and

$$\partial F(u) = \big\{ y : ||y||_\infty \leq 1,\, \langle y, u\rangle = ||x||_1 \big\}. \tag{2.15}$$

The following Propositions are elementary for computations later. We provide them without giving a proof and refer to [**?**] and [**?**].

Proposition 2.19 *Let* $F \in \Gamma_0(X)$*. Then*

- *the set of minimizers* $\arg\min_{u \in X} F(u)$ *is convex (possibly empty).*
- *if* $\hat{u}$ *is a local minimum of* F*, then* $\hat{u}$ *is in fact a global minimum, i.e.*

$$\hat{u} \in \arg\min_{u \in X} F(u).$$

Proposition 2.20 *For any* F *convex,* $p \in \partial F(u)$ *if and only if*

$$\langle p, u\rangle - F(u) = F^*(p).$$

Moreover, if $F \in \Gamma_0$*, so that* $F^{**} = F$*, then this is equivalent to* $u \in \partial F^*(p)$*.*

Proposition 2.21 *Let* F *be convex, then* $\hat{u} \in \arg\min_{u \in X} F(u)$ *if and only if* $0 \in \partial F(\hat{u})$*.*

The three propositions from above are fundamental for our work. If we find a (local) minimizer of a convex optimization problem, we already know that this minimizer is the global optimum. The last proposition assures, that if we find a global minimum of our convex optimization problem, then we have that 0 always is an element of the subdifferential.
The next theorem can be found in [**?**] and is used extensively in the following chapters. One can find different editions of the theorem, see for instance [**?**]. We provide it without a proof, for which we also refer to [**?**]. But, first we need:

Definition 2.22 (Projection Operator, Proximity Operator, Moreau Envelop) *For any non-empty closed set* C *the projection operator for an arbitrary point* $z \notin C$ *on the set* C *is defined by*

$$\Pi_C(z) = \arg\min_{u \in C} \frac{1}{2}||z-u||_2^2, \tag{2.16}$$

meaning the orthogonal projection onto the convex set C. We define the Proximity Operator in a similar fashion due to

$$\text{prox}_F^\alpha(z) = \arg\min_{u\in X} \frac{1}{2}||u-z||_2^2 + \alpha\, F(u), \tag{2.17}$$

and the Moreau envelop as

$$M_F^\alpha(z) = \min_{u\in X} \frac{1}{2}||u-z||_2^2 + \alpha\, F(u), \tag{2.18}$$

for any $\alpha > 0$.

Let us shortly view two useful examples for the projection operator.

Example 2.23 *The l_p-Projection of a point $p \in \mathbb{R}^n$ onto the l_p-unit sphere with $p \notin C = \{u : ||u||_p \leq 1\}$ is given by the following minimization problem:*

$$\begin{array}{ll} \min\limits_{u\in C} & \frac{1}{2}||u-p||_2^2 \\ \text{subject to} & ||u||_p \leq 1. \end{array}$$

We are looking at the l_2-Projection (euclidean projection) and the l_∞-Projection, where both convex optimization problems have unique solutions.

1. *The solution for the euclidean projection is given by*

$$u^* = \begin{cases} p, & \text{if } ||p||_2 \leq 1 \\ \frac{p}{||p||_2}, & \text{otherwise.} \end{cases}$$

   Or equivalently one gets $\Pi_{l_2\leq 1}(u) = \frac{u}{\max(1,|u|)}$.

2. *For the l_∞-Projection we observe*

$$u_k^* = \begin{cases} p_k, & \text{if } |p_k| \leq 1 \\ 1, & \text{otherwise,} \end{cases}$$

   for $k = 1, ..., n$ or $\Pi_{l_\infty\leq 1}(u) = \min(1, \max(-1, u))$.

Theorem 2.24 (Moreau's Theorem) *Let $F \in \Gamma_0(X)$ and $F^*(p) = \sup\limits_{u\in X} \langle u, p\rangle - F(u)$ be its Legendre-Fenchel conjugate. Then*

$$M_F^\alpha(z) + M_{F^*}^\alpha(z) = \frac{1}{2}||z||^2,$$

i.e. for each $z \in \mathbb{R}^n$ one has

$$\min_{u\in X} \frac{1}{2}||u-z||^2 + \alpha\, F(u) + \min_{p\in X^*} \frac{1}{2}||p-z||^2 + \alpha\, F^*(p) = \frac{1}{2}||z||^2, \tag{2.19}$$

where both minima are uniquely attained. The unique vectors u and p for which the respective minima are attained for a given z are the unique vectors u and p such that

$$z = u + p, \quad p \in \partial F(u), \tag{2.20}$$

and they are given by

$$u = \text{prox}_F^\alpha(z), \quad p = \text{prox}_{F^*}^\alpha(z).$$

This theorem was first proven in 1965 by Jean J. Moreau and can be found in [**?**].

Remark 2.25
- *Note that Equation* **??** *is equivalent to* $u \in \partial F^*(p)$ *or* $F(u) + F^*(p) = \langle u, p \rangle$*, by Proposition* **??**.
- *Another way to denote the proximity operator is given by*

$$\operatorname{prox}_F^\alpha = (\mathrm{Id} + \alpha \partial F)^{-1}. \tag{2.21}$$

This notation can be derived by Proposition **??**. *Here, zero is an element of the subdifferential of Equation* **??**, *namely*

$$0 \in \partial\big(\alpha F(u) + \frac{1}{2}||u - z||_2^2\big) = \alpha\, \partial F(u) + u - z.$$

But this can be rewritten as

$$z \in \alpha\, \partial F(u) + u \Longleftrightarrow u = (\mathrm{Id} + \alpha\, \partial F)^{-1}(z).$$

Furthermore, using Equation **??** *one gets*

$$u = (\mathrm{Id} + \alpha\, \partial F)^{-1}(u) + \alpha \left(\mathrm{Id} + \frac{1}{\alpha}\partial F^*\right)^{-1}\left(\frac{u}{\alpha}\right). \tag{2.22}$$

*We refer to [**?**] and [**?**] for more information.*

Example 2.26 *To this end we briefly discuss one example for the proximity operator. If we set* $F(u) = \delta_C(u)$ *then we observe*

$$\operatorname{prox}_F^\alpha(\tilde{u}) = \min_{u \in C} \frac{||u - \tilde{u}||_2^2}{2} + \alpha\, \delta_C(u) = \min_{u \in C} \frac{||u - \tilde{u}||_2^2}{2} = \Pi_C(\tilde{u}).$$

Here, one can see the connection between the two operators. Proximation of the indicator function is the same as doing a euclidean projection onto the corresponding set C.

2.3 Total Variation

The definitions of the total variation, varition of functions and functions of bounded variation can be found in [**?**], together with the first example in this section.

Definition 2.27 (Total Variation) *Let* Ω *be an open subset of* $\mathbb{R}^n$. *For a function* $u \in L^1(\Omega)$, *the* total variation *of* u *in* Ω *is defined as*

$$\mathrm{TV}(u) = \sup\left\{ -\int_\Omega u \operatorname{div} \varphi \, dx : \varphi \in C_0^1(\Omega, \mathbb{R}^n), |\varphi(x)| \leq 1, \forall x \in \Omega \right\}. \tag{2.23}$$

Example 2.28 *Let $u \in W_1^1(\Omega)$, then integration by parts and the fact that φ having compact support leads to*

$$\begin{aligned} -\langle u, \operatorname{div} \varphi \rangle &= -\int_\Omega u \operatorname{div} \varphi \, dx \\ &= -\Big(\underbrace{\int_{\partial\Omega} u \, \varphi \, d\mathcal{H}^{n-1}}_{=0} - \int_\Omega \nabla u \, \varphi \, dx \Big) = \int_\Omega \nabla u \, \varphi \, dx \\ &= \langle \nabla u, \varphi \rangle, \end{aligned} \tag{2.24}$$

for every $\varphi \in C_0^1(\Omega, \mathbb{R}^n)$ so that

$$\mathrm{TV}(u) = \int_\Omega |\nabla f| dx. \tag{2.25}$$

*The idea to derive Equation **??** is that we first evaluate the case where $\mathrm{TV}(u) \leq \int_\Omega |\nabla f| dx$. This can easily be seen by using Cauchy-Schwarz-inequality*

$$\begin{aligned} \int_\Omega \nabla u \, \varphi \, dx &\leq \left| \int_\Omega \nabla u \, \varphi dx \right| dx \\ &\leq \int_\Omega |\nabla u| \underbrace{|\varphi|}_{\leq 1} dx \leq \int_\Omega |\nabla u| \, dx. \end{aligned} \tag{2.26}$$

For the inequality in the other direction the key idea would be to set $\varphi = \frac{\nabla u}{|\nabla u|}$. Then clearly, $\varphi = 1$. Since $\varphi \notin C_0^1$, we would approximate φ in L^1. This space lies close in the space of smooth functions with compact support and for that satisfies the assumptions.

Remark 2.29 *Note that $W_1^1(\Omega)$ denotes the L^1-Sobolev-Space of Ω, which means that $u \in L^1(\Omega)$ and $Du \in L^1(\Omega)$. Here Du is meant in the sense of distributional derivatives. From now on, we assume that all functions u are in $W_1^1(\Omega)$, where Ω is the image domain.Further note, that the total variation is convex and lower-semicontinous. See [**?**] for details.*

To get a better intuition about the Total Variation, we also want to give the defintion of the variation of functions $u \in [a, b] \subset \mathbb{R}$ with $a, b \in \mathbb{R}$ and $a < b$.

Definition 2.30 (Variation of a function) *Let $u : \mathbb{R} \longrightarrow \mathbb{R}$ and $a < b$ be real numbers. Then define the variation of u on $[a, b]$ as*

$$V_a^b(u) = \sup \left\{ \sum_{i=1}^n |u(x_i) - u(x_{i-1})| : m \in \mathbb{N} \text{ and } a = x_0 < x_1 < ... < x_m = b \right\}.$$

This definition is well known and suited for functions form $\mathbb{R}$ to $\mathbb{R}$. According to [**?**] this definition is ill-posed in the sense of measures and integrals. But to make variation a bit more vivid it fits perfectly.

Example 2.31 *1. Let $a, b \in \mathbb{R}$ with $a < b$. If $u : [a, b] \longrightarrow \mathbb{R}$ is monotonically increasing, then for any $a = x_0 < x_1 < ... < x_n = b$ we observe*

$$\begin{aligned} & \sum_{i=1}^{n} |u(x_i) - u(x_{i-1})| = \sum_{i=1}^{n} u(x_i) - u(x_{i-1}) \\ = \;& u(x_1) - u(x_0) + u(x_2) - u(x_1) + ... + u(x_{n-1}) - u(x_{n-2}) + u(x_n) - u(x_{n-1}) \\ = \;& u(x_n) - u(x_0) = u(b) - u(a). \end{aligned}$$

It follows that $V_a^b(u) = \sup\{u(b) - u(a)\} = u(b) - u(a)$.

2. Define the function $u : [0, 1] \longrightarrow \mathbb{R}$ by

$$u(x) = \begin{cases} 0, & \text{if } x = 0, \\ x\cos(\frac{\pi}{x}), & \text{if } x \neq 0. \end{cases} \tag{2.27}$$

This function is continuous and we have $V_a^b(u) = \infty$. To see this, consider for each $m \in \mathbb{N}$, the partition $P_m = \{0, \frac{1}{2m}, \frac{1}{2m-1}, ..., \frac{1}{3}, \frac{1}{2}, 1\}$. The values of u at the points of this partition are $u(P_m) = \{0, -\frac{1}{2m}, \frac{1}{2m-1}, ..., -\frac{1}{3}, \frac{1}{2}, -1\}$.

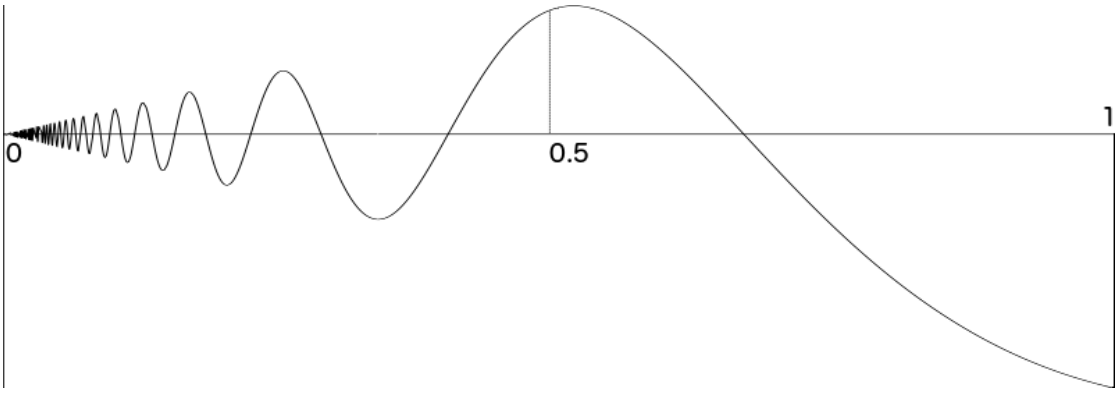


Figure 2.4: The plot of the function of equation **??**.

For this partition,

$$\begin{aligned} \sum_{i=1}^{n} |u(x_i) - u(x_{i-1})| &= |\frac{1}{2m} - 0| + |-\frac{1}{2m-1} - \frac{1}{2m}| + ... + |\frac{1}{2} + \frac{1}{3}| + |-1 - \frac{1}{2}| \\ &= \frac{1}{2m} + \frac{1}{2m-1} + \frac{1}{2m} + \frac{1}{2m-1} + ... + \frac{1}{2} + \frac{1}{3} + 1 + \frac{1}{2} \\ &= 2 * (\frac{1}{2m} + \frac{1}{2m-1} + ... + \frac{1}{2}) + 1. \end{aligned}$$

It is known, that the (harmonic) series $\sum_{k=2}^{\infty} \frac{1}{k}$ diverges. So given any m the partition P_m always ensures

$$V_a^b(u) = \sup\{2 * \sum_{k=2}^{\infty} \frac{1}{k} + 1\} = \infty.$$

Another concept which later is used in Chapter **??** are (special) functions of bounded variation. Even though we do not explicitely make use of this class of functions, we want to give the definition(s) for the sake of completeness. Further, (special) funcitons of bounded variation are strongly related to Total Variation as the following definition (see for instance [**?**]) ensures.

Definition 2.32 (Functions of bounded variation) *A function $u \in L^1(\Omega)$ is said to have bounded variation in Ω if $\mathrm{TV}(u) < \infty$. We define $BV(\Omega)$ as the space of all functions in $L^1(\Omega)$ with bounded variation. This space is equipped with the norm*

$$||u||_{BV} = ||u||_{L^1} + \int_\Omega |Du|,$$

and for that it is a Banach space.

As mentioned the Total Variation is the basis for this class of functions. Relating to example **??** we find that $u \in BV(\mathbb{R})$ in the first example, since we had that $V_a^b(u) = u(b) - u(a) < \infty$ for $a, b \in \mathbb{R}$, $a < b$. Furthermore u in the second example is not a function of bounded variation, because $V_a^b(u) = \infty$.
Additionally, BV-functions are proper and we have if $u \in BV(\Omega)$, then also $u \in \Gamma_0(\Omega)$, since the total varition is convex and l.s.c. (see for instance [**?**]).

Definition 2.33 (Special functions of bounded variation [?]) *SBV denotes the special functions of bounded variation, i.e. functions u of bounded variation for which the derivative Du is the sum of an absolutely continuous part $\nabla u \cdot dx$ and a discontinuous singular part S_u.*

3 A Primal-Dual Algorithm for (Convex) Saddle-Point Problems

3.1 The General Saddle-Point Problem

In this section we present the general class of problem we consider in this work. Therefore we define the map $K : X \longrightarrow Y$ as a continuous linear operator with induced norm

$$||K|| = \max\big\{||Kx||_Y : x \in X \text{ with } ||x||_X \leq 1\big\}. \tag{3.1}$$

Furthermore, we define the map $K^* : Y \longrightarrow X$ as the adjoint operator of K.
Now, let K be a linear operator, $u \in X$, $p \in Y$ and define $G : X \longrightarrow \mathbb{R}_+$ and $F^* : Y \longrightarrow \mathbb{R}_+$ where $G, F^* \in \Gamma_0$ and F^* being the Legendre-Fenchel conjugate of a convex, l.s.c. function F. We are trying to find the saddle point (u, p) of the following problem:

$$\min_{u\in X}\max_{p\in Y}\langle Ku, p\rangle + G(u) - F^*(p). \tag{3.2}$$

We call this problem also the primal-dual-problem, where u is the primal and p the dual variable. We define the corresponding primal-problem to this formulation by

$$\min_{u\in X} F(Ku) + G(u), \tag{3.3}$$

and the dual-problem by

$$\max_{p\in Y} -(G^*(-K^*p) + F^*(p)). \tag{3.4}$$

These three different classes of problems are equivalent, if F itself is convex. The Legendre-Fenchel conjugate assures, with $F(Ku) = \sup_{p\in Y}\langle p, Ku\rangle - F^*(p)$, that

$$\begin{aligned}
\min_{u\in X} F(Ku) + G(u) &= \min_{u\in X}\max_{p\in Y}\langle p, Ku\rangle - F^*(p) + G(u) \\
&= \min_{u\in X}\max_{p\in Y}\langle K^*p, u\rangle + G(u) - F^*(p) \\
&= \max_{p\in Y}\min_{u\in X}\langle K^*p, u\rangle + G(u) - F^*(p) \\
&= \max_{p\in Y} -\big(\max_{u\in X}\langle K^*p, u\rangle + G(u) - F^*(p)\big) \\
&= \max_{p\in Y} \underbrace{\max_{u\in X}\langle -K^*p, u\rangle - G(u)}_{=-(G^*(-K^*)p)} + F^*(p) \\
&= \max_{p\in Y} -(G^*(-K^*p) + F^*(p))
\end{aligned} \tag{3.5}$$

Note that we can swap min and max and exchange sup by max since we are acting on finite, bounded, normed vector spaces. It is equivalent to seek for the maximimum of a function u or instead seek for the minimum of the function $-u$.

Assuming that problem **??** has at least one solution, which we denote by $(\hat{u}, \hat{p}) \in X \times Y$, then $\hat{u}$ as the solution to the primal-problem satisfies

$$K\hat{u} \in \partial_p F^*(\hat{p}), \tag{3.6}$$

and $\hat{p}$ as the solution of the dual-problem satisfies

$$-(K^*\hat{p}) \in \partial_u G(\hat{u}), \tag{3.7}$$

where ∂_p and ∂_u, respectively, denote the subdifferential of the corresponding functions.

Proof If we define a function $L(u, p) := \langle K^*p, u\rangle - F^*(p) + G(u)$, we consider

$$\min_{u\in X} \max_{p\in Y} L(u, p).$$

Now, fixing the optimal value $\hat{p}$ over all p, we get the following minimzation problem

$$\min_{u\in X} L(u, \hat{p}).$$

According to Proposition **??** this is equivalent to $0 \in \partial\, L(\hat{u}, \hat{p})$, if $\hat{u}$ is a minimizer of this optimization problem. But, this leads us to equation **??**, since

$$0 \in \partial_u \left(\langle K^*\hat{p}, \hat{u}\rangle - F^*(\hat{p}) + G(\hat{u})\right) \Longleftrightarrow 0 \in K^*\hat{p} + \partial_u\, G(\hat{u}) \Longleftrightarrow -K^*\hat{p} \in \partial_u\, G(\hat{u}).$$

Doing the same by fixing the optimal value over all u, denoted by $\hat{u}$, we observe

$$\max_{p\in Y} L(\hat{u}, p) \Longleftrightarrow \min_{p\in Y} -L(\hat{u}, p).$$

Again, using Proposition **??** we get

$$0 \in \partial_p\, -\left(\langle \hat{p}, K\hat{u}\rangle - F^*(\hat{p}) + G(\hat{u})\right) \Longleftrightarrow 0 \in -K\hat{u} + \partial_p\, F^*(\hat{p}) \Longleftrightarrow K\hat{u} \in \partial_p F^*(\hat{p}),$$

which proves Equation **??**. ■

3.2 A First-Order Primal-Dual Algorithm

Our goal is to solve the saddle-point problem **??**. Therefore, one finds in a series of paper the celebrated (fast) first-order primal-dual algorithm. According to [**?**] the idea of this method of solving saddle-point problems in this manner goes back to Arrow and Hurwicz, see also [**?**]. For that reason these primal-dual approaches are sometimes also called Arrow-Hurwicz type methods. The first time this algorithm was stated in this framework was probably in [**?**]. The general idea of the proposed algorithm is to do a gradient descent in u, since this is the variable of the minimzation problem. And do,

simultaneously, a gradient ascent in p, because this is the variable of the maximization problem. Choosing time-steps $\sigma, \tau > 0$ one gets

$$p^{n+1} = (\mathrm{Id} + \sigma\, \partial\, F^*)^{-1}(p^n + \sigma\, K u^n)$$
$$u^{n+1} = (\mathrm{Id} + \tau\, \partial\, G)^{-1}(u^n - \tau\, K^* p^{n+1}).$$

The scheme was first proposed in a paper of Zhu and Chan in [**?**]. Unfortunatelly, there is no proof of convergence for it. This makes the approach poor. But in 2009 Pock, Cremers, Bischof and Chambolle published a paper, that we also consider in Chapter **??**, whose contribution was a provable extension of the above scheme. The idea here was to add an additional extrapolation step to the algorithm, as seen in line three of the primal-dual algorithm **??**. In [**?**] Pock and Chambolle generalized this algorithm. They also proposed some variations of the algorithm itself. Depending on the properties of the corresponding functions F^* and G one can derive a better convergence rate. We will not provide details and only make use of two of the proposed algorithms of [**?**]. In Chapter **??** we apply the them to different problems in imaging. Further, we will not provide a proof of convergence for these algorithms. For details we refer to [**?**] and [**?**]. The general fast primal-dual algorithm is as follows

Algorithm 3.1 (First-Order Primal-Dual Algorithm) *Choose $(u^0, p^0) \in X \times Y$ and let $\bar{u}^0 = u^0$. Further let $\tau, \sigma > 0$ with $\sigma\tau L^2 \leq 1$ and $\theta \in [0, 1]$. Then, we let for each $n \geq 0$*

$$\begin{cases} p^{n+1} = (\mathrm{Id} + \sigma\, \partial\, F^*)^{-1}(p^n + \sigma\, K \bar{u}^n) \\ u^{n+1} = (\mathrm{Id} + \tau\, \partial\, G)^{-1}(u^n - \tau\, K^* p^{n+1}) \\ \bar{u}^{n+1} = u^{n+1} + \theta(u^{n+1} - u^n). \end{cases} \tag{3.8}$$

These three lines provide a powerful tool. Computing $p^n + \sigma\, K\bar{u}^n$, respectively, $u^n - \tau\, K^* p^{n+1}$ is easy, since these two are sums of vectors. The last line of this scheme is again the summation of vectors. In the next section we introduce the notation for K and K^*. We will see that, computing the linear operator applied to u and p, is again a simple task. Computing the proximity operators in line one and two of the algorithm needs further work and varies within each model. We will compute the proximity operators for each model explicitly in each section.

Additionally, let us introduce the primal-dual gap, which is strongly related to the weak and strong duality theorems (found for instance in [**?**], which is a part of the convergence theorem **??**.

Definition 3.2 (Primal-Dual Gap) *Let $u \in X$, $p \in Y$ be the variables of the optimization problem in Equation **??**. Then we define the primal-dual gap of this problem by*

$$\mathcal{G}(u, p) = \max_{\tilde{p} \in Y} \langle \tilde{p}, Ku \rangle - F^*(\tilde{p}) + G(u) - \min_{\tilde{u} \in X} \langle p, K\tilde{u} \rangle - F^*(p) + G(\tilde{u}), \tag{3.9}$$

which has the property that $\mathcal{G}(u, p) \geq 0$ for all u, p and equality only holds if and only if (u, p) is a saddle-point. If $\hat{p}$ is a solution of the maximization problem and $\hat{u}$ a solution of the minimization problem the following inequality holds:

$$\mathcal{G}(u, p) \geq \langle \hat{p}, Ku \rangle - F^*(\hat{p}) + G(u) - \langle p, K\hat{u} \rangle - F^*(p) + G(\hat{u}) \geq 0$$

For this particular algorithm one can find a convergence theorem in [**?**], which we provide without a proof.

Theorem 3.3 *Let $L = ||K||$ and assume Equation* **??** *has a saddle-point $(\hat{u}, \hat{p})$. Choose $\theta = 1, \tau, \sigma, L^2 < 1$, and let $(u^n, \bar{u}^n, p^n)$ be defined by* **??**. *Then*

1. *For any n,*

$$\frac{||p^n - \hat{p}||^2}{2\sigma} + \frac{||u^n - \hat{u}||^2}{2\tau} \leq C\left(\frac{||p^0 - \hat{p}||^2}{2\sigma} + \frac{||u^0 - \hat{u}||^2}{2\tau}\right),$$

   where the constant $C \leq (1 - \tau\sigma L^2)^{-1}$.

2. *If we let $u^N = \left(\frac{\sum_{n=1}^{N} u^n}{N}\right)$ and $p^N = \left(\frac{\sum_{n=1}^{N} p^n}{N}\right)$, for any bounded $B_1 \times B_2 \subset X \times Y$ the restricted primal-dual gap has the following bound:*

$$\mathcal{G}_{B_1 \times B_2}(u^N, p^N) \leq \frac{D(B_1, B_2)}{N},$$

   where

$$D(B_1, B_2) = \sup_{(u,p) \in B_1 \times B_2} \frac{||u - u^0||^2}{2\tau} + \frac{||p - p^0||^2}{2\sigma}.$$

   Moreover, the weak cluster points of (u^N, p^N) are saddle-points of **??**.

3. *If the dimension of the spaces X and Y is finite, then there exists a saddle-point (u^*, p^*), such that $u^n \longrightarrow u^*$ and $p^n \longrightarrow p^*$.*

Remark 3.4 *What this theorem states is, that one needs to choose τ, σ carefully by initializing the algorithm. As long as the inequality $\tau\sigma L^2 < 1$ holds, convergence is guaranteed. The two parameters τ, σ are also called time-steps. The better the choice for these beforhand, the faster the algorithm converges. Unfortunatelly, the two time-step parameters also depend on the choice of the parameters of the underlying model, c.f. chapter* **??**. *Estimating the best time-steps is ongoing research. A first approach was a preconditioning scheme, published in [**?**]. Of course, it is not best practice having an algorithm which is dependent on manually choosen parameters, which further depend on other parameters. But, on the other hand, two parameters are highly controllable, with respect to the fact that σ can be computed from τ with $\sigma = \frac{1}{\tau * L^2}$. Other methods are almost as fast as the primal-dual algorithm, but depend on a couple of parameters, or they are independent of parameter choices and have a slow convergence rate.*

As mentioned, the proposed algorithm goes probably back to Arrow-Hurwicz. To derive their original algorithm one would only need to choose $\theta = 0$. Then the last line of algorithm **??** becomes $\bar{u}^{n+1} = u^n$. And for that, one can drop $\bar{u}$ and derive the Arrow-Hurwicz method. We state this for the sake of completeness but will not consider it in our computations.

The convergence rate for our primal-dual algorithm is $\mathcal{O}(\frac{1}{N})$. In the case that one function, F^* or G, is uniformly convex one gets the following variant of the primal-dual algorithm:

Algorithm 3.5 (F^* or G are uniformly convex) *Choose $(u^0, p^0) \in X \times Y$ and let $\bar{u}^0 = u^0$. Further let $\tau_0, \sigma_0, \gamma > 0$ with $\sigma\tau L^2 \leq 1$. Then, we let for each $n \geq 0$*

$$\begin{cases} p^{n+1} = (\mathrm{Id} + \sigma_n \, \partial \, F^*)^{-1}(p^n + \sigma_n \, K\bar{u}^n) \\ u^{n+1} = (\mathrm{Id} + \tau_n \, \partial \, G)^{-1}(u^n - \tau_n \, K^* p^{n+1}) \\ \theta_n = \frac{1}{\sqrt{1+2\gamma\tau_n}}, \; \tau_{n+1} = \theta_n \tau_n, \; \sigma_{n+1} = \frac{\sigma_n}{\theta_n} \\ \bar{u}^{n+1} = u^{n+1} + \theta_n(u^{n+1} - u^n). \end{cases} \tag{3.10}$$

This leads to a convergence rate of $\mathcal{O}(\frac{1}{N^2})$. These two types of convergence rates are called sublinear convergence. Fortunatelly, both primal-dual algorithm versions are highly parallelizable on a GPU using the CUDA framework. In Chapter **??** we provide in-depth details of this.

3.3 Discrete Setting

In our discrete setting we consider a regular pixel grid of size $N \times M$ and set

$$\mathcal{G} = \Big\{(i,j) : i = 1, ..., N \text{ and } j = 1, ..., M\Big\}, \tag{3.11}$$

where the indices (i, j) denote the discrete locations in the pixel grid. And we define an image $u \in X : \mathcal{G} \longrightarrow \mathbb{R}$ where $X \in \mathbb{R}^{N \cdot M}$ is a finite dimensional vector space equipped with the standard inner product

$$\langle u, v \rangle_X = u^T v = \sum_{i=1}^{N} \sum_{j=1}^{M} u_{i,j} v_{i,j}, \;\; u, v \in X, \tag{3.12}$$

and the standard euclidean norm

$$||u||_2 = \left(u^T u\right)^{\frac{1}{2}} = \langle u, u \rangle^{\frac{1}{2}}, \;\; u \in X.$$

Furthermore, let $Y = X \times X$ be the dual space of X, equipped with the inner product defined by

$$\langle p, q \rangle_Y = p^T q = \sum_{i=1}^{N} \sum_{j=1}^{M} p^1_{i,j} q^1_{i,j} + p^2_{i,j} q^2_{i,j},$$

with $p_{i,j} = \left(p^1_{i,j}, p^2_{i,j}\right)^T, q_{i,j} = \left(q^1_{i,j}, q^2_{i,j}\right)^T \in Y$ and also equipped with the euclidean norm.

For simplicity we let our normed vector space X be in $\mathbb{R}^n$ and swap the superscript $N \cdot M$ to n. If the exact space size is of importance, we will make this clear.

Because the total variation is associated with the gradient operator, we want to give some discretization for this operator. From Equation **??** we know that $\langle \nabla u, p \rangle = -\langle u, \text{div} p \rangle$ holds. In a finite dimensional vector space equipped with inner product one has $\langle \nabla u, p \rangle = \langle u, \nabla^T p \rangle$. Because of this, we set $\nabla^T = -\text{div}$. According to the linear operator K we set $K = \nabla$ and $K^* = \nabla^T$.

Definition 3.6 (Discrete gradient operator) *We define the discrete gradient of a vector $u \in X$ by $\nabla u = ((\partial_i u)_{i,j}, (\partial_j u)_{i,j})^T$ using forward differences with Neumann boundary conditions, i.e*

$$(\partial_i u)_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < N \\ 0 & \text{if } i = N \end{cases} \tag{3.13}$$

$$(\partial_j u)_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < M \\ 0 & \text{if } j = M \end{cases} \tag{3.14}$$

And the discrete divergence operator is defined in the following fashion:

Definition 3.7 (Discrete divergence operator) *We define the discrete divergence of a vector $p \in Y$ by $\nabla^T p = \partial_i p^1_{i,j} + \partial_j p^2_{i,j}$ using backward differences with Dirichlet boundary conditions, i.e*

$$(\partial_i p^1)_{i,j} = \begin{cases} p^1_{i,j} - p^1_{i-1,j} & \text{if } 1 < i < N \\ p^1_{i,j} & \text{if } i = 1 \\ -p^1_{i-1,j} & \text{if } i = N \end{cases} \tag{3.15}$$

$$(\partial_j p^2)_{i,j} = \begin{cases} p^2_{i,j} - p^2_{i,j-1} & \text{if } 1 < j < M \\ p^2_{i,j} & \text{if } j = 1 \\ -p^2_{i,j-1} & \text{if } j = M \end{cases} \tag{3.16}$$

We additionally compute the bound on the norm of these two operators.

Proposition 3.8 (Bound on the norm of ∇) *The bound on the norm of the proposed discrete linear operators is given by*

$$L^2 = ||\nabla|| = ||\text{div}|| \leq 8. \tag{3.17}$$

Proof With the definitions of **??**, **??** and **??** we get

$$\begin{aligned} L &= ||\nabla|| = \max_{||x||_X \leq 1} ||\nabla x||_Y \\ &= ||\mathrm{div}|| = \max_{||p||_Y \leq 1} ||\mathrm{div}\, p||_X \\ &= \max_{||p||_Y \leq 1} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(p^1_{i,j} - p^1_{i-1,j} + p^2_{i,j} - p^2_{i,j-1}\right)^2 \\ &\leq 4 \max_{||p||_Y \leq 1} \sum_{i=1}^{N} \sum_{j=1}^{M} (p^1_{i,j})^2 + (p^1_{i-1,j})^2 + (p^2_{i,j})^2 + (p^2_{i,j-1})^2 \\ &\leq 8 \max_{||p||_Y \leq 1} ||p||_Y^2 = 8. \end{aligned} \tag{3.18}$$

■

Since, there exists a couple of models to approximate an input image g by a function u, we want to make some conventions about this beforehand. Our models are based on the idea that a given image g consists of two things: data and noise. This can be expressed by

$$g = g_d + g_n,$$

where g_d denotes the data of the image and g_n the noise which should be removed.

An example model to efficiently remove Gaussian noise from an input image g would be the so called ROF model. Before introducing this model in detail, let us state one general property of all our underlying models. What they all have in common is the idea how they are set up:

$$\text{Model} = \text{Data Fidelity Term} + \text{Regularizer Term}.$$

The data fidelity term assures that the approximation u is as close to the input image g as possible. For instance, we set $G(u) = \frac{1}{2}||u - g||_2^2$ having the quadratic, euclidean distance as the data fidelity term. As a quadratic norm function this term is convex. For the regularizer we use convex terms, which are easy to handle, but also highly non-convex regularizers, like in the Mumford-Shah Functional, discussed in Chapter **??**.

3.4 The ROF Model

The first model we consider in this thesis is the ROF Model, named after Leonid I. Rudin, Stanley Osher and Emad Fatemi. They first proposed this model in 1992 in [**?**]. It is the prototype when talking about variational methods in image processing. Before we define this model in details, we define two important norms. The discrete isotropic total variation norm is given by

$$||\nabla u||_1 = \sum_{i=1}^{N} \sum_{j=1}^{M} |(\nabla u)_{i,j}|, \text{ where } |(\nabla u)_{i,j}| = \sqrt{((\nabla u)^1_{i,j})^2 + ((\nabla u)^2_{i,j})^2}. \tag{3.19}$$

Additionally, we define the discrete maximum norm by

$$||p||_\infty = \max_{i,j} |p_{i,j}|, \text{ where } |p_{i,j}| = \sqrt{(p^1_{i,j})^2 + (p^2_{i,j})^2}. \tag{3.20}$$

Definition 3.9 (ROF Model) *Let $\Omega \in \mathbb{R}^d$ be the d-dimensional image domain, $u \in W^1_1(\Omega)$ and $g \in L^1(\Omega)$ an (noisy) input image. Then the ROF model is defined as the minimization problem*

$$\min_u E_{ROF}(u) = \min_u \mathrm{TV}(u) + \frac{\lambda}{2}\int_\Omega |u-g|^2\,dx = \min_u \int_\Omega |\nabla u|\,dx + \frac{\lambda}{2}\int_\Omega |u-g|^2\,dx, \tag{3.21}$$

The appearing parameter λ is used to handle the tradeoff between the regularizer, namely the total variation, and the data fidelity term. Using a larger value for λ one gets an approximation u, which is closer to the input image g. If we choose λ to be small, then the weighting of the regularizer is higher and for that the approximation u is smoother.

Reformulation of Equation **??** into a discrete setting leads us to:

$$\min_{u\in X} E_{ROF}(u) = \min_{u\in X} ||\nabla u||_1 + \frac{\lambda}{2}||u-g||_2^2, \tag{3.22}$$

with $u \in X$ being the unknown approximation and $g \in X$ the given noisy data.

Remark 3.10 *In some literature, for instance [**?**], one finds a multiplicative factor h^2 in Equation **??**. This factor is due to discretization. Since we assume our image domain Ω having its pixel values as discrete locations, we do not make use of the additional factor in none of our models. Note, that the factor h^2 only rescales the energy function $E_{ROF}(u)$ and does not change the solution of the minimization problems.*

3.4.1 ROF Model as Saddle-Point Problem

Let us now rewrite the minimization problem in Equation **??** to derive a saddle-point formulation. For that, we set $F(\nabla u) = ||\nabla u||_1$ and $G(u) = \frac{\lambda}{2}||u-g||_2^2$. Hence, we are facing the following optimization problem

$$\min_{u\in X}\ F(\nabla u) + G(u) = \min_{u\in X}\ ||\nabla u||_1 + \frac{\lambda}{2}||u-g||_2^2. \tag{3.23}$$

Applying the Legendre-Fenchel conjugate, one has $F^{**}(\nabla u) = F(\nabla u) = \sup_{p\in Y}\langle p, \nabla u\rangle_Y - F^*(p)$, and we observe the saddle-point problem

$$\min_{u\in X}\max_{p\in Y}\ \langle p, \nabla u\rangle_X - F^*(p) + G(u) = \min_{u\in X}\max_{p\in Y}\ -\langle \nabla^T p, u\rangle_X - F^*(p) + G(u). \tag{3.24}$$

Now, it remains to show how $F^*(p)$ looks like. From Example **??** 2. and the fact that the conjugate norm of the l_1 norm is the l_∞ norm, we have

$$F^*(p) = \begin{cases} 0 & \text{if } ||p||_\infty \le 1, \\ \infty & \text{else,} \end{cases} \tag{3.25}$$

or equivalently $F^*(p) = \delta_P(p)$ for a set P given by

$$P = \big\{p \in Y : ||p||_\infty \leq 1\big\}. \tag{3.26}$$

Using this notation in the saddle-point problem leads us to

$$\min_{u\in X} \max_{p\in Y} \langle p, \nabla u\rangle_Y + \frac{\lambda}{2}||u-g||_2^2 - \delta_P(p). \tag{3.27}$$

We also want to propose the dual formluation of equation **??**. For that, we need to compute G^*.
Setting $v = u - g$, which is equivalent to $u = v + g$. Further,

$$\begin{aligned} G^*(p) &= \sup_{u\in X} \langle u, p\rangle - \frac{\lambda}{2}||u-g||_2^2 \\ &= \sup_{v\in X} \langle v+g, p\rangle - \frac{\lambda}{2}||v||_2^2 \\ &= \underbrace{\sup_{v\in X} \langle v, p\rangle - \frac{\lambda}{2}||v||_2^2}_{=\frac{1}{2\lambda}||p||_2^2 (*)} + \langle g, p\rangle \\ &= \frac{1}{2\lambda}||p||_2^2 + \langle g, p\rangle. \end{aligned}$$

In $(*)$ we used example **??** 2. and 4. Plugging F^* and G^*, respectively, into equation **??** and setting $K^* = -\nabla^T$ we get

$$\max_{p\in Y} -(G^*(-K^*p) + F^*(p)) = \max_{p\in Y} -\left(\frac{1}{2\lambda}||\nabla^T p||_2^2 + \langle g, \nabla^T p\rangle_X + \delta_P(p)\right). \tag{3.28}$$

which is the dual of the ROF model.

3.4.2 The Proximity Operators of the ROF Model

In Algorithm **??** we saw, how saddle-point problems can be solved efficiently. In two of the three steps in this scheme we needed to calculate the proximity operators $(\mathrm{Id} + \sigma\,\partial\, F^*)^{-1}$ and $(\mathrm{Id} + \tau\,\partial\, G)^{-1}$). Within the ROF Model one has $G(u) = \frac{\lambda}{2}||u-g||_2^2$ and $F^*(p) = \delta_P(p)$. Applying Equation **??** to F^* we get

$$(\mathrm{Id} + \sigma\,\partial\, F^*)^{-1}(\tilde{p}) = \arg\min_{p\in P} \frac{||p-\tilde{p}||_2^2}{2} + \sigma\,\delta_P(p) = \arg\min_{p\in P} \frac{||p-\tilde{p}||_2^2}{2}.$$

This is nothing but the euclidean projection of a vector $\tilde{p} \notin P$ onto the convex set P. From Example **??** 1. we have

$$p = (\mathrm{Id} + \sigma\,\partial\, F^*)^{-1}(\tilde{p}) = \Pi_P(\tilde{p}) \Longleftrightarrow p_{i,j} = \frac{\tilde{p}_{i,j}}{\max(1, |\tilde{p}_{i,j}|)}, \tag{3.29}$$

pointwise for all $i = 1, ..., N, j = 1, ..., M$. So, it only remains to compute the proximity operator for our function G. Here, we get

$$(\mathrm{Id} + \tau\, \partial\, G)^{-1}(\tilde{u}) = \arg\min_{u \in X} \frac{||u - \tilde{u}||_2^2}{2} + \frac{\tau\lambda}{2}||u - g||_2^2 = \arg\min_{u \in X} \mathcal{L}(u).$$

Since this minimization problem is convex, a local minimizer is also a global minimizer of this problem. So, let $\hat{u} \in \arg\min_{u \in X} \mathcal{L}(u)$. With proposition **??** we have

$$0 \in \partial\mathcal{L}(\hat{u}) \Longleftrightarrow 0 \in (\hat{u} - \tilde{u}) + \tau\lambda(\hat{u} - g) \Longleftrightarrow (1 + \tau\lambda)\hat{u} = \tilde{u} + \tau\lambda g,$$

and this implies $\hat{u} = \frac{\tilde{u} + \tau\lambda g}{1 + \tau\lambda}$.

Overall, the following equivalence holds

$$u = (\mathrm{Id} + \tau\, \partial\, G)^{-1}(\tilde{u}) \Longleftrightarrow u_{i,j} = \frac{\tilde{u}_{i,j} + \tau\lambda g}{1 + \tau\sigma} \tag{3.30}$$

for all $i = 1, ..., N$ and $j = 1, ..., M$ pointwise.

As mentioned in section **??**, each model has its proximity operators. Now, that we computed the versions for the ROF model, everything is set up and one can implement the primal-dual algorithm. Since the proximity operators of the ROF model have a very simple representation, all computations in the algorithm are highly parallelizable. We will discuss the implementation issues in chapter **??**.

3.5 The TVL1 Model

The idea within the TVL1 model is to replace the L_2 norm in the data fidelity term with the L^1 norm. This norm is more robust in removing the so called salt and pepper noise, meaning that it removes single pixels with extrem values white or black. The model is depicted as follows:

Definition 3.11 (The TVL1 Model) *Let $\Omega \in \mathbb{R}^d$ be the d-dimensional image domain, $u \in W_1^1(\Omega)$ and $g \in L^1(\Omega)$ an (noisy) input image. Then the TVL1 Model is defined as the optimization problem*

$$\min_u E_{TVL1}(u) = \min_u \mathrm{TV}(u) + \lambda \int_\Omega |u - g|\, dx = \min_{u \in X} \int_\Omega |\nabla u|\, dx + \lambda \int_\Omega |u - g|\, dx. \tag{3.31}$$

Note, that there is also the parameter λ to handle the tradeoff between both terms, as seen in the previous section for the ROF model. The primal formulation of the TVL1 model is represented by

$$\min_{u \in X} E_{TVL1}(u) = \min_{u \in X} ||\nabla u||_1 + \lambda||u - g||_1, \tag{3.32}$$

where u is the approximation of an input image g.

3.5.1 TVL1 as Saddle-Point Problem

We can rewrite this minimization problem into a saddle-point problem. Let us first state, that our function $F(\nabla u)$ remains the same as in the ROF model. We only have a change in G. Again using the Legendre-Fenchel conjugate for F - as in subsection **??** - we obtain

$$\min_{u\in X}\max_{p\in Y} \langle p, \nabla u\rangle_X - F^*(p) + G(u)$$

Writing F^* as the indicator function with $F^* = \delta_P(p)$ and the set P of Equation **??** we observe for the primal-dual formulation

$$\min_{u\in X}\max_{p\in Y} \langle p, \nabla u\rangle_Y + \lambda||u-g||_1 - \delta_P(p). \tag{3.33}$$

To give the representation of the dual-problem for the TVL1 model, we need to compute G^*. We find, that the Legendre-Fenchel conjugate of the function $G(u) = \lambda||u-g||_1$ is given by

$$G^*(q) = \begin{cases} \langle q, g\rangle & \text{if } ||q||_\infty \le \lambda, \\ \infty & \text{else,} \end{cases} \tag{3.34}$$

which means nothing but $G^*(q) = \delta_Q(q) + \langle q, g\rangle$ for a set

$$Q = \big\{q \in X : ||q||_\infty \le \lambda\big\}. \tag{3.35}$$

Proof To derive this representation of the conjugate function we set $z = u - g$, which is equivalent to $u = z + g$. Then with the definition of the Legendre-Fenchel conjugate we get

$$\begin{aligned} G^*(q) = \sup_{u\in X}\langle q, u\rangle - G(u) \Longleftrightarrow G^*(q) &= \sup_{z\in X}\langle q, z+g\rangle - G(z+g) \\ &= \sup_{z\in X}\langle q, z+g\rangle - \lambda||z||_1 \\ &= \sup_{z\in X}\frac{1}{\lambda}\big(\langle q, z\rangle + \underbrace{\langle q, g\rangle}_{=\text{const}}\big) - ||z||_1 \\ &= \left(\sup_{z\in X}\langle\frac{1}{\lambda}q, z\rangle - ||z||_1\right) + \langle q, g\rangle. \end{aligned} \tag{3.36}$$

Using again example **??** 2. and the fact that the conjugate norm of the l_1 norm is the l_∞ norm, gives us

$$G^*(q) = \begin{cases} \langle q, g\rangle & \text{if } ||\frac{1}{\lambda}q||_\infty \le 1, \\ \infty & \text{else,} \end{cases} \Longleftrightarrow G^*(q) = \begin{cases} \langle q, g\rangle & \text{if } ||q||_\infty \le \lambda, \\ \infty & \text{else.} \end{cases}$$

Or equivalently $G^*(q) = \delta_Q(q) + \langle q, g\rangle$. ■

We obtain the dual formulation of the TVL1 Model as

$$\max_{p\in Y} -(G^*(-K^*p) + F^*(p)) = \max_{p\in Y} -\Big(\delta_Q(\nabla^T q) + \langle\nabla^T q, g\rangle + \delta_P(p)\Big). \tag{3.37}$$

The Proximity Operators of the TVL1 Model

The proximity operator of F^* remains the same as in subsection **??**. We get again

$$p = (\mathrm{Id} + \sigma\,\partial\,F^*)^{-1}(\tilde{p}) = \Pi_P(\tilde{p}) \Longleftrightarrow p_{i,j} = \frac{\tilde{p}_{i,j}}{\max(1, |\tilde{p}_{i,j}|)}, \tag{3.38}$$

for all $i = 1, ..., N, j = 1, ..., M$.

To compute the proximity operator of the function G we get

$$(\mathrm{Id} + \tau\,\partial\,G)^{-1}(\tilde{u}) = \arg\min_{u \in X} \frac{||u - \tilde{u}||_2^2}{2} + \lambda\tau||u - g||_1.$$

Therefore, we define $\mathcal{L}(u) = \frac{||u-\tilde{u}||_2^2}{2} + \lambda\tau||u - g||_1$ and assume that $\hat{u} \in \arg\min_{u \in X} \mathcal{L}(u)$. Then with proposition **??** we have that

$$0 \in \partial\mathcal{L}(\hat{u}) \Longleftrightarrow 0 \in \hat{u} - \tilde{u} + \tau\lambda\partial(||\hat{u} - g||_1).$$

In example **??** 2. we already saw that the l^1 norm is not differentiable everywhere. To compute the subgradient we need the partial derivatives for each $k = 1, ..., N \cdot M$. Let us first take a look at the k-th row of $\partial\mathcal{L}(\hat{u})$:

$$\begin{aligned} 0 \in \partial_k\mathcal{L}(\hat{u}) &\iff 0 \in \hat{u}_k - \tilde{u}_k + \tau\lambda\partial_k(||\hat{u} - g||_1) \\ &\iff 0 \in \hat{u}_k - \tilde{u}_k + \tau\lambda\partial_k(|\hat{u}_k - g_k|). \end{aligned}$$

With this we see that we need to compute the subgradient of the absolute value function for all partial subderivatives $k = 1, ..., N \cdot M$. From example **??** 1. we have

$$y_k = \begin{cases} 1 & \text{if } \hat{u}_k - g_k > 0, \\ -1 & \text{if } \hat{u}_k - g_k < 0, \\ [-1, 1] & \text{if } \hat{u}_k - g_k = 0 \Longleftrightarrow \hat{u}_k = g_k, \end{cases}$$

where y_k is the subgradient of the k-th row. Let us check all three cases:

1. Let $y_k = 1$. Then we obtain

   $$0 \in \hat{u}_k - \tilde{u}_k + \tau\lambda \Longleftrightarrow \hat{u}_k = \tilde{u}_k - \tau\lambda.$$

   With $\hat{u}_k - g_k > 0$ and $\hat{u}_k = \tilde{u}_k - \tau\lambda$ we have that this equation holds if

   $$\tilde{u}_k - \tau\lambda - g_k > 0 \Longleftrightarrow \tilde{u}_k - g_k > \tau\lambda.$$

2. Now, let $y_k = -1$. The we get in row k

   $$0 \in \hat{u}_k - \tilde{u}_k - \tau\lambda \Longleftrightarrow \hat{u}_k = \tilde{u}_k + \tau\lambda.$$

   Rewriting the constraint in the definition of the subgradient leads us to

   $$\tilde{u}_k + \tau\lambda - g_k > 0 \Longleftrightarrow \tilde{u} - g_k > \tau\lambda.$$

3. Finally, $y_k \in [-1,1]$. Since, we know that $\hat{u}_k = g_k$ we get

$$0 \in g_k - \tilde{u}_k + \tau\lambda y_k \Longleftrightarrow \tilde{u}_k - g_k = \tau\lambda y_k.$$

We apply the absolute value function to each side of the equation, then with $|y_k| \leq 1$ we have

$$|\tilde{u}_k - g_k| = |\tau\lambda y_k| \leq \tau\lambda.$$

Overall, we observe for all $i = 1, ..., N$, $j = 1, ..., M$

$$u = (\mathrm{Id} + \tau\,\partial\, G)^{-1}(\tilde{u}) \Longleftrightarrow u_{i,j} = \begin{cases} \tilde{u}_{i,j} - \tau\lambda & \text{if } \tilde{u}_{i,j} - g_{i,j} > \tau\lambda, \\ \tilde{u}_{i,j} + \tau\lambda & \text{if } \tilde{u}_{i,j} - g_{i,j} < -\tau\lambda, \\ g_{i,j} & \text{if } |\tilde{u}_{i,j} - g_{i,j}| \leq \tau\lambda. \end{cases} \tag{3.39}$$

Note, that we used the discrete locations of the pixel (i, j) in this representation instead of the row-wise notation k. As we discussed earlier, these two notations are totally consistent.

3.6 The Mumford-Shah Functional

We saw that solving the ROF Model and TVL1 Model, respectively, depended mainly on finding the right choices of the convex conjugates and the proximity operator. Now, we introduce a highly non-convex functional. The problem of the proposed method for this kind of functional is, that there is - to date - no proof of convergence for it. Even though, one can see that it yields to high quality solutions. But, of course one wants to assure that convergence is given. Therefore, we discuss a second method to minimize the Mumford-Shah functional in the next chapter. For now, let us first give the definition of this functional.

Definition 3.12 (The Mumford-Shah Functional [?]) *Let $\Omega \subset \mathbb{R}^2$ be a rectangular image domain. In order to approximate an input image $g : \Omega \longrightarrow \mathbb{R}$ in terms of a piecewise smooth function $u : \Omega \longrightarrow \mathbb{R}$, the Mumford-Shah Functional is given by*

$$E(u) = \int_\Omega (u-g)^2 dx + R(u) = \int_\Omega (u-g)^2 dx + \lambda \int_{\Omega\setminus K} |\nabla u|^2 dx + \nu|K|, \tag{3.40}$$

where $\nu, \lambda > 0$ are weighting parameters, $K = K_1 \cup ... \cup K_N$ and $|K|$ denotes the length of the curves in K.

Remark 3.13 *In this section we follow the representation of the Mumford-Shah Functional of [?]. This is equivalent to Equation **??**, but has some advantages for our proposed method. In Chapter **??** we will state a second notation.*

This functional differs from the ones in the previous sections. The first term, the data fidelity term, remains the same as in the ROF or TVL1 Model. The approximation u should be as close to the input image g as possible. The regularizer consists of two terms. The first term of the also called Mumford-Shah regularizer, uses again the gradient, but not in the set Ω itself. Instead it states that the approximation u is not allowed to change too much in sets $\Omega \setminus K_k$. We call K_k the discontinouty sets (or jump sets) for all $k = 1, ..., P$ and curves u_k. The gradient is only taken into account in exactly these regions. The discontinouties of the sets K_k are also measured and taken into account in the energy $E(u)$. This means, that all curves K_k should be regular in the sense of measure theory. Here, we find two weighting parameters ν and λ. Where λ handles the tradeoff between the first term of the regularizer and the data fidelity term, ν controls the length of the discontinouty set K. A smaller ν yields to a smoother image, where a higher ν leads to sharper edges in our images. The parameter λ yet plays another important role. If one chooses a λ small enough, then our model is also called piecewise-smooth Mumford-Shah Model. On the other hand, in the limiting case $\lambda \longrightarrow \infty$, we can only attain a minimum if we set $\nabla u = 0$ in $\Omega \setminus K$. Then the model is known as the piecewise-constant Mumford-Shah Model. In [**?**] Strekalovskiy and Cremers proposed to rewrite this functional in a discrete setting by first defining the discrete regularizer function by

$$R_{MS}(f) = \min(\lambda ||f||_2^2, \nu). \tag{3.41}$$

Then the discrete Mumford-Shah Model can be expressed by

$$\min_{u \in X} E_{MS}(u) = \min_{u \in X} ||u - g||_2^2 + \sum_{i=1}^{N} \sum_{j=1}^{M} R_{MS}((\nabla u)_{i,j}). \tag{3.42}$$

According to [**?**] the idea behind this formulation is to model the discontinouity set K explicitly in terms of the function u. This means, that K is the set of all points where the minimum in **??** attains ν. In other words, if the gradient $(\nabla u)_{i,j}$ is large enough, we have for the explicit set K_{MS}:

$$K_{MS} = \left\{ (i,j) \in \Omega : |(\nabla u)_{i,j}|^2 \geq \sqrt{\frac{\nu}{\lambda}} \right\}. \tag{3.43}$$

We can check that for a point $(i,j) \in K_{MS}$ we observe

$$R_{MS}(\nabla u_{i,j}) = \min(\underbrace{\lambda |(\nabla u)_{i,j}|^2}_{\geq \lambda \sqrt{\frac{\nu}{\lambda}}^2 = \nu}, \nu) = \nu$$

and if $(i,j) \notin K_{MS}$ we have

$$R_{MS}(\nabla u_{i,j}) = \min(\underbrace{\lambda |(\nabla u)_{i,j}|^2}_{< \lambda \sqrt{\frac{\nu}{\lambda}}^2 = \nu}, \nu) = \lambda |(\nabla u)_{i,j}|^2.$$

Remark 3.14 *In the piecewise-constant case, where $\lambda \longrightarrow \infty$, equation* **??** *changes to*

$$R_{MS}(f) = \begin{cases} \nu & \text{if } f \neq 0, \\ 0 & \text{else.} \end{cases} \tag{3.44}$$

3.6.1 Mumford-Shah as Saddle-Point Problem

Again, we try to formulate the Mumford-Shah Model as a saddle-point problem to be able to apply the second primal-dual algorithm of section **??**. In the sense of our notations from the previous section we have

$$\min_{u\in X} F(\nabla u) + G(u) = \min_{u\in X} \sum_{i=1}^{N}\sum_{j=1}^{M} R_{MS}((\nabla u)_{i,j}) + ||u-g||_2^2 \tag{3.45}$$

This is the primal formulation for the discrete Mumford-Shah Model. In the earlier sections we could easily apply Legendre-Fenchel conjugate on the function F, since these functions where convex and for that we had that $F^{**} = F$. Unfortunatelly, the function F, namely R_{MS} is non-convex in this model. We could still compute the convex conjugate, but would not be able to compute an appropriate proximity operator for R_{MS}^*. Because of this reason, we assume that we consider a model with an arbitrary convex regularizer R. We consider

$$\min_{u\in X} \sum_{i=1}^{N}\sum_{j=1}^{M} R((\nabla u)_{i,j}) + ||u-g||_2^2. \tag{3.46}$$

Applying now the Legendre-Fenchel conjugate on R we get the primal-dual formulation by

$$\min_{u\in X}\max_{p\in Y} \sum_{i=1}^{N}\sum_{j=1}^{M} \langle p_{i,j}, (\nabla u)_{i,j}\rangle - R^*(p_{i,j}) + G(u).$$

Since, we will not compute R^* directly, we do not provide the dual formulation to this problem. Let us now compute the proximity operators.

3.6.2 The Proximity Operators of the Mumford-Shah Model

First we compute the proximity operator for the function G. We can partially adapt it from equation **??**, because the underlying data fidelity term is the same as in the ROF Model, excluding the scaling factor $\frac{\lambda}{2}$. For a function $\mathcal{L}(u) = \frac{||u-\tilde{u}||_2^2}{2} + \tau\frac{||u-g||_2^2}{2}$ we had

$$\left(\mathrm{Id} + \tau\partial G\right)^{-1}(\tilde{u}) = \arg\min_{uinX} \mathcal{L}(u) \Longleftrightarrow 0 \in \partial\mathcal{L}(\hat{u}),$$

if $\hat{u} \in \arg\min_{u\in X} \mathcal{L}(u)$. From this characterization it follows

$$0 \in \hat{u} - \tilde{u} + 2\tau(\hat{u} - g) \Longleftrightarrow (1+2\tau)\hat{u} = \tilde{u} + 2\tau g \Longleftrightarrow \hat{u} = \frac{\tilde{u} + 2\tau g}{1+2\tau}.$$

We then get pointwise for all $i = 1, ..., N$ and $j = 1, ..., M$

$$u = \left(\mathrm{Id} + \tau\,\partial\, G\right)^{-1}(\tilde{u}) \Longleftrightarrow u_{i,j} = \frac{\tilde{u}_{i,j} + 2\tau g_{i,j}}{1+2\tau}. \tag{3.47}$$

As we originally consider the regularizer R_{MS} and want to take it into account in the proximity operator, we compute the proximity operator for $R^*(p)$ by using Moreau's Identity **??**, which means we have

$$\left(\mathrm{Id} + \sigma\partial R^*\right)^{-1}(\tilde{p}) = \tilde{p} - \sigma\left(\mathrm{Id} + \frac{1}{\sigma}\partial R\right)^{-1}\left(\frac{\tilde{p}}{\sigma}\right). \tag{3.48}$$

Then we can plug the original function R_{MS} into this formula and observe the proximal operator to our problem.

Let us start by computing $\left(\mathrm{Id} + \gamma\partial R_{MS}\right)^{-1}(\tilde{x})$ for some parameter $\gamma > 0$. Since the minimum function in R_{MS} can attain two different values we need to do a case differentiation:

1. Assume that $R_{MS}(x) = \nu$. And set $\mathcal{L}(x) = \frac{||x-\tilde{x}||_2^2}{2} + \gamma\nu$. Then if $\hat{x} \in \arg\min_{x\in X} \mathcal{L}(x)$ we have

$$\begin{aligned} x = \left(\mathrm{Id} + \gamma R_{MS}\right)^{-1}(\tilde{x}) \quad &= \quad \arg\min_{x\in Y} \frac{||x-\tilde{x}||_2^2}{2} + \gamma\nu \\ &\iff \quad 0 \in \partial\left(\frac{||\hat{x}-\tilde{x}||_2^2}{2} + \gamma\nu\right) \\ &\iff \quad 0 \in \hat{x} - \tilde{x} \\ &\implies \quad \hat{x} = \tilde{x}. \end{aligned}$$

2. On the other hand, if $R_{MS}(x) = \lambda||x||_2^2$ we set $\mathcal{L}(x) = \frac{||x-\tilde{x}||_2^2}{2} + \gamma\lambda||x||_2^2$. Again, if $\hat{x} \in \arg\min_{x\in X} \mathcal{L}(x)$ we observe

$$\begin{aligned} x = \left(\mathrm{Id} + \gamma R_{MS}\right)^{-1}(\tilde{x}) \quad &= \quad \arg\min_{x\in Y} \frac{||x-\tilde{x}||_2^2}{2} + \gamma\lambda||x||_2^2 \\ &\iff \quad 0 \in \partial\left(\frac{||\hat{x}-\tilde{x}||_2^2}{2} + \gamma\lambda||\hat{x}||_2^2\right) \\ &\iff \quad \hat{x} - \tilde{x} + 2\gamma\lambda\hat{x} = 0 \\ &\iff \quad \hat{x}(1 + 2\gamma\lambda) = \tilde{x} \\ &\implies \quad \hat{x} = \frac{\tilde{x}}{(1+2\gamma\lambda)} \end{aligned}$$

It is left to show, for which case the energy in the proximity operator is minimal. By definition of R_{MS} the following inequality holds:

$$\min_{x\in Y} \frac{||x-\tilde{x}||_2^2}{2} + \gamma\lambda||x||_2^2 \leq \min_{x\in Y} \frac{||x-\tilde{x}||_2^2}{2} + \gamma\nu. \tag{3.49}$$

In the first case, where the minimum is attained for $x = \tilde{x}$, we get

$$\min_{x\in Y} \frac{||\tilde{x}-\tilde{x}||_2^2}{2} + \gamma\nu = \gamma\nu.$$

But if we need to set $x = \frac{\tilde{x}}{(1+2\gamma\lambda)}$ to attain the minimal value, as in the second case, we have

$$\begin{aligned}
\frac{||\frac{\tilde{x}}{1+2\gamma\lambda} - \tilde{x}||_2^2}{2} + \gamma\lambda||\frac{\tilde{x}}{1+2\gamma\lambda}||_2^2 &= \frac{\frac{||2\gamma\lambda\tilde{x}}{1+2\gamma\lambda}||_2^2}{2} + \gamma\lambda\frac{1}{(1+2\gamma\lambda)^2}||\tilde{x}||_2^2 \\
&= \frac{4\gamma^2\lambda^2}{2(1+2\gamma\lambda)^2}||\tilde{x}||_2^2 + \frac{\gamma\lambda}{(1+2\gamma\lambda)^2}||\tilde{x}||_2^2 \\
&= \frac{2\gamma^2\lambda^2+\gamma\lambda}{(1+2\gamma\lambda)^2}||\tilde{x}||_2^2 \\
&= \frac{(1+2\gamma\lambda)\gamma\lambda}{(1+2\gamma\lambda)^2}||\tilde{x}||_2^2 \\
&= \frac{\gamma\lambda}{1+2\gamma\lambda}||\tilde{x}||_2^2. \qquad (3.50)
\end{aligned}$$

Together with inequality **??** we conclude

$$\begin{aligned}
&\frac{\gamma\lambda}{1+2\gamma\lambda}||\tilde{x}||_2^2 \leq \gamma\nu \\
\iff \quad &\frac{\lambda}{1+2\gamma\lambda}||\tilde{x}||_2^2 \leq \nu \\
\iff \quad &||\tilde{x}||_2^2 \leq \frac{\nu}{\lambda}(1+2\gamma\lambda) \\
\iff \quad &||\tilde{x}||_2 \leq \sqrt{\frac{\nu}{\lambda}(1+2\gamma\lambda)} \qquad (3.51)
\end{aligned}$$

The proximity operator of the function R_{MS} is therefore given by

$$x = \left(\mathrm{Id} + \gamma R_{MS}\right)^{-1}(\tilde{x}) \iff x_{i,j} = \begin{cases} \frac{1}{1+2\gamma\lambda}\tilde{x}_{i,j} & \text{if } ||\tilde{x}_{i,j}||_2 \leq \sqrt{\frac{\nu}{\lambda}(1+2\gamma\lambda)} \\ \tilde{x}_{i,j} & \text{else.} \end{cases} \qquad (3.52)$$

This holds pointwise for all $i = 1, ..., N$ and $j = 1, ..., M$.

To derive the representation of the proximal operator for R_{MS}^* we now plug the operator of equation **??** into Moreau's identity formula of equation **??**. Again, we need to distinguish two cases.

1. Assume that $||\tilde{p}||_2 > \sqrt{\frac{\nu}{\lambda}(1+2\gamma\lambda)}$. Then

$$\sigma\left(\mathrm{Id} + \frac{1}{\sigma}\partial R_{MS}\right)^{-1}\left(\frac{\tilde{p}}{\sigma}\right) = \sigma\frac{\tilde{p}}{\sigma} = \tilde{p}.$$

And we observe

$$\begin{aligned}
p = \left(\mathrm{Id} + \sigma\partial R_{MS}^*\right)^{-1}(\tilde{p}) &= \tilde{p} - \sigma\left(\mathrm{Id} + \frac{1}{\sigma}\partial R_{MS}\right)^{-1}\left(\frac{\tilde{p}}{\sigma}\right) \\
&= \tilde{p} - \tilde{p} = 0.
\end{aligned}$$

2. Now, we assume that $||\tilde{p}||_2 \leq \sqrt{\frac{\nu}{\lambda}(1+2\gamma\lambda)}$. This leads to

$$\begin{aligned} p = \left(\mathrm{Id} + \sigma\partial R^*_{MS}\right)^{-1}(\tilde{p}) &= \tilde{p} - \sigma\left(\mathrm{Id} + \frac{1}{\sigma}\partial R_{MS}\right)^{-1}\left(\frac{\tilde{p}}{\sigma}\right) \\ &= \tilde{p} - \sigma\frac{\frac{\tilde{p}}{\sigma}}{1+2\frac{1}{\sigma}\lambda} \\ &= \tilde{p} - \frac{\sigma}{\sigma+2\lambda}\tilde{p} \\ &= \left(1 - \frac{\sigma}{\sigma+2\lambda}\right)\tilde{p} \\ &= \left(\frac{\sigma+2\lambda-\sigma}{\sigma+2\lambda}\right)\tilde{p} \\ &= \frac{2\lambda}{\sigma+2\lambda}\tilde{p}. \end{aligned} \tag{3.53}$$

As we did for the proximity operator for R_{MS}, we also need to show for which condition these two cases hold. Using equation **??**, inequality **??** and the fact that we have $\gamma = \frac{1}{\sigma}$ gives us

$$\begin{aligned} & \frac{\frac{1}{\sigma}\lambda}{1+2\frac{1}{\sigma}\lambda}||\tilde{p}||_2^2 \leq \frac{1}{\sigma}\nu \\ \iff & \frac{\lambda}{\sigma+2\lambda}||\tilde{p}||_2^2 \leq \sigma\nu \\ \iff & ||\tilde{p}||_2^2 \leq \frac{\nu}{\lambda}\sigma(\sigma+2\lambda) \\ \iff & ||\tilde{p}||_2 \leq \sqrt{\frac{\nu}{\lambda}\sigma(\sigma+2\lambda)}. \end{aligned} \tag{3.54}$$

Overall, the proximity operator for R^*_{MS} is defined by

$$p = \left(\mathrm{Id} + \sigma\partial R^*_{MS}\right)^{-1}(\tilde{p}) \iff p_{i,j} = \begin{cases} \frac{\lambda}{\lambda+\sigma}\tilde{p}, & \text{if } ||\tilde{p}||_2 \leq \sqrt{\frac{\nu}{\lambda}\sigma(\sigma+2\lambda)}, \\ 0 & \text{else,} \end{cases} \tag{3.55}$$

for all $i = 1, ..., N$, $j = 1, ..., M$.

Despite the fact, that there is no convergence theorem for using algorithm **??** together with the computed proximal operators, one can show the boundedness for u^n in the case $\lambda < \infty$.

Proposition 3.15 *The sequence (u^n, p^n) generated by algorithm **??** is bounded and thus compact for $\lambda < \infty$, for instance it has a convergent subsequence.*

This proposition can be found in [**?**], along with a proof in the supplementary material.

In this chapter we provided a framework to solve variational methods, as well as a possibility to minimize the Mumford-Shah Functional. Applying this framework to real

images and compare the results is then part of chapter **??**. But first, we want to show a second way to minimize the Mumford-Shah Functional, in which we have a proof of optimiality for the solution.

4 A Primal-Dual Algorithm for Minimizing the Mumford-Shah Functional

As we revisit the Mumford-Shah Functional, we also rewrite Definition **??** to be consistend with [**?**], which is the publication we mainly follow in this chapter. If other results are taken into account, we make this clear.

Definition 4.1 (Mumford-Shah Functional) *Let $\Omega \subset \mathbb{R}^2$ be a rectangular image domain. In order to approximate an input image $f : \Omega \longrightarrow [0,1]$ in terms of a piecewise smooth function $u : \Omega \longrightarrow [0,1]$, Mumford and Shah suggested to minimize the functional*

$$E(u) = \lambda \int_\Omega (f-u)^2 dx + \int_{\Omega \setminus S_u} |\nabla u|^2 dx + \nu \mathcal{H}^{n-1}(S_u), \tag{4.1}$$

where $\lambda, \nu > 0$ are weighting parameters, $S_u = S_u^{n-1} \cup ... \cup S_u^N$ and $\mathcal{H}^1(S_u)$ denotes the n-1-dimensional Hausdorff-measure of the curves in S_u.

The difference to Chapter 3 is, that we interchanged the set $K = K_1 \cup ... \cup K_N$ to $S_u = S_u^1 \cup ... \cup S_u^N$ and instead of computing $|K|$, the length of the curves, we use the more general notation of measure theory. In the case, where $u \in \Omega \subset \mathbb{R}^2$, the n-1-dimensional Hausdorff-measure of S_u is nothing but $|S_u|$. Another difference to definition **??** is the parameter λ, which handles the tradeoff between the data fidelity term and the first term in the regularizer. It is swapped and controls now the term where the image f enters the functional. Swapping this parameter does not change the energy at all. So both definitions are totally consistent. As mentioned in the previous chapter, this functional is highly non-convex. The idea to derive a convex saddle-point representation of the Mumford-Shah functional is to make use of some results presented by Alberti, Bouchitte and DalMaso. They applied in (papers...) convex relaxation techniques to approximate the Mumford-Shah Functional by a convex maximization problem. To get this representation, let us first start with introducing some functions.

4.1 Convex Relaxation

In order to state a convex relaxed model, we need to introduce the characteristic (level set) function.

Definition 4.2 *Let $\Omega \subset \mathbb{R}^2$ denote the image plane and let $u \in SBV(\Omega)$[1]. The upper level sets of u are denoted by the characteristic function $\mathbb{1}_u : \Omega \times \mathbb{R} \longrightarrow \{0,1\}$ of the subgraph of u:*

$$\mathbb{1}_u(x,t) = \begin{cases} 1, & \text{if } t < u(x), \\ 0, & \text{else.} \end{cases} \tag{4.2}$$

In figure **??** one can see an example with a characteristic function in $\mathbb{R}^2$ for a function $u \in SBV(\Omega)$[1]. So, for a one-dimensional signal $u(x)$ the characteristic function becomes a two-dimensional function. It extends the original from with one dimension. In our case, as we consider two-dimensional images, the characteristic function adds another label space and we then act in a cube.

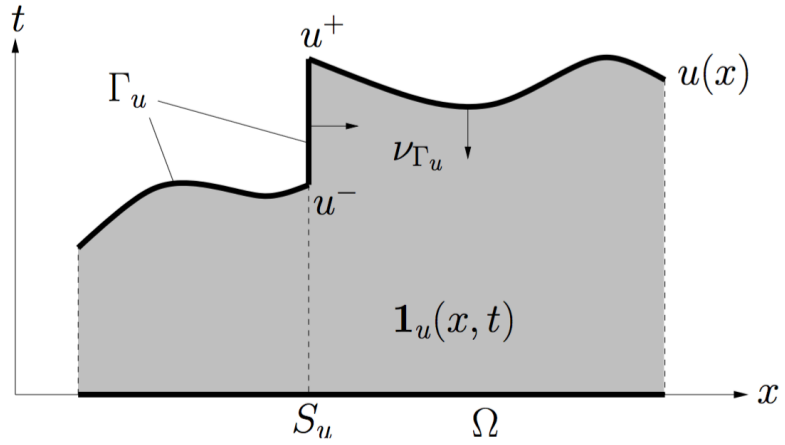


Figure 4.1: This picture (taken from [**?**]) shows the characteristic function $\mathbb{1}_u(x,t)$ of a function $u(x) \in SBV$.

The gray shaded area is the part where $\mathbb{1}_u(x,t) = 1$. Otherwise, the characteristic function is set to zero. The upper interface of the gray domain is denoted by Γ_u. This set is the set which holds all parts of the function $u(x)$ and each curve S_u which connects the points u^- and u^+. These two points are the last and the first point, respectively, of each partial curve of $u(x)$. Additionally, the notation ν_{Γ_u} denotes the normals on Γ_u.

Using $\mathbb{1}_u(x,t)$, one can find in [**?**] and [**?**] the proposed method of Alberti, Bouchitte and DalMaso along with a proof of this result. We state this theorem in the fashion of [**?**]. We provide it without giving a proof.

Theorem 4.3 (Convex Relaxation of the Mumford-Shah Functional) *For a function $u \in SBV(\Omega)$ the Mumford Shah functional can be written as*

$$E(u) = \sup_{\varphi \in K} \int_{\Omega \times \mathbb{R}} \varphi D\mathbb{1}_u, \tag{4.3}$$

with a convex set

$$K = \Big\{ \varphi \in C^0(\Omega \times \mathbb{R}, \mathbb{R}^2) \quad : \quad \varphi^t(x,t) \geq \frac{\varphi^x(x,t)^2}{4} - \lambda(t - f(x))^2, \tag{4.4}$$

$$\left| \int_{t_1}^{t_2} \varphi^x(x,s) ds \right| \leq \nu \Big\}, \tag{4.5}$$

where the inequalities in the definition of K hold for all $x \in \Omega$ and for all $t, t_1, t_2 \in \mathbb{R}$.

Remark 4.4
- *We used in the theorem, that we will represent a vector $\varphi \in K \subseteq \mathbb{R}^n$ by $\varphi(x,t) = \left(\varphi^x, \varphi^t\right)^T$, where $\varphi(x,t) \in \mathbb{R}^n$, $\varphi^x \in \mathbb{R}^{n-1}$ and $\varphi^t \in \mathbb{R}$.*
- *The idea to approximate the Mumford-Shah Functional is, to maximize the flux of a vector field φ through the interface Γ_u. The advantage of this technique is, to get a convex representation of the Mumford-Shah Functional and for this we derive in the following a convex saddle-point formulation. Then we can again make use of the primal-dual algorithm to solve this optimization problem.*

Our goal is to minimize **??**. We have

$$\min_u E(u) = \min_u \left(\sup_{\varphi \in K} \int_{\Omega \times \mathbb{R}} \varphi D \mathbb{1}_u \right) \tag{4.6}$$

To compute the minimizer of this formulation we first need to substitute $\mathbb{1}_u$ by a generic function

$$v(x,t) : \Omega \times \mathbb{R} \longrightarrow [0,1] \text{ and } \lim_{t \to -\infty} v(x,t) = 1, \quad \lim_{t \to +\infty} v(x,t) = 0. \tag{4.7}$$

Overall, we are going to face the following convex optimization problem:

$$\min_{v \in [0,1]} \sup_{\varphi \in K} \langle v, D\varphi \rangle = \min_{v \in [0,1]} \sup_{\varphi \in K} \int_{\Omega \times \mathbb{R}} \varphi D v. \tag{4.8}$$

According to [**?**] there is no proof, that finding an optimal pair (v^*, φ^*) for the optimization problem **??** leads to the global minimizer of equation **??**. Further, they state that only if v^* is binary one gets indeed the global minimum of the Mumford-Shah Functional.

In the next section we will consider this optimization problem **??** in the discrete version. Additionally, we propose the corresponding operators and compute the proximity operators - which then will be euclidean projections.

4.2 Discrete Setting

Using the characteristic function, respectively the function v means, that we are adding an additional space to our two dimensional image domain. This extra label space needs to be considered in the discrete setting. In [**?**] they consider $\Omega = [0,1]^2$ and for that the subgraph of u to be in $[0,1]^3$. This would imply that we discetize these two spaces by adding a step-size h, where for instance $h_N = \frac{1}{N}$ or $h_M = \frac{1}{M}$. Further, they consider all their operators without having this additional step-size. To be not confusing, we propose all our spaces and operators without needing a step size. Then the image domain is $\Omega = \{1, 2, ..., N\} \times \{1, 2, ..., M\}$ and the subgraph of the function $u : \mathbb{R}^2 \longrightarrow [0,1]$ is defined in the cube $\Omega \times \{1, 2, ..., S\}$. In this discrete setting we define the pixel grid $\mathcal{G}$ with size $N \times M \times S$ and the following notation

$$\mathcal{G} = \left\{ (i,j,k) : i = 1,2,...,N, j = 1,...,M, k = 1,2,...,S \right\} \tag{4.9}$$

where i, j, k are the discrete locations of each voxel. For a reformulation of **??** we also need to define the corresponding functions of v, φ. So, let $u \in X : \mathcal{G} \longrightarrow [0,1]$ and $p \in Y : \mathcal{G} \longrightarrow \mathbb{R}^3$ be the discrete versions of the continous functions in equation **??** where u corresponds to v and p to φ. If we replace the inner-product for infinite dimensions in equation **??** by the inner-product for finite dimensions and note that in finite, bounded normed vector spaces we can interchange sup and max we are going to face the saddle-point problem

$$\min_{u \in C} \max_{p \in K} \langle Au, p \rangle. \tag{4.10}$$

This notation looks now familiar to us. Here, A is the linear operator, earlier denoted with K. The set C of the minimization is defined as

$$C = \{ u \in X : u(i,j,k) \in [0,1], u(i,j,1) = 1, u(i,j,S) = 0 \}. \tag{4.11}$$

To take the limits of the function v into account in its discrete version u, we set the values in the first label space to 1 and those in the last label space to 0. We also stated, that the approximation u maps into $[0,1]$, which is modeled in the set C. The discrete version of the set K from equation **??** has the following notation:

$$K = \{ p = (p^x, p^t)^T \in Y \quad : \quad p^t(i,j,k) \geq \frac{||p^x||_2^2}{4} - \lambda(\frac{k}{S} - f(i,j))^2, \tag{4.12}$$

$$\left| \sum_{k_1 \leq k \leq k_2} p^x \right| \leq \nu \}, \tag{4.13}$$

whereas we define $p^x := (p^1, p^2)^T$ and $p^t := p^3$ and the vector p itself is an element of $\mathbb{R}^{N \cdot M \cdot S \cdot 3}$. For this, $p^x \in \mathbb{R}^{N \cdot M \cdot S \cdot 2}$ and $p^t \in \mathbb{R}^{N \cdot M \cdot S}$. The constraint in equation **??** goes pointwise for all $(i,j,k) \in \mathcal{G}$. The second constraint is more involved, since the constraint in **??** holds for all $i = 1,...,N$, $j = 1,...,M$ and all possible combinations (k_1, k_2) for all $k_1, k_2 = 1,...,S$. What looks like having a set K with two constraints, turns out that the set K is an intersection of a couple of convex sets. Namely, one has that for a fixed voxel (i,j,k) one can compute $\frac{S(S-1)}{2} + S + 1$ many convex sets. The amount of several convex sets will lead us to a long run-time in our algorithm. Before we discuss this in detail, we first continue with the definitions of the discrete setting.

Remark 4.5 *In addition to discretize the variable t in **??** one gets $\frac{k}{S}$ in the discrete version of **??**. Note, that t is a value in the continuous setting which determines at which point the characteristic function vanishes, i.e. is set to zero. The bound on the norm L depends on the discrete gradient operator. In [?] they proposed to set the discrete version of t to $\frac{k}{L}$, which is a mistake.*

We want to define the representation of the linear operator A is represented. It is the same as found in section **??**, namely the discrete gradient operator, but extended to the additional label space.

Definition 4.6 (Discrete gradient operator) *We define the discrete gradient of* $u \in X$ *by* $\nabla u = ((\partial_i u)_{i,j}, (\partial_j u)_{i,j}, (\partial_k u)_{i,j})^T$ *using forward differences with Neumann boundary conditions, i.e*

$$(\partial_i u)_{i,j} = \begin{cases} u_{i+1,j,k} - u_{i,j,k} & \text{if } i < N \\ 0 & \text{if } i = N \end{cases} \quad (\partial_j u)_{i,j,k} = \begin{cases} u_{i,j+1,k} - u_{i,j,k} & \text{if } j < M \\ 0 & \text{if } j = M \end{cases}$$

$$(\partial_k u)_{i,j,k} = \begin{cases} u_{i,j,k+1} - u_{i,j,k} & \text{if } k < S \\ 0 & \text{if } k = S \end{cases}$$

Again we have $\text{div} : Y \longrightarrow X$ as the discrete divergence operator. And it relates to ∇ with $\nabla^T = -\text{div}$ as seen before.

Definition 4.7 (Discrete divergence operator) *We define the discrete divergence of* $p \in Y$ *by* $\nabla^T p = \partial_i p^1_{i,j,k} + \partial_j p^2_{i,j,k} + \partial_k p^3_{i,j,k}$ *using backward differences with Dirichlet boundary conditions, i.e*

$$(\partial_i p^1)_{i,j,k} = \begin{cases} p^1_{i,j,k} - p^1_{i-1,j,k} & \text{if } 1 < i < N \\ p^1_{i,j,k} & \text{if } i = 1 \\ -p^1_{i-1,j,k} & \text{if } i = N \end{cases} \quad (\partial_j p^2)_{i,j,k} = \begin{cases} p^2_{i,j,k} - p^2_{i,j-1,k} & \text{if } 1 < j < M \\ p^2_{i,j,k} & \text{if } j = 1 \\ -p^2_{i,j-1,k} & \text{if } j = M \end{cases}$$

$$(\partial_k p^3)_{i,j,k} = \begin{cases} p^3_{i,j,k} - p^3_{i,j,k-1} & \text{if } 1 < k < S \\ p^3_{i,j,k} & \text{if } k = 1 \\ -p^3_{i,j,k-1} & \text{if } k = S \end{cases}$$

Proposition 4.8 (Bound on the norm of ∇) *The bound on the norm of the proposed discrete linear operator is given by*

$$L^2 = ||\nabla|| = ||\text{div}|| \leq 12. \tag{4.14}$$

Proof The proof is the same as in section **??** by adding the additional discretization variable $p^3_{i,jk}$. ■

4.3 Primal and Dual Formulation and the Proximity Operators

Recalling equation **??** and setting $A = \nabla$, we have that a minimizer of the Mumford-Shah Functional can be computed by solving

$$\min_{u \in C} \max_{p \in K} \langle \nabla u, p \rangle. \tag{4.15}$$

As in the chapter before we first want to formulate this saddle-point problem in the primal and the dual version. Afterwards, we want to compute the proximity operators for the corresponding functions.

We can write equation **??** in a slightly different way, since we can simply add the indicator functions of the corresponding sets C and K. We observe

$$\min_{u\in C}\max_{p\in K}\langle\nabla u, p\rangle - \delta_K(p) + \delta_C(u). \tag{4.16}$$

But then, we immediately can determine our functions F^* and G, which are given by

$$F^*(p) = \delta_K(p) \text{ and } G(u) = \delta_C(u).$$

With this, we first want to take a look at the primal formulation of the Mumford-Shah model. Therefore, it is left to compute F from F^*. Assume the function $F(g)$ to be convex, then the Legendre-Fenchel conjugate is given by

$$F^*(p) = \sup_{u\in C}\langle g, p\rangle - F(g).$$

For F convex we observe on the other hand

$$F(g) = \sup_{p\in Y}\langle g, p\rangle - F^*(p) = \sup_{p\in Y}\langle g, p\rangle - \delta_K(p).$$

We want to compute $F(g)$ and show that the function F is indeed convex. Computing F means solving

$$\sup_{p\in Y}\langle g, p\rangle - \delta_K(p) = \sup_{p\in K}\langle g, p\rangle.$$

We see that, since K is finite dimensional, we can change the supremum to the maximum and observe

$$\max_{p\in K}\langle g, p\rangle \Longleftrightarrow 0 \in \partial\,(g, p) \Longleftrightarrow 0 \in g \Longleftrightarrow g = 0.$$

But this means nothing that, if $g = 0$ then $F(g) = 0$. If $g < 0$ to attain a maximal value the only choice to let p go to $-\infty$. But if $g > 0$ then $p \longleftarrow \infty$. For an arbitray argument of F, we have

$$F(u) = \begin{cases} 0 & \text{if } u = 0, \\ \infty & \text{else.} \end{cases}$$

This is nothing but the indicator function of a single point. Then for the primal formulation we get

$$\min_{u\in C} F(\nabla u) + G(u) \iff \min_{u\in C} G(u) = 0 \tag{4.17}$$

$$\text{s.t.} \quad \nabla u = 0. \tag{4.18}$$

Finally, we also want to evaluate the dual formulation of the Mumford-Shah saddle-point problem. The dual of the saddle-point problem was given by

$$\max_{p\in Y} -(G^*(-K^*p) + F^*(p)).$$

We need to compute the Legendre-Fenchel conjugate of G. Since this is the indicator function of the set C we can make use of Example **??** 1. and see that $G^*(p) = \delta_C^*(p) = \sup_{u \in C} \langle p, u \rangle$, which is the support function of C. Overall, we obtain the dual problem by

$$\max_{p \in Y} -(G^*(-K^*p) + F^*(p)) = \max_{p \in K \subset Y} -(\langle \nabla^T p, u \rangle + \delta_K(p)) = \max_{p \in K} -\langle \nabla^T p, u \rangle. \quad (4.19)$$

It is left to compute the proximity operators. Since, the proximal operator for a indicator function is a euclidean projection onto the corresponding set, we need to describe how a projection on C and K can be derived. For that we want to rewrite our primal-dual algorithm to be consistent with [**?**]. We observe

Algorithm 4.9 *Choose $(u^0, p^0) \in C \times K$ and let $\bar{u}^0 = u^0$. We choose $\tau, \sigma > 0$. Then, we let for each $n \geq 0$*

$$\begin{cases} p^{n+1} = \Pi_K(p^n + \sigma K \bar{u}^n) \\ u^{n+1} = \Pi_C(u^n - \tau K^* p^{n+1}) \\ \bar{u}^{n+1} = 2u^{n+1} - u^n. \end{cases} \quad (4.20)$$

Here we denote Π_K and Π_C as the (euclidean) projections on the sets K and C. How we compute such projections, especially onto the set K, will be discussed in the next section.

4.4 Projection onto the convex sets and Dykstra's projection algorithm

In this section we start with the projection onto the set C and go on by stating an algorithm to project onto the convex set K.

4.4.1 Projection onto C

The projection onto C can efficiently be computed. By definition of the proximity operator we have

$$u = \arg\min_{u \in C} \frac{||u - \tilde{u}||_2^2}{2} + \tau \delta_C(u) = \arg\min_{u \in C} \frac{||u - \tilde{u}||_2^2}{2}.$$

Assume that $\tilde{u} \in C$. Then the best choice for u is of course $u = \tilde{u}$ itself, since the energy is then equal to zero, for which the quadratic function is minimal. On the other hand, if $\tilde{u} \notin C$, the euclidean distance to $C = [0, 1]$ is to clip $\tilde{u}$ onto the bound of C. This means if $\tilde{u} < 0$ then the shortest distance from $\tilde{u}$ to C is to set $u = 0$. Reversely, if $\tilde{u} > 1$ then the euclidean distance to C is given by setting $u = 1$. This idea is also illustrated in figure **??**. For an arbitrary intervall $[a, b]$ we have the following algorithm:

Algorithm 4.10 (Clipping) *The projection of a vector $u \in \mathbb{R}^n$ onto the intervall $[a, b]$, where $a, b \in \mathbb{R}$ and $a < b$ is given pointwise by*

$$u_i^{n+1} = \min\{b, \max\{a, u_i^n\}\} \quad (4.21)$$

for all $i = 1, ..., n$.

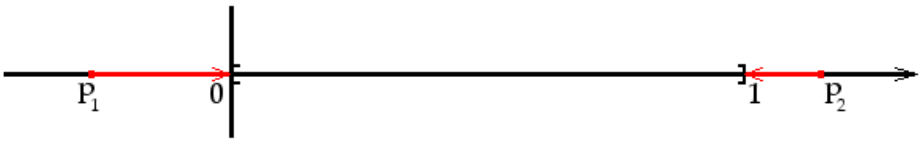


Figure 4.2: A point $p_1 < 0$ is clipped to 0, whereas a point $p_2 > 1$ is clipped to 1.

Remark 4.11

- *By projecting onto C, we also need to take care about the limits in* **??**. *For that we set $u(i,j,1) = 1$ and $u(i,j,S) = 0$ in each projection. Or similarly, using for instance the programming language C++, we have $u(i,j,0) = 1$ and $u(i,j,S-1) = 0$.*

- *In the case of our framework - acting in a discrete cube $\mathcal{G}$ - equation* **??** *can be regarded as*

$$u_{i,j,k}^{n+1} = \min\{1, \max\{0, u_{i,j,k}^{n}\}\}.$$

4.4.2 The projection onto K

Projecting onto K is more involved than the projection onto C, since K takes into account local and non-local constraints. In other words, the set K is an intersection of several convex sets. The projection onto the intersection of convex sets can be done by Dykstra's projection algorithm. The idea behind this algorithm is to project onto each set alternatingly in a step n and store the error, which is made in this iteration step. Before the projection in the $n+1$-th step is done, the vector which should be projected is reduced by the error made in the previous step. To fully understand this scheme, we first give the definition which was first proposed by Boyle and Dykstra in [**?**], where one can also find a proof of convergence. Afterwards, we will provide and discuss the algorithm and additionally show an example.

Algorithm 4.12 *Consider P convex sets with $\mathbb{R}^n \ni X = X_1 \cap X_2 \cap ... \cap X_P$. Let Π_i denote the projection onto the i-th set for $i = 1, ..., P$. And let $u_c \in \mathbb{R}^n$ be the current estimate with $u_c \notin X$, $u_i^k \in \mathbb{R}^n$ for $i = 0, ..., P$ and $v_i^k \in \mathbb{R}^n$ for $i = 1, ..., P$ and $k = 1, 2, ...$, where k denotes the number of iterations. Then the algorithm finds the (only) $u^* \in X$, such that*

$$||u^* - u_c||^2 \leq ||u - u_c||^2 \ \ \forall u \in X.$$

For $k = 1, 2, ...$ set $u_P^0 = u_c$ and $v_i^0 = 0$ for all $i = 1, ..., P$. Then iterate until convergence (e.g. $||u_0^k - u_P^k||_2 \leq \varepsilon$ with ε small):

$$\begin{aligned} u_0^k = \ & u_P^{k-1}, \\ \text{for} \ & i = 1, 2, ..., P: \\ & u_i^k = \Pi_i(u_{i-1}^k - v_i^{k-1}), \\ & v_i^k = u_i^k - (u_{i-1}^k - v_i^{k-1}). \end{aligned}$$

Theorem 4.13 (Convergence) *The sequence u_0^k in algorithm* **??** *converges to the (only) point $u \in X$.*

As mentioned, the proof for the theorem can be found in [**?**].

Now, that we know how the projection onto the entire set K can be implemented, we need to discuss how a projection onto the subsets of K can be computed. First let us give a decomposition of K into two several sets.

4.4.3 Decomposition of K

We will decompose the set K into sets K_p and K_{nl}, where the first resembles the local constraint, or more precisely, a parabola constraint and the second corresponds to the non-local constraint. Overall, we have $K = K_p \cap K_{nl}$ with

$$K_p = \left\{ p^t(i,j,k) \geq \frac{||p^x(i,j,k)||^2}{4} - \lambda(\frac{k}{M} - f(i,j))^2 \right\} \ \forall i,j,k \tag{4.22}$$

where $p^t(i,j,k) = p^3(i,j,k)$ and $p^x(i,j,k) = (p^1(i,j,k), p^2(i,j,k))^T$. For the non-local constraint we have

$$K_{nl} = \left\{ \left| \sum_{k_1 \leq k \leq k_2} p^x(i,j,k) \right| \leq \nu \right\} \ \forall i,j,k_1 \leq k \leq k_2. \tag{4.23}$$

We will now deduce the projection on these two sets. Let us start with the projection onto the parabola.

4.4.4 Projection onto K_p

Since the projection onto the set K_p is pointwise we want to drop the indices (i,j,k). Note that we do not necessarily need that a p^x is an element of $\mathbb{R}^2$. The following derivation holds for a larger class of problems namely having $p^x \in \mathbb{R}^n$. Let $\alpha > 0$, $p^x \in \mathbb{R}^n$, $p^t \in \mathbb{R}$ and $p = (p^x, p^t)^T \in \mathbb{R}^n \times \mathbb{R}$. Assume that $p_0^t < \alpha ||p_0^x||_2^2$ holds for a point $p_0 \in \mathbb{R}^n \times \mathbb{R}$. Then the projection of p_0 onto the parabola $\alpha ||p_0^x||_2^2$ can be written as the following minimization problem

$$\begin{aligned} \min_{p \in \mathbb{R}^n \times \mathbb{R}} \quad & \frac{1}{2}||p - p_0||_2^2 \\ \text{subject to} \quad & p^t \geq \alpha ||p^x||_2^2 \end{aligned}$$

To find the solution of this optimization problem we introduce a Lagrange Multiplier $\mu \in \mathbb{R}$ and define the Lagrangian as

$$\mathcal{L}(p,\mu) = \frac{(p-p_0)^2}{2} - \mu\Big(p^t - \alpha ||p^x||_2^2\Big). \tag{4.24}$$

We are seeking to minimize $\mathcal{L}(p,\mu)$ over all p and μ. The minimization problem we consider is convex, because our function to optimize is convex, the inequality constraint is convex and the feasible set $\mathbb{R}^n \times \mathbb{R}$ is also convex. For that, we know that computing a critical point of the Lagrangian function $\mathcal{L}$ also leads to a global optimum of the optimization problem itself. We compute a critical point with

$$\nabla\mathcal{L}(p,\mu) = \begin{pmatrix} \partial_{p^x}\mathcal{L}(p,\mu) \\ \partial_{p^t}\mathcal{L}(p,\mu) \\ \partial_{\mu}\mathcal{L}(p,\mu) \end{pmatrix} = \begin{pmatrix} p^x - p_0^x + \mu 2\alpha p^x \\ p^t - p_0^t - \mu \\ p^t - \alpha||p^x||_2^2 \end{pmatrix} = 0. \tag{4.25}$$

That means, we need to solve a linear system. The first equation gives us

$$p_0^x = (\mu 2\alpha + 1)p^x \iff p^x = \frac{p_0^x}{\mu 2\alpha + 1}, \tag{4.26}$$

and the second equation leads us to

$$p^t = p_0^t + \mu. \tag{4.27}$$

In the following we discuss two different possibilities how the system **??** can be solved.

1. Plugging the equalities **??** and **??** into the third line of equation **??**, we get

$$\begin{aligned} p^t - \alpha||p^x||_2^2 \quad &\iff \quad p_0^t + \mu - \alpha\left|\left|\frac{p_0^x}{\mu 2\alpha + 1}\right|\right|_2^2 = 0 \iff p_0^t + \mu - \frac{\alpha}{(\mu 2\alpha+1)^2}||p_0^x||_2^2 = 0 \\ &\overset{\cdot(\mu 2\alpha+1)^2}{\iff} \quad (\mu 2\alpha+1)^2 p_0^t + (\mu 2\alpha+1)^2\mu - \alpha||p_0^x||_2^2 = 0 \\ &\iff \quad (4\mu^2\alpha^2 + 4\mu\alpha + 1)p_0^t + 4\mu^3\alpha^2 + 4\mu^2\alpha + \mu - \alpha||p_0^x||_2^2 = 0 \\ &\iff \quad 4\alpha^2\mu^3 + \mu^2(4\alpha^2 p_0^t + 4\alpha) + \mu(4\alpha p_0^t + 1) + p_0^t - \alpha||p_0^x||_2^2 = 0. \end{aligned}$$

   Defining a function

$$h(\mu) = 4\alpha^2\mu^3 + \mu^2(4\alpha^2 p_0^t + 4\alpha) + \mu(4\alpha p_0^t + 1) + p_0^t - \alpha||p_0^x||_2^2,$$

   we are seeking for the zeroes of h. Computing the zeroes can efficiently be established by using Newton's algorithm defined by

$$\mu^{k+1} = \mu^k - \frac{h(\mu)}{h'(\mu)}, \tag{4.28}$$

   for $k = 1, 2, ...$

   If we set

$$h(\mu) = 4\alpha^2\mu^3 + \mu^2(4\alpha^2 p_0^t + 4\alpha) + \mu(4\alpha p_0^t + 1) + p_0^t - \alpha||p_0^x||_2^2,$$

the first derivative of h is given by

$$h^{'}(\mu) = 12\alpha^2\mu^2 + 2\mu(4\alpha^2 p_0^t + 4\alpha) + (4\alpha p_0^t + 1).$$

Overall, we get the update equation for a μ^{k+1} with

$$\mu^{k+1} = \mu^k - \frac{4\alpha^2\mu^3 + \mu^2(4\alpha^2 p_0^t + 4\alpha) + \mu(4\alpha p_0^t + 1) + p_0^t - \alpha||p_0^x||_2^2}{12\alpha^2\mu^2 + 2\mu(4\alpha^2 p_0^t + 4\alpha) + (4\alpha p_0^t + 1)}. \tag{4.29}$$

In [**?**] they suggest setting $\mu^0 = \max\{0, -\frac{2p_0^t}{3}\}$, where they state that Newton's method converges within 7-10 iterations to a quite accurate solution.
The projected vector p of our problem is then given by

$$p = \left(\frac{p_0^x}{\mu 2\alpha + 1}, p_0^t + \mu\right), \tag{4.30}$$

which we get from equations **??** and **??**. We did not apply this method to our framework, since the primal-dual algorithm will be extremley slow in the case of this framework. Having a few iterations of Newton's algorithm in each iteration step would generate more overhead and for that would decelerate the program. Further, Newton's method is inexact and as it turns out, the second approach to this problem will lead to an exact solution, which can be computed within one loop of straightforward computations.

2. For the second approach we note that **??** and **??** hold and the third equation in **??** can be expressed with

$$p^t = \alpha||p^x||_2^2 \Longleftrightarrow \alpha||p^x||_2^2 = \underbrace{p_0^t + \mu}_{p^t}. \tag{4.31}$$

With equation **??** we can also compute the solution of μ by using that

$$\begin{aligned}
p^x = \frac{p_0^x}{\mu 2\alpha + 1} \quad &\Longleftrightarrow \quad ||p^x||_2 = \left|\left|\frac{p_0^x}{1 + 2\alpha\mu}\right|\right|_2 \Longleftrightarrow ||p^x||_2 = \frac{1}{1 + 2\alpha\mu}||p_0^x||_2 \\
&\Longleftrightarrow \quad \frac{1}{1 + 2\alpha\mu} = \frac{||p^x||_2}{||p_0^x||_2} \\
&\Longleftrightarrow \quad 1 + 2\alpha\mu = \frac{||p_0||_2}{||p_x||_2} \\
&\Longleftrightarrow \quad 2\alpha\mu = \frac{||p_0^x||_2}{||p^x||_2} - 1 \\
&\Longleftrightarrow \quad \mu = \frac{1}{2\alpha}\left(\frac{||p_0^x||_2}{||p^x||_2}\right) - \frac{1}{2\alpha}.
\end{aligned}$$

Using the solution of μ in **??** we get

$$
\begin{aligned}
\alpha||p^x||_2^2 = p_0^t + \frac{1}{2\alpha}\left(\frac{||p_0^x||_2}{||p^x||_2}\right) - \frac{1}{2\alpha} \quad &\overbrace{\iff}^{\cdot 2\alpha||p^x||_2} \quad 2\alpha^2||p^x||_2^3 = 2\alpha||p^x||_2 p_0^t + ||p_0^x||_2 - ||p^x||_2 \\
&\iff \quad 2\alpha^2||p^x||_2^3 + 2\alpha||p^x||_2 - 2\alpha p_0^t||p^x||_2 - ||p_0^x||_2 = 0. \\
&\iff \quad 2\alpha^2||p^x||_2^3 + (1 - 2\alpha p_0^t)||p^x||_2 - ||p_0^x||_2 = 0. \\
&\overbrace{\iff}^{\cdot 4\alpha} \quad 8\alpha^3||p^x||_2^3 + 4\alpha(1 - 2\alpha p_0^t)||p^x||_2 - 4\alpha||p_0^x||_2 = 0. \\
&\iff \quad (2\alpha||p^x||_2)^3 + 2(1 - 2\alpha p_0^t)2\alpha||p^x||_2 - 4\alpha||p_0^x||_2 = 0. \\
&\iff \quad t^3 + 3bt - 2a = 0, \qquad (4.32)
\end{aligned}
$$

with $a = 2\alpha||p_0^x||_2$, $b = \frac{2}{3}(1 - 2\alpha p_0^t)$ and $t = 2\alpha||p^x||_2$.
The cubic equation **??** in t can efficiently be solved using the analytical formula for solving cubic equation published by J. P. McKelvey in 1984 in [**?**].

The result of the work mentioned is summarized in the following algorithm. Notice that we already computed the factors a and b. The others follow with [**?**].

Algorithm 4.14 *If already $p_0^t \geq \alpha||p_0^x||_2^2$, the solution is $(p^x, p^t) = (p_0^x, p_0^t)$. Otherwise, with $a = 2\alpha||p_0^x||_2$, $b = \frac{2}{3}(1 - 2\alpha p_0^t)$, and*

$$
d = \begin{cases} a^2 + b^3 & \text{if } b \geq 0 \\ (a - \sqrt{-b}^3)(a + \sqrt{-b}^3) & \text{else} \end{cases}
$$

set

$$
v = \begin{cases} c - \dfrac{b}{c} \text{ with } c = \sqrt[3]{a + \sqrt{d}} & \text{if } d \geq 0 \\ 2\sqrt{-b}\cos\left(\dfrac{1}{3}\arccos\dfrac{a}{\sqrt{-b}^3}\right) & \text{else.} \end{cases}
$$

If $c = 0$ in the first case, set $v = 0$. The solution is then given by

$$
p^x = \begin{cases} \dfrac{v}{2\alpha}\dfrac{p_0^x}{||p_0^x||_2} & \text{if } p_0^x \neq 0 \\ 0 & \text{else} \end{cases}
$$

and $p^t = \alpha||p^x||_2^2$.

The above method states that the projection onto the parabola can be done by one cycle of straightforward computations. The implementation of it is quite simple and computation time is fast. Compared to Newton's algorithm, which is inexact, we get a exact solution and save a huge amount of run-time.

4.4.5 Projection onto K_{nl}

This set is a combination of non-local constraints, meaning that in a fixed point (i, j, k) we sum up for all possible combinations $k_1 \leq k \leq k_2$ for all $k, k_1, k_2 = 1, ..., S$. For that reason you can not project pointwise. Let us first present the algorithm:

Algorithm 4.15 (Soft Shrinkage Scheme) *Let $p_k^i = (p^1(i,j,k), p^2(i,j,k))^T \in \mathbb{R}^2$, $p^i = (p_1^i, ..., p_S^i)^T \in \mathbb{R}^{2\times S}$ for all $i = 1, 2, ...$ Then the projection p^{n+1} of a p^n - for an arbitrary, fixed pair (k_1, k_2) with $1 \leq k_1 \leq k \leq k_2 \leq S$ - is computed by:*

$$p^{n+1} = \begin{cases} p^n + \dfrac{s - \tilde{s}}{k_2 - k_1 + 1} & \text{if } k_1 \leq k \leq k_2, \\ p^n & \text{else,} \end{cases}$$

where $s \in \mathbb{R}^2, \tilde{s} \in \mathbb{R}^2$ with

$$\tilde{s} = \sum_{k_1 \leq k \leq k_2} p_k^n$$

and

$$s = \begin{cases} \tilde{s} & \text{if } ||\tilde{s}||_2 \leq \nu, \\ \Pi_{||\cdot||_2 \leq \nu}(\tilde{s}) = \dfrac{\nu}{||\tilde{s}||_2}\tilde{s} & \text{else.} \end{cases}$$

Since this procedure needs a clarification, we need the KKT conditions. Following the notation, definition and theorem of [**?**], we consider the optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0 & i \in \mathcal{E}, \\ c_i(x) \geq 0 & i \in \mathcal{I}, \end{cases} \tag{4.33}$$

where f and for all $i \in \mathcal{E} \cup \mathcal{I}$ the functions c_i are smooth, real-valued functions on a subset of $\mathbb{R}^n$. Further, $\mathcal{E}$ and $\mathcal{I}$ denote two finite sets of indices. Here, f is the objective function, whereas the c_i for $i \in \mathcal{E}$ are the equality constraints and c_i with $i \in \mathcal{I}$ the inequality constraints. We further want to define the feasible set Ω. As stated in section **??** this set is the set of all points, which satisfy the constraints c_i for all i. Then we have

$$\Omega = \{x \mid c_i(x) = 0,\ i \in \mathcal{E};\ c_i \geq 0,\ i \in \mathcal{I}\}. \tag{4.34}$$

We rewrite our optimization problem in terms of the set Ω to

$$\min_{x \in \Omega} f(x).$$

Additionally, we want to define the so called active set for the constraint functions.

Definition 4.16 (Active Set) *The active set $\mathcal{A}(x)$ at any feasible x consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i, for which $c_i(x) = 0$, that is*

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(x) = 0\} \tag{4.35}$$

We call a inequality constraint active if $c_i = 0$ and inactive if $c_i > 0$, for a $i \in \mathcal{I}$. The definition of the active set is an important basis for the following definition.

Definition 4.17 (LICQ) *Given the point x and the active set $\mathcal{A}(x)$, we say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients*

$$\{\nabla c_i(x),\, i \in \mathcal{A}(x)\}$$

is linearly independent.

Then, we can propose the first-order necessary conditions, which are also well known as the Karush-Kuhn-Tucker (KKT) optimality conditions.

Theorem 4.18 (Karush-Kuhn-Tucker Optimality Conditions) *Suppose that x^* is a local solution of equation **??**, that the functions f and c_i are continuously differentiable, and that the LICQ holds at x^*. Then there is a Lagrange multiplier vector λ^*, with components $\lambda_i^*, i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at (x^*, λ^*).*

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0, \tag{4.36a}$$
$$c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E}, \tag{4.36b}$$
$$c_i(x^*) \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{4.36c}$$
$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{4.36d}$$
$$\lambda_i^* c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}. \tag{4.36e}$$

Remark 4.19 *The last condition **??** in the theorem is also known as the complementary slackness condition. It implies that either the i-th constraint is active or $\lambda_i^* = 0$. It is also possible, that both are zero.*

A proof of theorem **??** can, for instance, be found in [**?**]. Let us now proof, that the algorithm **??** is valid, by using the KKT conditions.

Proof Let $\tilde{s}, s, p_k, \tilde{p}_k \in \mathbb{R}^2$ and $p, \tilde{p} \in \mathbb{R}^{2\times M}$ have the same form as in algorithm **??**. For a fixed pair (k_1, k_2) we face the following optimization problem:

$$\min_{p \in \mathbb{R}^{2\times M}} \frac{1}{2} ||p - \tilde{p}||_2^2 \tag{4.37a}$$
$$\text{subject to } \left|\left| \sum_{k_1 \leq k \leq k_2} p_k \right|\right|_2 \leq \nu. \tag{4.37b}$$

The problem states that we try to find the closest p to $\tilde{p}$ whose components fullfil the inequality constraint in **??**, which can also be written as

$$\left|\left| \sum_{k_1 \leq k \leq k_2} p_k \right|\right|_2 - \nu \leq 0.$$

The feasible set for a point (k_1, k_2) of this optimization problem is given by

$$\mathcal{F}_p = \{p \in Y \mid \sum_{k_1 \leq k \leq k_2} p_k \; \forall \, k_1 \leq k \leq k_2 \, \text{with} \, k = 1, ..., S\}.$$

If $\tilde{p} \in \mathcal{F}_p$, the inequality constraint is fullfild and for that $p = \tilde{p}$ leads to the minimal energy, which is zero. For that reason, we only consider points $\tilde{p} \notin \mathcal{F}_p$.

Let us introduce a Lagrange multiplier variable $\lambda \in \mathbb{R}$ and observe the Lagrange function

$$\mathcal{L}(p, \lambda) = f(p) - \lambda g(p) = \sum_{k=1}^{M} \frac{(p_k - \tilde{p}_k)^2}{2} - \lambda \frac{1}{2}\nu^2 - \frac{1}{2} - \left|\left| \sum_{k_1 \leq k \leq k_2} p_k \right|\right|_2. \tag{4.38}$$

Considering equation **??** and assume that $p_k^* = \tilde{p}_k + \frac{s - \tilde{s}}{k_2 - k_1 + 1}$ is a feasible point, that satisfies LICQ and is a local solution of system **??**, together with a optimal λ^*. Then we have

$$\nabla_p \mathcal{L}(p^*, \lambda^*) = \nabla_p f(p^*) + \lambda \nabla_p g(p^*) = \underbrace{\begin{pmatrix} p_1^* - \tilde{p}_1 \\ \vdots \\ \\ \vdots \\ \\ \vdots \\ p_S^* - \tilde{p}_S \end{pmatrix}}_{\in \mathbb{R}^S} + \lambda^* \underbrace{\begin{pmatrix} 0 \\ \vdots \\ \sum\limits_{k_1 \leq k \leq k_2} p_k^* \\ \vdots \\ \sum\limits_{k_1 \leq k \leq k_2} p_k^* \\ 0 \\ \vdots \end{pmatrix}}_{\in \mathbb{R}^S} = 0, \tag{4.39}$$

where the zeroes in the last vector are obtain for all components where $k < k_1$ and $k > k_2$. For these lines we have

$$p_k^* = \tilde{p}_k \;\; \text{if} \; k < k_1 \; \text{and} \; k > k_2.$$

We also have for such a point that $p^* \in \mathcal{F}_p$. With this, let us take a closer look at the i-th line where $(\nabla_p g(p))_i \neq 0$. Then we have

$$p_i^* - \tilde{p}_i + \lambda^* \sum_{k_1 \leq k \leq k_2} p_k^* = 0. \tag{4.40}$$

Further, we observe for the feasible point p^* the following identity:

$$\begin{aligned} \frac{1}{2}\left|\left| \sum_{k_1 \leq k \leq k_2} p_k^* \right|\right|_2^2 - \frac{1}{2}\nu^2 &= \frac{1}{2(k_2 - k_1 + 1)}\left|\left| \sum_{k_1 \leq k \leq k_2} \tilde{p}_k + (s - \tilde{s}) \right|\right|_2^2 - \frac{1}{2}\nu^2 \\ &= \frac{1}{2}\left|\left| \sum_{k_1 \leq k \leq k_2} \tilde{s} + s - \tilde{s} \right|\right|_2^2 - \frac{1}{2}\nu^2 = \frac{1}{2}\underbrace{||s||_2^2}_{=\nu^2} - \frac{1}{2}\nu^2 = 0. \end{aligned}$$

This means, that the inequality constraint is active at the point p^*. Since, this is the case we show that the LICQ is fullfild for p^*. We have

$$\begin{aligned}(\nabla g(p^*))_i = -\sum_{k_1\leq k\leq k_2} p_k^* &= -\Big(\sum_{k_1\leq k\leq k_2}\tilde{p}_k + \frac{\tilde{s}-s}{k_2-k_1+1}\Big)\\ &= -\Big(\sum_{k_1\leq k\leq k_2}\tilde{p}_k + \sum_{k_1\leq k\leq k_2}\frac{\tilde{s}-s}{k_2-k_1+1}\Big)\\ &= -\Big(\tilde{s}+\frac{k_2-k_1+1}{k_2-k_1+1}s-\tilde{s}\Big)\\ &= -s \end{aligned} \tag{4.41}$$

Because, we only observe one vector, LICQ is always true for this p^*. It is left to compute λ^* and show that it is greater or equal to zero. We have

$$\begin{aligned} & p_i^* - \tilde{p}_i + \frac{\lambda^*}{k_2-k_1+1}\sum_{k_1\leq k\leq k_2} p_k^* = \\ = \;& \tilde{p}_i + \frac{s-\tilde{s}}{k_2-k_1+1} - \tilde{p}_i + \frac{\lambda^*}{k_2-k_1+1}\sum_{k_1\leq k\leq k_2}\tilde{p}_k + \frac{s-\tilde{s}}{k_2-k_1+1}\\ = \;& \frac{s-\tilde{s}}{k_2-k_1+1} + \frac{\lambda^*}{k_2-k_1+1}\Big(\underbrace{\sum_{k_1\leq k\leq k_2}\tilde{p}_k}_{=\tilde{s}} + \underbrace{\sum_{k_1\leq k\leq k_2}\frac{s-\tilde{s}}{k_2-k_1+1}}_{=\frac{(k_2-k_1+1)(s-\tilde{s})}{k_2-k_1+1}=s-\tilde{s}}\Big)\\ = \;& \frac{s-\tilde{s}}{k_2-k_1+1} + \frac{\lambda^*}{k_2-k_1+1}(\tilde{s}+s-\tilde{s})\\ = \;& \frac{s-\tilde{s}}{k_2-k_1+1} + \frac{\lambda^*}{k_2-k_1+1}s\\ = \;& \frac{1}{k_2-k_1+1}(s-\tilde{s}+\lambda^* s) = 0 \end{aligned} \tag{4.42}$$

Now we can solve for λ^* using that the last equation **??** is equivalent to

$$\begin{aligned}(s-\tilde{s}+\lambda^* s) = 0 \quad &\overset{\overbrace{||\tilde{s}||_2>\nu}}{\Longleftrightarrow} \quad \Big(\frac{\nu}{||\tilde{s}||_2}\tilde{s} - \tilde{s} + \lambda^*\frac{\nu}{||\tilde{s}||_2}\tilde{s}\Big) = 0\\ &\Longleftrightarrow \quad \tilde{s}\Big(\frac{\nu}{||\tilde{s}||_2} - 1 + \lambda^*\frac{\nu}{||\tilde{s}||_2}\Big) = 0\\ &\Longleftrightarrow \quad \frac{\nu}{||\tilde{s}||_2} - 1 + \lambda^*\frac{\nu}{||\tilde{s}||_2} = 0\\ &\Longleftrightarrow \quad \lambda^*\frac{\nu}{||\tilde{s}||_2} = 1 - \frac{\nu}{||\tilde{s}||_2}\\ &\Longleftrightarrow \quad \lambda^* = \underbrace{\frac{||\tilde{s}||_2}{\nu}}_{>1} - 1 > 0. \end{aligned}$$

In the last equation we used that $\tilde{s} = \sum_{k_1 \leq k \leq k_2} \tilde{p}_k$ and the fact we assumed $||\tilde{s}||_2 > \nu$.

Applying this procedure to each combination (k_1, k_2) and replacing p^* by p^{n+1}, $\tilde{p}$ by p^n respectively, we observe our algorithm. ■

We presented and proved several projection methods to project onto the sets C and K. As we will see in the chapter **??** this approach needs a huge amount of memory and is extremley slow, even on a GPU. In the next section, we will present an alternative approach.

4.5 An alternative approach

In this section we present a formulation to solve the discrete Mumford-Shah Functional of subsection **??**. The idea is to decouple the set K_{nl} to derive an alternative saddle-point problem, which can be solved with our primal-dual algorithm. Recalling the original problem, we had

$$\min_{u \in C} \max_{p \in K} \langle Au, p \rangle, \tag{4.43}$$

where we maximized over the whole set K, defined in equations **??** and **??**. As discussed in section **??**, this set is an intersection of several convex sets. We projected onto the non-local constraint, which was defined by

$$K_{nl} = \left\{ \left| \sum_{k_1 \leq k \leq k_2} p^x(i,j,k) \right| \leq \nu \right\} \ \forall i, j, k_1 \leq k \leq k_2,$$

where p^x is defined as in subsection **??**, using a soft-shrinkage scheme. This projection was used in each step of Dykstra's algorithm. We now want to solve this problem optimal by decoupling the non-local constraint. We note that in the following we drop the index representation for p^x. Instead of writting $p^x(i,j,k)$ we denote this by p^x_k for a pair (i,j) and all k. Decoupling means then, that we substitute p^x_k by s_{k_1,k_2} and introduce an additional constraint to take the bound on ν into account. We have

$$K_{nl} = \left\{ |s_{k_1,k_2}| \leq \nu \ \text{ subject to } \ s_{k_1,k_2} = \sum_{k_1 \leq k \leq k_2} p^x_k \right\} \tag{4.44}$$

for all combinations $1 \leq k_1 \leq k \leq k_2 \leq S$, $s_{k_1,k_2} \in \mathbb{R}^2$ and $s \in \mathbb{R}^{N \times M \times S \times 2}$, respectively. Alternatively, we can also denote this set by

$$K_{nl} = \left\{ |s_{k_1,k_2}| \leq \nu \ \text{ subject to } \ s_{k_1,k_2} - \sum_{k_1 \leq k \leq k_2} p^x_k = 0 \right\}$$

We further present an auxiliary variable $\mu_{k_1,k_2} \in \mathbb{R}^2$, which belongs to a $\mu \in \mathbb{R}^{N \times M \times S \times 2}$. We define a new function

$$\mathcal{L}(u, \mu, p, s) = \langle Au, p \rangle + \sum_{k_1=1}^{S} \sum_{k_2=k_1}^{S} \langle \mu_{k_1,k_2}, s_{k_1,k_2} - \sum_{k_1 \leq k \leq k_2} p^x_k \rangle, \tag{4.45}$$

in which we added to our original problem an enforced term corresponding to the constraint in K_{nl}. Taking now into account, that the constraint is an equality constraint, we have that all μ_{k_1,k_2} can either be greater, less than or equal to zero. Additionally, we need to be sure, that the inequality $|s_{k_1,k_2}| \leq \nu$ holds. Then we obtain the following equivalence:

$$\min_{u\in C}\max_{p\in K}\langle Au,p\rangle \Longleftrightarrow \min_{\substack{u\in C\\ \mu_{k_1,k_2}}}\max_{\substack{p\in K_p\\ |s_{k_1,k_2}|\leq\nu}} \mathcal{L}(u,\mu,p,s),$$

which complies to

$$\min_{u\in C}\max_{p\in K}\langle \nabla u,p\rangle \Longleftrightarrow \min_{\substack{u\in C\\ \mu_{k_1,k_2}}}\max_{\substack{p\in K_p\\ |s_{k_1,k_2}|\leq\nu}} \langle \nabla u,p\rangle + \sum_{k_1=1}^{S}\sum_{k_2=1}^{S}\langle \mu_{k_1,k_2}, \sum_{k_1\leq k\leq k_2} p_k^x - s_{k_1,k_2}\rangle, \quad (4.46)$$

if we use $A = \nabla$ as in the other chapters. Let us proof this equivalence in the general case, where A resemble a linear operator:

Proof As we did not change either the properties on u nor the set C, we assume for the rest of the proof, that we found an optimal value u^*. Further, assume we have a optimal value s^*, for which $|s^*_{k_1,k_2}| \leq \nu$ for all possible combinations (k_1, k_2). Then, we do a case analysis:

1. If the equality
$$\sum_{k_1\leq k\leq k_2} p_k^x - s^*_{k_1,k_2} = 0,$$
holds, the saddle-point problem reduces to
$$\max_{p\in K_p}\langle Au^*,p\rangle.$$
If the equality does not hold, we distinguish between two other cases:

2. Let $\sum_{k_1\leq k\leq k_2} p_k^x < s^*_{k_1,k_2}$, then first note, that this is equivalent to
$$\left|\left|\sum_{k_1\leq k\leq k_2} p_k^x\right|\right| < |s^*_{k_1,k_2}|.$$
But as $|s^*_{k_1,k_2}| \leq \nu$ we already have
$$\left|\left|\sum_{k_1\leq k\leq k_2} p_k^x\right|\right| \leq \nu.$$
Further, we obtain the minimal value over all possible μ_{k_1,k_2} in this case only if we let $\mu_{k_1,k_2} \longrightarrow \infty$, for which the whole energy becomes $-\infty$. Since, the conditions according to the set K_{nl} are met and to avoid having a energy with the value $-\infty$, we set $\mu_{k_1,k_2} = 0$ in this case. We get again
$$\max_{p\in K_p}\langle Au^*,p\rangle.$$

3. Now, let $\sum_{k_1 \le k \le k_2} p_k^x > s^*_{k_1,k_2}$. In this case we could again set $\mu_{k_1,k_2} = 0$. Then the last term in **??** will not be considered. But with this we are not able to tell anything about the sum over the p_k^x. We know that

$$\left\| \sum_{k_1 \le k \le k_2} p_k^x \right\| > |s^*_{k_1,k_2}|$$

and from our assumption that $|s^*_{k_1,k_2}| \le \nu$. The only choice we have is to set

$$\sum_{k_1 \le k \le k_2} p_k^x = s^*_{k_1,k_2}.$$

With this we get

$$\max_{p \in K_p} \langle Au^*, p \rangle,$$

and the μ_{k_1,k_2} can be chosen arbitrarily. This shows, that the two problems are equivalent.

■

Defining the function

$$\mathcal{M}(u, \mu, p, s) = \langle Au, p \rangle + \sum_{k_1=1}^{S} \sum_{k_2=1}^{S} \langle \mu_{k_1,k_2}, \sum_{k_1 \le k \le k_2} p_k^x - s_{k_1,k_2} \rangle$$

leads to

$$\frac{\partial \mathcal{M}(u, \mu, p, s)}{\partial u} = A^T p \tag{4.47}$$

$$\frac{\partial \mathcal{M}(u, \mu, p, s)}{\partial p_k} = \sum_{k_1,k_2} \mu_{k_1,k_2} \tag{4.48}$$

$$\frac{\partial \mathcal{M}(u, \mu, p, s)}{\partial \mu_{k_1,k_2}} = \sum_{k_1,k_2} \left(\sum_{k_1 \le k \le k_2} p_k^x - s_{k_1,k_2} \right) \tag{4.49}$$

$$\frac{\partial \mathcal{M}(u, \mu, p, s)}{\partial p} = Au + \hat{p}. \tag{4.50}$$

with

$$
\begin{aligned}
\frac{\partial \mathcal{M}(u,\mu,p,s)}{\partial p_l} &= (Au)_l + \frac{\partial}{\partial p_l}\Big(\sum_{k_1=1}^{S}\sum_{k_2=k_1}^{S}\langle \mu_{k_1,k_2}, s_{k_1,k_2} - \sum_{k_1\le k\le k_2} p_k^x\Big) \\
&= (Au)_l + \frac{\partial}{\partial p_l}\Big(\sum_{k_1=1}^{S}\sum_{k_2=k_1}^{S}\langle \mu_{k_1,k_2}, s_{k_1,k_2}\rangle - \langle \mu_{k_1,k_2}, \sum_{k_1\le k\le k_2} p_k^x\Big) \\
&= (Au)_l - \frac{\partial}{\partial p_l}\Big(\sum_{k_1=1}^{S}\sum_{k_2=k_1}^{S}\langle \mu_{k_1,k_2}, \sum_{k_1\le k\le k_2} p_k^x\Big) \\
&= (Au)_l - \begin{pmatrix} \sum_{k_1=1}^{l}\sum_{k_2=k_1}^{l}\mu^1_{k_1,k_2} \\ \sum_{k_1=1}^{l}\sum_{k_2=k_1}^{l}\mu^2_{k_1,k_2}\end{pmatrix} \qquad (4.51)
\end{aligned}
$$

With this it follows

$$
\tilde{p} = -\begin{pmatrix} \sum_{k_1=1}^{l}\sum_{k_2=k_1}^{l}\mu^1_{k_1,k_2} \\ \sum_{k_1=1}^{l}\sum_{k_2=k_1}^{l}\mu^2_{k_1,k_2}\end{pmatrix} \qquad (4.52)
$$

We have

Algorithm 4.20 *Choose* $(u^0,p^0,\mu^0,s^0)\in C\times K_p\times\mathbb{R}^{2\times N\times M\times S}\times\mathbb{R}^{2\times N\times M\times S}$ *and let* $\bar{x}^0=u^0,\bar{\mu}^0=\mu^0$. *We choose* $\tau_u=\frac{1}{6},\tau_\mu=\frac{1}{2+k_2-k_1},\sigma_p=\frac{1}{3+S},\sigma_s=1$. *Then, we let for each* $n\ge 0$

$$
\begin{cases}
p^{n+1} = \Pi_{K_p}(p^n+\sigma_p(A\bar{u}^n+\tilde{p})) \\
s^{n+1}_{k_1,k_2} = \Pi_{|\cdot|\le\nu}(s^n_{k_1,k_2}+\sigma_s\bar{\mu}^n_{k_1,k_2}) \\
u^{n+1} = \Pi_C(u^n-\tau_u A^* p^{n+1}) \\
\mu^{n+1}_{k_1,k_2} = \mu^n_{k_1,k_2} - \tau_\mu(s^{n+1}_{k_1,k_2} - \sum_{k_1\le k\le k_2} p^x_k \\
\bar{u}^{n+1} = 2u^{n+1}-u^n \\
\bar{\mu}^{n+1}_{k_1,k_2} = 2\mu^{n+1}_{k_1,k_2} - \mu^n_{k_1,k_2}.
\end{cases} \qquad (4.53)
$$

5 Applications to Imaging

In this chapter we present applications to imaging for the presented models. We will compare run-time of different models by looking at the pure image approximation. We will see that the best method to solve the convex relaxed Mumford-Shah model is the algorithm based on Lagrange multipliers. Further, we consider image cartooning by using the real-time minimizer for the Mumford-Shah model, the ROF model and the TVL1 model. The last two models together with the convex relaxed Mumford-Shah model are then taken into account, when we show the denoising case. Last but not least we will see the most impressive application: image inpainting. The ROF model is able to reconstruct an images, which has a data loss of 70 percent.

5.1 Linearized Storage of Images

Images in a discrete two-dimensional grid of the size $N \times M$ can be viewed like matrices. The first element, which can be accessed is the one at the top left corner. Then they are stored row-wise from the left to the right side. The last element of this matrix is then at the bottom right. As mentioned in chapter **??**, we do not store two-dimensional images in matrix form, but use a linearized version. In a programming language like C++, an image in matrix form would be accessed with

$$image[i][j] = value;,$$

where i resembles the i-th row of the pixel grid and j the j-th element in the corresponding row. It also admits, that we allocate a pointer to a pointer array and to access one element we need to look up two points in memory. This would slow down the code and for that we store all data only in one vector. We allocate a vector v of the size $N \cdot M$. We attach each line of the image matrix to this array. Then we can access elements by

$$v[j + i \cdot M] = value;.$$

This method works well for grayscaled images, where we approximate an input images $g : \mathcal{G} \longrightarrow C = [0, 1]$ by a function $u : \mathcal{G} \longrightarrow C = [0, 1]$. So we map from the discret pixel grid $\mathcal{G}$ (see also notation **??**) into the range from 0 to 1. Instead of using $[0, 1]$, we could also set $C = \{0, ..., 255\}$, where we assume our image and approximation to have a 8-bit depth (see remark **??**). Additionally note, that the vector $\bar{u}$ is stored in the same fashion, in the propose primal-dual algorithm.

In the case of color images, in our case only RGB (red-green-blue) images, we can extend this kind of linearized storage easily. For that, we assume that each single color

channel has its own pixel grid. We have for $k = 1, 2, 3$

$$\mathcal{G}_k = \Big\{(i, j) : i = 1, ..., N \text{ and } j = 1, ..., M\Big\}. \tag{5.1}$$

Overall, we attach each row of each grid separately to the vector img consecutively. Then, we have that $img \in \mathbb{R}^{N \cdot M \cdot 3}$ and we access one element by

$$img[j + i \cdot M + k \cdot M \cdot N] = value;.$$

Furthermore, we need to consider the variables p in the primal-dual algorithm. These are also stored as vectors, but with the extension, that for a fixed point (i, j) (or (i, j, k) in the case of color images) we have $p_{i,j(,k)} \in \mathbb{R}^2$. We have two possibilities how to handle the storage of these variables. The easiest way, which we choose to use, would be to allocate two vectors of the size $N \cdot M \cdot 3$ and call them for instance p_x and p_y, respectively. Another possibility is to do the same as before: attach the values of p_y to these of p_x and observe one large array, which then can be accessed by

$$p[j + i \cdot M + k \cdot M \cdot N + l \cdot M \cdot N \cdot 3] = value;,$$

with $l = 1, 2$. For us it seemed to be necessary to use the first method, to obtain a readable and maintainable code. At last, we want to mention, that in our case using C++ and the CUDA framework, the access of elements starts with 0 and the last element of a n-dimensional array is then accessed at the location $n - 1$.

5.2 Image Approximation using the ROF Model

In this section we show some approximations of input images. Besides we compare run-time and different parameters. We also provide the best estimates of parameters for each model. Further, we proof that using the Lagrange Multiplier method for the convex relaxed Mumford-Shah model leads to a run-time speed up of a factor 450 compared with the proposed method in [**?**].

The implementation for this model using the proposed primal-dual algorithm **??** is straightforward.

5.2.1 Implementation Issues

In our framework we propose both, a C++ and a CUDA implementation. Since, there is no big difference in the code itself, we discuss the implementation not programming language specific. We set $K = \nabla$ with the proposed discretization stated in section **??**. Further, we consider u, $\bar{u}$, etc. being accessed with $u_{i,j,k}$. For the approximation of color images we have that $k = 1, 2, 3$, in the case of grayscaled images $k = 1$.

Initialization

According to [**?**] the algorithm is independent of its initialization. The better the initialization at the beginning, the faster the convergence to a optimal solution. We ran numerous tests to find good initialization values, e.g. setting everything to zero or using the maximal value 255. It turned out, that the best choice is the input image g itself. In table **??** we show a few estimates of our tests. So, overall we initialize with $p = 0$ and $u = g$.

Initialization	Iterations	CPU time
0	593	5.13
124	575	4.88
255	562	4.84
g	510	4.22

Computing $p^{n+1} = (\mathrm{Id} + \sigma\,\partial\, F^*)^{-1}(p^n + \sigma\,\nabla\bar{u}^n)$

In the $n+1$-th iteration step we first need to estimate the gradient of $\bar{u}^n$. For this we allocate two variables dx and dy and compute under the premise that we discretized ∇ by forward differences with Neumann boundary conditions

$$dx = \bar{u}_{i+1,j,k} - \bar{u}_{i,j,k} \quad \text{and} \quad dy = \bar{u}_{i,j+1,k} - \bar{u}_{i,j,k},$$

where we set $dx = 0$ if $i+1 < M$ and $dy = 0$ if $j+1 < N$. After that, we multiply both values with σ and then add the old estimate of p, namely p^n to them. This means

$$dx = dx + p^n_{x_{i,j,k}} \quad \text{and} \quad dy = dy + p^n_{y_{i,j,k}}.$$

At the end we only need to apply the proximity operator of the function F^* to the variables dx and dy and save the observed value in p^{n+1}. We compute

$$p^{n+1}_{x_{i,j,k}} = \frac{dx}{\max(1, |dx|)} \quad \text{and} \quad p^{n+1}_{y_{i,j,k}} = \frac{dy}{\max(1, |dy|)}.$$

Algorithm 5.1 (Dual Ascent) *Summarizing this procedure in a function, we get*

```
template<typename T>
void dual_asc(T* p_x, T* p_y, T* u_bar, float sigma, int M, int N, int C) {
  T dx, dy;
  int index;
  for (int k = 0; k < C; k++) {
    for (int i = 0; i < M; i++) {
      for (int j = 0; j < N; j++) {
        index = j + i * N + k * M * N;
        dx = i+1<M ? u_bar[j + (i+1) * N + k * M * N]-u_bar[index] : 0;
        dy = j+1<N ? u_bar[(j+1) + i * N + k * M * N]-u_bar[index] : 0;
        dx = p_x[index] + sigma * dx;
        dy = p_y[index] + sigma * dy;
        p_x[index] = dx / fmax(1.f, fabs(dx));
```

```
          p_y[index] = dy / fmax(1.f, fabs(dy));
        }
      }
    }
  }
}
```

where C is the number of color channels and the template value T is mostly used as float.

These 18 lines of code are used to compute the update on p^{n+1}. To be able to use this function in another framework, like for minimizing the TVL1 energy, we only need to change the last two lines in the for-loops. These resemble the proximity operator and this operator can vary from model to model.

Computing $u^{n+1} = (\mathrm{Id} + \tau\,\partial\, G)^{-1}(u^n - \tau\,\mathrm{div} p^{n+1})$

The first thing to mention for this line of pseudo code is, that we set $\nabla^T = -\mathrm{div}$. But then we compute

$$u^{n+1} = (\mathrm{Id} + \tau\,\partial\, G)^{-1}(u^n + \tau\,\nabla^T p^{n+1}).$$

We follow the same procedure as before: we allocate values dx, dy and this time also sum, in which we store the sum of the partial derivatives. Using backward differences with Dirichlet boundary conditions, like in definition **??**, we compute

$$dx = p_x i,j,k - p_x i-1,j,k,\;\; dy = p_y i,j,k - p_y i,j-1,k \;\text{ and }\; sum = \tau \cdot (dx + dy).$$

And we additional take into account that if $i+1 < M$ or $j+1 < N$ we have

$$dx = -p_x i-1,j,k \quad \text{or} \quad dy = -p_y i,j-1,k$$

and if $i > 0$ or $j > 0$ we compute

$$dx = p_x i,j,k \quad \text{or} \quad dy = p_y i,j,k.$$

As we already multiplied the discrete divergence with the parameter τ, it is left to add the previous estimate u^n by

$$sum = u_{i,j,k} + sum.$$

In a last step we apply the proximity operator for the function G and observe

$$u_{i,j,k}^{n+1} = \frac{sum + \tau\lambda g_{i,j,k}}{1 + \tau\sigma}.$$

Algorithm 5.2 (Primal Descent) *Summarizing again in a function, we have*

```
template<typename T>
void primal_desc(T* p_x, T* p_y, T* u, T* g, float tau, float lambda, int M, int N, in
  T dx, dy, sum;
  int index;
  for (int k = 0; k < C; k++) {
    for (int i = 0; i < M; i++) {
```

```
          for (int j = 0; j < N; j++) {
            index = j + i * N + k * M * N;
            dx = (i+1<M ? p_x[index] : 0.f) -
                 (i>0 ? p_x[j + (i-1) * N + k * M * N] : 0.f);
            dy = (j+1<N ? p_y[index] : 0.f) -
                 (i>0 ? p_y[(j-1) + i * N + k * M * N] : 0.f);
            sum = tau * (dx + dy);
            sum += u[index];
            u[index] = (sum + tau * lambda * g[index]) / (1.f + tau * lambda);
          }
        }
      }
    }
```

where C is the number of color channels and the template value T is mostly used as float.

It is only left to compute the extrapolation step. This is a straightforward computation, since we just add vectors.

Algorithm 5.3 (Extrapolation) *In a function we have*

```
template<typename T>
void extrapolation(T* u_bar, T* u, T* u_prev, float theta, int M, int N, int C) {
  for (int i = 0; i < N*M*C; i++) {
    u_bar[i] = u[i] + theta * (u[i] - u_prev[i]);
    u_prev[i] = u[i];
  }
}
```

where C is the number of color channels and the template value T is mostly used as float.

We also set u_{prev} to the current estimate u, because we need this in each step of the extrapolation function. As a last function we provide the primal-dual algorithm, which makes use of the stated functions:

Algorithm 5.4 (Primal-Dual Algorithm) *In a function we have*

```
template<typename T>
void ROF(T* u, T* g, float lambda, float tau, int M, int N, int C) {
  float sigma = 1.f / (tau * 8.f);
  float theta = 2.f;
  T* u_bar = new T[M*N*C];
  T* u_prev = new T[M*N*C];
  T* p_x = new T[M*N*C];
  T* p_y = new T[M*N*C];

  void dual_asc(p_x, p_y, u_bar, sigma, M, N, C);
  void primal_desc(p_x, p_y, u, g, tau, lambda, M, N, C);
  void extrapolation(u_bar, u, u_prev, theta, M, N, C);

  delete [] u_bar;
  delete [] u_prev;
  delete [] p_x;
  delete [] p_y;
}
```

where C is the number of color channels and the template value T is mostly used as float.

We derive the computation of σ, because of the convergence theorem **??**. It states, that we have a convergent algorithm if $\sigma\tau L^2 < 1$. For that we get

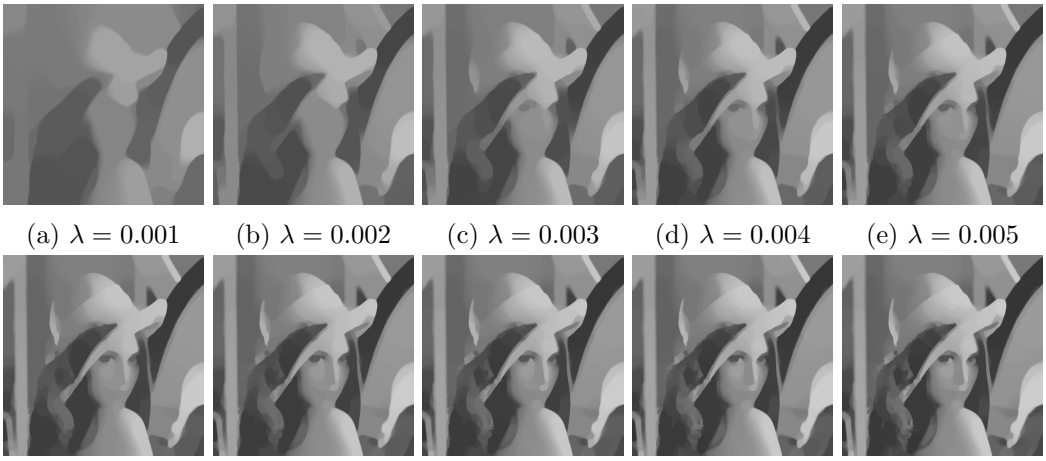
$$\sigma = \frac{1}{\tau * L^2} = \frac{1}{\tau * 8},$$

by proposition **??**. Further, we set $\theta = 2$ to derive the version suggested in [**?**].

Now, that we know how the implementation of the ROF model can be done, we want to compare some outcomes of the algorithm. We ran two separate parameter estimations: first we looked for the perfect λ for a minimal energy, a fast convergence and a visible good approximation u. Afterwards, we estimated the τ, for which the algorithm converges quickly with respect to the optimal λ. In figure **??** one can see the evolution of the Lena image using different parameters λ and estimating the perfect fit.

5.2.2 Best λ estimation

To estimate the possible best λ we started by testing all values in the range from 0.001 to 0.01 by adding in each approximation a factor 0.001 to the parameter λ. In figure **??** one can see, that an increasing value for λ makes the image more visible. But, it turned out that the algorithm took a long time till convergence - differing from over 1000 to almost 6000 iterations. Additionally, the PSNR is then between 18 and 25, which is too bad for a good approximation of an image.



(a) $\lambda = 0.001$ (b) $\lambda = 0.002$ (c) $\lambda = 0.003$ (d) $\lambda = 0.004$ (e) $\lambda = 0.005$

(f) $\lambda = 0.006$ (g) $\lambda = 0.007$ (h) $\lambda = 0.008$ (i) $\lambda = 0.009$ (j) $\lambda = 0.01$

Figure 5.1: Approximation of the Lena image with the ROF Model. Again, the higher λ the closer the outcome of the algrithm to the original image.

It seemed not to be appropriable to go one one-thousandth steps and we already learned, that a small parameter does not lead to the desired results. We then turned our interest to the other extrem setting. We used the range from 0.1 to 1 in one-tenth

steps. It not only lead to the perfect fit, it also produced good approximations u of our input image g. This can also be seen in figure **??**. We also want to show the table with some data like PSNR, iterations, energy and run-time.

(a) $\lambda = 0.1$ (b) $\lambda = 0.2$ (c) $\lambda = 0.3$ (d) $\lambda = 0.4$ (e) $\lambda = 0.5$

(f) $\lambda = 0.6$ (g) $\lambda = 0.7$ (h) $\lambda = 0.8$ (i) $\lambda = 0.9$ (j) $\lambda = 1$

Figure 5.2: Approximation of the Lena image with the ROF Model. Again, the higher λ the closer the outcome of the algrithm to the original image.

Having a PSNR over 40 means, that we are extremely close to the original image. As discussed before, an image consists of data and noise and we seek to remove that noise from our images. We assumed to only consider approximation u, which have a PSNR less than 40. Then there were only the choices 0.1 and 0.2, but as the energy in the case $\lambda = 0.1$ was significantly smaller, we choose this λ as our best estimate. We also ran some comparison for $\lambda = 0.09$ down to $\lambda = 0.06$, but nothing fitted better than this estimate.

λ	Iterations	Run-Time	MSE	PSNR	Energy
0.1	505	3.84	23	34	1,384,210
0.2	505	3.94	12	38	1,595,940
0.3	502	4.02	8	39	1,717,590
0.4	502	3.96	6	41	1,802,040
0.5	502	4.07	4	42	1,865,850
0.6	504	4.29	3	43	1,916,280
0.7	502	3.97	3	44	1,957,350
0.8	502	4.00	2	44	1,991,410
0.9	505	4.10	2	45	2,020,130
1.0	503	4.00	2	46	2,044,670

To the end of this subsection let us mention that in [**?**] they proposed other estimates for λ, since they discretized the domain Ω by a factor $h_N = \frac{1}{N}$ and $h_M = \frac{1}{M}$, respectively.

Lena Image					Hepburn Image			
No.	τ	Iterations	Run-Time	Energy	τ	Iterations	Run-Time	Energy
1	0.93	49	0.70	1.383.850	0.86	57	1.91	5.219.660
2	0.73	52	0.75	1.384.130	0.84	65	2.08	5.219.470
3	0.69	53	0.77	1.384.190	0.85	71	2.26	5.219.340
4	0.61	65	0.94	1.384.200	0.93	72	2.34	5.219.120
5	0.77	67	0.95	1.384.110	0.73	76	2.43	5.219.680

Van Gogh Image				
No.	τ	Iterations	Run-Time	Energy
1	0.81	75	1.23	3.993.880
2	0.85	78	1.34	3.993.780
3	0.76	79	1.23	3.993.910
4	0.73	93	1.45	3.993.770
		⋮	⋮	
15	0.93	123	1.79	3.993.660

Table 5.1: Where the estimate $\tau = 0.93$ is under the five fastest approximations for the Lena and Hepburn image, it is only on fifteenth position using the Van Gogh image. Setting $\tau = 0.73$ gives us a better guarantee for fast convergence.

This factor enters our function in the discrete version of ∇ and ∇^T. It does not change the energy, since its a constant factor, but influences the scaling factor λ.

5.2.3 Best τ estimation

Unfortunattely, estimating the best τ is also a difficult task. It turns out, that τ couples with λ and depends on the image itself. Finding a τ, which delivers fast convergence is not impossible, but one has to guess instead of being able to verify a perfect estimate. We found, that for our perfect fit $\lambda = 0.1$ the best choice for the time-step can either be $\tau = 0.73$ or $\tau = 0.93$. We found these two values by running tests for τ starting at 0.01 and inreasing it by 0.01 till we reach the value 0.99. For the tests we used three different images: Lena (grayscaled) and Hepburn, Van Gogh (both RGB). Since, there was no big difference in the estimated energy, we only focussed on iteration steps in our comparison. In

We also tested some other images to see, if these two values are really the best option. It turned out, that it completely depends on the underlying image. In some cases convergence was attained within few iterations, but in other cases it took up to 200 iterations. Of course, this is not a large amount of iterations, but knowing nothing about the best time-step τ beforehand makes the proposed approach a bit incosistent. Nonetheless, having a stable framework like this to minimize the energy for the ROF model is a great approach and yields good approximations u of the input images g.

5.3 Image Approximation using the TVL1 Model

We now turn our focus on the TVL1 model. Fortunately, we can adapt a lot of the previous section. The gradient operators and computation of the proximity operator for $F^*(p) = \delta_P(p)$ remain completely the same. Also the primal-dual algorithm with the extrapolation step are consistent and can be used. The only difference to the ROF model is the computation of $(\mathrm{Id} + \tau\,\partial\, G)^{-1}(\tilde{u})$. In equation **??** we showed, that it is of the form

$$u = (\mathrm{Id} + \tau\,\partial\, G)^{-1}(\tilde{u}) \Longleftrightarrow u_{i,j} = \begin{cases} \tilde{u}_{i,j} - \tau\lambda & \text{if } \tilde{u}_{i,j} - g_{i,j} > \tau\lambda, \\ \tilde{u}_{i,j} + \tau\lambda & \text{if } \tilde{u}_{i,j} - g_{i,j} < -\tau\lambda, \\ g_{i,j} & \text{if } |\tilde{u}_{i,j} - g_{i,j}| \leq \tau\lambda. \end{cases}$$

Now, the only thing we need to change in computing the function in algorithm **??** is the last line of code in the inner for-loop. In this line we compute the proximity operator for that change it to the one of the TVL1 model. We obtain:

Algorithm 5.5 (Primal Descent) *Summarizing again in a function, we have*

```
template<typename T>
void primal_desc(T* p_x, T* p_y, T* u, T* g, float tau, float lambda, int M, int N, int C)
  T dx, dy, sum;
  int index;
  for (int k = 0; k < C; k++) {
    for (int i = 0; i < M; i++) {
      for (int j = 0; j < N; j++) {
        index = j + i * N + k * M * N;
        dx = (i+1<M ? p_x[index] : 0.f) -
             (i>0 ? p_x[j + (i-1) * N + k * M * N] : 0.f);
        dy = (j+1<N ? p_y[index] : 0.f) -
             (i>0 ? p_y[(j-1) + i * N + k * M * N] : 0.f);
        sum = tau * (dx + dy);
        sum += u[index];
        if (sum - g[i] > tau*lambda) u[i] = sum - tau*lambda;
        if (sum - g[i] < -tau*lambda) u[i] = sum + tau*lambda;
        if (fabs(sum - g[i]) <= tau*lambda) u[i] = g[i];
      }
    }
  }
}
```

where C is the number of color channels and the template value T is mostly used as float.

This substitution is all it takes and we are ready to run the code and approximate the image g by u.

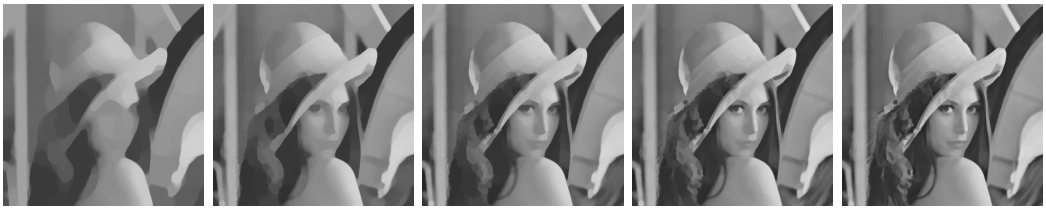
5.3.1 Best λ estimation

For the TVL1 model, estimating a good λ was an easy task compared to the ROF model. The reason for this is, that in [**?**] they proposed this model without the scaling factor h,

Lena Image			
τ	Iterations	Run-Time	Energy
0.98	245	2.27	1.576.060
Hepburn Image			
τ	Iterations	Run-Time	Energy
0.98	328	7.17	5.635.150
Van Gogh Image			
τ	Iterations	Run-Time	Energy
0.98	380	3.76	3.987.510

Table 5.2: The results for the best estimate $\tau = 0.98$.

like we did in this work. They already suggested to set $\lambda = 0.7$. We adapted this idea and tested all values from 0.1 to 1 by increasing λ for each approximation by 0.1. This evolution process is shown in figure **??**. At the end, we can verify that $\lambda = 0.7$ is the best choice.



(a) $\lambda = 0.1$ (b) $\lambda = 0.2$ (c) $\lambda = 0.3$ (d) $\lambda = 0.4$ (e) $\lambda = 0.5$

(f) $\lambda = 0.6$ (g) $\lambda = 0.7$ (h) $\lambda = 0.8$ (i) $\lambda = 0.9$ (j) $\lambda = 1$

Figure 5.3: Approximation of the Lena image with the TVL1 Model. Again, the higher λ the closer the outcome of the algrithm to the original image.

5.3.2 Best τ estimation

We applied the same procedure to estimate the best time-step parameter τ as in subsection **??**. We used again the images Lena, Hepburn and Van Gogh for the evaluation. There is only one parameter, which appeared under the sixth fastest convergence rates: $\tau = 0.98$, c.f. table **??**.

Testing this framework with other images lead again to different results. As before, finding a best fit for the time-step parameter is an impossible task.

5.4 Approximation using the Real-Time Minimizer

In the case of the real-time minimizer for the Mumford-Shah model we can again adapt most of the code from the ROF model. Since, the function G is almost the same, except the scaling parameter $\frac{\lambda}{2}$, we only exchange one line of code. The line

$$u[index] = (sum + tau * lambda * g[index])/(1.f + tau * lambda);$$

becomes

$$u[index] = (sum + 2 * tau * g[index])/(1.f + 2 * tau);.$$

Everything else in algorithm **??** remains the same.

In algorithm **??** we only need to exchange two lines of code, where we take into account that the proximity operator for $R^*_{MS}(p)$ is computed by

$$p = \left(\mathrm{Id} + \sigma \partial R^*_{MS}\right)^{-1}(\tilde{p}) \Longleftrightarrow p_{i,j} = \begin{cases} \dfrac{\lambda}{\lambda + \sigma}\tilde{p}, & \text{if } ||\tilde{p}||_2 \leq \sqrt{\frac{\nu}{\lambda}\sigma(\sigma + 2\lambda)}, \\ 0 & \text{else.} \end{cases}$$

We observe the following code:

Algorithm 5.6 (Dual Ascent) *Summarizing this procedure in a function, we get*

```
template<typename T>
void dual_asc(T* p_x, T* p_y, T* u_bar, float sigma, float lambda, float nu, int M, int N,
  T dx, dy;
  int index;
  aType factor = (2 * lambda) / (sigma + 2 * lambda);
  aType bound = sqrt((nu / lambda) * sigma * (sigma + 2 * lambda));
  for (int k = 0; k < C; k++) {
    for (int i = 0; i < M; i++) {
      for (int j = 0; j < N; j++) {
        index = j + i * N + k * M * N;
        dx = i+1<M ? u_bar[j + (i+1) * N + k * M * N]-u_bar[index] : 0;
        dy = j+1<N ? u_bar[(j+1) + i * N + k * M * N]-u_bar[index] : 0;
        dx = p_x[index] + sigma * dx;
        dy = p_y[index] + sigma * dy;
        p_x[index] = (dx*dx+dy*dy) <= bound ? factor * dx : 0;
        p_y[index] = (dx*dx+dy*dy) <= bound ? factor * dy : 0;
      }
    }
  }
}
```

where C is the number of color channels and the template value T is mostly used as float.

As we used the second algorithm of section **??**, namely algorithm **??**, we additionally propose this version in C++ code:

Algorithm 5.7 (Primal-Dual Algorithm) *In a function we have*

```
template<typename T>
void RealTimeMinimizer(T* u, T* g, float lambda, float nu, int M, int N, int C) {
  float tau = 1.f / 4.f;
  float sigma = 1.f / 2.f;
  float theta = 2.f;
  T* u_bar = new T[M*N*C];
  T* u_prev = new T[M*N*C];
  T* p_x = new T[M*N*C];
  T* p_y = new T[M*N*C];

  void dual_asc(p_x, p_y, u_bar, sigma, lambda, nu, M, N, C);
  void primal_desc(p_x, p_y, u, g, tau, M, N, C);
  theta = (aType)1 / sqrt(1 + 4 * tau); tau *= theta; sigma /= theta;
  void extrapolation(u_bar, u, u_prev, theta, M, N, C);

  delete [] u_bar;
  delete [] u_prev;
  delete [] p_x;
  delete [] p_y;
}
```

where C is the number of color channels and the template value T is mostly used as float.

It is left to evaluate a good λ and ν.

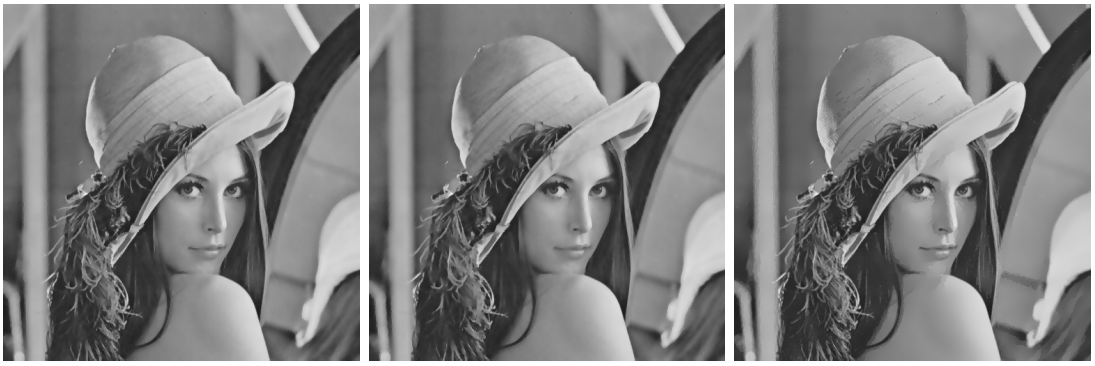
5.4.1 Best λ and ν estimation

For this model we have two parameters which can be combined in a lot of ways. Finding parameters λ and ν took several estimation steps. We will not provide the whole procedure of this process, since it could fill a huge amount of pages. But, we show and discuss the results of the estimation.

λ	ν	Iterations	Run-Time	Energy
2	0.02	260	1.72	527,42
2	0.03	260	1.71	579,89
20	0.03	260	2.05	1.746,89
20	0.04	260	1.95	1.895,69
500	0.07	260	1.93	3.094,23
500	0.08	260	1.81	3.132,59

The corresponding images to the proposed values for λ and ν can be seen in figure **??**.

We tested the cases in which $\lambda = \{2, 20, 500\}$. Choosing $\lambda = 2$ yields to the piecewise smooth approximation u. It is close to the input image g and the values set for ν seek for sharp edges in u. If $\lambda = 20$ the approximation turns slowly to a piecewise constant case. The output of the algorithm is still a quite smooth approximation, but some constant parts slightly appear. Again, ν controls the edges being sharp. The last case resembles the piecewise constant case. We see constant areas in the approximation. This setting is also useful for the cartooning case. For this, we then additionally propose a method for edge highlighting using the set K_{MS}.



(a) $\alpha = 2, \lambda = 0.02$ (piecewise smooth) (b) $\alpha = 2, \lambda = 0.03$ (piecewise constant) (c) $\alpha = 20, \lambda = 0.03$ (piecewise constant)

(d) $\alpha = 20, \lambda = 0.04$ (piecewise smooth) (e) $\alpha = 500, \lambda = 0.07$ (piecewise constant) (f) $\alpha = 500, \lambda = 0.08$ (piecewise constant)

Figure 5.4: Approximation of Lena with the Mumford-Shah Model.

There is no need to find a good estimation of the time-step τ, because this value is updated in each iteration seperately and is coupled with the parameter θ.

We proposed several image approximations u of a input image g. Further, we showed good estimates for the parameters in each model and the time-step value τ. We turn our interest no to one application to imaging, namely cartooning.

5.5 Image Cartooning

In this section we present a technique to turn an input image g into a cartooned image u. For this we will make use of the ROF Model, TVL1 Model and most of all the real-time minimizer for the Mumford-Shah Function, presented in section **??**. Using the model of Rudin, Osher and Fatemi and the TVL1 Model meant for us to find the right parameter λ to smooth the image enough, but still preserve edges. We will exchange the data fidelity term G, by another term G_γ^q. This leads to a nice cartoon representation of our image. For this, we first propose the new data fidelity term and then we compute the corresponding proximity operator, which depends on the function G_γ^q.

5.5.1 A new data fidelity term

Recalling the Mumford-Shah energy function we had

$$E_{MS}(u) = ||u-g||_2^2 + \sum_{i=1}^{N}\sum_{j=1}^{M} \min(\lambda|(\nabla u)_{i,j}|, \nu),$$

where we set the data fidelity term to $G(u) = ||u-g||_2^2$. The idea for cartooning is now to replace G with a function G_γ^q defined by

$$G_\gamma^q(u) := ||u-g||_2^2 + \gamma||u-u^{\text{prev}}||_q^q. \tag{5.2}$$

Here, u^{prev} is the u, which was obtained in the previous iteration, hence the notation. The additional l_q regularization aims to handle the tradeoff between the current and the previous estimate of u. It forces u not to change too much during the iterations process. But as mentioned above, we need to compute the proximity operator with respect to the new function G_γ^q. Using again **??**, we obtain

$$u = (\text{Id} + \tau\partial G_\gamma^q)^{-1}(\tilde{u}) = \arg\min_{u\in X} \frac{||u-\tilde{u}||_2^2}{2} + \tau\left(||u-g||_2^2 + \gamma||u-u^{\text{prev}}||_q^q\right). \tag{5.3}$$

As one can see, the new proximity operator is dependent on how the norm in the additional term is chosen. We are looking at the cases where we have $q = 1$ and $q = 2$. In the appendix of [**?**] they suggest using $q = 1.5$ and therefore also compute the proximal operator. In our tests, this did not have that big impact, for that we choose to use the l_1 and l_2 norm, respectively.

Let us first consider the case where $q = 2$. Then we obtain for the proximal operator

$$\begin{aligned}
u = (\text{Id} + \tau\partial G_\gamma^q)^{-1}(\tilde{u}) \quad &= \quad \arg\min_{u\in X} \frac{||u-\tilde{u}||_2^2}{2} + \tau\left(||u-g||_2^2 + \gamma||u-u^{\text{prev}}||_2^2\right) \\
&\iff \quad 0 \in \partial\left(\frac{||u-\tilde{u}||_2^2}{2} + \tau\left(||u-g||_2^2 + \gamma||u-u^{\text{prev}}||_2^2\right)\right) \\
&\iff \quad 0 \in u - \tilde{u} + 2\tau(u-g) + 2\tau\gamma(u-u^{\text{prev}}) \\
&\iff \quad u\big(1 + 2\tau + 2\tau\gamma\big) = \tilde{u} + 2\tau g + 2\tau\gamma u^{\text{prev}} \\
&\iff \quad u = \frac{\tilde{u} + 2\tau g + 2\tau\gamma u^{\text{prev}}}{1 + 2\tau + 2\tau\gamma}
\end{aligned}$$

Overall, we obtain pointwise for all $i = 1, ..., N$ and $j = 1, ..., M$

$$u_{i,j} = \frac{\tilde{u}_{i,j} + 2\tau g_{i,j} + 2\tau\gamma u^{\text{prev}_{i,j}}}{1 + 2\tau + 2\tau\gamma} \tag{5.4}$$

The other option was to set $q = 1$. In this case we obtain the non-differentiable l_1 norm and we need to make use of the subgradient. We need to do a case analysis.

For this, let y_i be the subgradient of the term $||u - u^{\text{prev}}||_1$ in the i-th component with $y_i \in [-1, 1]$. Then we get with the the equations for the proximity operator:

$$\begin{aligned} u = (\mathrm{Id} + \tau \partial G_\gamma^q)^{-1}(\tilde{u}) \quad &= \quad \arg\min_{u \in X} \frac{||u - \tilde{u}||_2^2}{2} + \tau \left(||u - g||_2^2 + \gamma ||u - u^{\text{prev}}||_1^1 \right) \\ &\iff \quad 0 \in \partial \left(\frac{||u - \tilde{u}||_2^2}{2} + \tau \left(||u - g||_2^2 + \gamma ||u - u^{\text{prev}}||_1 \right) \right) \\ &\iff \quad 0 \in u - \tilde{u} + 2\tau(u - g) + \tau\gamma y \end{aligned}$$

We will look at the i-the row of this equation.

1. Consider the first case and assume that $u_i - u_i^{\text{prev}} > 0$. Then clearly, $y_i = 1$ and

$$\begin{aligned} 0 \in u_i - \tilde{u}_i + 2\tau(u_i - g_i) + \tau\gamma \quad &\iff \quad u_i(1 + 2\tau) = \tilde{u}_i + 2\tau g_i - \tau\gamma \\ &\iff \quad u_i = \frac{\tilde{u}_i + 2\tau g_i - \tau\gamma}{1 + 2\tau}. \end{aligned}$$

   Then, the last equation implies

$$\begin{aligned} \frac{\tilde{u}_i + 2\tau g_i - \tau\gamma}{1 + 2\tau} - u^{\text{prev}} > 0 \quad &\iff \quad \tilde{u}_i + 2\tau g_i - \tau\gamma - u^{\text{prev}}(1 + 2\tau) > 0 \\ &\iff \quad \tilde{u}_i + 2\tau g_i - u^{\text{prev}}(1 + 2\tau) > \tau\gamma. \end{aligned} \tag{5.5}$$

2. Assuming that $u_i - u_i^{\text{prev}} < 0$ leads to a similar expression, but we need to exchange the sign in front of the term $\tau\gamma$, since in this case it holds that $y_i = -1$. Overall, we obtain

$$u_i = \frac{\tilde{u}_i + 2\tau g_i + \tau\gamma}{1 + 2\tau}.$$

   As in 1. we can compute

$$\begin{aligned} \frac{\tilde{u}_i + 2\tau g_i + \tau\gamma}{1 + 2\tau} - u^{\text{prev}} < 0 \quad &\iff \quad \tilde{u}_i + 2\tau g_i + \tau\gamma - u^{\text{prev}}(1 + 2\tau) < 0 \\ &\iff \quad \tilde{u}_i + 2\tau g_i - u^{\text{prev}}(1 + 2\tau) < -\tau\gamma. \end{aligned} \tag{5.6}$$

3. In the last case we assume that the equality $u_i = u_i^{\text{prev}}$ holds. Then we have that $y_i \in [-1, 1]$ can be set arbitrary. To show in which case this setting holds we plug y_i into the equation, set $u_i = u_i^{\text{prev}}$ and observe

$$\begin{aligned} 0 \in u_i^{\text{prev}} - \tilde{u}_i + 2\tau(u_i^{\text{prev}} - g_i) + \tau\gamma y_i \quad &\iff \quad u_i^{\text{prev}} - \tilde{u}_i + 2\tau(u_i^{\text{prev}} - g_i) = -\tau\gamma y_i \\ &\iff \quad \tilde{u}_i - u_i^{\text{prev}}(1 + 2\tau) + 2\tau g_i = \tau\gamma y_i \\ &\iff \quad |\tilde{u}_i - u_i^{\text{prev}}(1 + 2\tau) + 2\tau g_i| = |\tau\gamma y_i| \leq \tau\gamma \end{aligned}$$

Overall, we have the following representation of the proximity operator to this problem:

$$u = (\mathrm{Id} + \tau\, \partial\, G_\gamma^q)^{-1}(\tilde{u}) \iff \tag{5.7a}$$

$$u_{i,j} = \begin{cases} \dfrac{\tilde{u}_{i,j} + 2\tau g_{i,j} - \tau\gamma}{1 + 2\tau} & \text{if } \tilde{u}_{i,j} + 2\tau g_{i,j} - u^{\text{prev}}(1 + 2\tau) > \tau\gamma, \\ \dfrac{\tilde{u}_{i,j} + 2\tau g_{i,j} + \tau\gamma}{1 + 2\tau} & \text{if } \tilde{u}_{i,j} + 2\tau g_{i,j} - u^{\text{prev}}(1 + 2\tau) < -\tau\gamma, \\ u_{i,j}^{\text{prev}} & \text{if } |\tilde{u}_{i,j} - u_{i,j}^{\text{prev}}(1 + 2\tau) + 2\tau g_{i,j}| \leq \tau\gamma, \end{cases} \tag{5.7b}$$

for all $i = 1, ..., N$, $j = 1, ..., M$.

5.5.2 Edge Highlighting

Another idea Strekalovskiy and Cremers proposed in [**?**] was to use the edge set K_{MS} in order to highlight edges in the smoothed images.

Assume that $x \in K_{MS}$ then we have $|\nabla u(x)| \geq \sqrt{\frac{\nu}{\lambda}}$ by definition. This notation is equivalent to setting

$$\frac{|\nabla u(x)|}{\sqrt{\frac{\nu}{\lambda}}} \geq 1.$$

Further, we have by discretizing the operator ∇ with forward differences that

$$|\nabla u(x)| = \max_{|u(x)| \leq 1} |\nabla u(x)| = \max_{|x| \leq 1} |\frac{u(x + \delta x) - u(x)}{\delta x}|.$$

This means we have $|\nabla u(x)| \leq \sqrt{2}$ and for that

$$1 \leq \frac{|\nabla u(x)|}{\sqrt{\frac{\nu}{\lambda}}} \leq \frac{\sqrt{2}}{\sqrt{\frac{\nu}{\lambda}}}.$$

Applying the logarithm to each of the terms we get

$$0 \leq \log\left(\frac{|\nabla u(x)|}{\sqrt{\frac{\nu}{\lambda}}}\right) \leq \log\left(\frac{\sqrt{2}}{\sqrt{\frac{\nu}{\lambda}}}\right).$$

If we now divide each term by the last one, we observe

$$0 \leq \frac{\log\left(\frac{|\nabla u(x)|}{\sqrt{\frac{\nu}{\lambda}}}\right)}{\log\left(\frac{\sqrt{2}}{\sqrt{\frac{\nu}{\lambda}}}\right)} \leq 1.$$

With these calculations, we now define a $v \in [0, 1]$ by

$$v := \frac{\log\left(\frac{|\nabla u(x)|}{\sqrt{\frac{\nu}{\lambda}}}\right)}{\log\left(\frac{\sqrt{2}}{\sqrt{\frac{\nu}{\lambda}}}\right)},$$

then, since ν and λ are constant values, this parameter v only increases if $|\nabla u(x)|$ increases and decreases if $|\nabla u(x)|$ decreases. For this reason v can serve as an edge indicator. Then the idea proposed by [**?**] is to multiply each RGB value $u(x)$ by

$$1 - v \in [0, 1],$$

if $x \in K_{MS}$. On the other hand, if $x \notin K_{MS}$ the value $u(x)$ will not be changed. Then points in the imaged domain Ω with strong edges are painted darker, as those where we do not find edges.

5.5.3 Image Comparison

In this subsection we want to present solutions of the three models: ROF, TVL1 and real-time Mumford-Shah minimizer. We will compare run-time of the models - on a CPU and GPU - and provide the best estimations for the parameters λ, ν and γ. We start with the ROF Model:

(a) $\lambda = 0.01$ (b) $\lambda = 0.35$

Figure 5.5: Comparison of cartooning applying the ROF Model (left) and the TVL1 Model (right) to the Audrey Hepburn image

(a) $\lambda = 0.01$ (b) $\lambda = 0.35$

Figure 5.6: Comparison of cartooning applying the ROF Model (left) and the TVL1 Model (right) to the Lena image

One can see that the difference of these two models becomes visible at the edges. As the ROF Model really smoothes the image and provides pretty good cartooned images, the TVL1 Model preserves more edges and details in the images. But, what both models

have in common: the cartoon images do not fit the imagination one has when thinking about cartoons. Therefore, we not only applied the real-time minimzer for the Mumford-Shah model to the images, we also made use of the above data fidelity term G_γ^q and the method for edge highlighting. We start with $q = 1$.

(a) $\alpha = 500, \lambda = 0.4, \gamma = 1$ (b) $\alpha = 500, \lambda = 0.4, \gamma = 10$

(c) $\alpha = 500, \lambda = 0.4, \gamma = 20$ (d) $\alpha = 500, \lambda = 0.5, \gamma = 50$

Figure 5.7: Cartooning of.

By estimating the best fits for our parameters we ran a huge amount of tests. Using the framework presented for the Mumford-Shah Model, we observed a lot of possible images and chose the ones, which fitted best for us. Finding nicely cartooned images for the ROF and TVL1 Model, respectively, was a harder task. We obtained also a large number of images, but only a few fitted to be considered as cartoon images. Overall, the real-time minimizer is the model to choose, if one seeks for cartooned images or movies.

5.5.4 Computational Comparison

In this part of the section we present the run-times for the three models for the best fitted parameters for cartooning. We also provide the best time-step parameter τ for the ROF and TVL1 model, the iterations to convergence and the final (minimal) energy of the model. Further, we discuss some implementation issues and the comparison of running the algorithms on a CPU and GPU.

For this computational comparison we use the Audrey Hepburn image with the parameters presented in subsection **??**. For the Mumford-Shah Model the parameter τ is set to 0.25 by definition of th proposed algorithm in section **??**.

Model	τ	CPU time	GPU time	Iterations	Energy
ROF	0.25				
ROF	0.5				
ROF	0.97				
TVL1	0.25				
TVL1	0.5				
TVL1	0.97				
Mumford-Shah	0.25				
Mumford-Shah	0.25				
Mumford-Shah	0.25				

6 Conclusion