Georg-August-Universität Göttingen
Fakultät für Mathematik und Informatik

Inertial Proximal Algorithms in Diffusion-based Image Compression

Master's Thesis

submitted by
Rebecca Nahme
born in Göttingen

November 2015

Supervisor:
Prof. Dr. David Russell Luke

Second Assessor:
Prof. Dr. Gerlind Plonka-Hoch

Contents

1 Introduction

In the 19th century, the method of gradient descent emerged. Since then it has been improved and specialized in several ways to solve smooth minimization problems. The assumption concerning the objective function to be differentiable heavily restricts its scope of application. Though, this classical method of gradient descent still lays foundation for recent research.

A similarly good and general approach to nonsmooth problems is harder to find. It requires additional assumptions. Beyond differentiability, there exist properties of objective functions that make solving the corresponding minimization problem more accessible. Back to the 1960s, the topic of convex analysis arose from nonlinear analysis ([Roc70]). Within this framework, the proximal map was originally developed ([Mor65]). It facilitates to solve the special class of nonsmooth problems minimizing a convex objective function.

With an expansion of the proximal map to nonconvex settings, it also enables minimizing nonsmooth nonconvex functions [RW98, BC11]. But, its computation may be found difficult.

In general, it stays challenging to create an efficient solver for nonsmooth nonconvex optimization problems. Nowadays, some approachs combine the proximal map with gradient methods to take advantage of special structures of the initial problem formulation.

In this thesis, we present two algorithms to solve specific classes of nonsmooth nonconvex optimization problems in Chapter 4. The first algorithm, the "Inertial Proximal Algorithm for Nonconvex Optimization" (iPiano), is introduced in Section 4.1. It solves the minimization problem concerning a sum of a smooth, possibly nonconvex function and a convex, possibly nonsmooth function. To take advantage of the properties of each function, the algorithm performs a forward and a backward step in every iteration. It combines a refinement of the gradient descent applied to the differentiable function and the proximal map of the convex function. The algorithm iPiano assured remarkable performance at the time of its publication in [OCBP14].

The second algorithm, the "Proximal Alternating Linearized Minimization" (PALM), follows in Section 4.2. It can be applied to a more general class of nonsmooth nonconvex optimization problems than iPiano. Its basic idea is to optimize with respect to only a group of the variables and not to all simultaneously. The variables are separated into groups that are minimized consecutively and alternately. Each update step combines the gradient descent and the proximal map as described in [BST14].

In addition, we suggest a slight modification of the algorithm PALM. It is extended by adding an inertial term to the part based on the gradient descent. This "Inertial Proximal Alternating Linearized Minimization" (iPALM) is introduced in Section 4.2.4.

In practice, algorithms minimizing nonsmooth nonconvex sums of finitely many functions are of great interest for various applications. These composite minimization problems can be found in many areas like machine learning, signal and image analysis.

The concrete example stating the initial problem for this thesis is the linear diffusion-based image compression ([HSW13, HMH+15]). We derive and describe it in Chapter 3. The major goal is to achieve optimal image compression with minimal loss of information and quality.

In Chapter 5, we apply all three algorithms, iPiano, PALM and iPALM, to the problem of linear diffusion-based image compression. We express the initial problem formulation in a suitable setup for each algorithm and calculate the required operators. Afterwards, we analyze the algorithms in this framework with concrete numerical experiments.

But first of all, in the following Chapter 2, we lay the general theoretical foundation for the further work.

2 Preliminaries

In this chapter, we introduce notations and the theoretical foundation of this thesis. We start by defining some properties and concepts of functions. Afterwards, we use them to formulate the initial optimization problem for this thesis and provide optimality conditions. In the end, we present some basic optimization tools and methods which can be used to solve various special cases of the initial optimization problem.

2.1 Functions

Throughout the whole thesis, we work on the underlying space $\mathbb{R}^n$, $n \in \mathbb{N}$, equipped with the 2-norm (Euclidean norm) given by $\|x\|_2 := \sqrt{\sum_{i=1}^n |x_i|}$ for $x \in \mathbb{R}^n$. This setting suffices for our later applications. Furthermore, let

$$\overline{\mathbb{R}} := [-\infty, +\infty] = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$$

denote the extended real line.

Let $X \subset \mathbb{R}^n$ and $Y \in \mathbb{R}^m$, $m \in \mathbb{N}$, be nonempty sets. By using the familiar notation $f : X \to Y$ for a function f from X to Y, we mean a single-valued function, i.e. every $x \in X$ is mapped to the singleton $\{y\} \in Y$.

Definition 2.1.1 (domain, lower level set [BC11, Def. 1.4]). *Let $f : X \to \overline{\mathbb{R}}$ be a function. The* (effective) domain *of f is defined by*

$$\operatorname{dom} f := \{x \in X \mid f(x) < +\infty\}$$

and the lower level set *of f at height $t \in \mathbb{R}$ is given by*

$$\operatorname{lev}_{\leq t} f := \{x \in X \mid f(x) \leq t\}.$$

The function f is said to be proper, *if* $\operatorname{dom} f \neq \emptyset$ *and $f(x) > -\infty$ for all $x \in X$.*

Demanding a function $f : X \to \overline{\mathbb{R}}$ to be proper guarantees the existence of at least one element $x \in X$ with a finite function value $f(x)$. This is very beneficial, especially considering an optimization problem that aims at minimizing a given function - what is our intention in later chapters.

The following definition provides a group of functions which is already well-analyzed. Convex functions possess nice properties with regard to continuity, differentiability and other aspects. More details on convex analysis can be found in [Roc70].

Definition 2.1.2 (convex set, convex function [RW98, Def. 2.1])**.**

1. *A subset $C \subset \mathbb{R}^n$ is called* convex *if for every pair $x, y \in C$ also their connecting line segment $[x, y] := \{\lambda x + (1-\lambda)y \mid \lambda \in [0,1]\}$ lies in C.*

2. *A function f on a convex set C is said to be* convex *relative to C if for every pair $x, y \in C$ it holds that*
$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y).$$

Example 2.1.3 (convexity of 1-norm)**.** *Let the 1-norm of a vector $x \in \mathbb{R}^n$ be given by the function $\|\cdot\|_1 : \mathbb{R}^n \to \mathbb{R}$,*
$$\|x\|_1 := \sum_{i=1}^{n} |x_i|.$$
Then, by the triangle inequality on $\mathbb{R}$, the 1-norm $\|\cdot\|_1$ is convex.

In further applications, we especially deal with nonconvex objective functions. This makes the respective optimization problem more challenging.

But, in general, we require the objective function to be at least lower semicontinuous.

Definition 2.1.4 (lower semicontinuity [BC11, Thm. 1.24])**.** *A function $f : X \to \overline{\mathbb{R}}$ is called* lower semicontinuous *at every point in X if the lower level sets $\operatorname{lev}_{\leq t} f$ are closed in X for all t.*

The concept of lower semicontinuity describes a weaker form of the usual continuity. In contrast, the following Lipschitz continuity defines stronger requirements towards a function's behavior.

Definition 2.1.5 (Lipschitz continuity [RW98, Def. 9.1])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function and $D \subset \mathbb{R}^n$ a subset. Function f is* Lipschitz continuous *on D if there exists a nonnegative finite constant $L \in \mathbb{R}$ with*
$$\|f(x) - f(y)\|_2 \leq L\|x - y\|_2 \text{ for all } x, y \in D.$$
If L is the smallest real number with this property, then L is called the Lipschitz constant *for f on D.*

Intuitively, Lipschitz continuity limits how fast a function can change over any interval of its domain. Let us illustrate this with an example.

Example 2.1.6 (Lipschitz continuity of 2-norm)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = \|x\|_2$ be the 2-norm. Then by the triangle inequality*
$$|f(x) - f(y)| = |\|x\|_2 - \|y\|_2| \leq \|x - y\|_2 \text{ for all } x, y \in \mathbb{R}^n.$$
Thus, f is Lipschitz continuous with Lipschitz constant $L \leq 1$. Since equality is achieved for example in the case of $(x, y) \in \mathbb{R}^n \times \{0\}^n$, the Lipschitz constant is given by $L = 1$.

There exists a stronger property of functions than the Lipschitz continuity to describe how fast a function decreases or increases. This leads us to the concept of gradients. Later, we use them to find the direction of the steepest descent of a function at a given point. Further, we can tackle optimization problems with them to minimize a function.

Definition 2.1.7 (Fréchet differentiability, gradient [BC11, Def. 2.45]). *Let $x \in \mathbb{R}^n$, $C \subset \mathbb{R}^n$ be a neighbourhood of x and $f : C \to \mathbb{R}^m$. Then f is* Fréchet differentiable *at x if there exists a bounded linear operator $\mathrm{D}f(x) : \mathbb{R}^n \to \mathbb{R}^m$ with $\operatorname{dom} \mathrm{D}f(x) = \mathbb{R}^n$, called the* Fréchet derivative *of f at x, such that*

$$\lim_{0 \neq \|y\| \to 0} \frac{\|f(x+y) - f(x) - \mathrm{D}f(x)y\|_2}{\|y\|_2} = 0.$$

The Fréchet gradient *of f at x is the unique vector $\nabla f(x) \in \mathbb{R}^n$ which fulfills for all $y \in \mathbb{R}^n$ the relation*

$$\mathrm{D}f(x)y = \langle y, \nabla f(x) \rangle.$$

In the case of functions which are not Fréchet differentiable, the subgradients defined next generalize the idea of gradients.

Definition 2.1.8 (subdifferential, subgradient [RW98, Def. 8.3]). *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a function with $\overline{x} \in \operatorname{dom} f$. For a vector $v \in \mathbb{R}^n$, define the following*

1. *The vector v is called a* regular subgradient *of f at $\overline{x}$, if*

$$\liminf_{x \to \overline{x}, x \neq \overline{x}} \frac{f(x) - f(\overline{x}) - \langle v, x - \overline{x} \rangle}{\|x - \overline{x}\|_2} \geq 0.$$

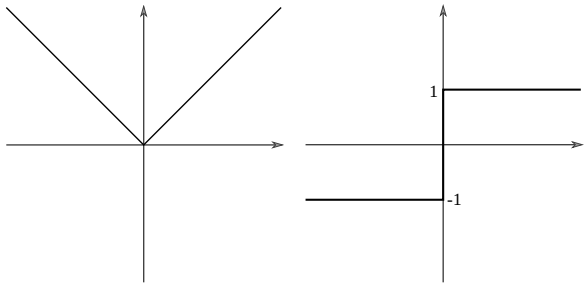
   The set of all regular subgradients v at $\overline{x}$ is called regular subdifferential *and denoted by $\hat{\partial} f(\overline{x})$.*

2. *The vector v is a* limiting subgradient *of f at $\overline{x}$, written $v \in \partial f(\overline{x})$, if there are sequences $x^k \to \overline{x}$ and $v^k \in \hat{\partial} f(\overline{x})$ with $v^k \to v$. The set of all limiting subgradients v at $\overline{x}$ is called the* limiting subdifferential *and denoted by $\partial f(\overline{x})$.*

The regular subgradient and the regular subdifferential are sometimes also called *Fréchet subgradient* and *Fréchet subdifferential*, respectively. From now on, if nothing else is said or logical, we use the short terms like (sub-) differentiability to refer to Fréchet (sub-) differentiability.

Note that, in general, the Fréchet as well as the limiting subdifferential define set-valued mappings. This means that at least for one $x \in \operatorname{dom} f$ we have $\#\{\hat{\partial} f(x)\} > 1$ or $\#\{\partial f(x)\} > 1$.

Let us illustrate the difference of gradients and subgradients given by the Definition 2.1.7 and 2.1.8 with the following example.



(a) graph of $|x|$ (b) graph of $\partial|x|$

Figure 2.1: absolute value function $|x|$ in $\mathbb{R}$ and its subdifferential $\partial|x|$

Example 2.1.9 (subdifferentiability of absolute value function)**.** *Let $g : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be the absolute value function or 1-norm in $\mathbb{R}$, namely*

$$g(x) = \|x\|_1 = |x| \text{ for } x \in \mathbb{R}.$$

Then, g is differentiable everywhere except at $x = 0$. There exists no gradient at this point. But, subgradients exist everywhere for g and the corresponding subdifferential looks like

$$\partial g(x) = \begin{cases} -1 & , x < 0 \\ [-1, 1] & , x = 0 \\ 1 & , x > 0. \end{cases}$$

Figure 2.1 shows the graphs of the absolute value function $g(x) = |x|$ and its set-valued subdifferential.

The next property provides a notion to bound the subgradient of a function from below. We need this sort of functions during the presentation of the convergence theory concerning the specific optimization algorithms.

Definition 2.1.10 (KL-property, KL-function [BST14, Def. 2.3])**.** *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper, lower semicontinuous function.*

1. *The function f is said to have the* KL-property *at a $\overline{x} \in \operatorname{dom} f$ if there exist a positive number $q \in (0, +\infty]$, a neighbourhood U of $\overline{x}$ and a continuous concave function $\varphi : [0, q) \to \mathbb{R}_{\geq 0}$ with $\varphi(0) = 0$, $\varphi \in C^1(0, q)$ and $\varphi'(y) > 0$ for all $y \in (0, q)$ such that for all x with*

$$x \in U \cap [f(\overline{x}) < f(x) < f(\overline{x}) + q]$$

the following inequality holds

$$\varphi'(f(x) - f(\overline{x})) \cdot \operatorname{dist}(0, \partial f(x)) \geq 1.$$

2. *If function f has the KL-property at each point of* $\operatorname{dom} \partial f$*, then f is called a* KL-function.

2.2 Optimization Problems

Gradients and subgradients are closely related to optimization problems. As they describe how a function behaves, they can be used to formulate optimality conditions.

The general formulation of the initial optimization problem for this thesis is the following.

Definition 2.2.1 (nonsmooth nonconvex optimization problem)**.** *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be proper, lower semicontinuous functions. Moreover, let $H : \mathbb{R}^n \times \mathbb{R}^m \to \overline{\mathbb{R}}$ define a so-called coupling function for $x \in X$, $y \in Y$ and nonempty sets $X \subset \mathbb{R}^n, Y \subset \mathbb{R}^m$. Then, the* nonsmooth nonconvex optimization problem *with respect to f, g and H is given by*

$$\min_{(x,y)\in X\times Y} f(x) + g(y) + H(x,y). \tag{2.1}$$

A special case of 2.1 is the problem in one variable, i.e. $x = y$, with a smooth function $f : \mathbb{R}^n \to \mathbb{R}$ and without coupling function, i.e. $H \equiv 0$,

$$\min_{x\in X} f(x) + g(x). \tag{2.2}$$

In the next theorem, we provide a generalized version of Fermat's rule to get a first-order optimality condition. Considering the specific form 2.2 of a nonsmooth nonconvex optimization problem, we also implement Fermat's rule to that case.

Theorem 2.2.2 (Fermat's rule [RW98, Thm. 10.1])**.** *Let $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper function having a local minimum at $\overline{x}$. Then*

$$0 \in \hat{\partial} h(\overline{x}) \text{ and } 0 \in \partial h(\overline{x}). \tag{2.3}$$

If h is convex, then 2.3 states not only a necessary but also sufficient condition for a global minimum. If $h = f + g$ with f being smooth, then

$$-\nabla f(\overline{x}) \in \partial g(\overline{x}). \tag{2.4}$$

Definition 2.2.3 (critical point)**.** *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper and lower semicontinuous function. A point $\overline{x} \in \mathbb{R}^n$ is called* critical point *if $0 \in \partial f(\overline{x})$ and limiting critical point if $0 \in \hat{\partial} f(\overline{x})$.*

Note that, by Theorem 2.2.2, each local minimum is a critical point. But, in general, a critical point can be a local minimum, maximum or saddle point. For more details on optimization problems and optimality conditions see for example [Pol97].

2.3 Optimization Tools and Methods

In this section, we provide basic methods to solve optimization problems. Each tool proposed has its advantages regarding simplicity, generality or performance in comparison to the others. Later, we see that a combination of the tools can compose algorithm setups which benefit from each different framework.

2.3.1 Gradient Methods

Gradient methods are the probably most famous methods to solve an optimization problem formulated with differentiable functions. They use the gradient of the objective function in each update step to compute the next iterate. The general initial optimization problem is to minimize a given proper differentiable function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, formally

$$\min_{x \in \mathbb{R}^n} f(x).$$

Definition 2.3.1 (gradient descent)**.** *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be differentiable, $x^0 \in \mathbb{R}^n$ and $(\alpha_k)_{k \in \mathbb{N}} \subset \mathbb{R}$. The method of the* gradient descent *iteratively performs the following update step for $k \geq 0$*

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k). \tag{2.5}$$

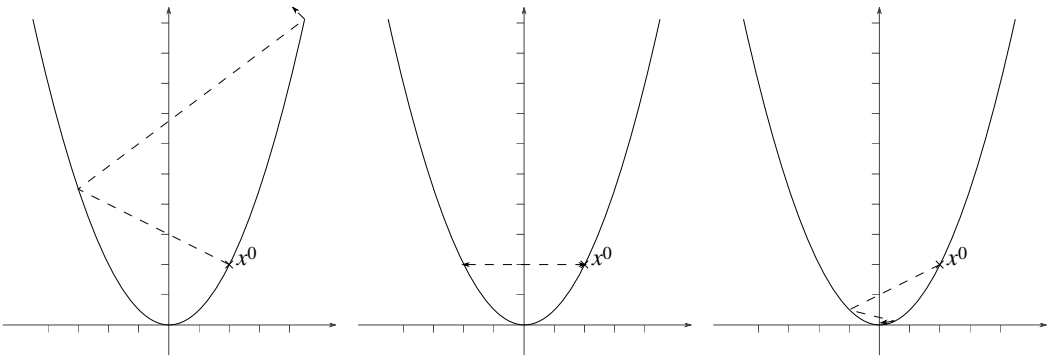
The idea is that, for updating x^{k+1} as suggested in 2.5, the gradient descent step yields $f(x^{k+1}) \leq f(x^k)$ for small α_k. So the next iterate is never worse than the current one. Thus it is likely that the generated sequence $(x^k)_{k \in \mathbb{N}}$ converges to the desired minimum of function f. The convergence heavily depends on the choice of the parameter α_k in each step what is illustrated in the following example.

Example 2.3.2 (convergence of gradient descent for $\frac{1}{2}x^2$)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be the function defined by $f(x) = \frac{1}{2}x^2$. Then, differentiation yields $\nabla f(x) = x$ and substitution into the general update step 2.5 results in $x^{k+1} = (1 - \alpha)x^k$. By induction, it follows $x^k = (1 - \alpha)^k x^0$ for all $k \geq 0$. In the following, we fix the parameter α, but examine the method's convergence in case of different choices for α. For each of the three cases, the corresponding sequence of iterates is indicated in Figure 2.2.*

1. *For $|1 - \alpha| > 1$, it is $|x^k| = |(1 - \alpha)^k x^0| = |(1 - \alpha)^k| \cdot |x^0| \to \infty$ for $k \to \infty$. Thus, the sequence diverges.*

2. *For $|1 - \alpha| = 1$, it is $|x^{k+1}| = |x^k|$ and $x^k \in \{-x^0, x^0\}$ for all $k \in \mathbb{N}$. Thus, the sequence stagnates at x^0 or alternates between x^0 and $-x^0$.*

3. *For $|1 - \alpha| < 1$, it follows that $|x^{k+1}| = |(1 - \alpha)x^k| = |(1 - \alpha)^k| \cdot |x^0| \to 0$ for $k \to \infty$. Moreover, $f(x^{k+1}) = (x^{k+1})^2 < (x^k)^2 = f(x^k)$. Hence, $(f(x^k))_{k \in \mathbb{N}} \to 0$ for $k \to \infty$ and the sequence of generated function values converges to the minimum.*



(a) $\alpha = 2.5$ implies diverging iterates (b) $\alpha = 2$ implies alternating iterates between $-3, 3$ (c) $\alpha = 1.5$ implies converging iterates

Figure 2.2: convergence of gradient descent for $f(x) = \frac{1}{2}x^2$ and starting point $x^0 = 2$

There exist several modifications and improvements of the gradient descent ([BT09], [GSJ13], [Pol97]). The variant we are interested in is the heavy-ball method ([ZK93]). It combines the usual gradient descent as in Definition 2.3.1 with an inertial term, a weighted difference of the current and the previous step of the generated sequence.

Definition 2.3.3 (heavy-ball method)**.** *Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be differentiable, $x^0 \in \mathbb{R}^n$ be some initial point and $(\alpha_k)_{k\in\mathbb{N}}, (\beta_k)_{k\in\mathbb{N}} \subset \mathbb{R}$ be sequences. Then, the* heavy-ball method *repeats the following update step*

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k) + \beta_k \left(x^k - x^{k-1}\right).$$

In physics, inertia describes the resistance of an object to any change in its state of motion involving speed and direction or just state of rest.

The convergence theory of the heavy-ball method has already been much discussed. In the general case, the convergence theory assumes at least that the objective function f possesses a Lipschitz continuous gradient ∇f. This general setting is what we are interested in for our later applications. But then, the choice of good step sizes α_n, β_n remains challenging. In the convex setting, optimal choices of parameters α_n, β_n can be determined by using the Lipschitz constant as well as the (strong) convexity parameter.

2.3.2 Proximal Methods

The whole setup of gradient methods is based on the existence of the gradient of the objective function. To overcome the assumption that the function is differentiable, we introduce the concept of the proximal map.

Therefore, let $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be a function which is not necessarily differentiable.

Definition 2.3.4 (proximal map [RW98, Def. 1.22]). *Let $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be a proper, lower semicontinuous function. For $\overline{x} \in \mathbb{R}^m$ and $\alpha > 0$, the* proximal map *of g is defined by*

$$\operatorname{prox}_{\alpha,g}(\overline{x}) := \operatorname*{arg\,min}_{x \in \mathbb{R}^m} \left(\frac{\alpha}{2} \|x - \overline{x}\|_2^2 + g(x) \right).$$

Note that there exist slightly different notions of the definition of the proximal map in the literature (compare for example [Mor65], [BC11, Def. 12.23], [OCBP14]). Sometimes, the proximal map of a function g is defined in terms of the resolvent with respect to ∂g. We want to distinguish these two terms by defining the resolvent as follows.

Definition 2.3.5 (resolvent [BC11, Def. 23.1]). *Let $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be a proper, lower semicontinuous function. For $\overline{x} \in \mathbb{R}^m$ and $\alpha > 0$, the* resolvent *of ∂g is given by $(Id + \alpha \cdot \partial g)^{-1}(\overline{x})$ where Id denotes the identity.*

Generally, the resolvent of a subdifferential of a function contains the proximal map of this function.

Lemma 2.3.6 (relation of proximal map and resolvent [BC11, Ex. 23.3]). *Let $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be a proper lower semicontinuous function, $\overline{x} \in \mathbb{R}^m$ and $\alpha > 0$. Then,*

$$\operatorname{prox}_{\alpha,g}(\overline{x}) \subset (Id + \alpha \cdot \partial g)^{-1}(\overline{x}).$$

In fact, for a proper, lower semicontinuous function that is also convex, its proximal map and the resolvent of its subdifferential coincide. This comes along with the correspondence of subdifferential and proximal map in the next proposition.

In the following, we present some properties of the proximal map. First of all, we want to mention that, in general, the proximal map defines a set-valued map, i.e. for a point $\overline{x} \in \mathbb{R}^m$, the respective proximal map $\operatorname{prox}_{\alpha,g}(\overline{x})$ could consist of more than one element of $\mathbb{R}^m$.

Since we aim at utilizing the proximal map in the context of optimization methods, the following relation to subdifferentials may be useful. Note that the proximal map turns out to be single-valued for convex functions. For more details on the proximal map and a wealth of fundamental results dealing with proximal algorithms in the convex setting, we refer to [BC11].

Proposition 2.3.7 (relation of proximal map and subdifferential in the convex setting [BC11, Prop. 16.34]). *Let $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be a proper, lower semicontinuous, convex function and $u, x \in \mathbb{R}^m$. Then,*

$$u = \operatorname{prox}_{1,g}(x) \;\Leftrightarrow\; x - u \in \partial g(u).$$

Thus, the equality $\operatorname{prox}_{1,f}(y) = (Id + \partial g)^{-1}(y)$ holds for all $y \in \mathbb{R}^m$.

The next example illustrates what the proximal map becomes for the trivial function $g \equiv 0$.

Example 2.3.8 (proximal map for $g \equiv 0$)**.** *Let $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be a function with $g \equiv 0$. Then, for every $\overline{x} \in \mathbb{R}^m$ and $\alpha > 0$, the* proximal map *of g equals the identity, i.e.*

$$\operatorname{prox}_{\alpha,g}(\overline{x}) = \operatorname{prox}_{\alpha,0}(\overline{x}) = \overline{x}.$$

Concluding, we want to interpret the proximal map from Definition 2.3.4 as a gradient descent step given in Definition 2.3.1. The necessary assumption is that, now, we are given a differentiable function g with positive definite second-order derivative $\nabla^2 g(x) = \nabla(\nabla g(x))$.

Theorem 2.3.9 (convergence of proximal map to gradient descent [PB13, sec. 3.3])**.** *Let g be a proper lower semicontinuous function. If g is twice differentiable at x with positive definite $\nabla^2 g(x)$, then, as $\lambda \to 0$, it holds that $\operatorname{prox}_{\lambda,g}(x) = x - \lambda \nabla g(x) + o(\lambda)$.*

For further reading about alternating proximal minimization algorithms to solve problems of Definition 2.2.1, see for example [ABRS10]. The authors assume H to be a C^1-function with Lipschitz continuous gradient ∇H on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$. We meet the latter assumption concerning Lipschitz continuity of the gradient ∇H on bounded domains again when presenting the convergence analysis of one of our algorithms.

2.3.3 Projection Methods

Until now, we have only introduced optimization methods that calculate a solution over the whole domain of the objective function. But the defined nonsmooth nonconvex optimization problem in Definition 2.2.1 also allows minimization of a function with respect to subsets of the domain. Thus, we need an operator to realize the limitation to a subset. This is where the following projection operator comes in.

Definition 2.3.10 (projection operator [BC11, Def. 3.7])**.** *Let $X \subset \mathbb{R}^n$ be a nonempty closed set. The* projection operator *onto X for every $v \in \mathbb{R}^n$ is defined by*

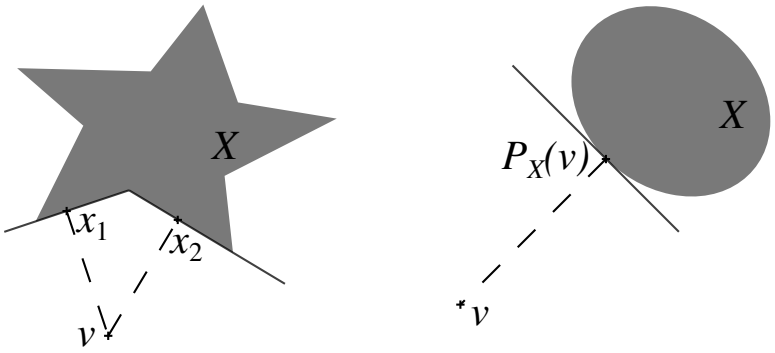
$$\mathrm{P}_X(v) := \operatorname*{arg\,min}_{x \in X} \left(\|x - v\|_2 \right).$$

Note that, in the framework of this thesis, the definition of the projection operator relies on the 2-norm of $\mathbb{R}^n$. Of course, a projection can also be defined in terms of any other norm of the underlying space.

In general, the projection operator is a set-valued map, i.e. there can exist several points x_1, x_2 in X which have the same shortest distance to v, i.e.

$$\|x_1 - v\|_2 = \|x_2 - v\|_2 = \min_{x \in X} \left(\|x - v\|_2 \right).$$

In the special case of a nonempty closed convex set C, the projection onto C becomes single-valued ([BC11, Thm. 3.14]). See Figure 2.3 for a visualization of projections onto



(a) projection onto a nonconvex set

(b) projection onto a convex set

Figure 2.3: set- and single-valued projection operators

convex and nonconvex sets. The following example shows the unique projection of a point onto any ball in $\mathbb{R}^n$.

Example 2.3.11 (projection onto ball)**.** *Let $y \in \mathbb{R}^n$ be the center of the ball $\mathrm{B}_\rho(y)$ with radius $\rho > 0$ and based on the 2-norm. Then, for any $x \in \mathbb{R}^n \backslash \mathrm{B}_\rho(y)$*

$$\mathrm{P}_{\mathrm{B}_\rho(y)}(x) = \rho \frac{x-y}{\|x-y\|_2} + y.$$

The projection operator can be used independently from other operators, as in alternating projections (like in [HLN14]) to tackle an optimization problem. But it can also be combined with other methods like a gradient step ([BC11, Cor. 27.10]). We are primarily interested in the relation of projection and proximal map as well as the combination of the proximal map with gradient methods.

Finally, we introduce the indicator function of a set. The respective proximal map of this function reduces to the projection onto the set. This relation is used in calculations of operators for optimization methods that are presented later in this thesis.

Definition 2.3.12 (indicator function)**.** *Let $X \subset \mathbb{R}^n$ be a set. The* indicator function *of X is given by*

$$\iota(x) = \begin{cases} 0 & , x \in X \\ +\infty & , x \notin X. \end{cases}$$

Example 2.3.13 (proximal map of indicator function)**.** *Let $C \subset \mathbb{R}^n$ be a nonempty closed set. For its corresponding indicator function ι_C, the proximal map equals the projection operator onto the set C, i.e. for $\alpha \in \mathbb{R}$ and $v \in \mathbb{R}^n$, it is*

$$\mathrm{prox}_{\alpha,\iota_C}(v) = \mathrm{P}_C(v).$$

3 Linear Diffusion-based Image Compression

This chapter outlines the idea of image analysis with a special focus on the process of inpainting. First, this process is formulated by using the framework of diffusion. Then, we discretize and optimize that formulation to get the homogeneous diffusion inpainting in the form of a constrained nonsmooth nonconvex optimization problem. The derived problem formulation serves as initial optimization problem which we want to solve with the algorithms introduced afterwards.

The idea of linear diffusion-based image compression presented throughout this chapter relies on the work of [HSW13] and [HMH+15].

3.1 Image Compression and Inpainting

A major challenge in data analysis is the reconstruction of a signal from a few given data points. This procedure is called inpainting. It refills data where information about the signal is missing for example due to interference or destructive compression.

In the digital world of images, the process of inpainting is also known as image interpolation. Nowadays, we store lots of images digitally instead of keeping them in a printed version. To save storage, they are compressed. That means that just a few information about an image are provided to enable its reconstruction. While decompressing, a gap can be refilled or interpolated by using the known information of its surroundings. The quality of the image restored depends on the level of destruction which comes along with compression.

The probably most famous and commonly used standard to store images are JPEG (Joint Photographic Experts Group) and the advanced JPEG 2000. The first public release as ISO/IEC standard occured in 1992. JPEG and JPEG 2000 depend on the discrete cosine and wavelet transform, respectively. Therefore, these well-analyzed methods are of great interest from the theoretical as well as practical point of view ([Str09]). Modern designs, such as JPEG 2000, provide in addition to a lossy image compression also a lossless modus.

An overview about some more image analysis tools, image denoising and inpainting techniques can be found in [CS05].

There are two different application approachs to optimize the process of image compression. We can improve the reconstruction model or the choice of the interpolation data. In this thesis, we focus on the choice of suitable interpolation data. To realize this, we combine partial differential equations (PDE's) and variational methods.

3.2 Combining Diffusion and Inpainting

In the following section, we consider PDE's to derive our initial problem formulation. First, we present the basic terms of Laplace interpolation based on standard partial differential equations. Then, these terms are discretized and optimized to achieve a suitable optimization problem for our further application.

3.2.1 Combination Approach

Before we formulate the first model, we need to introduce some basic notations. For more details, we refer to [Cia13].

Let $\Omega \subset \mathbb{R}^n$ be a subset with nonempty, sufficiently regular boundary $\partial\Omega$. The term "sufficiently regular" means that the boundary can locally be described by Lipschitz continuous functions. The Laplace operator is given by $\Delta := \sum_{k=1}^{n} \partial_{kk}$ which acts on functions in Ω. For a function $u : \Omega \to \mathbb{R}$, Δu is called the Laplacian of u. The term $\partial_{kk} := \partial_k \partial_k$ denotes the second-order partial derivative. Sometimes, it may be convenient to call ∂_k directional derivative where k denotes the direction.

Furthermore, let $\partial_\nu := \sum_{k=1}^{n} \nu_k \partial_k$ define the outer normal derivative operator where $(\nu_k)_{k=1}^{n}$ denotes the unit outer normal vector field along $\partial\Omega$. This operator acts on functions defined on the boundary $\partial\Omega$. For the function u, $\partial_\nu u$ is called the outer normal derivative of u.

Definition 3.2.1 (Laplace interpolation)**.** *Let $f : \Omega \to \mathbb{R}$ be smooth on a bounded domain $\Omega \subset \mathbb{R}^n$ and represent the initial data. The domain $\Omega \subset \mathbb{R}^n$ has a sufficiently regular boundary $\partial\Omega$ and $\Omega_K \subsetneq \Omega$ is a subset with positive measure. Let u define a reconstruction of f. The* homogeneous diffusion inpainting *is given by*

$$-\Delta u = 0, \text{ on } \Omega \backslash \Omega_K \tag{3.1}$$

$$u = f, \text{ on } \Omega_K \tag{3.2}$$

$$\partial_\nu u = 0, \text{ on } \partial\Omega \backslash \partial\Omega_K \tag{3.3}$$

where the term $\partial_\nu u$ denotes the derivative of u in outer normal direction.

Note that the just defined Laplace interpolation 3.2.1 is sometimes already presented as homogeneous diffusion inpainting. To distinguish this problem formulation and the corresponding discrete optimization problem introduced later, we always refer to 3.2.1 as Laplace interpolation and to 3.2.2 as homogeneous diffusion inpainting.

The Laplace interpolation 3.2.1 belongs to the boundary value problems. It consists of the Laplace equation 3.1 as well as the Dirichlet 3.2 and Neumann 3.3 boundary conditions. Since the Neumann boundary condition is required to be 0, we call it homogeneous.

From now on, we assume that the sets $\partial\Omega_K$ and $\partial\Omega\backslash\partial\Omega_K$ are nonempty. Note that due to Definition 3.2.1 the set Ω_K has positive measure and, thus, is nonempty as well. Within the context of inpainting, we interpret Ω_K to represent the set of known data points after a compression. Thus, the Dirichlet boundary condition 3.2 affects these points that are used to reconstruct the missing data.

In the next step, we introduce a binary valued function $c : \Omega \to \{0, 1\}$ which takes the value 1 only for points in Ω_K and 0 otherwise, i.e.

$$c(x) = \begin{cases} 1 & , x \in \Omega_K \\ 0 & , \text{otherwise.} \end{cases}$$

Since this function c separates Ω into Ω_K and $\Omega\backslash\Omega_K$, it is said to be a mask on Ω with respect to Ω_K.

Using mask c, we can rewrite the Laplace equation 3.1 and the Dirichlet boundary condition 3.3 to $-(1 - c(x))\Delta u(x) = 0$ and $c(x)(u(x) - f(x)) = 0$, respectively. Combining these new formulations leads to the following problem which is equivalent to the Laplace interpolation 3.2.1

$$\begin{aligned} c(x)(u(x) - f(x)) - (1 - c(x))\Delta u(x) = 0 \text{ for } x \in \Omega \\ \partial_\nu u(x) = 0 \text{ for } x \in \partial\Omega\backslash\partial\Omega_K. \end{aligned} \tag{3.4}$$

Our goal is to optimize the mask c with respect to accuracy of the reconstruction as well as sparsity of the interpolation data. These two aspects contradict each other. A perfect reconstruction requires $c \equiv 1$, which is not sparse at all. But the sparsest possible mask is given by $c \equiv 0$, which does not allow any reconstruction.

3.2.2 Discrete Problem Formulation

The following approach of discretizing and optimizing the reformulated Laplace interpolation 3.4 can be found in [HMH+15].

Let $J := \{1, ..., N\}$ be the set of finitely many indices which enumerate all discrete sample points. In case of an image $f \in \mathbb{R}^{n\times n}$, its pixels are counted columnwise such that sample point $(x, y) \in \mathbb{R}$ gets numbered with $x + n(y - 1) \in J$ and $N := n^2$ as shown in Figure 3.1. Denote the set of indices of the known sample points by $K \subseteq J$ as Ω_K represents the known data. Then, we can discretize image, reconstruction and mask and write $f, u, c \in \mathbb{R}^N$ where

$$c_i = \begin{cases} 1 & , i \in K \\ 0 & , \text{otherwise.} \end{cases}$$

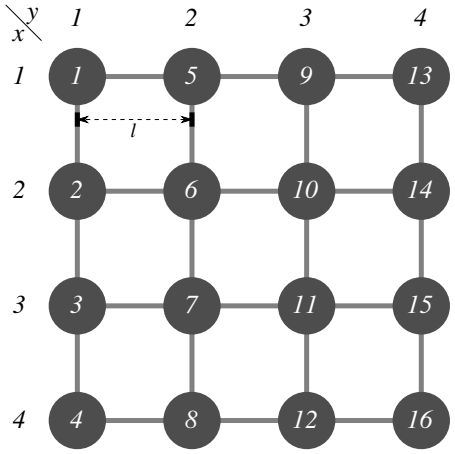


Figure 3.1: regular 4×4 -grid with size $l := 1$ and pixels numbered by $1, ..., 16$

Furthermore, we work on a regular grid as presented in Figure 3.1. In this setting, let each sample point represent a pixel of an image. An important characterictic of such a grid is its size l, the maximal distance between two neighboured pixels. For simplicity, we set the grid size to 1 so that we do not have to consider it in the further discretization. To see the influence of a grid size $l \neq 1$, we refer to [HMH+15].

Having a look at problem formulation 3.4, we still need to discretize the Laplace operator Δ with homogeneous Neumann boundary conditions on $\partial\Omega \backslash \partial\Omega_K$. Remember that this operator approximates the sum of the second-order partial derivatives as introduced earlier. To visualize the idea of discretization, we consider the set of neighbours $\mathcal{N}(i)$ for a pixel $i = x + n(y-1) \in J$ corresponding to sample point (x, y),

$$\mathcal{N}(i) = \{j \in J \mid \exists x_j, y_j \in \{1, ..., n\} : \begin{array}{l} j = x_j + n(y_j - 1) \\ \wedge\ |x - x_j| + |y - y_j| = 1\}. \end{array} \tag{3.5}$$

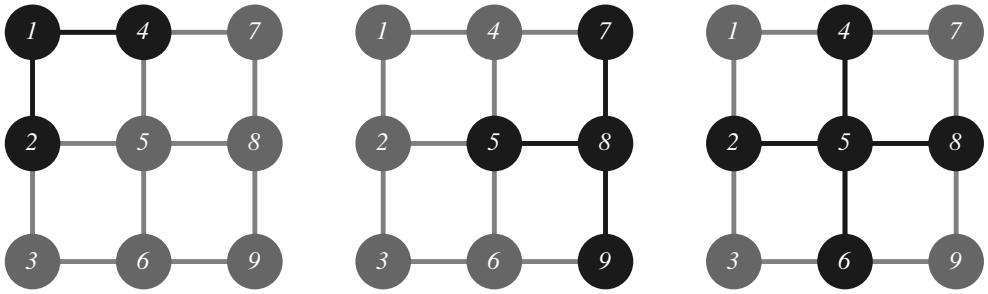
Since $\Delta := \partial_{xx} + \partial_{yy}$ for a function in two variables x, y, we are interested in the approximation of these two second-order partial derivatives. First, we assume that the sample point (x, y) lies in $\Omega \backslash \partial\Omega$. Then the point (x, y) has four neighbours, one in each direction as illustrated in Figure 3.2c. By the central step method, we get

$$\begin{aligned} \partial_{xx} f(x, y) &\approx \frac{f(x+h, y) - 2f(x, y) + f(x-h, y)}{h^2} \\ \partial_{yy} f(x, y) &\approx \frac{f(x, y+h) - 2f(x, y) + f(x, y-h)}{h^2} \end{aligned}$$

where $h \in \mathbb{R}$ realizes a step in both possible directions, i.e. positive 1 and negative -1, of each variable x, y. Parameter h can also be seen as the distance to the neighboured point used for approximation.

Thus, for points in $\Omega \backslash \partial\Omega$, the discrete Laplacian operator can be approximated by

$$\Delta f(x, y) \approx f(x+1, y) + f(x, y+1) - 4f(x, y) + f(x-1, y) + f(x, y-1).$$



(a) $\mathcal{N}(1) = \{2, 4\}$ (b) $\mathcal{N}(8) = \{5, 7, 9\}$ (c) $\mathcal{N}(5) = \{2, 4, 6, 8\}$

Figure 3.2: neighbours in a regular 3×3 -grid with size 1

Identify x and y with the indices of a matrix like in Figure 3.1. This leads to the following kernel D

$$D := \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Next, consider the cases where one or more of the neighboured pixels do not exist, like for pixels at the boundary of an image like in Figure 3.2a or Figure 3.2b. We can still approximate the second-order partial derivatives $\partial_{xx} f, \partial_{yy} f$ by using the set of neighbours $\mathcal{N}$ defined in 3.5. We take only one existing neighbour of the center point what leads to

$$\begin{aligned} \partial_{xx} f(x,y) &\approx \frac{f(x+h,y) - f(x,y)}{h} \\ \text{or } \partial_{xx} f(x,y) &\approx \frac{-f(x,y) + f(x-h,y)}{h} \end{aligned}$$

for $h \in \mathbb{R}$ and analogously $\partial_{yy}(x,y)$. Note that this approximation using only one direction, either h or $-h$, comes along with less accuracy compared to the central step method taking both possible directions $h, -h$.

Finally, we can write the Laplace operator with Neumann boundary conditions as the matrix $L \in \mathbb{R}^{N \times N}$ with entries

$$L(i,j) = \begin{cases} 1 & , j \in \mathcal{N}(i) \\ -|\mathcal{N}(i)| & , j = i \\ 0 & , \text{otherwise.} \end{cases} \tag{3.6}$$

Now, we formulate the initial discrete constrained optimization problem. It optimizes mask c with respect to accuracy of the reconstruction as well as sparsity of the

interpolation data. We use 3.4 as constraint to correlate mask c and reconstruction u. Note that the Neumann boundary condition is included in the Laplace operator 3.6.

Definition 3.2.2 (homogeneous diffusion inpainting)**.** *Let $f \in \mathbb{R}^N$ be the initial data. Moreover, let $c \in \{0,1\}^N$ define a mask on each sample point and let the corresponding reconstruction be given by $u \in \mathbb{R}^N$. Then, the* homogeneous diffusion inpainting *for this data is the following constrained optimization problem*

$$\begin{aligned} &\underset{(u,c)\in\mathbb{R}^N\times\mathbb{R}^N}{\arg\min} \ \frac{1}{2}\|u-f\|_2^2 + \lambda\|c\|_1 \\ &\text{s.t. } \operatorname{diag}(c)(u-f) - (Id - \operatorname{diag}(c))Lu = 0 \end{aligned}$$

where $\lambda \geq 0$ and L denotes the discrete Laplacian with entries defined by 3.6.

The first term penalizes the difference between reconstruction u and initial image f. The second term weights the mask c and parameter λ steers the sparsity of c. For $\lambda = 0$, the optimal mask can have maximum weight and the corresponding reconstruction is perfect, i.e. $c \equiv 1$ and $u = f$. But when λ tends to infinity, the sparsity of mask c is enforced such that $c_i \to 0$ for $\lambda \to \infty$ and all indices $i \in \{1, ..., N\}$.

The constraint equation assures the dependence of mask and reconstruction in this model. Fixing the mask, it has a unique solution for the reconstruction ([MBWF11]).

3.2.3 Outlook

Many mathematical models especially in natural sciences rest upon the widely- researched theory of PDE's. The process of diffusion which we are interested in can be modeled in terms of them, as done in Definition 3.2.1. It is closely related to the well-known heat equation ([Eva10]).

The combination of diffusion and inpainting was already very successful on cartoon-like images ([MW09]). These images are piecewise flat and thus easier to reconstruct. The authors show that their approach can outperform even the advanced JPEG 2000 standard.

Another approach uses anisotropic diffusion to compress general images like photographs ([GWW+08]). It is already better than JPEG and close to the quality of the JPEG 2000 standard. This work gets improved with recognizable success in [SWB09]. The authors demonstrate that it beats the quality of the JPEG 2000 standard.

4 Optimization Algorithms

This chapter is divided into two parts. Each of them presents an optimization algorithm applying the method of proximal mapping combined with a gradient method. Though, the algorithms differ in their concepts and the classes of initial problems.

4.1 Inertial Proximal Algorithm for Nonconvex Optimization (iPiano)

The first algorithm we introduce is the Inertial Proximal Algorithm for Nonconvex Optimization (iPiano). It solves a special class of nonsmooth nonconvex optimization problems, namely the sum of a differentiable, possibly nonconvex function and a convex, possibly nondifferentiable function. This algorithm combines a gradient method with a proximal step to utilize the special structure of the initial problem.

We start by motivating the idea of iPiano and by introducing the general problem setup. After a brief description of the algorithm, we give a short overview of the convergence analysis.

The whole section about iPiano is based on [OCBP14].

4.1.1 Initial Problem

Due to practical problems like in signal or image processing and machine learning, there has been an increased interest in partly smooth, partly nonsmooth objective functions. The goal is to solve a sum of a smooth and a nonsmooth, but convex function. This leads to a special case of the nonsmooth nonconvex optimization problems given in Definition 2.2.1.

Thus, let $g : \mathbb{R}^N \to \overline{\mathbb{R}}$, $N \geq 1$, be a proper nonsmooth convex function. Define $f : \mathbb{R}^N \to \mathbb{R}$ to be a smooth C^1-function with Lipschitz continuous gradient ∇f on $\operatorname{dom} g$ and Lipschitz constant L.

Composing and minimizing these two functions yields the following structured nonsmooth nonconvex optimization problem

$$\min_{x \in \mathbb{R}^N} h(x) = \min_{x \in \mathbb{R}^N} f(x) + g(x) \tag{4.1}$$

where the proper, lower semicontinuous, extended-valued function $h : \mathbb{R}^N \to \overline{\mathbb{R}}$ is called a composite objective function. It is assumed to be coercive, i.e. $\|x\|_2 \to +\infty$ implies $h(x) \to +\infty$, and to be bounded from below by some $\underline{h} > -\infty$.

Within this setup, function f is not necessarily convex. But if it is so, there already exists an improvement of the algorithm iPiano for this special case in [OBP15].

The algorithm iPiano combines the concepts of gradient methods and proximal mapping to solve the structured nonconvex nonsmooth optimization problem 4.1.

4.1.2 The iPiano-algorithm

Now, we present the generic algorithm to tackle optimization problem 4.1. The approach starts with the forward step in terms of the heavy-ball method 2.3.3 for the smooth function f. The next backward step consists of the proximal map 2.3.4 of the convex function g applied to the result of the forward step.

Algorithm 4.1.1 (iPiano)**.** *Inertial Proximal Algorithm for Nonconvex Optimization*

- *Initialization: Choose $c_1, c_2 > 0$ close to 0, $x^0 \in \operatorname{dom} h$ and set $x^{-1} = x^0$.*
- *Iterations ($k \geq 0$): Update*

$$x^{k+1} = (I + \alpha_k \partial g)^{-1}(x^k - \alpha_k \nabla f(x^k) + \beta_k (x^k - x^{k-1})) \tag{4.2}$$

where $L_k > 0$ is the local Lipschitz constant of ∇f satisfying

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{L_k}{2} \|x^{k+1} - x^k\|_2^2$$

and $\alpha_k \geq c_1, \beta_k \geq 0$ are chosen such that $\delta_k \geq \gamma_k \geq c_2$ defined by

$$\delta_k := \frac{1}{\alpha_k} - \frac{L_k}{2} - \frac{\beta_k}{2\alpha_k} \text{ and } \gamma_k := \frac{1}{\alpha_k} - \frac{L_k}{2} - \frac{\beta_k}{\alpha_k}$$

and $(\delta_k)_{k=0}^{\infty}$ is monotonically decreasing.

It is important to choose the step sizes appropriately to achieve (good) convergence results. The proposed rule above suffices to guarantee convergence. There are several special instances shown in [OCBP14, sec. 4.4] which exactly describe how the step sizes can be updated during each iteration. For our application, we take constant parameters as also proposed by the authors. Although, it is the simplest way to choose α_k and β_k, the final results turn out to be already good enough for our purpose.

Therefore, we fix $\beta_k = \beta \in [0, 1)$ and $\alpha_k = \alpha < 2\frac{1-\beta}{L}$ where L denotes the Lipschitz constant of ∇f. The constants can be chosen arbitrarily, as long as they fulfill the step size constraints in Algorithm 4.1.1.

4.1.3 Convergence Analysis

During this part, we summarize the results of the convergence analysis achieved by [OCBP14]. For further details and the entire derivation, we refer to this paper.

First of all, the authors verify the existence of parameters $\alpha_k, \beta_k, \delta_k, \gamma_k$ for given $L_k > 0$ which satisfy the defined constraints in Algorithm 4.1.1. Hence, the requirements for the parameters are not contradictory and the algorithm makes sense.

Now, let us state the general convergence results for the Algorithm 4.1.1 iPiano. The theorem does not make any further assumptions concerning the generated sequence or the objective functions.

Theorem 4.1.2 (general convergence results of iPiano [OCBP14, Thm. 4.8]). *Let $(x^k)_{k\in\mathbb{N}}$ be a sequence generated by the iPiano-algorithm.*

1. *The sequence $(h(x^k))_{k=0}^{\infty}$ converges.*
2. *There exists a converging subsequence $(x^{k_j})_{j=0}^{\infty}$.*
3. *Any limit point $x^* := \lim_{k\to\infty} x^{k_j}$ is a critical point of $\min_{x\in\mathbb{R}^N} h(x)$ and $h(x^{k_j}) \overset{j\to\infty}{\to} h(x^*)$.*

Note that a critical point $\overline{x} \in \operatorname{dom} h$ fulfills the necessary first-order optimality condition $0 \in \partial h(\overline{x})$. By Fermat's rule 2.2.2, this is equivalent to $-\nabla f(\overline{x}) \in \partial g(\overline{x})$ for $h(x) = f(x) + g(x)$ and a smooth function f.

In addition, under some assumptions, it is even shown that not only subsequences, but each possible sequence $(x^k)_{k\in\mathbb{N}}$ generated by iPiano converges to a critical point x^* of the objective function h, i.e. $x^* = \lim_{k\to\infty} x^k$. If the starting point x^0 is close enough to a global minimum, iPiano converges to that global minimizer.

Theorem 4.1.3 (convergence of iPiano to critical point [OCBP14, Thm. 4.9]). *Let $(x^k)_{k\in\mathbb{N}}$ be a sequence generated by the iPiano-algorithm and let $\delta_k = \delta$ for all $k \in \mathbb{N}$. Moreover, define the function $h_\delta : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ with $(x, y) \mapsto h(x) + \delta\|x - y\|_2^2$. If $h_\delta(x, y)$ has the KL-property at a cluster point (x^*, y^*), then the sequence $(x^k)_{k\in\mathbb{N}}$ has finite length, $x^k \to x^*$ for $k \to \infty$, and (x^*, x^*) is a critical point of h_δ. Hence, x^* is a critical point of the objective function h.*

Last, let us also mention the convergence rate. The authors of [OCBP14] analyzed it with respect to the proximal residual $r(x) := x - (Id + \partial g)^{-1}(x - \nabla f(x))$. Since $\|x^k - x^{k+1}\|_2^2 = \|r(x^k)\|_2^2$, this residual provides an error measure of tending to a critical point of the objective function for a sequence $(x^k)_{k\in\mathbb{N}}$ generated by iPiano. The squared normed residuum $\|r(x^k)\|_2^2$ achieves a global convergence with rate roughly bounded from above by $\mathcal{O}(1/k)$. Hence, the iterates generated by Algorithm 4.1.1 iPiano converge with rate $\mathcal{O}(1/\sqrt{k})$.

The proposed Algorithm 4.1.1 iPiano has many favourable theoretical properties as demonstrated in [OCBP14]. It is simple and efficiently solves the structured nonsmooth nonconvex optimization problem 4.1. These are the reasons why the authors recommend it as standard solver for this class of optimization problems.

4.2 Proximal Alternating Linearized Minimization (PALM)

During this section, we present the second algorithm of interest, the Proximal Alternating Linearized Minimization (PALM). Compared to iPiano, this algorithm was developed to solve a wider class of nonsmooth nonconvex optimization problems, namely a sum of finitely many functions with weaker assumptions. It also combines a gradient method with a proximal step.

Again, we start by presenting the problem setting with rather general assumptions concerning the objective functions. Then, we introduce the algorithm followed by a short convergence analysis.

The part dealing with PALM relies on [BST14].

4.2.1 Initial Problem

The development of this algorithm can be motivated by problems with practical applications. Like in compressed sensing, sparse approximation of signals and images as well as in blind decomposition, we are interested in minimizing a sum of functions which are neither smooth nor convex similarly in all variables. The setup built in [BST14] addresses a quite general formulation of nonsmooth nonconvex optimization problems as defined in Definition 2.2.1.

Assume $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ to be two proper, lower semicontinuous functions. Let $H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a C^1-function coupling the domains of f and g. Define the composite objective function as $\Phi(x, y) := H(x, y) + f(x) + g(y)$. This leads to the formally unconstrained initial problem formulation

$$\begin{aligned} &\operatorname*{arg\,min}_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^m} \Phi(x, y) \\ = &\operatorname*{arg\,min}_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^m} H(x, y) + f(x) + g(y). \end{aligned} \tag{4.3}$$

For further analysis of the algorithm, we assume that the functions $x \mapsto H(x, y)$ and $y \mapsto H(x, y)$ are continuously differentiable with Lipschitz continuous gradient for fixed y or x, respectively. If there exist variables in the domain of H for which this assumption does not hold, we refer to [HLST15]. In this paper, the functions f and g define indicator functions (Definition 2.3.12) on the corresponding domains. Then,

the algorithm PALM was adopted as a part of the introduced Proximal Block Implicit-Explicit algorithm (PBIE).

The algorithm PALM combines the concepts of gradient methods and proximal mapping to solve a broad class of nonsmooth nonconvex optimization problems 4.3.

4.2.2 The PALM-algorithm

Before we can present the algorithm, we need to introduce the notation of the partial gradient. Therefore, consider the map $x \mapsto H(x,y)$ for the block of variables x and fixed y. Its corresponding gradient, denoted by $\nabla_x H(x,y)$, is the partial gradient of H in x and is assumed to be Lipschitz continuous with modulus $L_x(y)$. Analogously, we get the partial gradient $\nabla_y H(x,y)$ for the other block of variables y and fixed x as well as its Lipschitz constant $L_y(x)$.

Now, we can describe the generic algorithm. The approach relies on an alternating minimization. An iteration updates each block of variables x and y separately with respect to the objective function. Each update contains a forward step in terms of the gradient descent 2.5 for the partial gradients of H. The next backward step consists of the proximal map 2.3.4 of function f for block x or g for block y which is applied to the result of the forward step.

The algorithm PALM solving optimization problem 4.3 is given by the following alternating forward-backward scheme.

Algorithm 4.2.1 (PALM). *Proximal Alternating Linearized Minimization algorithm*

- *Initialization: Choose* $\alpha, \beta > 1$ *and any* $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$.
- *Iterations* ($k \geq 0$)*: Update as follows*
  1. *Set* $\alpha_k = \alpha L_x(y^k)$ *and compute*

$$x^{k+1} \in \operatorname{prox}_{\alpha_k, f}\left(x^k - \frac{1}{\alpha_k}\nabla_x H(x^k, y^k)\right) \tag{4.4}$$

  2. *Set* $\beta_k = \beta L_y(x^{k+1})$ *and compute*

$$y^{k+1} \in \operatorname{prox}_{\beta_k, g}\left(y^k - \frac{1}{\beta_k}\nabla_y H(x^{k+1}, y^k)\right) \tag{4.5}$$

4.2.3 Convergence Analysis

Throughout this part, we summarize the results of the convergence analysis attained in [BST14]. For further details and the entire derivation, we refer to this paper.

Let the following assumptions hold for further analysis.

A1 Let the objective function Φ as well as f and g be bounded from below, i.e.

$$\inf_{\mathbb{R}^n \times \mathbb{R}^m} \Phi > -\infty, \ \inf_{\mathbb{R}^n} f > -\infty \text{ and } \inf_{\mathbb{R}^m} g > -\infty.$$

A2 The partial gradients $\nabla_x H(x,y)$ and $\nabla_y H(x,y)$ are Lipschitz continuous with moduli $L_x(y,z)$ and $L_y(x,z)$, respectively.

A3 There exist bounds $\lambda_x^-, \lambda_x^+, \lambda_y^-, \lambda_y^+ > 0$ such that

$$\inf \left\{ L_x(y^k) \mid k \in \mathbb{N} \right\} \geq \lambda_x^- \text{ and } \inf \left\{ L_y(x^k) \mid k \in \mathbb{N} \right\} \geq \lambda_y^-,$$
$$\sup \left\{ L_x(y^k) \mid k \in \mathbb{N} \right\} \leq \lambda_x^+ \text{ and } \sup \left\{ L_y(x^k) \mid k \in \mathbb{N} \right\} \leq \lambda_y^+.$$

A4 The gradient $\nabla H(x,y)$ is Lipschitz continuous on bounded domains in $\mathbb{R}^n \times \mathbb{R}^m$.

A5 The iterates $(x^k, y^k)_{k \in \mathbb{N}}$ are bounded.

Before we state the general convergence results for PALM, let us introduce the notion of the set of all limit points and denote the set by $\omega(x^0, y^0) \subset \mathbb{R}^n \times \mathbb{R}^m$. We have $(x^*, y^*) \in \omega(x^0, y^0)$ if there exists a subsequence of $(x^k, y^k)_{k \in \mathbb{N}}$ generated by PALM with starting point (x^0, y^0) that converges to (x^*, y^*).

Lemma 4.2.2 (properties of limit points of PALM [BST14, Lemma 3.5])**.** *Let the initial problem 4.3 be given and let $(x^k, y^k)_{k \in \mathbb{N}}$ be a sequence generated by PALM. Moreover, assume that assumptions* A1-A5 *are fulfilled. Then the following hold*

1. *The set $\omega(x^0, y^0)$ is nonempty, compact, connected and contained in the set of the critical points of the objective function Φ.*
2. *It is*
$$\lim_{k \to \infty} \| (x^k, y^k) - \omega(x^0, y^0) \| = 0.$$
3. *The objective function Φ is finite and constant on $\omega(x^0, y^0)$.*

Note that a critical point $(\overline{x}, \overline{y}) \in \operatorname{dom} \Phi$ fulfills the necessary first-order optimality condition $0 \in \partial \Phi(\overline{x}, \overline{y})$ by Fermat's rule 2.2.2. Since $\Phi(x,y) = H(x,y) + f(x) + g(y)$ and considering the partial gradients of H, we have

$$\partial \Phi(x,y) = \left(\nabla_x H(x,y) + \partial f(x), \nabla_y H(x,y) + \partial g(y) \right).$$

In addition, under some more assumptions, it is even shown that not only subsequences, but each possible sequence $(x^k, y^k)_{k\in\mathbb{N}}$ generated by PALM converges to a critical point (x^*, y^*) of the objective function Φ, i.e. $(x^*, y^*) = \lim_{k\to\infty}(x^k, y^k)$.

Theorem 4.2.3 (convergence of PALM to critical point [BST14, Thm. 3.1]). *Suppose that the objective function Φ is a KL-function and that $(x^k, y^k)_{k\in\mathbb{N}}$ is a sequence generated by the Algorithm 4.2.1 PALM. Let the assumptions* A1-A5 *hold. Then the following yields*

1. *The sequence $(x^k, y^k)_{k\in\mathbb{N}}$ has finite length.*
2. *The sequence $(x^k, y^k)_{k\in\mathbb{N}}$ converges to a critical point (x^*, y^*) of Φ.*

4.2.4 Inertial PALM (iPALM)

A combination of an inertial term and a proximal algorithm for nonsmooth nonconvex optimization problems has already been realized for example in [BC15] or [BCL15].

This motivates the consideration of an inertial version of the algorithm PALM.

We extend the forward step concerning the coupling function H in each update of PALM by an inertial term such that it becomes a heavy-ball step as in Definition 2.3.3. Thus, we obtain the following scheme.

Algorithm 4.2.4 (iPALM). *Inertial Proximal Alternating Linearized Minimization algorithm*

- *Initialization: Choose $\alpha, \beta > 1$ and any $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$.*
- *Iterations ($k \geq 0$): Update as follows*

  1. *Set $\alpha_k = \alpha L_x(y^k)$, choose $\mu_k > 0$ and compute*

$$x^{k+1} \in \operatorname{prox}_{\alpha_k f}\left(x^k - \frac{1}{\alpha_k}\nabla_x H(x^k, y^k) + \mu_k(x^k - x^{k-1})\right) \tag{4.6}$$

  2. *Set $\beta_k = \beta L_y(x^{k+1})$, choose $\eta_k > 0$ and compute*

$$y^{k+1} \in \operatorname{prox}_{\beta_k g}\left(y^k - \frac{1}{\beta_k}\nabla_y H(x^{k+1}, y^k) + \eta_k(y^k - y^{k-1})\right) \tag{4.7}$$

Yet there exists no convergence theory for this inertial version of PALM.

In the numerical experiments presented in the following chapter, we also apply the algorithm iPALM to the homogeneous diffusion inpainting given in Definition 3.2.2. This shall serve as a first evaluation of behaviour and convergence in practical application.

5 Numerical Experiments

In this chapter, we apply the presented algorithms iPiano, PALM and iPALM to the problem of linear diffusion-based image compression. Founded on several experiments, we analyze results and performance of these algorithms within this setup.

5.1 Notations

For analyzing the results of the algorithms, we need some tools to quantize and measure different attributes.

We evaluate the actual results of the experiments. That means accuracy of reconstruction and sparsity of mask. To measure the quality of the reconstruction, we determine the mean square error of its difference to the original image. This tool is not used as stopping criterion since none of the algorithms explicitly minimizes that value.

Definition 5.1.1 (mean square error (MSE))**.** *Let $u^0, u \in \mathbb{R}^N$ be given. The* mean square error *(MSE) of u to u^0 is determined by*

$$\mathrm{MSE}\left(u^0, u\right) := \frac{1}{N}\sum_{i=1}^{N}(u_i^0 - u_i)^2.$$

Since the mask should be as sparse as possible, we are interested in its nonzero elements as well as the ratio between nonzero and all elements.

Definition 5.1.2 (1-norm density)**.** *Let $c \in \mathbb{R}^N$ be given. The 1-norm is the sum of the absolute values of all entries in c, i.e.*

$$\|c\|_1 := \sum_{i=1}^{N}|c_i|$$

and the (1-norm) density *of c equals the ratio $\frac{\|c\|_1}{N}$.*

Additionally, we measure the step length between two successive iterates of a sequence $(x^k)_{k\in\mathbb{N}} \subset \mathbb{R}^N$ by the normed proximal residual

$$\|r(x^k)\|_2 = \|x^k - x^{k+1}\|_2.$$

This helps to define a stopping criterion for an algorithm with respect to the development of its iterates. If the step length between two successive iterates becomes smaller than a given positive tolerance $\mathrm{TOL} \in \mathbb{R}$, the algorithm stops.

5.2 iPiano

The problem of linear diffusion-based image compression needs to fit into the framework of the algorithm iPiano to allow its application. Hence, we start by reformulating the homogeneous diffusion inpainting to achieve a suitable setup for the algorithm. We also calculate the operators required. Afterwards, the algorithm iPiano is applied during several experiments. We use different problem setups to be able to evaluate as well-founded as possible.

5.2.1 Specialization to Linear Diffusion-based Image Compression

The calculations and derivations throughout this subsection rely on [OCBP14].

Recall the linear diffusion-based image compression as a constrained discrete optimization problem from Chapter 3 in Definition 3.2.2.

We need to express the homogeneous diffusion inpainting in terms of an unconstrained problem in one variable.

Let u^0 be the ground truth image and u its reconstruction based on mask c. For the following conversions, let $c \in [0,1]^N$ be continuously-valued, as relaxation of its initial formulation. Then, the latter problem still makes sense in the context of inpainting.

Now, we solve u from the equality constraint of Definition 3.2.2 what leads us to the relation $u = A^{-1}Cu^0$ with $A := C + (C - Id)L$ and $C := \operatorname{diag}(c)$. This representation of u as a function of c emphasizes their dependence. Note that A is invertible if $c \neq 0$.

Substituting u into the objective function yields the following optimization problem in one variable

$$\min_{c \in [0,1]^N} \frac{1}{2}\|A^{-1}Cu^0 - u^0\|_2^2 + \lambda\|c\|_1. \tag{5.1}$$

Define the functions

$$\begin{aligned} f(c) &:= \frac{1}{2}\|A^{-1}Cu^0 - u^0\|_2^2 \\ g(c) &:= \lambda\|c\|_1 \end{aligned}$$

to obtain a suitable structured nonsmooth nonconvex optimization problem for the Algorithm 4.1.1 iPiano. Next, we calculate the gradient of f as well as the proximal map of g.

Lemma 5.2.1 (gradient of $f(c) = \frac{1}{2}\|A^{-1}Cu^0 - u^0\|_2^2$ [OCBP14, Lemma 5.2]). *Let the function $f : \mathbb{R}^N \to \mathbb{R}$ be real-valued and given by $f(c) = \frac{1}{2}\|A^{-1}\operatorname{diag}(c)u^0 - u^0\|_2^2$. Then*

$$\nabla f(c) = \operatorname{diag}(-(Id + L)u + u^0)(A^T)^{-1}(u - u^0). \tag{5.2}$$

Proof. First of all, we formulate function $f(c)$ in terms of the scalar product which induces the 2-norm on $\mathbb{R}^N$

$$f(c) = \frac{1}{2}\|u - u^0\|_2^2 = \frac{1}{2}\langle u - u^0, u - u^0\rangle.$$

Remember that u depends on c, but u^0 is just a constant independent of c. By Definition 2.1.7, we get $\mathrm{D}f(c) : \mathbb{R}^N \to \mathbb{R}$. Hence, deriving yields

$$\begin{aligned} \mathrm{D}f(c) &= \mathrm{D}\left(\frac{1}{2}\langle u - u^0, u - u^0\rangle\right) \\ &= \frac{1}{2}\left(\langle \mathrm{D}u, u - u^0\rangle + \langle u - u^0, \mathrm{D}u\rangle\right) \\ \Rightarrow \mathrm{D}f(c) &= \langle \mathrm{D}u, u - u^0\rangle \end{aligned} \tag{5.3}$$

Thus, deriving $f(c)$ requires deriving u with respect to c. With the product rule and the fact that $0 = \mathrm{D}Id = \mathrm{D}(A^{-1}A) = (\mathrm{D}A^{-1})A + A^{-1}\mathrm{D}A$, it follows that

$$\begin{aligned} \mathrm{D}u := \mathrm{D}(A^{-1}Cu^0)^{-1} &= \mathrm{D}A^{-1} \cdot Cu^0 + A^{-1} \cdot \mathrm{D}C \cdot u^0 \\ &= -A^{-1}\mathrm{D}A \cdot A^{-1}Cu^0 + A^{-1} \cdot \mathrm{D}C \cdot u^0 \\ &= -A^{-1}\mathrm{D}(C(Id + L))u + A^{-1} \cdot \mathrm{D}C \cdot u^0 \\ &= A^{-1} \cdot \mathrm{D}C(-(Id + L)u + u^0). \end{aligned}$$

Since $\mathrm{D}C \cdot t = \mathrm{diag}(t) \cdot \mathrm{D}c$ for $t \in \mathbb{R}^N$ and $-(Id + L)u + u^0 \in \mathbb{R}^N$, it is

$$\mathrm{D}u = A^{-1}\,\mathrm{diag}(-(Id + L)u + u^0)\mathrm{D}c. \tag{5.4}$$

Substituting 5.4 into 5.3 yields

$$\begin{aligned} \mathrm{D}f(c) &= \langle A^{-1}\,\mathrm{diag}(-(Id + L)u + u^0)\mathrm{D}c, u - u^0\rangle \\ &= \langle \mathrm{D}c, (A^{-1}\,\mathrm{diag}(-(Id + L)u + u^0))^T(u - u^0)\rangle \end{aligned}$$

and thus by Definition 2.1.7 and the uniqueness of the gradient

$$\nabla f(c) = \mathrm{diag}(-(Id + L)u + u^0)(A^T)^{-1}(u - u^0).$$

□

It remains to calculate the proximal map of $g(c) = \lambda\|c\|_1$. Note that the authors of [OCBP14] take the term of the resolvent $(Id + \alpha\partial g)^{-1}(y)$ instead of the proximal map $\mathrm{prox}_{\alpha^{-1},g}(y)$. But since g is convex, these two expressions coincide as stated in Lemma 2.3.6.

Lemma 5.2.2 (shrinkage operator). *Let $g : \mathbb{R}^N \to \mathbb{R}$ be the real-valued function defined by $g(c) = \lambda\|c\|_1$. Then, for any $\alpha > 0$, the proximal map of g is given by*

$$\text{prox}_{\alpha^{-1},g}(y) = \max(0, |y| - \alpha\lambda) \cdot sgn(y). \tag{5.5}$$

Proof. By the Definition 2.3.4 of the proximal map

$$\text{prox}_{\alpha^{-1},g}(y) = \underset{x\in\mathbb{R}^m}{\arg\min} \left(\frac{1}{2}\|x - y\|_2^2 + \alpha\lambda\|x\|_1 \right).$$

Since both terms are separable in x, we can minimize each element of x individually. Hence, $\min_{x_i\in\mathbb{R}} \frac{1}{2}(x_i - y_i)^2 + \alpha\lambda|x_i|$ for all $i \in \{1, ..., N\}$. Derive

$$\begin{array}{rrcl} \frac{\partial}{\partial x_i} & \frac{1}{2}(x_i - y_i)^2 & = & x_i - y_i \\ \frac{\partial}{\partial x_i} & \alpha\lambda|x_i| & = & \begin{cases} \alpha\lambda & , x_i > 0 \\ -\alpha\lambda & , x_i < 0 \\ [-\alpha\lambda, \alpha\lambda] & , x_i = 0. \end{cases} \end{array}$$

Sum up and find the minimum by using Theorem 2.2.2

$$0 \in \begin{cases} x_i - y_i + \alpha\lambda & , x_i > 0 \\ x_i - y_i - \alpha\lambda & , x_i < 0 \\ x_i - y_i + d \quad , d \in [-\alpha\lambda, \alpha\lambda] & , x_i = 0 \end{cases}$$

which leads to

$$x_i = \begin{cases} y_i - \alpha\lambda & , y_i > \alpha\lambda \\ y_i + \alpha\lambda & , y_i < -\alpha\lambda \\ 0 & , -\alpha\lambda \leq y_i \leq \alpha\lambda. \end{cases}$$

Since $\alpha\lambda \geq 0$, the claim follows.

□

Remember that the goal of the problem of homogeneous diffusion inpainting 3.2.2 is to find a binary-valued mask c. It should consist of as few nonzero elements as possible and still allow a good reconstruction of the original image. Since the shrinkage operator returns continuously-valued solutions out of $[0, 1]^N$ for each iteration of iPiano, we simply round the very last iterate of the algorithm entrywise to the closest integer.

5.2.2 Initialization of Experiments

During our experiments, we want to test several different initial setups to allow a well-founded evaluation of the algorithm. But since there exist four customizable parame-

ters, changing all of them would not really help us to get a general idea of the framework. Hence, we fix the two parameters α, β which primarily determine the step size of each iteration.

The parameter β weights the inertial term of the forward gradient step. We fix $\beta = 0.8$ since the authors of [OCBP14] indicate this choice as good in practice and since we received a similar impression during our experiments.

Dependent on β, we choose $\alpha = 2\frac{1-\beta}{L+1}$. This parameter affects the algorithm during the forward step as the length in front of the gradient and during the backward step as part of the range of the shrinkage (see Lemma 5.2.2).

Remember that L denotes the Lipschitz constant of ∇f. We also need to guess a value for L. It should rather be taken too big. Consequently, the restricting bound for step size α becomes smaller than necessary. But choosing L too small could violate the conditions concerning the parameters in Algorithm 4.1.1. During an experiment, this would raise α such that the corresponding iteration returns a new iterate which has a worse objective function value compared to the current value. To prevent the latter case, we implemented a stopping criterion that breaks the algorithm as soon as the function value of the new iterate gets worse again.

In addition, there exist two more stopping criteria. We can set a maximal number of iterations and a tolerance TOL concerning the difference between two successive iterates. We restrict this normed proximal residual by TOL $= 10 \times 10^{-10}$. As long as the maximal number of iterates is not reached and the difference between the iterates stays greater than the tolerance, the algorithms proceeds.

Moreover, we initialize the starting point x^0 with the full mask $c \equiv 1$.

All in all, if nothing else is indicated, the experiments are initialized with the following parameters

$$\begin{aligned} L &= 3 \\ \beta &= 0.8 \\ \alpha &= \tfrac{2}{55} \\ x^0 &\equiv 1. \end{aligned}$$

For each experiment, we vary the weight λ of the mask in the objective function to steer sparsity.

5.2.3 Results

First of all, we start with a simple image consisting of five stripes coloured in black, white and gray (see Figure 5.1a). The edges between the different colors are straight and sharp.

The results of our experiments in Figure 5.1 clearly illustrate the edge-detecting char-

λ	MSE	density
0.001	0	1
0.002	4.297×10^{-32}	0.08
0.04	4.297×10^{-32}	0.08
0.485	0.031	0.06
0.05	0.05	0.04

Table 5.1: experiments for iPiano and test image stripes and different parameters λ: MSE and density

λ	MSE	density	iterations	CPU-time (min)
0.0024	4.821×10^{-4}	0.0637	2000	1322.5

Table 5.2: experiment for iPiano and test image trui

acter of the algorithm caused by the utilization of the Laplace matrix. Moreover, it shows that smooth areas can be efficiently reconstructed from the data points of its border. We just need 8% of the pixels to allow a perfect reconstruction with MSE of the reconstruction of almost 0 (compare Table 5.1).

Comparing Figure 5.1b and Figure 5.1d, we recognize that edges between colors with smaller difference are the first omitted. The remaining interpolation data determines the way of reconstruction. In the example of the "stripes", the amount of interpolation data is reduced by omitting the inner edges belonging to the white stripes (Figure 5.1c). An even greater λ makes a smooth white area replacing the whole gray stripe (Figure 5.1d). This indicates the lost of information about that gray stripe.

Beyond this first simple figure, Algorithm 4.1.1 iPiano can also be applied to images like photographs. Therefore, we want to test iPiano on such an image. We take the standard test image "trui" from Matlab which consists of 256×256 pixels (Figure 5.3a). Our implementation of Algorithm 4.1.1 iPiano is able to solve the homogeneous diffusion inpainting for those big images as shown in Figure 5.3b. But the run time takes relatively long with 22 hours Table 5.2 in comparison to smaller images.

For further and a more detailed analysis, we use an extract of trui of 100×100 pixels as given in Figure 5.4a.

Again, the algorithm iPiano determines the edges within the test image and takes pixels of them as interpolation data. The Figure 5.4b shows the final binary-valued

λ	MSE	density	iterations	CPU-time (min)
0.0043	1.047×10^{-3}	0.0705	3060	121.1

Table 5.3: experiments for iPiano and test image scarf of trui

(a) test image stripes (100×100 pixels)

(b) mask and corresponding reconstruction for $\lambda = 0.002$ and $\lambda = 0.04$

(c) mask and corresponding reconstruction for $\lambda = 0.0485$

(d) mask and corresponding reconstruction for $\lambda = 0.05$

Figure 5.1: results of algorithm iPiano, test image "stripes" and different parameters λ

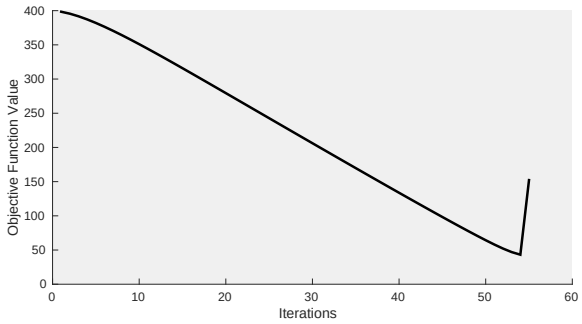
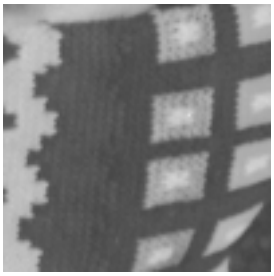


Figure 5.2: evolution of objective function value for test image stripes in iPiano

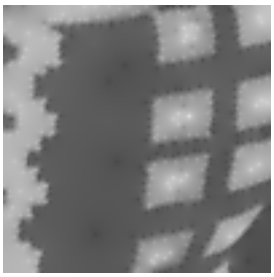
(a) test image trui (256 × 256 pixels)

(b) mask and corresponding reconstruction

Figure 5.3: results for algorithm iPiano and test image “trui”



(a) test image scarf of trui (100×100 pixels)



(b) mask and corresponding reconstruction

Figure 5.4: results for algorithm iPiano and test image "scarf of trui"

mask for the specification given in Table 5.3. We can easily recognize the smoother structure of the scarf in the reconstruction in comparison to the original image.

5.2.4 Performance

The performance clearly depends on the size of the input image. But the performance also depends on the choice of parameters in the initial problem setup.

As already seen in Figure 5.1b, the algorithm iPiano returns the same binary-valued mask for $\lambda = 0.002$ and $\lambda = 0.04$ (all other parameters fixed). But the amount of

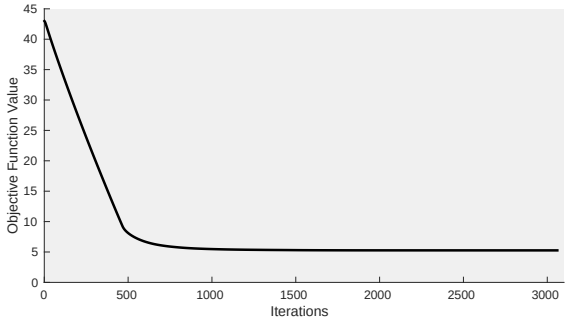


Figure 5.5: evolution of objective function value for test image scarf of trui in iPiano

λ	iterations	CPU-time (min)
0.001	1000	42.01
0.002	1000	41.77
0.04	55	2.01
0.485	46	1.79
0.05	45	1.63

Table 5.4: experiments for iPiano and test image stripes with different parameters λ: iterations and CPU-time

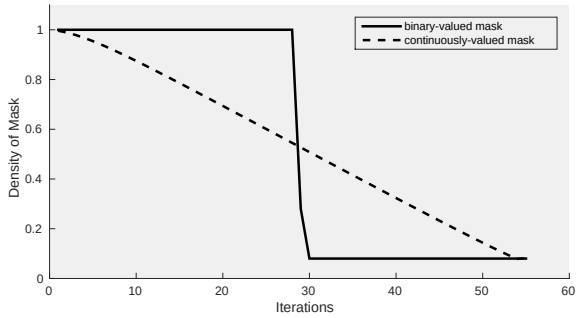
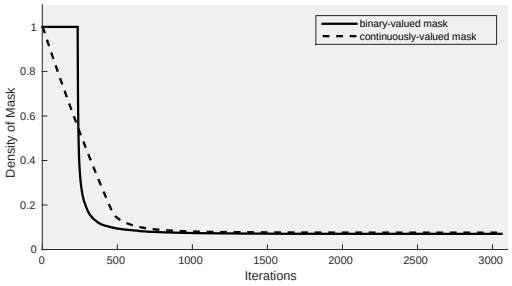


Figure 5.6: evolution of 1- and 0-norm of the iterates for test image stripes in iPiano with $\lambda = 0.04$

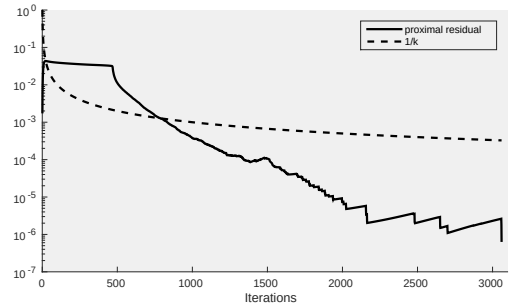
iterations and the associated CPU-times given in Table 5.4 considerably deviate by a factor of about 21. This phenomenon can be explained with the inclusion of parameter λ into the shrinkage operator (Lemma 5.2.2). The greater λ, the bigger is the value $\alpha\lambda$ subtracted from each entry to get the next iterate.

We test several values of fixed parameters λ in the interval $[0.002, 0.04]$. As expected, we recognize that the final binary-valued mask stays the same for all these input datasets. The obvious difference between the runs lies in the number of iterations until the algorithm terminates. Again, the bigger λ is chosen from this interval, the faster iPiano seems to converge. Let us have a look at the development of the entries of the mask during a run of iPiano. Exemplified for test image stripes and $\lambda = 0.04$, Figure 5.6 illustrates the density of the binary-valued as well as continuously-valued mask in each iteration until termination. The density of the actual iterate of iPiano linearly decreases over the number of iterations. But the corresponding binary-valued mask changes abruptly from the initial point (in iteration 28) to the final result (achieved in iteration 30). In the end, the algorithm terminates since the objective function value of the next iterate gets worse compared to the current state what we clearly see in Figure 5.2 from iteration 54 to 55.

In case of the test image scarf of trui, the algorithm also terminates since the objective function value becomes worse (increase of about 7.6×10^{-6}, Figure 5.5). The decrease of the density of the iterates is linear for the first 500 iterations as the plot in Figure 5.7a visualizes. But the density of the corresponding binary-valued mask rapidly falls from



(a) evolution of 1- and 0-norm of the iterates



(b) convergent squared normed proximal residual of iterates

Figure 5.7: iterates of algorithm iPiano for test image "scarf of trui"

the initial full mask (in iteration 236) to less than about 0.3 (in iteration 257), slows down and keeps slightly decreasing until termination of the algorithm (in iteration 3060).

Considering the proximal residual of each iterate, i.e. $r(c^k) = c^k - c^{k+1}$, we obtain the convergence of the generated sequence. Figure 5.7b visualizes that

$$\|r(c^k)\|_2^2 \leq \frac{1}{k} \text{ for all } k \geq 787$$

and, thus, the generated sequence $(c^k)_{k\in\mathbb{N}}$ has a convergence rate of $\mathcal{O}(1/\sqrt{k})$, i.e.

$$\|c^k - c^{k+1}\|_2 \leq \frac{1}{\sqrt{k}} \text{ for all } k \geq 787.$$

Next, we adjust the step size parameters α and β for image scarf of trui (Figure 5.4a). Since we want to raise the effect of the shrinkage operator, we test greater α which can only be taken for a smaller choice of β due to the assumed restriction in Algorithm 4.1.1. At the same time, a smaller β means less influence of the inertial force. Keeping the relation of α, β from the general initialization of our experiments, both $\beta = 0.6$ and $\beta = 0.1$ cause slower convergence to the same result (data not provided).

Both plots, Figure 5.6 and Figure 5.7a, indicate that the algorithm iPiano realizes the most computations affecting the binary-valued mask within just a few iterations in comparison to its whole run time.

5.2.5 Commentary

For further experiments and to improve the results, our implementation of iPiano should be extended at least by an update of the Lipschitz constant L in each iteration. This could be done by simple backtracking (see [OCBP14]). Since L affects parameter α, an iterative update of L would adapt the step size to the actual occurrence. The algorithm could start with a rather big step size. This would be decreased every iteration that requires a smaller step to achieve a better result than the current.

The main challenge during our implementation of iPiano was to find a good balance between the run time of the algorithm and the amount of data provided at the same time. The iterative update of L as just suggested could improve the results, but would also cause more computations during one iteration of the algorithm.

Another challenge is to determine a good choice of parameters for an experiment. The density of a mask with acceptable MSE of the corresponding reconstruction depends on the number and sharpness of edges in the image.

5.3 PALM

The linear diffusion-based image compression needs to be implemented as a nonsmooth nonconvex optimization problem that fits into the framework of the algorithm PALM. We proceed by specializing the update steps of PALM to this problem and calculate the operators. Then, the algorithm PALM is applied to several instances of the initial problem formulation. We provide different setups to be able to evaluate as well-founded as possible.

5.3.1 Specialization to Linear Diffusion-based Image Compression

Recall the linear diffusion-based image compression as a constrained discrete optimization problem from Chapter 3 in Definition 3.2.2.

We want to adopt the idea of the homogeneous diffusion inpainting and express it as a formally unconstrained problem of a sum of functions in two variables. Therefore, let $u^0 \in [0,1]^N$ be the ground truth image and $u \in [0,1]^N$ its reconstruction based on mask $c \in [0,1]^N$. The setup of PALM allows the definition of a function coupling the two variables u and c like in the constraint equation of homogeneous diffusion inpainting. Moreover, we can add functions in one variable as a restriction of the domain or as a weight. By using the notion of the indicator function given in Definition 2.3.12, we define the domains for u and c. Therefore, let $\rho B_2(u^0) := \{v \in \mathbb{R}^N \mid \|v - u^0\|_2 \leq \rho\}$ denote the closed ball around the initial data u^0.

Hence, define the coupling function $H : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ as well as the constraints $f : \mathbb{R}^N \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^N \to \overline{\mathbb{R}}$ by

$$\begin{aligned} H(u,c) &= \|\operatorname{diag}(c)(u-u^0) + (\operatorname{diag}(c) - Id)Lu\|_2^2 \\ f(u) &= \iota_{\rho B_2(u^0)\cap[0,1]^N}(u) \\ g(c) &= \iota_{[0,1]^N}(c) + \frac{\tau}{2}\|c\|_2^2. \end{aligned}$$

Next, we formulate the formally unconstrained nonsmooth nonconvex optimization problem in two variables as follows

$$\begin{aligned} &\min_{(u,c)\in\mathbb{R}^N\times\mathbb{R}^N} && H(u,c) + f(u) + g(c) \\ = &\min_{(u,c)\in\mathbb{R}^N\times\mathbb{R}^N} && \|\operatorname{diag}(c)(u-u^0) + (\operatorname{diag}(c) - Id)Lu\|_2^2 \\ & && + \iota_{\rho B_2(u^0)\cap[0,1]^N}(u) + \iota_{[0,1]^N}(c) + \tfrac{\tau}{2}\|c\|_2. \end{aligned} \tag{5.6}$$

This is our initial problem formulation for the experiments.

The next Lemma helps us during the calculation of the update steps of PALM. It provides the gradient of the squared 2-norm of a linear equation in $\mathbb{R}^N$.

Lemma 5.3.1 (gradient of squared normed linear equation [BC11, Ex. 2.48]). *Let the matrix $M \in \mathbb{R}^{N\times N}$ be real-valued and let $b \in \mathbb{R}^N$. Then,*

$$\nabla_x \|Mx + b\|_2^2 = 2M^T(Mx + b).$$

Now, we are prepared to calculate the operators for each update of PALM specialized to the linear diffusion-based image compression.

Let the optimization problem 5.6 be given. Then, the algorithm PALM updates the iterates u^k, c^k for $k \in \mathbb{N}$ in each step with the following operators. Set $\alpha_k = \alpha L_c(u^{k+1})$ and $\beta_k = \beta L_u(c^k)$.

We start by computing the update to the next reconstruction u^{k+1}. By applying *Example* 2.3.13, the proximal map of the indicator function $\iota_{\rho B_2(u^0)\cap[0,1]^N}$ reduces to the projection onto $\rho B_2(u^0) \cap [0,1]^N$, i.e.

$$\begin{aligned} u^{k+1} &\in \operatorname{prox}_{\beta_k,\, \iota_{\rho B_2(u^0)\cap[0,1]^N}} \left(u^k - \frac{1}{\beta_k}\nabla_u H(u^k, c^k)\right) \\ &= \mathrm{P}_{\rho B_2(u^0)\cap[0,1]^N} \left(u^k - \frac{1}{\beta_k}\nabla_u H(u^k, c^k)\right) \\ &= \mathrm{P}_{[0,1]^N} \left(\mathrm{P}_{\rho B_2(u^0)} \left(u^k - \frac{1}{\beta_k}\nabla_u H(u^k, c^k)\right)\right). \end{aligned}$$

We get the partial gradient of coupling function H by Lemma 5.3.1,

$$\nabla_u H(u, c) = 2A^T \left(Au - \operatorname{diag}(c)u^0\right)$$

where $A := \operatorname{diag}(c) + (\operatorname{diag}(c) - Id)L$.

Substitution into $u^{k+\frac{1}{2}} = \mathrm{P}_{\rho B_2(u^0)} \left(u^k - \frac{1}{\beta_k}\nabla_u H(u^k, c^k)\right)$ yields

$$u^{k+\frac{1}{2}} = \begin{cases} \rho \frac{u^k - \frac{1}{\beta_k}\nabla_u H(u,c) - u^0}{\|u^k - \frac{1}{\beta_k}\nabla_u H(u,c) - u^0\|_2} + u^0 & , \|u^k - \frac{1}{\beta_k}\nabla_u H(u, c) - u^0\|_2 > \rho \\ u^k - \frac{1}{\beta_k}\nabla_u H(u, c) & , \text{otherwise} \end{cases}$$

and the final iterate $u^{k+1} = \mathrm{P}_{[0,1]^N}\left(u^{k+\frac{1}{2}}\right)$ can be calculated entrywise by

$$u_j^{k+1} = \begin{cases} 0 & , u_j^{k+\frac{1}{2}} < 0 \\ u_j^{k+\frac{1}{2}} & , 0 \le u_j^{k+\frac{1}{2}} \le 1 \\ 1 & , u_j^{k+\frac{1}{2}} > 1 \end{cases}.$$

We continue with the calculation of the update to the next mask c^{k+1}. Again by

Example 2.3.13, and proximal regularization, it is

$$\begin{aligned} c^{k+1} &\in \operatorname{prox}_{\alpha_k \iota_{[0,1]^N} + \frac{\tau}{2}\|\cdot\|_2^2}\left(c^k - \frac{1}{\alpha_k}\nabla_c H(u^{k+1}, c^k)\right) \\ &= \mathrm{P}_{[0,1]^N}\left(\frac{1}{1+\alpha_k\tau}\left(c^k - \frac{1}{\alpha_k}\nabla_c H(u^{k+1}, c^k)\right)\right). \end{aligned}$$

For $M_c := \operatorname{diag}(u - u^0) + \operatorname{diag}(Lu)$ and by Lemma 5.3.1, the partial gradient of H in c is given by

$$\nabla_c H(u, c) = 2M_c^T(M_c c - Lu).$$

Concluding, for $c^{k+\frac{1}{2}} = \frac{1}{1+\alpha\tau}\left(c^k - \frac{1}{\alpha_k}\nabla_c H(u^{k+1}, c^k)\right)$, the final iterate is given by

$$c_j^{k+1} = \begin{cases} 0 & , \ c_j^{k+\frac{1}{2}} < 0 \\ c_j^{k+\frac{1}{2}} & 0 \le c_j^{k+\frac{1}{2}} \le 1 \\ 1 & , \ c_j^{k+\frac{1}{2}} > 1. \end{cases}$$

Now, we limit the Lipschitz constants of the partial gradients $\nabla_u H(u, c), \nabla_c H(u, c)$ from above and determine rough upper bounds.

Let $L_u(c)$ be the Lipschitz constant of $\nabla_u H(u, c)$ for a fixed c. Then,

$$\begin{aligned} \|\nabla_u H(u_1, c) - \nabla_u H(u_2, c)\|_2 &= \|2A^T(Au_1 + b) - 2A^T(Au_2 + b)\|_2 \\ &= \|2A^T A(u_1 - u_2)\|_2 \\ &\le \|2A^T A\|_2 \cdot \|u_1 - u_2\|_2 \end{aligned}$$

and thus $L_u(c) \le \|2A^T A\|_2$.

Let $L_c(u)$ be the Lipschitz constant of $\nabla_c H(u, c)$ for a fixed u. Then,

$$\begin{aligned} \|\nabla_c H(u, c_1) - \nabla_c H(u, c_2)\|_2 &= \|2M_c^T(M_c c_1 - Lu) - 2M_c^T(M_c c_2 - Lu)\|_2 \\ &= \|2M_c^T M_c(c_1 - c_2)\|_2 \\ &\le \|2M_c^T M_c\|_2 \cdot \|c_1 - c_2\|_2 \end{aligned}$$

and thus $L_c(u) \le \|2M_c^T M_c\|_2$.

Note that we chose continuously-valued domains for both iterate groups u^k, c^k. But the final mask should be binary-valued with entries in $\{0, 1\}$. Therefore, we round the resulting mask c^* entrywise to obtain $\overline{c}^* \in \{0, 1\}^N$ and compute the corresponding reconstruction by

$$\overline{u}^* = (\operatorname{diag}(\overline{c}^*) + (\operatorname{diag}(\overline{c}^*) - Id)L)^{-1} \operatorname{diag}(\overline{c}^*)u^0.$$

Unfortunately, this new reconstruction $\overline{u}^*$ need not lie any more in the pre-defined error-ball around the initial image u^0.

5.3.2 Initialization of Experiments

During our experiments for PALM, we test different initial setups to allow a well-founded evaluation of the algorithm. But since there exist several customizable parameters, we fix some of them to be able to keep track of the experiments.

First, we set the parameters α, β used to determine the step sizes for the groups of variables c^k, u^k, respectively. By assumption, they need to be chosen greater than 1. The bigger they are, the smaller becomes the gradient step. We take $\alpha = 2.5$ and $\beta = 1.5$.

The maximal number of iterations is limited to 2000. Additionally, we define a second stopping criterion dealing with the finite length of the generated sequence of PALM (see Theorem 4.2.3). Therefore, we define a tolerance, i.e. $\text{TOL} = 1 \times 10^{-6}$. In every iteration, we calculate the 2-norm of the current and previous iterates for each group of variables c^k, u^k separately. As long as one of the groups changes enough that means the calculated 2-norm is greater than the tolerance, the algorithm proceeds.

Note that we do not define a stopping criterion concerning the objective function value for every iteration. This is different to our implementation of iPiano. During computations, the step sizes of PALM are adjusted to the current state in every iteration. The sequence of objective function values corresponding to the iterates generated by PALM is not supposed to increase at any time.

Moreover, we initialize the starting point (u^0, c^0) with the original image u^0 and the empty mask $c^0 \equiv 0$. This means for the coupling function $H(u^0, c^0) \neq 0$.

Finally, the bounded domain of the sequence of reconstructions is defined as $\rho \mathrm{B}_2(u^0)$ with radius $\rho = 2.5 \times 10^{-4}$.

All in all, if nothing else is indicated, the experiments are initialized with the following parameters

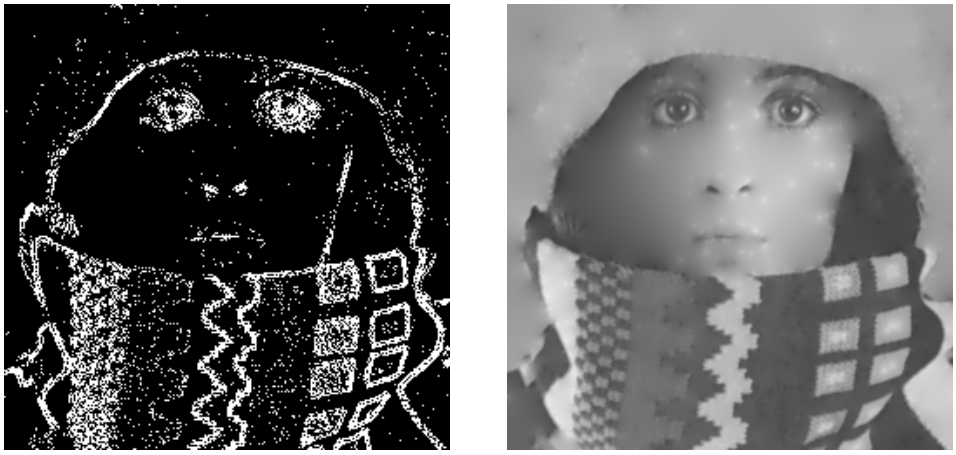
$$\begin{aligned} \alpha &= 2.5 \\ \beta &= 1.2 \\ u^0 &= u^0 \\ c^0 &\equiv 0. \end{aligned}$$

For each experiment, we vary the regularization parameter τ for the mask in the objective function to steer sparsity.

5.3.3 Results

Our implementation of Algorithm 4.2.1 PALM can be run for the standard test images with a size of 256×256 pixels like "trui" (Figure 5.8a). We present the result of one experiment in Figure 5.8b and Table 5.5. We can easily recognize the higher priority of the sharper edges in the scarf for the interpolation data in comparison to the edges in

(a) test image trui (256 × 256 pixels)



(b) mask and corresponding reconstruction

Figure 5.8: results for algorithm PALM and test image "trui"

the hat.

To facilitate comparisons to iPiano, we consider the same two test images "stripes " in Figure 5.9a and "scarf of trui " in Figure 5.11a for further analysis. The algorithm PALM clearly detects the edges in both images. A smooth area can be perfectly reconstructed from its boundary (Figure 5.9b). With raising regularization parameter τ, information of edges with comparatively low contrast are dropped first, as it was the case for iPiano in Figure 5.1 (not all data provided).

Again, the algorithm PALM determines the edges within the test image and takes pixels of them as interpolation data. The Figure 5.11 shows final binary-valued masks for the specifications given in Table 5.7.

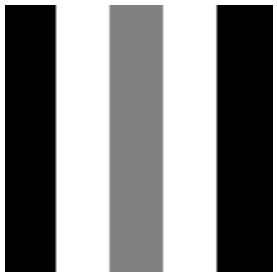
If the regularization parameter τ for the mask equals 0, there is no weight at all on the mask in the initial problem. We stopped the experiment in Figure 5.11b after the maxi-

τ	MSE	density	iterations	CPU-time (min)
0.0046	3.304×10^{-3}	0.1314	2000	1055.9

Table 5.5: experiment for PALM and test image trui

τ	MSE	density
0.01	4.288×10^{-32}	0.08
0.02	4.961×10^{-2}	0.04

Table 5.6: experiments for PALM, test image stripes and different parameters τ: MSE and density



(a) test image stripes (100×100 pixels)

(b) mask and corresponding reconstruction for $\tau = 0.01$

Figure 5.9: algorithm PALM for test image "stripes"

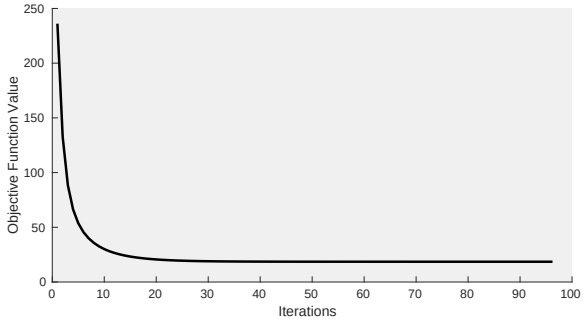


Figure 5.10: evolution of objective function value for test image stripes in PALM with regularization parameter $\tau = 0.01$

τ	MSE	density	iterations	CPU-time
0	1.230×10^{-6}	0.7566	2000	44.52
0.01	2.3×10^{-3}	0.1411	766	12.86

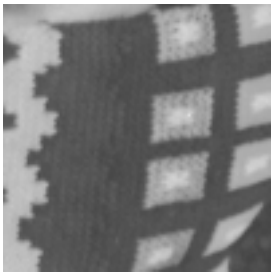
Table 5.7: experiments for PALM and test image scarf of trui and different proximal regularizations

mal number of iterations (2000) to keep the edge-detection visible in the binary-valued mask. If the entries of the iterates c^k are weighted with a regularization parameter greater than 0, the masks c^k need to be adjusted to the coupling function $H(u, c)$ as well as this additional weight. For $\tau = 0.01$ in Figure 5.11c, we can already easily recognize the smoother structure of the scarf in the reconstruction compared to the original image in Figure 5.11a.

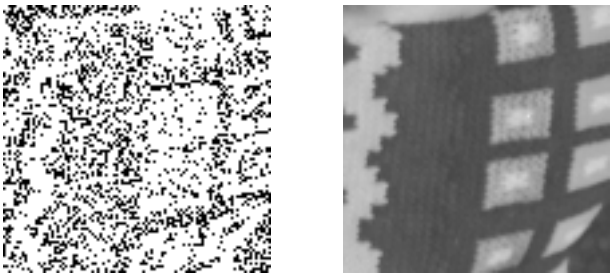
5.3.4 Performance

In our experiments, we choose a small threshold TOL. This causes longer run times than necessary with regard to the binary-valued mask. For the test image stripes, the final binary-valued mask is hit in iteration 9 as illustrated in Figure 5.13. Afterwards, only the continuously-valued mask, so the actual iterate, slightly changes over the remaining iterations. The algorithm terminates after iteration 96 (Table 5.8) since the proximal residuals of both groups of variables fall below the tolerance TOL. The corresponding objective function value nearly stagnates from iteration 25 on (Figure 5.10).

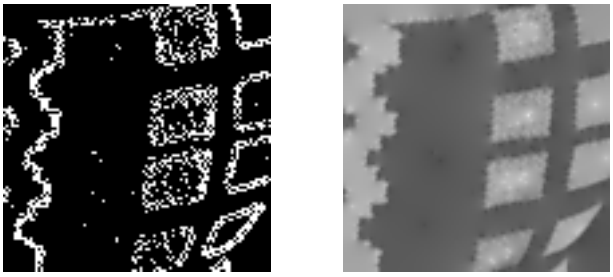
Moreover, looking at Figure 5.13, we notice that the binary-valued mask is filled in stages. First, in iteration 2, the interpolation data for the dark black stripes are determined at once. After a short waiting time, in iteration 10, the data for the gray-coloured stripe in the middle of the image are added. The actual iterates of the experiment develop in a similar but smooth way. First, their density is raised very fast, slows down and, finally from iteration 20 on, converges to a value of about 0.06. This shows that the darker an edge is, the higher is its priority for the interpolation data.



(a) test image scarf of trui (100×100 pixels)



(b) mask and corresponding reconstruction for $\tau = 0$



(c) mask and corresponding reconstruction for $\tau = 0.01$

Figure 5.11: algorithm PALM for test image "scarf of trui"

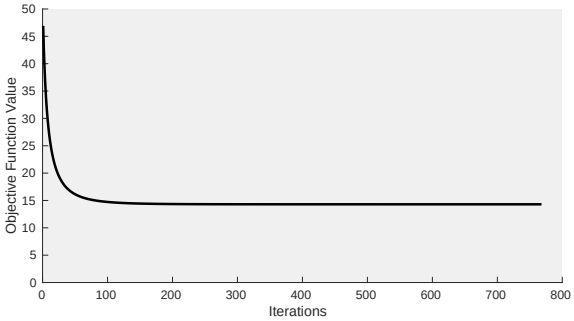


Figure 5.12: evolution of objective function value for test image scarf of trui in PALM with regularization parameter $\tau = 0.01$

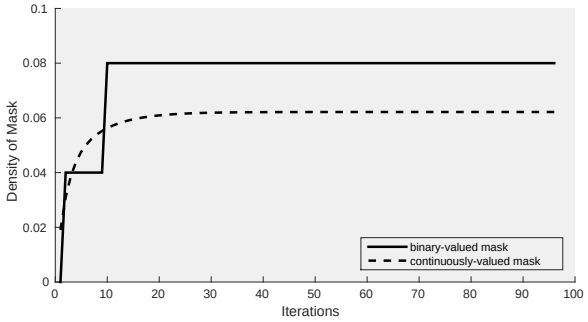


Figure 5.13: evolution of 1- and 0-norm density of the iterates for test image stripes in PALM with regularization parameter $\tau = 0.01$

τ	iterations	CPU-time
0.01	96	1.59
0.02	169	4.20

Table 5.8: experiments for PALM and test image stripes and different regularization parameters: iterations and CPU-time

For test image scarf of trui, the same initialization (like for stripes) ends up in a worse solution compared to the result of iPiano. But the CPU-time of the experiment for PALM takes about one-tenth of the time of iPiano. See Table 5.7 for the result. From iteration 200 on, the 1- and 0-norm density of the mask iterates converges to the final solution (Figure 5.14a). Moreover, the proximal residuals of the iterates for the two groups of variables, mask c^k and reconstruction u^k, in Figure 5.14b seem not to be worse in comparison to iPiano since

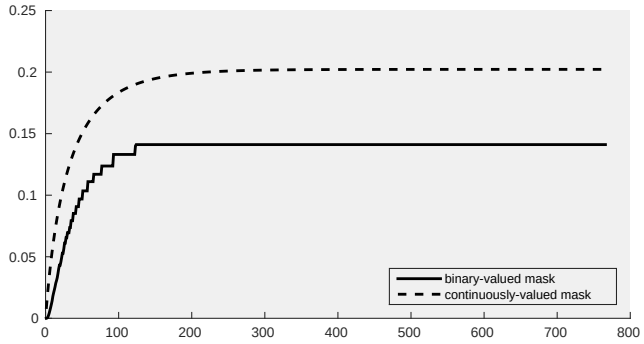
$$\|r(c^k)\|_2^2 \leq \frac{1}{k} \text{ for all } k \geq 227$$
$$\|r(u^k)\|_2^2 \leq \frac{1}{k} \text{ for all } k \in \mathbb{N}.$$

Hence, $\|c^k - c^{k+1}\|_2$ and $\|u^k - u^{k+1}\|_2$ converge in $\mathcal{O}(1/\sqrt{k})$.

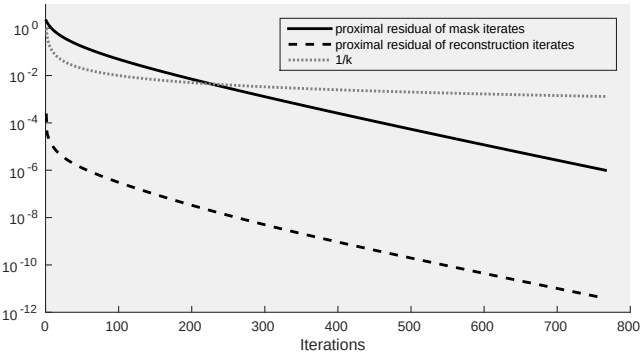
The plots in Figure 5.13 and Figure 5.14a indicate that the algorithm PALM realizes the most computations affecting the binary-valued mask within the first few iterations in comparison to its whole run time.

5.3.5 Commentary

To improve the results, further experiments could utilize the feature of additional blocking in PALM as it is suggested for the Proximal Heterogeneous Block Implicit-Explicit



(a) evolution of 1- and 0-norm density of the mask iterates



(b) convergent squared normed proximal residuals of mask and reconstruction iterates

Figure 5.14: iterates of PALM for test image scarf of trui with $\tau = 0.01$

Algorithm in [HLST15]. The idea is to split each group of variables once more into subblocks which are updated successively. In general, this allows parallel computations what speeds up the algorithm. In our problem of linear diffusion-based image compression, we could optimize mask and reconstruction for smaller pieces of the initial image (like "trui"). This would allow to precisely fit the step sizes to each single part.

As for iPiano, it is challenging to determine a good choice of parameters for an experiment with algorithm PALM. The density of a mask with acceptable MSE of the corresponding reconstruction depends on number and sharpness of edges in the image. The method of additional blocking mentioned above could aid in finding suitable parameters for each subblock independently.

5.4 iPALM

To construct the experimental setting for the new algorithm iPALM, we adopt the setup built for the algorithm PALM and expand it with the inertial force. Afterwards, we analyze the implementation of iPALM on the problem of linear diffusion-based image compression.

5.4.1 Specialization to Linear Diffusion-based Image Compression

Since iPALM is an extension of algorithm PALM and solves the same class of optimization problems, we take the initial problem formulated for PALM.

As usual, let u^0 be the ground truth image and u its reconstruction based on mask c. Define the coupling function $H : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ as well as the constraints $f : \mathbb{R}^N \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^N \to \overline{\mathbb{R}}$ by

$$\begin{aligned} H(u,c) &= \| \operatorname{diag}(c)(u-u^0) + (\operatorname{diag}(c) - Id)Lu\|_2^2 \\ f(u) &= \iota_{\rho \mathrm{B}_2(u^0)\cap[0,1]^N}(u) \\ g(c) &= \iota_{[0,1]^N}(c) + \frac{\tau}{2}\|c\|_2^2. \end{aligned}$$

The initial optimization problem for the experiments testing iPALM is the following

$$\begin{aligned} & \min_{(u,c)\in\mathbb{R}^N\times\mathbb{R}^N} && H(u,c) + f(u) + g(c) \\ = & \min_{(u,c)\in\mathbb{R}^N\times\mathbb{R}^N} && \| \operatorname{diag}(c)(u-u^0) + (\operatorname{diag}(c) - Id)Lu\|_2^2 \\ & && +\iota_{\rho \mathrm{B}_2(u^0)\cap[0,1]^N}(u) + \iota_{[0,1]^N}(c) + \tfrac{\tau}{2}\|c\|_2^2. \end{aligned} \tag{5.7}$$

The operators for updating the iterates can be calculated analogously to those of PALM in Section 5.3.1.

Let the optimization problem 5.7 be given. Then, the Algorithm 4.2.4 iPALM updates the iterates u^k, c^k for $k \in \mathbb{N}$ in each step with the following operators.

First, we compute the update to the next reconstruction u^{k+1}, i.e.

$$\begin{aligned} u^{k+1} &\in \operatorname{prox}_{\beta_k,\, \iota_{\rho \mathrm{B}_2(u^0)\cap[0,1]^N}} \left(u^k - \frac{1}{\beta_k}\nabla_u H(u^k,c^k) + \eta_k(u^k - u^{k-1}) \right) \\ &= \mathrm{P}_{[0,1]^N} \left(\mathrm{P}_{\rho \mathrm{B}_2(u^0)} \left(u^k - \frac{1}{\beta_k}\nabla_u H(u^k,c^k) + \eta_k(u^k - u^{k-1}) \right) \right). \end{aligned}$$

Let $H_{hb}(u^k,c^k) := u^k - \frac{1}{\beta_k}\nabla_u H(u^k,c^k) + \eta_k(u^k - u^{k-1})$ be the heavy-ball update step for the smooth coupling function $H(u,c)$. The projection onto the ball $\rho \mathrm{B}_2(u^0)$ is given

by

$$u^{k+\frac{1}{2}} = \begin{cases} \rho \frac{H_{hb}(u^k,c^k)-u^0}{\|H_{hb}(u^k,c^k)-u^0\|_2} + u^0 & , \|H_{hb}(u^k,c^k)-u^0\|_2 > \rho \\ H_{hb}(u^k,c^k) & , \text{otherwise} \end{cases}$$

and the final iterate $u^{k+1} = \mathrm{P}_{[0,1]^N}\left(u^{k+\frac{1}{2}}\right)$ can be calculated entrywise by

$$u_j^{k+1} = \begin{cases} 0 & , u_j^{k+\frac{1}{2}} < 0 \\ u_j^{k+\frac{1}{2}} & , 0 \leq u_j^{k+\frac{1}{2}} \leq 1 \\ 1 & , u_j^{k+\frac{1}{2}} > 1. \end{cases}$$

Next, we compute the update to the next mask c^{k+1}, i.e.

$$\begin{aligned} c^{k+1} &\in \mathrm{prox}_{\alpha_k, \iota_{[0,1]^N} + \frac{\tau}{2}\|\cdot\|_2^2}\left(c^k - \frac{1}{\alpha_k}\nabla_c H(u^{k+1},c^k) + \mu_k(c^k - c^{k-1})\right) \\ &= \mathrm{P}_{[0,1]^N}\left(\frac{1}{1+\alpha_k\tau}\left(c^k - \frac{1}{\alpha_k}\nabla_c H(u^{k+1},c^k) + \mu_k(c^k - c^{k-1})\right)\right). \end{aligned}$$

Finally, for $c^{k+\frac{1}{2}} = \frac{1}{1+\alpha_k\tau}\left(c^k - \frac{1}{\alpha_k}\nabla_c H(u^{k+1},c^k) + \mu_k(c^k - c^{k-1})\right)$, the final iterate is entrywise given by

$$c_j^{k+1} = \begin{cases} 0 & , c_j^{k+\frac{1}{2}} < 0 \\ c_j^{k+\frac{1}{2}} & 0 \leq c_j^{k+\frac{1}{2}} \leq 1 \\ 1 & , c_j^{k+\frac{1}{2}} > 1. \end{cases}$$

5.4.2 Initialization of Experiments

To permit a direct comparison of the algorithms PALM and iPALM, we initialize the experiments for iPALM with the same datasets as used for PALM. The essential difference are the additional nonzero weights of the inertial terms which we hold constant during an experiment, i.e. $\eta_k = \eta$ and $\mu_k = \mu$.

More precisely, algorithm iPALM is initialized with the following parameters

$$\begin{aligned} \alpha &= 2.5 \\ \beta &= 1.2 \\ u^0 &= u^0 \\ c^0 &\equiv 0 \\ \rho &= 2.5 \times 10^{-4} \\ \tau &= 0.01. \end{aligned}$$

We vary the tolerance TOL to get a better idea of the motion of the sequence generated by iPALM.

μ	η	TOL	MSE	density	iterations
0.5	0.5	1×10^{-4}	4.288×10^{-32}	0.08	7
0.8	0.8	1×10^{-4}	4.288×10^{-32}	0.08	3

Table 5.9: experiments for iPALM with test image stripes and different parameters μ, η

μ	η	MSE	density	iterations	CPU-time
0.5	0.5	1.57×10^{-3}	0.1592	8	0.3

Table 5.10: experiments for iPALM and test image scarf of trui

5.4.3 Results and Performance

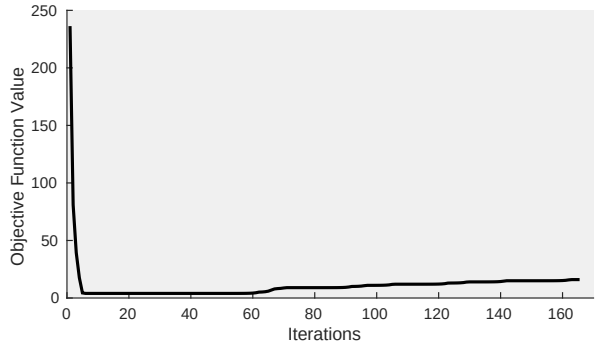
First, we set $\eta = \mu = 0.5$, TOL $= 1 \times 10^{-32}$ and the maximal number of iterations to 2000. Though, the algorithm stops after only 165 iterations (see Figure 5.15). But, unfortunately, the generated sequence ended in a worse solution compared to PALM in Figure 5.13. Considering the function value of the initial problem in Figure 5.15a, we recognize that the sequence seems to hit a (at least) local minimum in iteration 7. Then, the corresponding function value grows again, just slightly in the beginning, then faster. When we compare this plot to the evolution of the corresponding density of the iterates in Figure 5.15, we also find a decreasing density of the iterates which starts to grow again after iteration 7.

If we set the tolerance TOL to 1×10^{-4} or stop the algorithm as soon as the objective function value gets worse, algorithm iPALM stops after 7 iterations. The solution consists of a binary-valued mask with a density of 0.08 and a corresponding reconstruction with a MSE of 4.288×10^{-32} (Table 5.9). This is the same solution that PALM reaches after 10 iterations (compare Table 5.6, Figure 5.13 and Table 5.8).

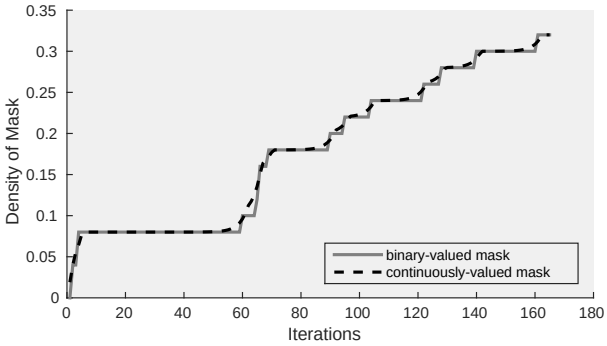
Raising the effect of the inertial terms by setting $\eta = \mu = 0.8$, the generated sequence reaches the desired solution already in iteration 3 (Table 5.9).

The smaller tolerance TOL $= 1 \times 10^{-4}$ is not enough for the experiment with test image "scarf of trui" to stop the algorithm as soon as it hits the (local) minimum. We test the same initialization as for PALM but with nonzero inertial force. The (local) minimum is reached in iteration 8 (Figure 5.16). Afterwards, the objective function value increases again. The solution achieved by iPALM and given in Figure 5.17b and Table 5.10 is different from the corresponding solution of algorithm PALM (compare Table 5.7).

An inertial force can affect the motion of a sequence generated by the algorithm such that it may leave a (local) minimum again. This is exactly what we observe in our experiment in Figure 5.15. The determination of an universal stopping criterion to prevent from ending up in a worse solution like in our example remains as an open task.



(a) evolution of objective function value



(b) evolution of density of iterates

Figure 5.15: evolution of function value and density of the iterates for test image stripes in iPALM with stopping criterion TOL ≈ 0

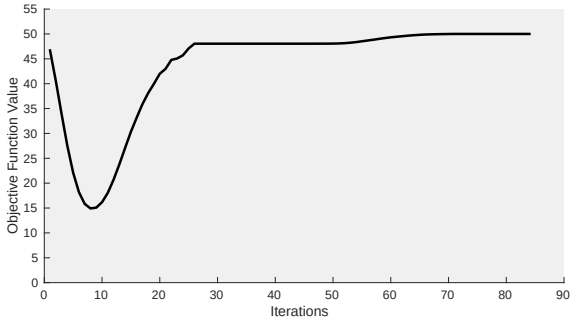
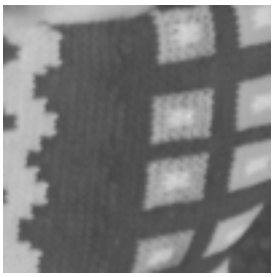
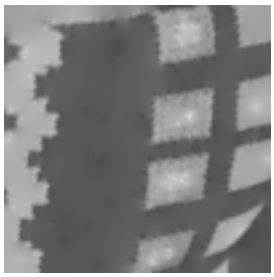


Figure 5.16: evolution of objective function value for test image scarf of trui in iPALM with inertial forces $\mu = \eta = 0.5$



(a) test image scarf of trui (100×100 pixels)



(b) mask and corresponding reconstruction for $\mu = \eta = 0.5$

Figure 5.17: algorithm iPALM for test image "scarf of trui"

6 Conclusion

This thesis investigated two conceptually different algorithms to solve the same optimization problem. However, the two algorithms iPiano and PALM rely on the same basic idea of forward-backward updates combining a gradient method with a proximal mapping.

We tested the different algorithm setups on several input data of a diffusion-based image compression problem. To facilitate comparisons, we run iPiano and PALM on the same images and under as similar conditions as possible.

The reasons to prefer iPiano as standard solver for a special class of nonsmooth nonconvex optimization problems are its simpleness and efficiency - to cite the developers. But during the numerical experiments, we could not verify a significant advantage in results or performance of algorithm iPiano compared to PALM.

A good choice of parameters that make the corresponding algorithm efficiently terminate in a favourable solution is often challenging to find. Regarding this challenge, there was no big difference between iPiano and PALM during our experiments. Quality of results and performance heavily depend on the initialization.

For both algorithms, we achieved similar results with regard to the quality of the output. Some experiments for iPiano ended in a better solution than for PALM. The amount of interpolation data is smaller whereas the corresponding reconstructed image has a worse MSE.

But with regard to the performance, algorithm PALM significantly beats iPiano. Algorithm PALM needs less computing power and time for all instances tested. Therefore, it could be interesting to extend the implementation of PALM by an additional blocking of the variables as already suggested. Probably, this makes the choice of parameters easier and the quality of results better than at present.

Another advantage of algorithm PALM is that it solves a more general class of nonsmooth nonconvex optimization problems than iPiano. Hence, this fact and our experiments suggest to take PALM as the first choice from the theoretical as well as practical point of view.

In addition, we implemented and tested a new variant of the algorithm PALM, an inertial PALM (iPALM). It combines PALM with an inertial force in the forward step. As observed during the experiments, algorithm iPALM generates sequences converging to solutions with comparable quality as achieved by iPiano or PALM. The addi-

tional inertial term caused a slightly better performance in our experiment compared to PALM. Hence, a more detailed analysis, theoretical convergence theory and universal strategies to set stopping criteria for iPALM would be worthwhile.

List of Theorems

List of Figures

List of Tables

Bibliography

[ABRS10] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal Alternating Minimization and Projection Methods for Nonconvex Problems: An Approach Based on the Kurdyka-Lojasiewicz Inequality. *Math. Oper. Res.*, 35(2):438–457, May 2010.

[BC11] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[BC15] RaduIoan Boţ and ErnöRobert Csetnek. An Inertial Tseng's Type Proximal Algorithm for Nonsmooth and Nonconvex Optimization Problems. *Journal of Optimization Theory and Applications*, pages 1–17, 2015.

[BCL15] Radu Ioan Boţ, Ernö Robert Csetnek, and Szilárd Csaba László. An inertial forward–backward algorithm for the minimization of the sum of two nonconvex functions. *EURO Journal on Computational Optimization*, pages 1–23, 2015.

[BST14] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal Alternating Linearized Minimization for Nonconvex and Nonsmooth Problems. *Math. Program.*, 146(1-2):459–494, August 2014.

[BT09] Amir Beck and Marc Teboulle. Gradient-based algorithms with applications to signal-recovery problems. In Daniel P. Palomar and Yonina C. Eldar, editors, *Convex Optimization in Signal Processing and Communications*, pages 42–88. Cambridge University Press, 2009.

[Cia13] Philippe G. Ciarlet. *Linear and Nonlinear Functional Analysis with Applications*. Applied mathematics. SIAM, Philadelphia, 2013.

[CS05] Tony Chan and Jianhong Shen. *Image Processing And Analysis: Variational, PDE, Wavelet, And Stochastic Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[Eva10] Lawrence C. Evans. *Partial differential equations*. Graduate studies in mathematics. American Mathematical Society, Providence (R.I.), 2nd edition, 2010.

[GSJ13] E. Ghadimi, I. Shames, and M. Johansson. Multi-Step Gradient Meth-

ods for Networked Optimization. *Signal Processing, IEEE Transactions on*, 61(21):5417–5429, 2013.

[GWW+08] Irena Galić, Joachim Weickert, Martin Welk, Andrés Bruhn, Alexander Belyaev, and Hans-Peter Seidel. Image Compression with Anisotropic Diffusion. *Journal of Mathematical Imaging and Vision*, 31(2-3):255–269, 2008.

[HLN14] R. Hesse, D.R. Luke, and P. Neumann. Alternating Projections and Douglas-Rachford for Sparse Affine Feasibility. *Signal Processing, IEEE Transactions on*, 62(18):4868–4881, Sept 2014.

[HLST15] Robert Hesse, D. Russell Luke, Shoham Sabach, and Matthew K. Tam. Proximal Heterogeneous Block Implicit-Explicit Method and Application to Blind Ptychographic Diffraction Imaging. *SIAM J. Imaging Sciences*, 8(1):426–457, 2015.

[HMH+15] Laurent Hoeltgen, Markus Mainberger, Sebastian Hoffmann, Joachim Weickert, Ching Hoo Tang, Simon Setzer, Daniel Johannsen, Frank Neumann, and Benjamin Doerr. Optimising Spatial and Tonal Data for PDE-based Inpainting. *CoRR*, abs/1506.04566, 2015.

[HSW13] Laurent Hoeltgen, Simon Setzer, and Joachim Weickert. An Optimal Control Approach to Find Sparse Data for Laplace Interpolation. In *EMMCVPR*, pages 151–164, 2013.

[MBWF11] Markus Mainberger, Andrés Bruhn, Joachim Weickert, and Søren Forchhammer. Edge-based compression of cartoon-like images with homogeneous diffusion, 2011.

[Mor65] J.J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299, 1965.

[MW09] Markus Mainberger and Joachim Weickert. Edge-Based Image Compression with Homogeneous Diffusion. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, CAIP '09, pages 476–483, Berlin, Heidelberg, 2009. Springer-Verlag.

[OBP15] P. Ochs, T. Brox, and T. Pock. iPiasco: Inertial Proximal Algorithm for strongly convex Optimization. *Journal of Mathematical Imaging and Vision*, 2015.

[OCBP14] Peter Ochs, Yunjin Chen, Thomas Brox, and Thomas Pock. iPiano: Inertial Proximal Algorithm for Nonconvex optimization. *SIAM J. Imaging Sciences*, 7(2):1388–1419, 2014.

[PB13] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231, 2013.

[Pol97] Elijah Polak. *Optimization: Algorithms and Consistent Approximations*. Applied Mathematical Sciences. Springer, 1997.

[Roc70] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton Mathematical Series. Princeton University Press, 1970.

[RW98] R Tyrrell Rockafellar and RJB Wets. *Variational Analysis*, volume 317. Springer: Grundlehren der Math. Wissenschaften., 1998.

[Str09] Th. Strutz. *Bilddatenkompression. Grundlagen, Codierung, Wavelets, JPEG, MPEG, H.264. 4. Aufl.* Vieweg Verlag, 2009.

[SWB09] Christian Schmaltz, Joachim Weickert, and Andrés Bruhn. Beating the quality of jpeg 2000 with anisotropic diffusion. In Joachim Denzler, Gunther Notni, and Herbert Süße, editors, *DAGM-Symposium*, volume 5748 of *Lecture Notes in Computer Science*, pages 452–461. Springer, 2009.

[ZK93] S.K. Zavriev and F.V. Kostyuk. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.

Eidesstattliche Erklärung

Mit dieser Unterschrift versichere ich,

Rebecca Nahme

dass ich die vorliegende Masterarbeit mit dem Titel

Inertial Proximal Algorithms
in Diffusion-based Image Compression

selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel verwendet habe.

Göttingen, 13.11.2015