Advanced Imaging

Tomas Sauer

Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverarbeitung
FORWISS
University of Passau
Innstr. 43
94032 Passau

Version 2.0
Last modified: February 20, 2021

Links were electronic now, not narrative … Until the advent of hyperlinks, only God had been able to see simultaneously into past, present and future alike; human beings were imprisoned in the calendar of their days.

S. Rushdie, *Fury*

Nothing spoils numbers faster than a lot of arithmetic.

Peppermint Patty, *The Peanuts, 4.12.1968*

[…] if you don't understand the math you can't write the code.

N. Stephenson, *Cryptonomicon*

The family is full of scientists. Mathematicians. The least intelligent of us become engineers.

N. Stephenson, *Cryptonomicon*

Don't ask me things I don't know. I can't tell you the answers. And don't ask me things I do know, because I won't tell you the answers.

R. Chandler, *The High Window*

To isolate mathematics from the practical demands of the sciences is to invite the sterility of a cow shut away from the bulls.

P. Chebyshev

Contents

Imaging principles | 1

Reality is software. What does it matter what system it's running on?

(R. Rucker, *Postsingular*)

In Image Processing the main task nowadays is to eventually extract *information* from data, in our case images. Ideally, this process takes into account the nature of the image, i.e., the acquisition process, the dimensionality[1] and the meaning of the picture elements, like pixels and voxels, and so on. In addition, even the context of the image may be relevant, like information when, how and for which purpose the image has been recorded. Moreover, images may have to be enhanced or cleaned in order to make the information extraction possible.

To do all that, we will need quite substantial methodology from mathematics. The idea of this lecture is to provide those mathematical tools and to relate them to task in image processing. This is what makes it "advanced" since nowadays image processing methods by far exceed the simple pushing around of pixels and require mathematics. To make it more challenging, it is not only about providing mathematical methods, it is also about implementing them efficiently and in sufficient precision on digital computers.

We will begin this lecture by giving a model for an image and by identifying some basic and fundamental concepts and ideas that will be developed in detail in the chapters to follow.

1.1 What is an image?

The simple and most classical idea of an image $\boldsymbol{B}$ is as an $m \times n$ matrix of (gray) values, i.e.,

$$\boldsymbol{F} = \begin{pmatrix} f_{11} & \cdots & f_{1n} \\ \vdots & \ddots & \vdots \\ f_{m1} & \cdots & f_{mn} \end{pmatrix} \in \mathbb{X}^{m\times n} \subset \mathbb{R}^{m\times n} \tag{1.1.1}$$

of QUANTIZED values; here $\mathbb{X}$ is a usually finite and thus discrete set of admissible values. The elements of the matrix are called PIXEL as abbreviation for "'picture element"' and one usually speaks of "'pixel (j,k)"' instead of f_{jk}.

Example 1.1.1. For an 8 bit GRAYSCALE IMAGE, we have $\mathbb{X} = \{0,\dots,255\}$.

This, "'doubly discrete"' approach has two quite obvious problems:

[1] Usual images are 2d, video sequences or volume measurements are 3d, time series of volumes even 4d.

1. Quantization is rather hard to maintain. Analytical operations on $\mathbb{X}$ like averaging, multiplication with numbers, even addition, may at least to strange behavior[2]. To overcome that, we always assume that pixels are real valued; not because quantization is irrelevant but because it is too complex to be treated here in full generality.

2. To focus on the RESOLUTION (m,n) is also not a good idea since it is very restrictive with respect to the image acquisition process. If we change the sensor[3] a change of resolution could cause problems and stop algorithms from working. And what if images from different sources have to be compared?

This makes it reasonable to consider the following mathematical model of an image.

Definition 1.1.2. An IMAGE is a function $\boldsymbol{f} : \mathbb{D} \to \mathbb{R}^d$, where $\mathbb{D}$ is the IMAGE DOMAIN and f_j, $j = 1,\dots,d$, are the IMAGE CHANNELS. The RECORDED IMAGE is the restriction of $\boldsymbol{f}$ to a GRID $\Gamma \subset \mathbb{D}$.

Example 1.1.3. For a usual pixeled image $\boldsymbol{F}$, with $m \times n$ pixels, one uses the grid

$$\Gamma = \{0,\dots,m-1\} \times \{0,\dots,n-1\} =: \mathbb{Z}_m \times \mathbb{Z}_n$$

and the matrix

$$\boldsymbol{F} = \boldsymbol{f}(\Gamma) = \begin{pmatrix} \boldsymbol{f}(m-1,0) & \dots & \boldsymbol{f}(m-1,n-1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{f}(0,0) & \dots & \boldsymbol{f}(0,n-1) \end{pmatrix},$$

where one usually puts the origin to the lower left corner of the image[4].

1.1.1 Vectorizing images and Linear Algebra

Since the values of $\boldsymbol{f}$ can be vectors[5], and usual are for the most common images, namely color photography, the matrix $\boldsymbol{F}$ is, strictly spoken, a TENSOR, that is, a higher dimensional matrix or a stack of matrices. This is also the way how Matlab or Octave handle images:

```
>> A = imread( "PassauCol.png" );
>> size (A)
ans =
864   1536      3
```

The image and its three channels are shown in Fig. 1.1.1. In the usual RGB representation, the first channel corresponds to the red content of the image, the second to green and the last one to blue. As can be easily seen, the three images are highly correlated.

[2] Just imagine 8 bit unsigned integer addition which gives, for example $231 + 109 = 85$ which is not what one might expect of the brightness of a pixel.

[3] Which could be camera or the CT equipment or whatever.

[4] This is pure convention, for standard matrix operations the upper left corner would be more consistent.

[5] To be precise: they are tuples and only become vectors once VECTOR SPACE OPERATIONS, namely addition and multiplication by scalars, are applied

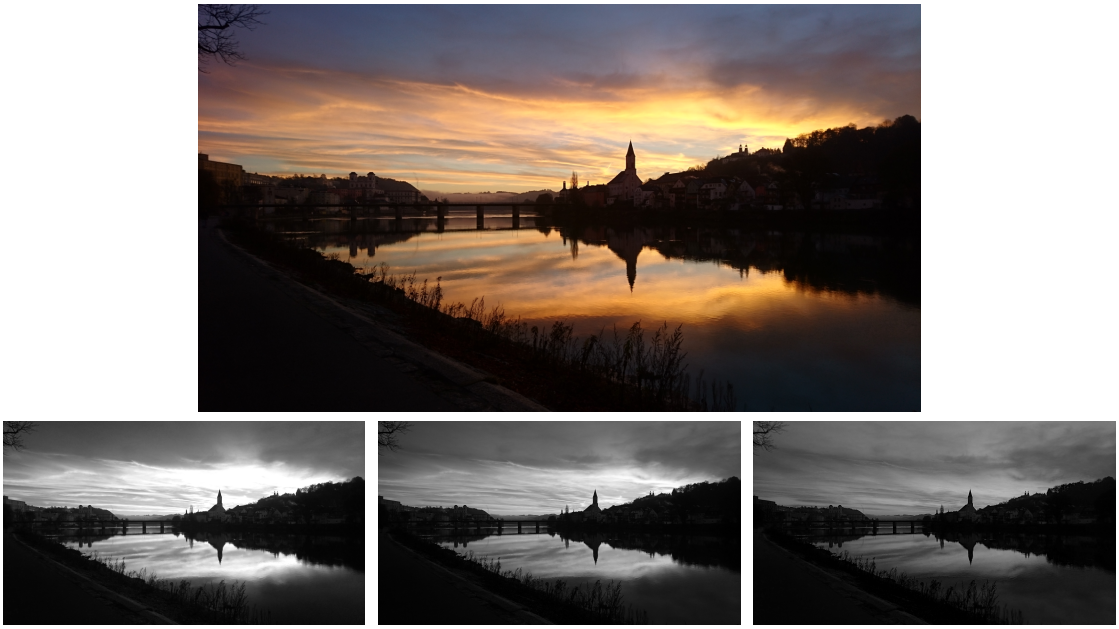

Figure 1.1.1: A color picture and its three color channels, RGB, left to right. Obviously, there is a lot of red in this image.

Definition 1.1.4. The VECTORIZATION of a matrix $\boldsymbol{F} \in \mathbb{R}^{m\times n}$ is the vector

$$v(\boldsymbol{F}) := \begin{pmatrix} f_{00} \\ \vdots \\ f_{m-1,0} \\ \vdots \\ f_{0,n-1} \\ \vdots \\ f_{m-1,n-1} \end{pmatrix}$$

obtained by stacking its columns on top of each other. The CORRELATION of two images (as matrices) is the normalized inner product of the vectorizations:

$$c(\boldsymbol{F},\boldsymbol{F}') = \frac{v(\boldsymbol{F})^T v(\boldsymbol{F}')}{\|v(\boldsymbol{F})\|_2 \|v(\boldsymbol{F}')\|_2} = \frac{1}{\|v(\boldsymbol{F})\|_2 \|v(\boldsymbol{F}')\|_2} \sum_{j,k=0}^{m-1,n-1} f_{jk} f'_{jk}. \tag{1.1.2}$$

Due to the Cauchy–Schwarz inequality we have that

$$-1 \le c(\boldsymbol{F},\boldsymbol{F}') \le 1, \tag{1.1.3}$$

and $\boldsymbol{F}$, $\boldsymbol{F}'$ are called DECORRELATED if $c(\boldsymbol{F},\boldsymbol{F}') = 0$. Moreover, $c(\boldsymbol{F},\boldsymbol{F}') = c(\boldsymbol{F}',\boldsymbol{F})$ is obvious.

Example 1.1.5. The correlations between the subimages of the example in Fig. 1.1.1 are

	R	G	B
R	1	.99028	.93169
G		1	.96124
B			1

where the subdiagonal values follow by symmetry.

The correlation means that the three channels contain a lot of *redundant* information which can become a problem, for example in compression. To overcome this, one can decorrelate the three images.
The central question is of course: how to decorrelate? The trick is by means of Linear Algebra. If we take the three channels $\boldsymbol{F}_r, \boldsymbol{F}_g, \boldsymbol{F}_b$, then we have to find three linearly independent, normalized and decorrelated image vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ such that $\|\boldsymbol{x}_j\| = 1$

$$v(\boldsymbol{F}_r) = \alpha_{r1}\boldsymbol{x}_1 + \alpha_{r2}\boldsymbol{x}_2 + \alpha_{r3}\boldsymbol{x}_3 =: \begin{bmatrix}\boldsymbol{x}_1\, \boldsymbol{x}_2\, \boldsymbol{x}_3\end{bmatrix} \alpha_r =: \boldsymbol{X}\alpha_r, \tag{1.1.4}$$

and, of course, the same for $\boldsymbol{F}_g$ and $\boldsymbol{F}_b$. That the vectors $\boldsymbol{x}_j$ are decorrelated means that the matrix $\boldsymbol{X} \in \mathbb{R}^{mn\times 3}$ that appears in all the versions of (1.1.4) and is independent of the color channel is an ORTHOGONAL MATRIX[6], that is,

$$\boldsymbol{X}^T\boldsymbol{X} = \left[\boldsymbol{x}_j^T\boldsymbol{x}_k : j,k = 1,2,3\right] = \boldsymbol{I}. \tag{1.1.5}$$

Given the matrix $\boldsymbol{X}$, the color channels of the original image are then completly described by the vectors $\alpha_r, \alpha_g, \alpha_b \in \mathbb{R}^3$ which can be easily computed by observing that (1.1.4) yields

$$\boldsymbol{X}^T v(\boldsymbol{F}_*) = \boldsymbol{X}^T\boldsymbol{X}\alpha_* = \boldsymbol{I}\alpha_* = \alpha_*, \tag{1.1.6}$$

where "$*$" stands for any of r, g, b. But (1.1.6) holds for *any* orthogonal matrix $\boldsymbol{X}$, so which one to choose? It is always a good idea to do it in an optimal way, so that the components of $\boldsymbol{X}$ distinguish between more and less relevant information. So the task is as follows:

1. Choose the first image vector $\boldsymbol{x}_1$ in such a way that it maximizes the expression

   $$\left(\boldsymbol{x}^T v(\boldsymbol{F}_r)\right)^2 + \left(\boldsymbol{x}^T v(\boldsymbol{F}_g)\right)^2 + \left(\boldsymbol{x}^T v(\boldsymbol{F}_b)\right)^2, \qquad \|\boldsymbol{x}\|_2 = 1,$$

   that is, in such a way that $\boldsymbol{x}_1$ is MAXIMALLY CORRELATED with all the color channels simultaneously,

2. choose $\boldsymbol{x}_2$ decorrelated from $\boldsymbol{x}_1$ such that it maximizes

   $$\left(\boldsymbol{x}^T v(\boldsymbol{F}_r)\right)^2 + \left(\boldsymbol{x}^T v(\boldsymbol{F}_g)\right)^2 + \left(\boldsymbol{x}^T v(\boldsymbol{F}_b)\right)^2, \qquad \boldsymbol{x}^T\boldsymbol{x}_1 = 0, \qquad \|\boldsymbol{x}\|_2 = 1,$$

3. and $\boldsymbol{x}_3$ as the maximizer of

   $$\left(\boldsymbol{x}^T v(\boldsymbol{F}_r)\right)^2 + \left(\boldsymbol{x}^T v(\boldsymbol{F}_g)\right)^2 + \left(\boldsymbol{x}^T v(\boldsymbol{F}_b)\right)^2, \qquad \boldsymbol{x}^T\boldsymbol{x}_1 = \boldsymbol{x}^T\boldsymbol{x}_2 = 0, \qquad \|\boldsymbol{x}\|_2 = 1.$$

Note that in the second and third step, the vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ computed before only enter in the SIDE CONDITIONS or RESTRICTIONS of the optimization problem, not in the target function.

Since we only have three channels, the procedure would stop here[7], in general the number of steps coincides with the dimension of the space spanned by the image components. The good thing is that the above problem has an explicit solution.

[6] Keep in mind that $\boldsymbol{X}$ is not a square matrix and therefore (1.1.5) only gives an identity matrix for $\boldsymbol{X}^T\boldsymbol{X}$ but not for $\boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{mn\times mn}$.

[7] No, this is not obvious, but we are not going to prove it here. You can find the result, for example, in the lecture notes [Sauer, 2015].

Theorem 1.1.6 (Thin SVD)**.** *Any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ can be written as*

$$\boldsymbol{A} = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \qquad r = \operatorname{rank}(\boldsymbol{A}) \le \min(m,n), \tag{1.1.7}$$

where the matrices

$$\boldsymbol{U} = \begin{bmatrix}\boldsymbol{u}_1,\dots,\boldsymbol{u}_r\end{bmatrix} \in \mathbb{R}^{m\times r} \qquad \textit{and} \qquad \boldsymbol{V} = \begin{bmatrix}\boldsymbol{v}_1,\dots,\boldsymbol{v}_r\end{bmatrix} \in \mathbb{R}^{n\times r}$$

are orthogonal[8] *and*

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0. \tag{1.1.8}$$

Moreover, the decomposition in (1.1.7) *is unique if and only if strict inequality holds in* (1.1.8)*.*

Exercise 1.1.1 Show that $\boldsymbol{u}_1, \boldsymbol{v}_1$ solve the bilinear optimization problem

$$\max \boldsymbol{u}^T \boldsymbol{A}\boldsymbol{v}, \qquad \text{subject to } \|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1. \tag{1.1.9}$$

Use this fact to derive an algorithm that computes all singular values and singular vectors, provided you can solve (1.1.9). Will this be a stable algorithm? Why? ♢

Definition 1.1.7. The decomposition (1.1.7) is called the (thin) SINGULAR VALUE DECOMPOSITION or SVD[9] of the matrix $\boldsymbol{A}$ and any of the vectors $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ is called LEFT SINGULAR VECTOR and RIGHT SINGULAR VECTOR of $\boldsymbol{A}$, respectively.

One more application of the SVD is that it can help us to make every matrix somewhat invertible. In fact, we can define the following simple inversion.

Definition 1.1.8 (Pseudoinverse)**.** Given a matrix $\boldsymbol{A}$ with SVD (1.1.7), its PSEUDOINVERSE $\boldsymbol{A}^+$ is defined as

$$\boldsymbol{A}^+ := \sum_{j=1}^{r} \frac{1}{\sigma_j} \boldsymbol{v}_j \boldsymbol{u}_j^T. \tag{1.1.10}$$

In the case where the inverse of a matrix exists, which implies that the matrix is a square one, the "pseudo" can be omitted as then it computes the inverse.

Theorem 1.1.9. *If $\boldsymbol{A}$ is invertible, then $\boldsymbol{A}^+ = \boldsymbol{A}^{-1}$.*

The proof is extremely simple, so we can give it here.
Proof: If $\boldsymbol{A}$ is invertible, it is a square matrix of full rank, i.e., $r = m = n$. Then

$$\boldsymbol{A}^+\boldsymbol{A} \;=\; \sum_{j=1}^{n} \frac{1}{\sigma_j} \boldsymbol{v}_j \boldsymbol{u}_j^T \sum_{k=1}^{n} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T = \sum_{j,k=1}^{n} \frac{\sigma_k}{\sigma_j} \boldsymbol{v}_j \underbrace{\boldsymbol{u}_j^T \boldsymbol{u}_k}_{=\delta_{j,k}} \boldsymbol{u}_k^T = \sum_{j=1}^{n} \boldsymbol{v}_j \boldsymbol{v}_j^T.$$

Now write $\boldsymbol{x} \in \mathbb{R}^n$ as $\sum x_k \boldsymbol{v}_k$ an simply note that

$$\left(\sum_{j=1}^{n} \boldsymbol{v}_j \boldsymbol{v}_j^T\right)\boldsymbol{x} = \sum_{j,k=1}^{n} x_k \boldsymbol{v}_j \underbrace{\boldsymbol{v}_j^T \boldsymbol{v}_k}_{=\delta_{j,k}} = \sum_{j,k=1}^{n} x_j \boldsymbol{v}_j = \boldsymbol{x}$$

[8] The same warning as above – they are not necessarily square matrices.
[9] Of course, the abbreviation is what you have to use if you wish to appear professional, though it never hurts to know its meaning or even the mathematical concept.

Figure 1.1.2: Decorrelation: the dominant image *(left)* and the absolute values of the orthogonal images.

to conclude that $\left(\boldsymbol{A}^{+}\boldsymbol{A}\right)\boldsymbol{x}=\boldsymbol{x}$, hence $\boldsymbol{A}^{+}\boldsymbol{A}=\boldsymbol{I}$. □

Exercise 1.1.2 Repeat the proof above to show that also $\boldsymbol{A}\boldsymbol{A}^{+}=\boldsymbol{I}$. ♢

Exercise 1.1.3 Show that if $m>n$ and $\boldsymbol{A}$ has full rank m, then $\boldsymbol{A}^{+}\boldsymbol{A}=\boldsymbol{I}$. What happens with $\boldsymbol{A}\boldsymbol{A}^{+}$? ♢

It may come as no surprise that the singular value decomposition comes to our rescue for the decorrelation problem; this is the basic idea behind the concepts of PRINCIPAL COMPONENT ANALYSIS (PCA) and INDEPENDENT COMPONENT ANALYSIS (ICA). Here we compute the thin SVD

$$\left[v(\boldsymbol{F}_r),v(\boldsymbol{F}_g),v(\boldsymbol{F}_b)\right]=\sum_{j=1}^{r}\sigma_j\,\boldsymbol{u}_j\,\boldsymbol{v}_j^T,$$

and have $\boldsymbol{u}_1$ as the dominant common part of the image, $\boldsymbol{u}_2$ as the decorrelated second most important part and $\boldsymbol{u}_3$ as the least relevant one. In `octave` this is very easy: we vectorize the channels and compute an SVD in the following way[10]

```
>> F = [ vec( A(:,:,1) ), vec( A(:,:,2) ), vec( A(:,:,3) ) ];
>> [U,S,V] = svd( F,1);
>> imwrite( reshape(ImScale8( U(:,1) ), size( A(:,:,1) ) ),
   "PassauColCor1.png" );
```

The last command normalizes the image and writes it as a `png` file with maximal contrast. The self written function `ImScale8` just scales it linearly to `uint8` integer space $0,\ldots,255$.

The images are shown in Fig. 1.1.2, the left hand side picture shows the common content of the color channels, the other two images tell us where the coloring is different. The math behind the relevance of these decompositions is the following result.

Theorem 1.1.10. *For $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ and $1\le k\le\operatorname{rank}(\boldsymbol{A})$, the solution of the optimization problems*

$$\max\sum_{j=1}^{k}\left\|\boldsymbol{A}\boldsymbol{w}_j\right\|_2^2,\qquad \textit{subject to}\qquad \boldsymbol{W}^T\boldsymbol{W}=\boldsymbol{I},\quad \boldsymbol{W}:=\left[\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k\right] \tag{1.1.11}$$

and

$$\max\sum_{j=1}^{k}\left\|\boldsymbol{w}_j^T\boldsymbol{A}\right\|_2^2,\qquad \textit{subject to}\qquad \boldsymbol{W}^T\boldsymbol{W}=\boldsymbol{I},\quad \boldsymbol{W}:=\left[\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k\right] \tag{1.1.12}$$

[10] The flag in the `svd` call is important as otherwise `octave` would a compute a FULL SVD $\boldsymbol{A}=\boldsymbol{U}\Sigma\boldsymbol{V}^T$ where $\boldsymbol{U}\in\mathbb{R}^{mn\times mn}$ would be slightly too large for the memory.

are given by $\boldsymbol{w}_j = \boldsymbol{v}_j$, *and* $\boldsymbol{w}_j = \boldsymbol{u}_j$, $j = 1, \ldots, k$, *from* (1.1.7), *respectively.*

Exercise 1.1.4 Prove Theorem 1.1.10. This is not so extremely difficult. ♢

Exercise 1.1.5 Show that $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ solve the optimization problems

$$\max \|\boldsymbol{x}^T \boldsymbol{A}\|_2 \quad \text{and} \quad \max \|\boldsymbol{A}\boldsymbol{x}\|_2, \qquad \text{subject to} \qquad \|\boldsymbol{x}\|_2 = 1,$$

respectively. ♢

Exercise 1.1.6 Show that if $\boldsymbol{A} \geq 0$ in the sense that $a_{jk} \geq 0$, then also $\boldsymbol{u}_1 \geq 0$ and $\boldsymbol{v}_1 \geq 0$. ♢

Remark 1.1.11. The result of Exercise 1.1.6 has a meaning in the context of image processing: if the matrix $\boldsymbol{A}$ consists of *nonnegative* pixel values which are usually integers between 0 and 255 or 65535 or floating point numbers between 0 and 1, then at least the dominant parts obtained by the SVD are images as well.

Example 1.1.12. The optimal decorrelation of the images into three images has been shown in Fig. 1.1.2. Clearly, two nonzero images are optimally decorrelated if[11]

$$0 = c(\boldsymbol{F}, \boldsymbol{F}') \qquad \Rightarrow \qquad 0 = v(\boldsymbol{F})^T v(\boldsymbol{F}') = \sum_{j,k=0}^{m-1,n-1} f_{jk} f'_{jk}.$$

The first image contributes around 84% of the original three images, the second one about 13% and the third one is almost irrelevant. That means that with a loss of only 16% of color image quality, a compression by a factor of 3 can be achieved without even loosing pixel information. If we simple project all three channels on the dominant decorrelated value and then recombine them into a color image, we get the picture shown in Fig 1.1.3.

Remark 1.1.13. This process is not really a clever way to compress color images. Indeed, in the JPEG standard the image is first transformed into a different color space, YCBCR, consisting of brightness and two types of CHROMINANCE, that automatically decorrelates the color information, and then even quantizes the three channels differently according to human perception. For color spaces and the respective transformations see [Foley, 1993, Foley et al., 1990].

Example 1.1.14. It is tempting to think that decorrelation may serve as a measure how similar or different images are. That this is not the case shows Fig. 1.1.4 which shows two totally decorrelated images: one is white where the other is black. Nevertheless it is hard to distinguish them visually.

[11] This defines an inner product between matrices of the same size, $(\boldsymbol{A}, \boldsymbol{B}) := v(\boldsymbol{A})^T v(\boldsymbol{B})$ that turns the matrix space into a Hilbert space whose norm is the FROBENIUS NORM

$$\|\boldsymbol{A}\|_F = \left(\sum_{j,k=1}^{m,n} a_{jk}^2 \right)^{1/2}, \qquad \boldsymbol{A} \in \mathbb{R}^{m \times n}.$$

One can have a lot of fun with this concept.

Figure 1.1.3: Reconstruction from projection on the dominant color information.

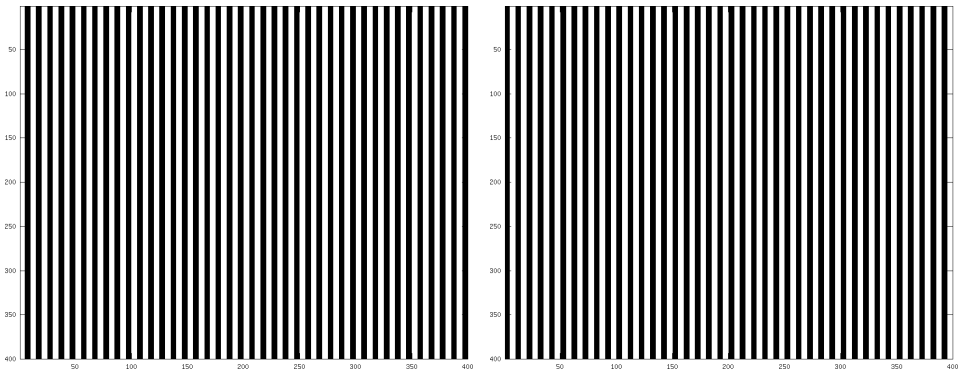

Figure 1.1.4: Two perfectly decorrelated images that are hard to distinguish

Figure 1.1.5: Frames from a surveillance video of a cat crossing a garden activating the automatic light switch on the way. The cat is small relative to the image size.

Figure 1.1.6: The singular vectors from the cat video. The first one ist nonnegative and shows the background *(left)*, the second one is "light on/light off" *(center)* and has posiive and negative entries *(right)*.

1.1.2 Vectors for videos

We can also make use of the concept of decorrelation for videos. A video is defined as a set of frames, $\boldsymbol{F}_j \in \mathbb{R}^{m\times n}$, $j = 1,\dots,N$, where each FRAME $\boldsymbol{F}_j$ is a single image of constant size[12] $m \times n$, which, for simplicity, we assume to be monochromatic or grayscale. Color could either be decorrelated as before or just treated channel by channel.

We can use the SVD concept for videos that have a large amount of static content like surveillance videos from a fixed camera where most of the image consists of BACKGROUND that only changes slowly.

Example 1.1.15. We consider a video[13] of a cat moving through a garden and build the singular vectors of matrix

$$\left[\nu(\boldsymbol{F}_1),\dots,\nu(\boldsymbol{F}_n)\right] \in \mathbb{R}^{307200\times 527}$$

The first singular vector is, as expected, the background of the scenery, cf. Fig. 1.1.6. All other singular vectors correspond to the moving cat, some shown in Fig 1.1.7.

One advantage of the singular vectors is that they allows to BACKGROUND SUBRTRACTION. Indeed, the cat is easily removed from the video by projecting any single frame on the first two singular vectors:

$$\nu(\tilde{\boldsymbol{F}}_j) = \left(\boldsymbol{u}_1\boldsymbol{u}_1^T + \boldsymbol{u}_2\boldsymbol{u}_2^T\right)\nu(\boldsymbol{F}_j) =: \boldsymbol{P}_{1:2}\,\nu(\boldsymbol{F}_j). \tag{1.1.13}$$

The projection in (1.1.13) makes sense because the singular vectors are *orthonormal*. And, of course, one does not compute the projection matrix $\boldsymbol{P}_{1:2} \in \mathbb{R}^{307200\times 307200}$ explicitly but evaluates the projection by inner products

$$\boldsymbol{P}_{1:k}\,\nu(\boldsymbol{F}) = \sum_{j=1}^{k} \boldsymbol{u}_j\left(\boldsymbol{u}_j^T\,\nu(\boldsymbol{F})\right)$$

[12] The video does not change size over time.

[13] https://www.forwiss.uni-passau.de/~sauer/leovid.avi

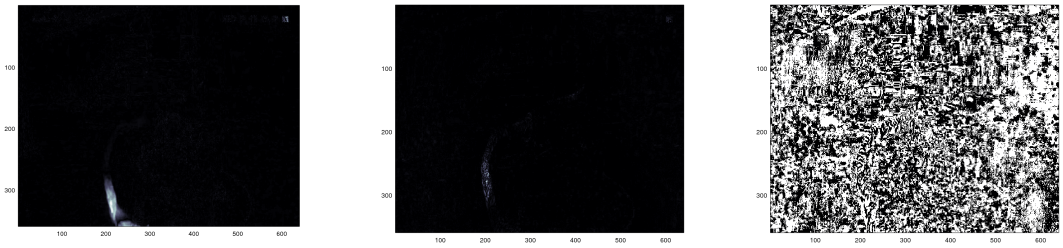

Figure 1.1.7: The singular vectors $\boldsymbol{u}_4$ *(left)* and $\boldsymbol{u}_{195}$ *(center)* with its quite erratic sign distribution *(right)*.

which needs something like $3kN$ operations for an image with N pixels. Choosing k relatively large can also be used for denoising the video.

The rest of the content, i.e., the video without background formed by the first k components is then obtained by computing

$$v(\tilde{\boldsymbol{F}}_j) = (\boldsymbol{I} - \boldsymbol{P}_{1:k})\, v(\boldsymbol{F}_j) = v(\boldsymbol{F}_j) - \sum_{j=1}^{k} \boldsymbol{u}_j \left(\boldsymbol{u}_j^T v(\boldsymbol{F})\right)$$

for any frame.

Example 1.1.16. Applying this operation directly to the cat video of Example 1.1.15 even just in a naive way already gives surprisingly satisfactory results for background[14] and the cat[15] itself.

But still there is the issue of how to compute the singular value decomposition of the image matrix and how to do this in an online way. Especially background subtraction should be a job that is done online. The first issue, efficiency, is a mathematical one and involves AUGMENTED MATRICES: given $\boldsymbol{A} \in \mathbb{R}^{N\times m}$, $N \gg m$, for which we know an SVD

$$\boldsymbol{A} = \sum_{j=1}^{m'} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \qquad m' = \operatorname{rank} \boldsymbol{A},$$

and $\mathbb{B} \in \mathbb{R}^{N\times n}$, $n \ll N$, find the SVD of $\boldsymbol{C} = [\boldsymbol{A}\,\boldsymbol{B}]$. In our video analogy this corresponds to *adding frames to an already analyzed video*. An efficient algorithm for that purpose[16] can be found in [Peña and Sauer, 2019] and the cat videos are simply a by-product of testing this method.

The method also works with more complex videos, for example a day in Passau[17] with the original webcam "video" on top and background and moving objects on the bottom. For the slowly varying lighting effects on the background, 15 singular vectors are necessary. This also means that the video on the lower left can be compressed very efficiently: one only needs 15 images and 15 numbers per frame.

Remark 1.1.17. In general, background subtraction is not so easy, starting with the nontrivial question of how many singular vectors make up the background. Analyzing the SVD process

[14] https://www.forwiss.uni-passau.de/~sauer/leovid+2.avi
[15] https://www.forwiss.uni-passau.de/~sauer/leovid-2.avi
[16] Developed, however, for a completely different application of a theoretical nature.
[17] https://www.forwiss.uni-passau.de/~sauer/passau_15.avi

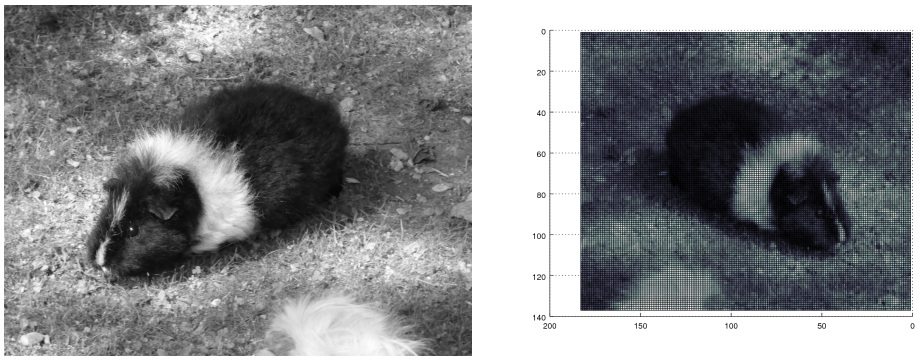

Figure 1.1.8: A grayscale image and somewhat coarser sampling.

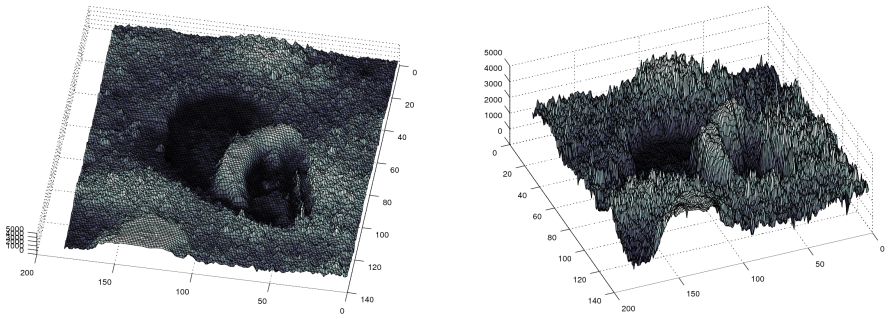

Figure 1.1.9: The image from Fig. 1.1.8 as height fields.

more carefully, one can however devise algorithms that can essentially be run in real time thanks to the efficient update and also work as excellent unsupervised methods for learning the background in videos, see [**?**].

Remark 1.1.18. There is much more that could and maybe should be said about the SVD or PCA. As long as one is interested in *linear* projections only, it is still one of the most efficient ways to extract *relevant* information from data. This changes, of course, if the data lives on more complex manifolds.

1.1.3 Continuous images

Returning to Definition 1.1.2, we now consider a single channel of the image, thus assuming that $f : \mathbb{D} \to \mathbb{R}$. This turns the image into a function whose graph is a height field over the image domain $\mathbb{D}$, see Fig 1.1.9.

This observation may appear unnecessarily fancy and just as a motiviation to plot cute 3D pictures. Nevertheless, it has the great advantage of allowing us to use techniques from Analysis in image processing. The concrete recorded image $f(\Gamma)$ is then a DISCRETIZATION, a concept very well known in and from Numerical Analysis.

Example 1.1.19. For $x \in \mathbb{D}$ the GRADIENT[18]

$$\nabla f := \frac{\partial f}{\partial x} := \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \tag{1.1.14}$$

[18]This is a good occasion to introduce the notation.

gives valuable information about the edges of the image. Since a directional derivative

$$D_y f = y^T \nabla f, \qquad y \in \mathbb{R}^2, \quad \|y\| = 1,$$

becomes maximal if $y = \nabla f / \|\nabla f\|$ and minimal[19] if $y = -\nabla f / \|\nabla f\|$ the gradient and negative gradient are the directions of STEEPEST ASCENT and STEEPEST DESCENT, respectively. Thus, locally an edge would run *orthogonal* to the gradient and a LEVEL SET where the image is constant is of the form $\{x : \nabla f(x) = 0\}$.

In many cases, the gradient of a pixeled image is simply discretized as

$$\nabla f(j,k) := \begin{pmatrix} f(j+1,k) - f(j,k) \\ f(j,k+1) - f(j,k) \end{pmatrix} =: \begin{pmatrix} f_{j+1,k} - f_{jk} \\ f_{j,k+1} - f_{jk} \end{pmatrix}, \qquad 1 \le j < m, 1 \le k < m, \tag{1.1.15}$$

by means of a FIRST ORDER DIFFERENCE. We will see later in Fig. 1.2.14 that this natural or naive discretization does not always capture all the properties of the derivative and thus has to be applied with care.

Exercise 1.1.7 Write a Matlab program that reads an image and plots the gradient directions. ♢

Images can be of even higher dimension[20]. The most important instances of a three dimensional image is

1. a VIDEO $\boldsymbol{f}(t)$ where the third dimension is time,
2. a VOLUME ELEMENT, where $f : \mathbb{R}^3 \to \mathbb{R}$ and $f(x)$ mostly describes the density or another material property of the object. Such images are generated routinely in COMPUTER TOMOGRAPHY.

There are even four dimensional objects, for example volume flow, some sort of three dimensional video. In medical imaging those can, for example, be generated by scanning a heart "'in vivo"'.

1.2 Imaging concepts

Next, we recall some fundamental concepts that will be useful in the treatment of images. The first is measure of similarity of images.

Definition 1.2.1. For two discrete images $\boldsymbol{F}, \boldsymbol{F}' \in \mathbb{R}^{m \times n}$, the PEAK SIGNAL TO NOISE RATIO or PSNR for short, is defined as

$$20 \log_{10} \left(\frac{I}{\sqrt{mn}} \left\| v(\boldsymbol{F}) - v(\boldsymbol{F}') \right\|_2 \right)^{-1}, \tag{1.2.1}$$

where I denotes the maximal pixel intensity of the images[21]. It is measured in dB (DECIBEL).

[19] And negative.

[20] Not counting channels, we talk about dimensionality of $\mathbb{D}$.

[21] It is the maximal value a pixel can assume which depends strongly on the data type of the image, in particular on whether these values are integers or floating point numbers; in the latter case one usually has $I = 1$. The reason for including I into the definition is to make it *independent* of the underlying data type.

The PSNR acts on a logarithmic scale, is normalized by the image size and the larger it is, the more similar are the pictures. For two completely identical images, the PSNR is ∞. While the PSNR ist the best established and most common similarity measure for images, it is not flawless: the two images in Fig 1.1.4 have a PSNR of zero.

The next concept, due to [Horn and Schunck, 1981] is mathematically a little bit more interesting, but also considers the difference between two images, motivated by successive frames in a video. We will derive it and then use it as an example how discretization concepts can affect the outcome of the method.

Definition 1.2.2 (Optical flow)**.** Given two images $f, g : \mathbb{D} \to \mathbb{R}$ where $f(x)$ denotes the INTENSITY[22] at the point x, the DISPLACEMENT $\phi : \mathbb{D} \to \mathbb{D}$ associates to any point x a DIRECTION[23] $y = \phi(x)$ such that

$$g(x + \phi(x)) = f(x), \qquad x \in \mathbb{D}. \tag{1.2.2}$$

Given a family f_t, $t \in [0, T]$, of images[24], the displacement can be written as a function of t,

$$f_t(x + \phi(t, x)) = f_0(x), \qquad x \in \mathbb{D}, \quad t \in [0, T]. \tag{1.2.3}$$

The OPTICAL FLOW is the *velocity*, i.e. time derivative of the displacement ϕ:

$$\phi'(t, x) = \frac{\partial \phi}{\partial t}(t, x) = \begin{pmatrix} \phi_1'(\cdot, x) \\ \phi_2'(\cdot, x) \end{pmatrix}(t) \tag{1.2.4}$$

Remark 1.2.3. A displacement $\phi(x)$ does not have to exist for any point $x \in \mathbb{D}$ in practice, only if the "pixel" appears in both or all images. In that case, one considers $\mathbb{D}$ as the set of all "trackable" elements of the image. We will not dwell on these – practically quite relevant – issues here and just assume that the displacement is defined for any $x \in \mathbb{D}$ and any t that we are interested in.

The concept of optical flow from Definition 1.2.2, considers the optical flow for a moving pixel $x(t) = x + \phi(t, x)$ in a time varying image $f(t, \cdot) := f_t$, with intensity

$$b_x(t) := b(t, x) := f\big(t, x + \phi(t, x)\big).$$

By the definition of the displacement, the intensity stays constant over time and the chain

[22] This is the same as the gray levels before.

[23] A direction is the difference between two points or, something that can be added to a point in the domain giving yet another point in the domain. Note that directions usually come from a different domain.

[24] For example a continuous video.

rule of differentiation for the function $f(t,x)$ yields that[25]

$$\begin{aligned}\frac{d}{dt}b_x &= \frac{d}{dt}f\big(t,x+\phi(t,x)\big) = \underbrace{\left[\frac{\partial f}{\partial t},\frac{\partial f}{\partial x}\right]}_{=f'}\frac{d}{dt}\begin{bmatrix}t\\ x+\phi(t,x)\end{bmatrix}\\ &= \left[\frac{\partial f}{\partial t},\frac{\partial f}{\partial x}\right]\begin{bmatrix}1\\ \frac{\partial\phi}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial t}+\frac{\partial f}{\partial x}\frac{\partial\phi}{\partial t}\end{aligned}$$

and the fact that $b_x(t)$ is the *constant* gray value of the pixel x from the original image f_0 yields for any starting point $x\in\mathbb{D}$ and any $t\in[0,T]$ that

$$0=\frac{d}{dt}b_x(t)=\frac{\partial f}{\partial t}(t,x)+\frac{\partial f}{\partial x}(t,x)\frac{\partial\phi}{\partial t}(t,x)$$

has to hold. This leads to the DIFFERENTIAL EQUATION

$$\big(\nabla_x f\big)^T\phi' = -\frac{\partial f}{\partial t},\qquad \phi(0,\cdot)=0, \tag{1.2.5}$$

for the optical flow of a continuous series of images, a so-called CONSERVATION LAW as it describes an invariance property of the process. Keep in mind that here the intensity function $f(x,t)$ is given and the equation has to be solved for the flow ϕ'.

Remark 1.2.4. The optical flow is at least consistent in the sense that a constant sequence of images, i.e., one with $f_t(x)=f_0(x)$ for all t, has $\phi'=0$ as a possible solution. If the image does not change, no flow could be an explanation.

Assuming that the image is infinite[26], the simplest discretization for this problem in space and time would be to replace derivatives by differences and get, for two successive discrete 2D images f and g,

$$\big(f(j+1,k)-f(j,k),f(j,k+1)-f(j,k)\big)\begin{pmatrix}\phi'_1\\ \phi'_2\end{pmatrix}=f(j,k)-g(j,k), \tag{1.2.6}$$

which can be easily extended to arbitrary dimensional objects into

$$\sum_{j=1}^{d}\big(f(t,\alpha+\epsilon_j)-f(t,\alpha)\big)\psi_j(\alpha)=f(t,\alpha)-f(t+1,\alpha),\qquad \psi_j=\phi'_j, \tag{1.2.7}$$

in standard MULTIINDEX notation. Note here that for the flow between two images the solution of (1.2.6) or (1.2.7), respectively, is proportional to the flow due to the initial condition in (1.2.5). Let us make this slightly more formal.

[25] Here f' denotes the TOTAL DERIVATIVE or JACOBIAN of f which is the *transpose* of the gradient. If you do not understand why this is so (and really want to know), check a good book or lecture on Analysis. The reason is that the derivative of the bivariate scalar function f is a linear map from $\mathbb{R}^3=\mathbb{R}^1\times\mathbb{R}^2$ to $\mathbb{R}$ which is represented by a 3×1 matrix, i.e., a "column vector" that consists of the *scalar* derivative $\frac{\partial f}{\partial t}\in\mathbb{R}^{1\times1}$ and the row $\frac{\partial f}{\partial x}=\big(\nabla_x f\big)^T\in\mathbb{R}^{1\times2}$. Since, on the other hand, the function $t\mapsto\phi(t,x)$ is from $\mathbb{R}$ to $\mathbb{R}^2$, its derivative $\phi'=\frac{\partial}{\partial t}\phi$ belongs to $\mathbb{R}^{2\times1}$, hence is a column vector and the product $\frac{\partial f}{\partial x}\frac{\partial\phi}{\partial t}$ is well-defined as a scalar product. This is one reason why mathematicians are so picky about notation, terminology and definition: if done right, they allow for identities that are simple to write down without being obscured by detail intricacies. Those can be explained in overlong footnotes.

[26] Otherwise we get an extra headache due to *boundary effects* of the image where the "'$k+1$"' in (1.2.6) may not be defined any more.

Definition 1.2.5. For an function $f : \mathbb{R}^d \to \mathbb{R}$ the kth PARTIAL DIFFERENCE of STEPSIZE h is defined as

$$\underbrace{f(x_1,\dots,x_k-1,x_k+h,x_{k+1},\dots,x_d)}_{:=f(x+h\epsilon_k)} - \underbrace{f(x_1,\dots,x_k-1,x_k,x_{k+1},\dots,x_d)}_{:=f(x)}.$$

Here (1.2.7) is the straightforward discretization of the optical flow equation for a sequence $f(t,x)$ of images where t and x are discrete, i.e., integers and multiintegers.

Remark 1.2.6. This derivation of the concept of optical flow nicely fits into the model of mathematical image processing: we started with a continuously changing continuous image and derived in (1.2.5) a description of this process. Only afterwards, the discretization of the differential equation gave us a simple rule for the discrete recorded object. Now, of course, we can vary the discretization for images of different resolution or involve more sophisticated numerical techniques to obtain the solution faster or with better accuracy.

Remark 1.2.7. Unfortunately, the equation (1.2.5) is what is known as an ILL POSED PROBLEM: the solution is not unique and can thus depend in a very tricky way on the input data that is usually extremely sensitive to noise.

Moreover, the optical flow is in no way unique.

Example 1.2.8. Consider the two images in Fig 1.1.4 and an optical flow between them. Suppose that $f(t,\cdot) : [0,1]^2 \to \mathbb{R}$ and that time also runs from $t = 0$ to $t = 1$ where with $t = 0$ we have the first, with $t = 1$ the second image and suppose that the image $f(1,\cdot)$ is a shift of τ. This means that

$$f(t,\cdot) = f(0,\cdot + t\tau(1,0)), \qquad t \in [0,1].$$

Then

$$\begin{aligned}\phi(t,x) &= (x_1 + t\tau, x_2),\\ \tilde{\phi}(t,x) &= (x_1 + t\tau, (1-t)x_2 + t(1-x_2)),\end{aligned}$$

are both valid optical flows. Which one is correct?

This is a general principle. Equation (1.2.5) says that the flow can only be determined up to a component perpendicular to $\frac{\partial f}{\partial x}$. This is known as the APERTURE PROBLEM and says that optical flow can only be determined *across* edges, but not *along* edges. Even worse: in a region where the image intensity is constant, i.e. $\frac{\partial f}{\partial x} = 0$, the optical flow is either *arbitrary* if $\frac{\partial f}{\partial t} = 0$ or *undefined*, if $\frac{\partial f}{\partial t} \neq 0$.

Example 1.2.9. The "'fade out"' $f(t,x) = 1 - t$, $t \in [0,1]$, $x \in [0,1]^2$, leads to the flow equation

$$0^T \phi' = 1$$

which is unsolvable.

In optical flow we thus deal with a problem that sometimes has many solutions and sometimes no solution at all. How can we treat such problems numerically? The answer is: by optimization. Instead of trying to find a solution, we just try to do our best[27]; to that end,

[27] This used to be a very American approach to problems.

we replace the *exact* solution of (1.2.5) by something that minimizes, for any TIME STEP t an ERROR FUNCTIONAL like

$$E(\phi') = \int_{[0,1]^2} \left(\left(\frac{\partial f}{\partial x} \right)^T \phi' + \frac{\partial f}{\partial t} \right)^2 (x)\, dx. \tag{1.2.8}$$

This "variational" functional measures the SQUARED ENERGY of the deviation of the flow from being ideal; the interpretation of the euclidean norm as energy has a physical background and is widely used in signal processing, cf. [Hamming, 1989, Schüßler, 1992]. There are lots of possible choices for the error functional and the particular choice of the error functional indeed can strongly influence the result. They only have to satisfy two fundamental requirements.

Definition 1.2.10. A functional $E : \mathscr{F} \to \mathbb{R}$ on a function space $\mathscr{F}$ is called an ERROR FUNCTIONAL for a functional equation if

1. $E(f) \geq 0$, $f \in \mathscr{F}$.
2. $E(f) = 0$ if and only if f satisfies the functional equation.

The QUADRATIC FUNCTIONAL (1.2.8) has a simple discretization, namely

$$\hat{E}(\psi) = \sum_{j,k} \left(\begin{pmatrix} f(j+1,k) - f(j,k) \\ f(j,k+1) - f(j,k) \end{pmatrix}^T \begin{pmatrix} \psi_1(j,k) \\ \psi_2(j,k) \end{pmatrix} + \big(g(j,k) - f(j,k) \big) \right)^2. \tag{1.2.9}$$

which is minimized by any solution of the decoupled linear systems

$$\begin{aligned} &\begin{pmatrix} f(j+1,k) - f(j,k) \\ f(j,k+1) - f(j,k) \end{pmatrix} \begin{pmatrix} f(j+1,k) - f(j,k) \\ f(j,k+1) - f(j,k) \end{pmatrix}^T \begin{pmatrix} \psi_1(j,k) \\ \psi_2(j,k) \end{pmatrix} \\ &\qquad = -\big(g(j,k) - f(j,k) \big) \begin{pmatrix} f(j+1,k) - f(j,k) \\ f(j,k+1) - f(j,k), \end{pmatrix}, \qquad \begin{matrix} j = 1,\dots,m, \\ k = 1,\dots,n. \end{matrix} \end{aligned} \tag{1.2.10}$$

Once $\boldsymbol{\psi}$ is computed from (1.2.10), the displacement function ϕ can be obtained by integration over t, discretized for frame ℓ as

$$\phi(\ell, j, k) = \sum_{t=0}^{\ell} \psi(t, j, k),$$

which can be seen as the EULER METHOD for the INTEGRATION of the ordinary differential equation $\phi' = \psi$.

Since (1.2.10) consists of $n \times m$ equations in $2(m \times n)$ unknowns, it is clear that there cannot be a unique solution, so we may still need more restrictions.

Example 1.2.11 (Example 1.2.8 continued)**.** The difference between the two choices of the displacement function is that

$$\begin{aligned} \phi'(t,x) &= (\tau, 0), \\ \tilde{\phi}'(t,x) &= (\tau, 1 - 2x_2), \end{aligned}$$

repectively, hence

$$\|\phi'\|_2 = \tau \leq \sqrt{\tau^2 + (1 - 2x_2)^2} = \|\tilde{\phi}'\|_2$$

and in particular

$$\sum_{j,k} |\phi'(j,k)|^2 < \sum_{j,k} |\tilde{\phi}'(j,k)|^2.$$

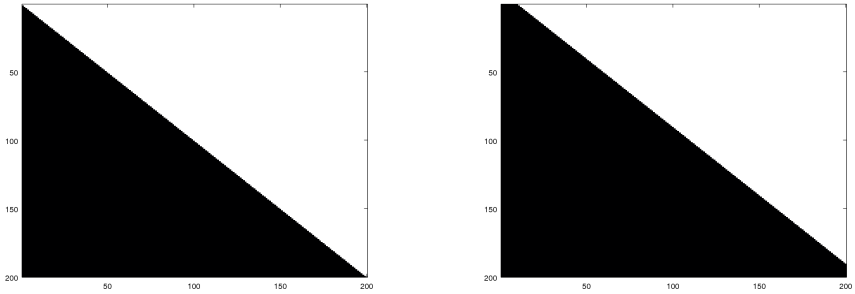

Figure 1.2.10: Two images for flow computation, shift along the edge. Black pixels stand for 0, whites one for 1.

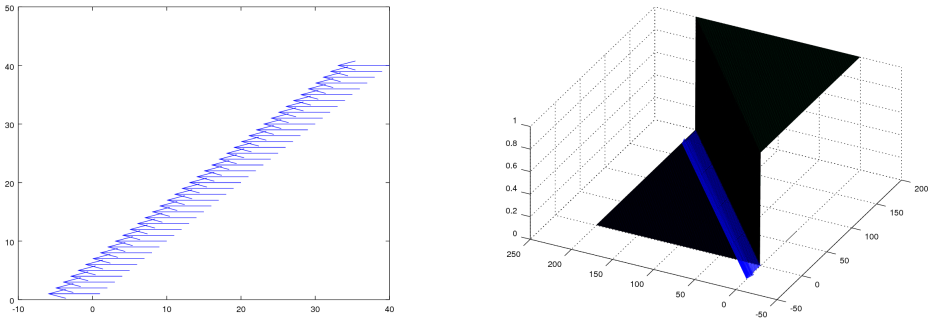

Figure 1.2.11: The flow goes downhill as the 3d image *(right)* shows.

Hence, one suggestion is to choose among the many possible solutions[28] of (1.2.10) the one that is of minimal norm. Indeed, this is what Matlab does automatically when "'solving"' the equation

$$\big(f(j+1,k)-f(j,k)\big)\,x_1+\big(f(j+1,k)-f(j,k)\big)\,x_2=-\big(g(j,k)-f(j,k)\big).$$

Let us test this by means of a very simple example. We define two images as shown in Fig 1.2.10, and compute their optical flow by means of a very simple implentation of (1.2.10).

```
>> F = triu( ones(200,200) );
>> G = triu( ones(200,200),10 );
>> X = SimpleFlow( F,G );
```

The computed flow is depicted in Fig. 1.2.11 and at least qualitatively it makes sense, pointing across the edge and "'downhill"', i.e., from value 1 to value 0. But what is the minimal norm solution of (1.2.10)? Writing the inner product there as

$$\underbrace{\begin{pmatrix} f(j+1,k)-f(j,k) \\ f(j,k+1)-f(j,k) \end{pmatrix}^T}_{=:\nabla f_{jk}^T} \boldsymbol{x} = \|\nabla f_{jk}\|\,\|\boldsymbol{x}\|\cos\theta, \qquad \theta\in(-\pi,\pi],$$

[28] The minimization problem $\min_\psi \hat{E}(\psi)$ always has at least one solution and all solutions are characterized by (1.2.10).

```
%% *** Advanced Imaging ***
%% SimpleFlow.m
%%
function B = SimpleFlow( F,G )
  [m,n] = size( F );
  G = F - G;          %% Simplify for right hand side
  B = zeros( m,n,2 );
  for j=1:m-1
    for k=1:n-1
      B(j,k,:) = [ F(j+1,k) - F(j,k), F(j,k+1)-F(j,k) ] \ G(j,k);
    end
  end
```

Figure 1.2.12: `SimpleFlow.m`: Matlab code for simple flow computation.

we get that

$$\|\boldsymbol{x}\| = \frac{f_{jk} - g_{jk}}{\|\nabla f_{jk}\| \cos\theta}$$

which is minimized with respect to θ if $\cos\theta \in \{\pm 1\}$ or $\theta \in \{0, \pi\}$ which means that $\boldsymbol{x} = \lambda \nabla f_{jk}$, $\lambda \in \mathbb{R}$, and therefore

$$\lambda \|\nabla f_{jk}\|^2 = \nabla f_{jk}^T \boldsymbol{x} = f_{jk} - g_{jk} \qquad \Rightarrow \qquad \lambda = \frac{f_{jk} - g_{jk}}{\|\nabla f_{jk}\|^2},$$

hence

$$\boldsymbol{x} = \frac{f_{jk} - g_{jk}}{\|\nabla f_{jk}\|^2} \nabla f_{jk}. \tag{1.2.11}$$

This is almost correct, except that we will have a problem in places where the intensity is constant since we would divide by zero. But then the shortest choice is indeed to choose the gradient as zero. We can formulate this in the following way.

Theorem 1.2.12. *The pointwise smallest minimizer $\boldsymbol{\psi}$ of E from* (1.2.8) *is the function*

$$\psi(x) = \begin{cases} \frac{\partial f}{\partial t} \frac{\nabla f}{\|\nabla f\|^2}, & \nabla f(x) \neq 0, \\ 0, & \nabla f(x) = 0, \end{cases} \qquad x \in [0,1]. \tag{1.2.12}$$

This looks quite good, simple and even intuitive[29], but unfortunately it presents us with more problems.

If we look carefully at Fig 1.2.11, we may realize that the flow is located quite well, but does actually not point in the right direction which would be, according to Theorem 1.2.12 *orthogonal* to the edge. The first idea could be that the solution of the ambiguous linear system may not be the shortest one, as it actually can happen with `Matlab`. But this is not the case since the program shown in Fig. 1.2.13 produces the same result, see Fig. 1.2.15.

To understand what really happens here, we have a look at Fig 1.2.14. Indeed the gradient computation by means of simple forward differences along the edge either compares identical or complementary pixels. In this case the complementary one is found in the y–direction

[29] The flow goes downhill in the stepest direction.

```
%% *** Advanced Imaging ***
%% SimpleFlowMin.m
%%
function B = SimpleFlowMin( F,G )
  [m,n] = size( F );
  G = F - G;          %% Simplify for right hand side
  B = zeros( m,n,2 );
  for j=1:m-1
    for k=1:n-1
      gF = [ F(j+1,k) - F(j,k); F(j,k+1) - F(j,k) ];
      if norm( gF,2 ) < eps    %% Never check for 0
        B(j,k,:) = 0;
      else
        B(j,k,:) = gF / ( gF'*gF) * G(j,k);
      end
    end
  end
```

Figure 1.2.13: `SimpleFlowMin.m`: Matlab code for simple flow computation with explicit minimization of length of the vector, still avoiding `pinv`.

while the identical one is located in the x–direction. To overcome this, we can really try to estimate a gradient based on *all* surrounding pixels by means of the first order Taylor identities

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & 1 \\ -1 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & -1 \end{pmatrix} \nabla f_{jk} = \begin{pmatrix} f_{j+1,k} - f_{jk} \\ \sqrt{2}(f_{j+1,k+1} - f_{jk}) \\ f_{j,k+1} - f_{jk} \\ \sqrt{2}(f_{j-1,k+1} - f_{jk}) \\ f_{j-1,k} - f_{jk} \\ \sqrt{2}(f_{j-1,k-1} - f_{jk}) \\ f_{j,k-1} - f_{jk} \\ \sqrt{2}(f_{j+1,k-1} - f_{jk}) \end{pmatrix} \tag{1.2.13}$$

and an approximate, for example least squares, solution of this overdetermined problem. This means to solve

$$\boldsymbol{A} \nabla f_{jk} = \boldsymbol{b},$$

where

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & 1 \\ -1 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & -1 \end{pmatrix}^T \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & 1 \\ -1 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 6 & 0 \\ 0 & 6 \end{pmatrix}$$

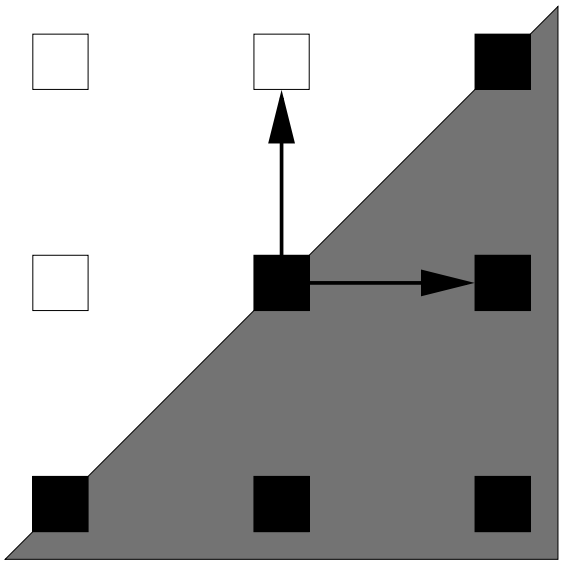

Figure 1.2.14: Discrete gradient computation leads to a coordinate direction in naive implementation.

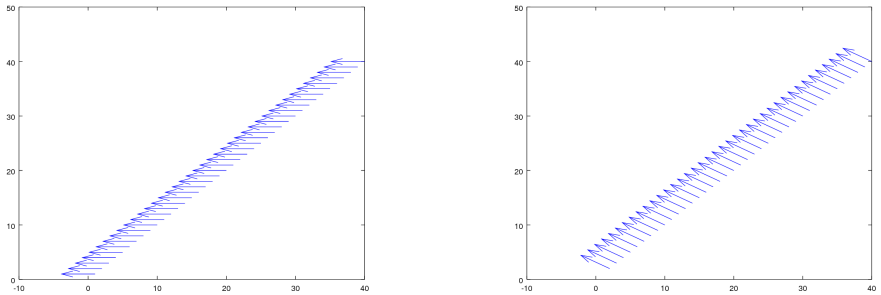

Figure 1.2.15: Minimization results without *(left)* and with *(right)* gradient estimate.

and

$$\begin{aligned} \boldsymbol{b} &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & 1 \\ -1 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & -1 \end{pmatrix}^T \begin{pmatrix} f_{j+1,k} - f_{jk} \\ \sqrt{2}(f_{j+1,k+1} - f_{jk}) \\ f_{j,k+1} - f_{jk} \\ \sqrt{2}(f_{j-1,k+1} - f_{jk}) \\ f_{j-1,k} - f_{jk} \\ \sqrt{2}(f_{j-1,k-1} - f_{jk}) \\ f_{j,k-1} - f_{jk} \\ \sqrt{2}(f_{j+1,k-1} - f_{jk}) \end{pmatrix} \\ &= \begin{pmatrix} f_{j+1,k} - f_{j-1,k} + \sqrt{2}\left(f_{j+1,k+1} - f_{j-1,k+1} + f_{j+1,k-1} - f_{j-1,k-1}\right) \\ f_{j,k+1} - f_{j,k-1} + \sqrt{2}\left(f_{j+1,k+1} - f_{j+1,k-1} + f_{j-1,k+1} - f_{j-1,k-1}\right) \end{pmatrix}. \end{aligned}$$

yielding

$$\nabla f_{jk} = \frac{1}{6}\begin{pmatrix} f_{j+1,k} - f_{j-1,k} + \sqrt{2}\left(f_{j+1,k+1} - f_{j-1,k+1} + f_{j+1,k-1} - f_{j-1,k-1}\right) \\ f_{j,k+1} - f_{j,k-1} + \sqrt{2}\left(f_{j+1,k+1} - f_{j+1,k-1} + f_{j-1,k+1} - f_{j-1,k-1}\right) \end{pmatrix} \tag{1.2.14}$$

as an estimator for the gradient that does not even involve the value of the pixel f_{jk}.

The adapted improved estimate of the gradient which, for example, has the property of being invariant to admissible rotations[30] is finally capable of recovering the flow direction properly. This is another reason why there should be a *continuous* model first and then an *appropriate* discretization of this model.

What we have shown this way by example, can and should be recorded as follows.

Observation 1.2.13. *Discretization can have significant effects on the result.*

Next, consider Fig. 1.2.10 which is a piecewise constant function whose gradient vanishes almost everywhere and therefore its gradient is not even a function[31] but only a DISTRIBUTION, see [Yosida, 1965]. Moreover, besides the conceptional problem[32], there is serious *numerical*[33] problem as well. Let us apply a small random perturbation to the two images, changing every pixel by just 1% of its value:

```
>> G1 = G.*(1+(1-2*rand(size(G))/100));
>> F1 = F.*(1+(1-2*rand(size(F))/100));
>> X1 = SimpleFlow( F1,G1 );
>> quiver( X1(1:5:end,1:5:end,1),X1(1:5:end,1:5:end,2));
```

The result as shown in Fig 1.2.17 is not too nice: the flow is distorted along the boundary and there is "'phantom flow"' in the image where it had value one[34]. Even if the differences between the images are small, the division by the also small nonzero quantity $\|\nabla f\|^2$ enlarges these values. And moreover the effect is not going to disappear if we reduce the amplitude of the perturbation.

Exercise 1.2.1 Experiment with different types of perturbation and analyze the result. ♢

[30] A discrete, pixeled image can only be rotated by multiples of 90 degrees.
[31] More precisely: as a function in the L_p sense it is zero.
[32] For that only mathematicians care, but they usually know why.
[33] This is why mathematicians care.
[34] No surprise again: any *relative* perturbation of zero still gives zero.

```
%% *** Advanced Imaging ***
%% SimpleFlowMin.m
%%
function B = SimpleFlowMinGrad( F,G )
  [m,n] = size( F );
  G = F - G;          %% Simplify for right hand side
  B = zeros( m,n,2 );
  for j=2:m-1
    for k=2:n-1
      f = F(j,k);
      grF = [ 1 0; -1 0; 0 1; 0 -1 ;
              1 1; 1 -1; -1 1;
              -1 -1 ] \ [
                          [ F(j+1,k) - f; F(j-1,k) - f;
                            F(j,k+1) - f; F(j-1,k) - f; ];
                          [ F(j+1,k+1) - f; F(j+1,k-1) - f;
                            F(j-1,k+1) - f; F(j-1,k-1) - f;
                          ] * sqrt(2)
            ];
      if norm( grF ) > 3*eps  %% Solve only if gradient != 0
        B(j,k,:) = ([ eye(2) , grF;
                      grF', 0
                    ] \ [ zeros(2,1);G(j,k)]) (1:2);
      end
    end
  end
```

Figure 1.2.16: `SimpleFlowMinGrad.m`: Estimate the gradient in a more elaborate way and minimize the length of the vector. Of course, one could simply use the gradient then.

The problem is due to the pointwise selection of the flow that does not take into account information of neighboring pixels and thus becomes very sensitive to small changes in the gradient. The way to avoid this is to require certain "'smoothness"' of the gradient, but we have to define smoothness appropriately and not only in the sense of Analysis, that is, as differentiability.

Example 1.2.14. The functions $f_k(x) = \sin k\pi x$ are all infinitely often differentiable and satisfy $f_k(\mathbb{Z}) = 0$, but the higher we choose the value of k, the more often the function is going to oscillate and therefore becomes less and less smooth in a geometric sense.

To introduce smoothness as another optimization problem, we use the next concept which is not standard, but will turn out to be useful for the context of this lecture.

Definition 1.2.15. A REGULARIZATION FUNCTIONAL $R : \mathscr{F} \to \mathbb{R}$ is a nonnegative, symmetric functional, i.e.,

$$0 \le R(f) = R(-f), \qquad f \in \mathscr{F}. \tag{1.2.15}$$

Example 1.2.16. A typical choice of a regularity functional in many application is to use a SEMINORM based on derivatives like

$$R(f) = \int \left| f^{(j)}(x) \right|^2 dx, \qquad j \in \mathbb{N}. \tag{1.2.16}$$

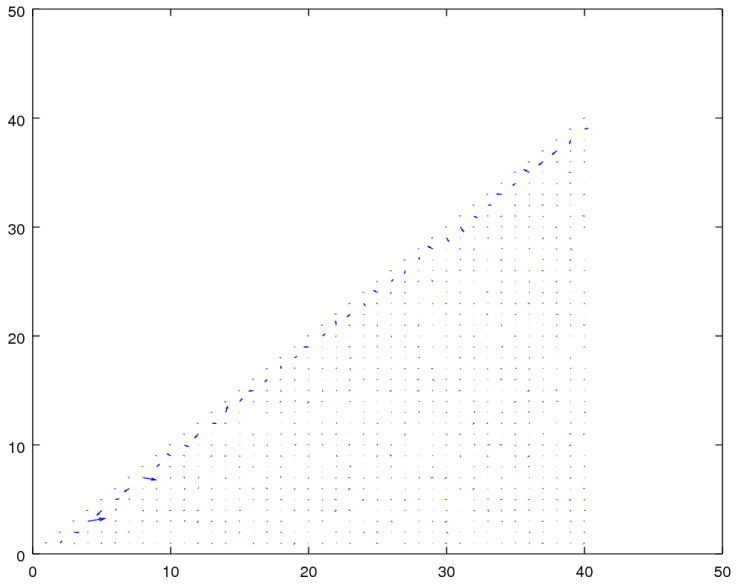

Figure 1.2.17: Flow after 1% perturbation, totally disoriented along the boundary and random in regions where the unperturbed gradient was zero.

The most popular choice of second order derivatives can be related to bending energy and leads to solutions of minimal *oscillation*. This plays a particular role in spline theory, cf. [Boor, 1978].

Example 1.2.17. In the context of Exampler 1.2.14, we could use

$$R(f) = \int_0^{2\pi} \left| f_k^{(j)}(x) \right|^2 dx = k^j \int_0^{2\pi} \left| f_k^{(j)}(x) \right|^2 dx,$$

and the obvious minimizer is the constant function $\sin 0x = 0$.

The simplest way to regularize the optical flow of our example is by considering the derivative of the flow function. Since $\psi := \phi'$ is a function in x, its derivative is the JACOBIAN

$$\psi' := \left(\frac{\partial \psi_j}{\partial x} : j = 1, \ldots, d \right)^T = \left(\frac{\partial \psi_j}{\partial x_k} : j,k = 1, \ldots, d \right)$$

which discretizes in our running example to

$$\Psi_{jk} = \begin{pmatrix} \psi_1(j+1,k) - \psi_1(j,k) & \psi_1(j,k+1) - \psi_1(j,k) \\ \psi_2(j+1,k) - \psi_2(j,k) & \psi_2(j,k+1) - \psi_2(j,k) \end{pmatrix}. \tag{1.2.17}$$

Using the convenient FROBENIUS NORM

$$\|\boldsymbol{A}\|_F^2 = \sum_{j=1}^{p} \sum_{k=1}^{q} a_{jk}^2, \qquad \boldsymbol{A} \in \mathbb{R}^{p \times q}, \tag{1.2.18}$$

we minimize the regularity functional

$$\begin{aligned} R(\psi) &= \sum_{j=1}^{m}\sum_{k=1}^{n} \|\Psi_{jk}\|_F^2 \\ &= \sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{\ell=1}^{s} \left(\left(\psi_{\ell,j+1,k} - \psi_{\ell,j,k}\right)^2 + \left(\psi_{\ell,j,k+1} - \psi_{\ell,j,k}\right)^2 \right) \\ &= \sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{\ell=1}^{s} \left\|\nabla\psi_{\ell,j,k}\right\|_2^2 \end{aligned} \tag{1.2.19}$$

subject to the side condition $\widehat{E}'(\psi) = 0$ that corresponds to solving the flow in the best possible way. This function in the discrete variables $\psi_{\ell,j,k}$ has now to be derived with respect to each of these variables.

To that end, we first note that $R(\psi) = \|\nabla\psi\|_2^2$ and that there exists a matrix $\boldsymbol{A}_\nabla \in \mathbb{R}^{2mn\times 2mn}$ such that

$$\left(\nabla\boldsymbol{\psi}\right)_{\ell,j,k} = \left(\boldsymbol{A}_\nabla\boldsymbol{\psi}\right)_{\ell,j,k}, \qquad \ell = 1,2, \quad \begin{matrix} j = 1,\dots,m \\ k = 1,\dots,n. \end{matrix}$$

The matrix $\boldsymbol{A}_\nabla$ only contains the values ± 1 and just covers the geometry of the discrete gradient. Therefore has -1 on the diagonal in the naive discretization of the gradient. It could also be replaced by the more complex formula obtained when solving (1.2.13), i.e., (1.2.14), but then $\boldsymbol{A}_\nabla$ actually is zero on the diagonal.

In either case, we have that

$$R(\psi) = \|\nabla\psi\|_2^2 = \left(\nabla\boldsymbol{\psi}\right)^T\left(\nabla\boldsymbol{\psi}\right) = \boldsymbol{\psi}^T\boldsymbol{A}_\nabla^T\boldsymbol{A}_\nabla\boldsymbol{\psi} \qquad \Rightarrow \qquad \frac{\partial R(\psi)}{\partial\boldsymbol{\psi}} = 2\,\boldsymbol{A}_\nabla^T\boldsymbol{A}_\nabla\boldsymbol{\psi}. \tag{1.2.20}$$

From the definition (1.2.9) we also get that

$$\frac{\partial\hat{E}(\psi)}{\partial\psi_{\ell,j,k}} = 2\left(\nabla f_{jk}^T\begin{pmatrix}\psi_{1,j,k}\\ \psi_{2,j,k}\end{pmatrix} - (f_{jk} - g_{jk})\right)\begin{cases} (f_{j+1,k} - f_{jk}), & \ell = 1, \\ (f_{j,k+1} - f_{jk}), & \ell = 2, \end{cases} \tag{1.2.21}$$

These two computations are needed to apply the following fundamental concept from nonlinear optimization.

Theorem 1.2.18. *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ be a differentiable* TARGET FUNCTION *and* CONSTRAINT*, respectively. Under certain technical constraints[35] on g there exists, for any solution x^* of the minimization problem*

$$\min_x f(x), \qquad \textit{subject to } g(x) = 0, \tag{1.2.22}$$

a LAGRANGE MULTIPLIER *$\lambda \in \mathbb{R}^p$ such that*

$$\nabla f(x^*) + \left(g'(x^*)\right)^T\lambda = 0. \tag{1.2.23}$$

Remark 1.2.19. Whenever Lagrange multipliers are introduced[36] and used, some things should be made clear.

1. (1.2.23) is a (often nonlinear) system of equation whose solution is a candidate for an extremum. In many applications, solving this equation is all what people care for.

[35] See, for example [Nocedal and Wright, 1999, Sauer, 2013b, Spellucci, 1993].
[36] They also appear in Analysis II when considering extrema with side conditions.

2. Since $g' : \mathbb{R}^n \to \mathbb{R}^{p\times n}$ is a matrix valued function whose columns are the gradients of the components of g, the multiplication with λ^T turns this into an n–vector valued function again. Hence, (1.2.23) is a "'square"' nonlinear system and can be approached by Newton's method which works very fast if f, g are well-behaved and if one has a good initial guess for the solution.

3. The most important thing is that (1.2.23) is only a *necessary condition* for the existence of a minimum, being a generalization of the usual $f'(x) = 0$ one knows from univariate functions. In other words: a solution of (1.2.23) is only a *candidate* for a minimum.

4. More precisely it is only a candidate for an EXTREMUM. Since f has a (local or global) minimum at x^* if and only if $-f$ has a maximum at x^* and since $\nabla f(x^*) + \lambda^T g'(x^*) = 0$ if and only if

$$0 = -\left(\nabla f(x^*) + \lambda^T g'(x^*)\right) = \nabla(-f)(x^*) + (-\lambda)^T g'(x^*),$$

   (1.2.23) holds for maxima as well.

5. There is also a version of Theorem 1.2.18 with INEQUALITY CONSTRAINTS $h : \mathbb{R}^n \to \mathbb{R}^q$ which request $h(x) \geq 0$ meaning that all components of h have to be nonnegative.

In order to be a solution of

$$\min_{\psi} R(\psi), \qquad \text{subject to } \hat{E}'(\psi) = 0, \tag{1.2.24}$$

for any optimal ψ^* there must exist a LAGRANGE MULTIPLIER $\lambda \in \mathbb{R}$ such that[37]

$$0 = \begin{pmatrix} \nabla R(\psi^*) + \lambda \nabla \hat{E}'(\psi) \\ \hat{E}'(\psi) \end{pmatrix}. \tag{1.2.25}$$

This is nice in the sense that it gives us the general approach for arbitrary regularization and energy function functionals, as long as they can be differentiated with respect to the function parameters. And we have seen this and the explicit result in (1.2.20). But in our situation here it is not a really good idea for two reasons. First, even if the gradient of both functionals is a rather simple linear function[38], the addition of $\widehat{E}$ to the function in (1.2.25) makes everything a *nonlinear* problem and requires more effort. Second, and this is more serious, we already know that in some situations it is *impossible* to find a function ψ such that $\widehat{E}(\psi) = 0$, for example when the images change in constant regions. In the language of optimization this means that the FEASIBLE SET of the optimization problem (1.2.24) can be the empty set which makes the problem meaningless and lets algorithms fail or return strange results[39].

Once more we almost had a solution and once more it is still not satisfactory. But we can relax the problem a bit and try the best possible solution that at least minimizes the error function, looking for the "best among the best"[40]. Rewriting our good old (1.2.10) as

$$0 = \nabla f_{jk}\left(\nabla f_{jk}^T \begin{pmatrix} \psi_{1,j,k} \\ \psi_{2,j,k} \end{pmatrix} + (g_{jk} - f_{jk})\right) =: G_{jk}, \qquad \begin{matrix} j = 1,\dots,m, \\ k = 1,\dots,n, \end{matrix}$$

[37] We must not forget to also encode the side condition!

[38] It can easily become more complicated.

[39] In many cases this is not the fault of the optimization algorithm but of the user who applies it incorrectly.

[40] Which does not have to be good.

we easily see that the only nonzero derivatives are

$$\frac{\partial G_{jk}}{\partial \psi_{1,j,k}} = \nabla f_{jk}\left(f_{j+1,k} - f_{jk}\right), \qquad \frac{\partial G_{jk}}{\partial \psi_{2,j,k}} = \nabla f_{jk}\left(f_{j,k+1} - f_{jk}\right),$$

or

$$\frac{\partial G_{jk}}{\partial \boldsymbol{\psi}_{jk}} = \nabla f_{jk} \nabla f_{jk}^T \tag{1.2.26}$$

If we denote again by $\boldsymbol{\psi} \in \mathbb{R}^{2mn}$ the vector of the $\boldsymbol{\psi}_{jk}$, we have to solve the system[41]

$$\begin{aligned} \boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla \boldsymbol{\psi} + \nabla f \nabla f^T \lambda &= 0, \\ \nabla f_{jk} \nabla f_{jk}^T \boldsymbol{\psi}_{jk} &= (f_{jk} - g_{jk}) \nabla f_{jk}, \qquad j,k. \end{aligned}$$

The matrix form

$$\boldsymbol{A}\begin{pmatrix} \boldsymbol{\psi} \\ \lambda \end{pmatrix} := \begin{pmatrix} \boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla & \nabla f \nabla f^T \\ \nabla f \nabla f^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ (f-g)\nabla f \end{pmatrix} \tag{1.2.27}$$

shows us that we now face a system of $3mn$ equations in $3mn$ variables, hence a matrix with about $10N^2$ entries if N denotes the number of pixels. For example, a 10 Megapixel image would result in a matrix with about 10^{15} entries when done in the naive way. It seems that we have to look for something non–naive again.

Definition 1.2.20. A matrix $A \in \mathbb{R}^{N\times N}$ is called SPARSE if it has only few nonzero entries, i.e., if

$$\#\operatorname{supp}(A) \ll N^2, \qquad \operatorname{supp}(A) := \{(j,k) : a_{jk} \neq 0\}. \tag{1.2.28}$$

For sparse matrices $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ one uses ITERATIVE METHODS for the linear system that compute a sequence of vectors

$$\mathbb{R}^N \ni \boldsymbol{x}^{(k+1)} = \boldsymbol{A}\boldsymbol{x}^{(k)} + \boldsymbol{b}, \qquad k \in \mathbb{N}_0,$$

initialized with any starting vector $\boldsymbol{x}^{(0)}$, see, for example, [Golub and van Loan, 1996, Sauer, 2013a]. Any such iteration step has a computational effort of $N + \#\operatorname{supp}(A)$. Unfortunately, classical methods like Gauss–Seidel iteration or Jacobi iteration cannot be applied here directly since they require matrices which are nonzero on the diagonal, a property that our $\boldsymbol{A}$ does not have, as can be easily seen in (1.2.27).

However, recalling the idea of iterative schemes, it is still relatively easy to build an adapted scheme: wanting to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, we choose an arbitary NONSINGULAR MATRIX $\boldsymbol{B}$ and note that the trivial equivalence

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} = (\boldsymbol{A} - \boldsymbol{B} + \boldsymbol{B})\boldsymbol{x} = (\boldsymbol{A} - \boldsymbol{B})\boldsymbol{x} + \boldsymbol{B}\boldsymbol{x} \qquad \Leftrightarrow \qquad \boldsymbol{x} = \boldsymbol{B}^{-1}\left(\boldsymbol{b} - (\boldsymbol{A} - \boldsymbol{B})\boldsymbol{x}\right) =: \boldsymbol{F}(\boldsymbol{x})$$

turns our linear system into a FIX POINT PROBLEM of the form

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x}$$

and solve it via $\boldsymbol{x}^{(n+1)} = \boldsymbol{F}(\boldsymbol{x}^{(n)})$. An appropriate $\boldsymbol{B}$ for (1.2.27) is to use, like in the JACOBI METHOD, an extended diagonal of $\boldsymbol{A}$, namely

$$\boldsymbol{B} = \begin{pmatrix} \operatorname{diag}(\boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla) & \\ & \mu I \end{pmatrix}, \qquad \mu > 0. \tag{1.2.29}$$

[41] Try to clarify by yourself where the factor 2 from (1.2.20) went and why we are permitted to drop it.

The addition of μ has the purpose to ensure that $\boldsymbol{B}$ is invertible. In fact, for any matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_n]$ the diagonal elements

$$\left(\boldsymbol{X}^T \boldsymbol{X}\right)_{jj} = \boldsymbol{x}_j^T \boldsymbol{x}_j = \|\boldsymbol{x}_j\|_2^2, \qquad j = 1, \dots, n,$$

are strictly positive iff the matrix has no zero columns. This is the case for $\boldsymbol{A}_\nabla$ since each point contributes to some difference computation and thus $\boldsymbol{B}$ from (1.2.29) is nonsingular as soon as $\mu \neq 0$.

Now the iteration takes, with $\boldsymbol{x}^{(k)} = \begin{pmatrix} \boldsymbol{\psi}^{(k)} \\ \lambda^{(k)} \end{pmatrix}$, the form

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \boldsymbol{B}^{-1} \left(\begin{pmatrix} 0 \\ (f-g)\nabla f \end{pmatrix} - \begin{pmatrix} \boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla - \operatorname{diag}(\boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla) & \nabla f \nabla f^T \\ \nabla f \nabla f^T & -\mu I \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi}^{(k)} \\ \lambda^{(k)} \end{pmatrix} \right) \\ &= \begin{pmatrix} 0 \\ \frac{f-g}{\mu} \nabla f \end{pmatrix} - \begin{pmatrix} \left(\operatorname{diag}(\boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla)^{-1} \boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla - I\right) \boldsymbol{\psi}^{(k)} + \operatorname{diag}(\boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla)^{-1} \nabla f \left(\nabla f^T \lambda^{(k)}\right) \\ \frac{1}{\mu} \nabla f \left(\nabla f^T \boldsymbol{\psi}^{(k)}\right) - \lambda^{(k)} \end{pmatrix}. \end{aligned}$$

This is doable! $\operatorname{diag}(\boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla)^{-1} \boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla - I$ is a sparse matrix and the products cost us $4mn$ operations. The SCALAR PRODUCT $\nabla f^T \boldsymbol{\psi}^{(k)}$ can be done with mn operations as well as $\nabla f^T \lambda^{(k)}$. Thus each iteration is a modest $O(mn)$.

Exercise 1.2.2 Implement the iterative solver and test it on images. ♢

There is yet another, even more popular, way to attack regularization problems which is based on *penalization*. In contrast to optimization with side conditions, it has the advantage of not being troubled by feasibility question, i.e., by the issue whether there exists something at all that satisfies the side condition.

Returning to our general idea with the error functional from Definition 1.2.10 and the regularization functional from Definition 1.2.15, we combine them, for an arbitrary value $\lambda > 0$, into an optimization problem

$$\min_f \left(E(f) + \lambda R(f)\right). \tag{1.2.30}$$

The first term measures the DATA FIDELITY, i.e., to which the minimizer really is a solution of our problem, while the second term takes into account how "nice" the function is.

Example 1.2.21. One classical example for a *penalized regularization* is the SMOOTHING SPLINE, cf. [Boor, 1978, Craven and Wahba, 1979], which finds, for given points $x_1, \dots, x_n \in [a,b]$ and values $y_1, \dots, y_n$ a SPLINE FUNCTION[42] $f : [a,b] \to \mathbb{R}$ which solves

$$\min_f \sum_{j=1}^n \left(f(x_j) - y_j\right)^2 + \lambda \int_a^b \left|f''(x)\right|^2 dx, \qquad \lambda > 0. \tag{1.2.31}$$

The REGULARITY PARAMETER $\lambda > 0$ then allows to balance between the two usually contradictionary goals of data fidelity and smoothness. If $\lambda \to 0$ then the solution of (1.2.31) will usually[43] become an INTERPOLANT, satisfying $f(x_j) = y_j$, $j = 1, \dots, n$, while for $\lambda \to \infty$ one obtains the linear function that best fits the data at the points.

[42] Whatever that is. In fact, the idea works with *any* linear space of functions.

[43] The associate spline space has to satisfy a certain relationship with the points x_j to enable interpolation.

In our case, the functional would become[44]

$$P_\lambda(\psi) := \left(\sum_{j,k=1}^{m,n} \nabla f_{jk}^T \boldsymbol{\psi}_{jk} + g_{jk} - f_{jk} \right)^2 + \lambda \sum_{j,k=1}^{n} \|\nabla \boldsymbol{\psi}_{jk}\|_2^2. \tag{1.2.32}$$

Also here, and this is the main reason why the 2–norms are used, the minimization problem is turned into a system of linear equations. Indeed, using (1.2.21), we get

$$\frac{1}{2} \frac{\partial P_\lambda}{\partial \boldsymbol{\psi}_{jk}} = \left(\sum_{j,k=1}^{m,n} \nabla f_{jk}^T \boldsymbol{\psi}_{jk} + g_{jk} - f_{jk} \right) \nabla f_{jk} + \lambda \left(\widehat{\nabla} \boldsymbol{\psi} \right)_{jk},$$

and the necessary condition $\frac{\partial P_\lambda}{\partial \boldsymbol{\psi}} = 0$ for the minimum becomes the linear system

$$\left(\nabla f \nabla f^T + \lambda \boldsymbol{A}_\nabla^T \boldsymbol{A}_\nabla \right) \boldsymbol{\psi} = (f - g) \nabla f. \tag{1.2.33}$$

The matrix that appears here has only half the size of the matrix in (1.2.27), has nonzero diagonal elements and is, with a bit of luck[45], positive definite. Under these conditions the GAUSS–SEIDEL method converges and yields a solution of (1.2.33).

Remark 1.2.22. Implementing and optimizing the Gauss–Seidel method is an art by itself. It also involves techniques like *overrelaxion* and, in particular, *preconditioning* to turn it into a really efficient method. Moreover, the choice of the parameter λ in (1.2.33) is by no way obvious.

It is time for summing up. Optical flow as a continuous model can be described as a partial differential equation (1.2.5) that has either no or too many solutions. To overcome this problem, we use a *variational* formulation and minimize an *error functional* $E(\psi)$ instead, giving us the best possible solution where the problem is unsolvable and chooses one solution where we have the choice. These solutions, however, can be very sensitive to noise since many of the problems are still ill-conditioned. To overcome this, we again use optimization and add a *regularization functional* $R(\psi)$ that measures how well–behaved the solution is. We can either solve it with precise side conditions

$$\min_\psi R(\psi) \qquad \text{subject to } E(\psi) = 0$$

or in the relaxed way

$$\min_\psi E(\psi) + \lambda R(\psi), \qquad \lambda > 0.$$

Finally we have to discretize the optimization problems and solve them numerically. This will be the recurrent scheme in this lecture.

[44] Now λ is not a Lagrange multiplier any more, but the regularity parameter. Since there is no penalized approximation with side conditions, this should not cause any confusion.

[45] In mathematics one formulates this as condition, in engineering one simply runs the algorithm and hopes for convergence.

Function space methods | 2

And thus there seems a reason in all things, even in law.

(H. Melville, *Moby Dick: or, the White Whale*)

2.1 Spaces & Distributions

Function spaces are an important concept in mathematics and thus it is no surprise that they also play a fundamental role in mathematical imaging. Let us recall some mathematical background in this section.

Definition 2.1.1. By a FUNCTION SPACE $\mathscr{F}$ we will mean a LINEAR SPACE or VECTOR SPACE of functions over[1] $\mathbb{R}$ which means that

$$f, f' \in \mathscr{F} \qquad \Rightarrow \qquad af + a'f' \in \mathscr{F}, \quad a, a' \in \mathbb{R}.$$

It is called a NORMED SPACE if there is a NORM $\|\cdot\| : \mathscr{F} \to \mathbb{R}$, i.e., a mapping such that[2]

1. $\|f\| \geq 0$ and $\|f\| = 0$ iff[3] $f = 0$,
2. $\|cf\| = |c|\,\|f\|$, $c \in \mathbb{R}$,
3. $\|f + f'\| \leq \|f\| + \|f'\|$,

such that $\mathscr{F} = \{f : \|f\| < \infty\}$. It is a BANACH SPACE if it is COMPLETE with respect to the norm, i.e., if any CAUCHY SEQUENCE f_n defined by

$$\lim_{n\to\infty} \sup_{m\geq n} \|f_m - f_n\| = 0, \tag{2.1.1}$$

has a limit in $\mathscr{F}$.

Example 2.1.2 (Banach spaces)**.**

1. If $\Omega \subset \mathbb{R}^s$ is compact, then the space $C(\Omega)$ of all continuous functions $\Omega \to \mathbb{R}$ is a Banach space for the norm
$$\|f\|_\infty := \|f\|_{\Omega,\infty} := \max_{x\in\Omega} |f(x)|. \tag{2.1.2}$$

[1] Sometimes $\mathbb{C}$ can also be a reasonable choice, but we will remain realistic here.
[2] These are the NORM AXIOMS.
[3] This abbreviation of *"if and only if"* is the invention of Paul Halmos, see [Halmos, 1988].

2. For $1 \le p < \infty$ and $\Omega \subseteq \mathbb{R}^n$ the L_p SPACE $L_p(\Omega)$ consisting of a all functions such that the LEBESGUE INTEGRAL

$$\|f\|_p := \|f\|_{\Omega,p} = \left(\int_\Omega |f(x)|^p \, dx \right)^{1/p} < \infty \tag{2.1.3}$$

is finite, is a Banach space.

3. The continously differentiable functions $C^1(\Omega)$ are *no* Banach space with respect to $\|\cdot\|_\infty$, but with respect to

$$\|f\| := \|f\|_\infty + \|Df\|_\infty \qquad \text{or} \qquad \|f\| := \max\left(\|f\|_\infty, \|Df\|_\infty\right). \tag{2.1.4}$$

Note that $\|Df\|_\infty = \|\nabla f\|_\infty$ depends on a vector norm for the gradient and is no norm, but only a SEMINORM as it vanishes for any constant function.

Things are getting sightly more tricky with infinitely differentiable functions. We define by

$$C^\infty_{00}(\Omega) = \left\{ f \in C^\infty(\Omega) : \operatorname{supp}(f) \text{ compact} \right\},$$

where the SUPPORT of a function is defined as the CLOSURE

$$\operatorname{supp}\,(f) = \overline{\{x : f(x) \neq 0\}}$$

of the set of points where the function does not vanish. Now set, for a compact $K \subseteq \Omega$

$$D_K(\Omega) = \left\{ f \in C^\infty_{00}(\Omega) : \operatorname{supp}\,(f) \subseteq K \right\}$$

and define a topology on that space by using open sets of the form[4]

$$U_{n,\varepsilon} = \left\{ f : \max_{j=0,\dots,n} \|D^j f\|_{K,\infty} < \varepsilon \right\}, \qquad n \in \mathbb{N}_0, \varepsilon > 0,$$

where

$$D^j f \simeq \left[\frac{\partial^j}{\partial x^\alpha} f : |\alpha| = j \right], \qquad j \in \mathbb{N}_0,$$

stands for the jth (total) DERIVATIVE of f. The limit of D_K over increasing compact subsets[5] of Ω then defines a space $D(\Omega)$ with a nice metric.

Lemma 2.1.3 (see [Yosida, 1965], I.1.Proposition 7)**.** *A sequence $\phi_n \in D(\Omega)$, $n \in \mathbb{N}$, converges to $\phi \in D(\Omega)$ if there is a compact $K \subset \Omega$ such that supp $(f_n) \subseteq K$ and*

$$\lim_{n\to\infty} \left\| D^k(\phi_n - \phi) \right\|_{K,\infty} = 0, \qquad k \in \mathbb{N}_0.$$

Definition 2.1.4. The space $D(\Omega) \subset C^\infty_{00}(\Omega)$ is called the space of TEST FUNCTIONS[6]. A DISTRIBUTION or GENERALIZED FUNCTION T is a CONTINUOUS LINEAR FUNCTIONAL on $D(\Omega)$, i.e., a continuous linear mapping $D(\Omega) \to \mathbb{R}$. The space of all generalized functions is denoted by $D(\Omega)'$.

[4] For the precise details see [Yosida, 1965].
[5] This is relevant if Ω is unbounded, in particular $\Omega = \mathbb{R}^s$, if Ω is compact, we do not need the K.
[6] They are distinguished by the fact that they are very *nice* functions.

Example 2.1.5. Any LOCALLY INTEGRABLE[7] function $f : \Omega \to \mathbb{R}$ defines a distribution

$$T_f : \phi \mapsto \int_\Omega f(x)\phi(x)\,dx. \tag{2.1.5}$$

This is the canonical imbedding of integrable functions or the interpretation of a function as generalized function.

The main point of generalized functions in image processing is that, seen as a distribution, any function is infinitely differentiable. To see this, suppose that $f \in C^1(\Omega)$ let ϕ be any test function such that[8] $K = \text{supp}\,(\phi) \subset \Omega^\circ$. Then we get by partial integration that

$$T_{\partial f/\partial x_k}(\phi) = \int_\Omega \frac{\partial f}{\partial x_k}(x)\,\phi(x)\,dx = -\int_\Omega f(x)\frac{\partial \phi}{\partial x_k}(x)\,dx.$$

Definition 2.1.6. The DISTRIBUTIONAL DERIVATIVE $\frac{\partial f}{\partial x_k}$ of the function f is the functional

$$\phi \mapsto -\int_\Omega f(x)\frac{\partial \phi}{\partial x_k}(x)\,dx, \qquad \phi \in D(\Omega).$$

If the distributional derivative is REGULAR, i.e., can be written in the form (2.1.5) for some locally integrable f, then its interpretation as a function makes sense, if not, it should be taken with care. The following standard example shows this.

Example 2.1.7. The distributional derivative of the function $f(x) = |x|$, $x \in \mathbb{R}$, is the SIGN FUNCTION

$$f'(x) = \begin{cases} -1, & x < 0 \\ 1, & x > 0, \end{cases} \qquad x \in \mathbb{R}, \qquad f(0) = 0,$$

where the value at 0 is irrelevant[9]. To compute the derivative of the sign function, we note that for $K \subset (0,\infty)$

$$T_{f''}(\phi) = -\int_K f'(x)\,\phi'(x)\,dx = \int_0^\infty \underbrace{\frac{d}{dx}1}_{=0}\phi(x)\,dx = 0,$$

and the same holds for $K \subset (-\infty, 0)$. For $K = [a,b]$, $a < 0 < b$, we then have that

$$\begin{aligned} T_{f''}(\phi) &= -\int_a^b f'(x)\,\phi(x)\,dx = \int_a^0 \phi'(x)\,dx - \int_0^b \phi'(x)\,dx \\ &= \phi(0) - \underbrace{\phi(a)}_{=0} - \underbrace{\phi(b)}_{=0} + \phi(0) = 2\phi(0), \end{aligned}$$

the second derivative of the absolute value or the derivative of the Heaviside function is therefore the DIRAC DISTRIBUTION δ, defined by $\delta f = f(0)$. This is not a function any more and to treat it like a function is a popular misbehavior among physicists and engineers.

Exercise 2.1.1 The HEAVISIDE FUNCTION is defined as

$$H(x) = \begin{cases} 0, & x < 0, \\ 1, & x > 0, \end{cases} \qquad x \in \mathbb{R}.$$

[7] This means that $\int_K |f(x)|\,dx < \infty$ for any compact $K \subset \Omega$.
[8] For $\Omega = \mathbb{R}^n$ this is always the case.
[9] The f of (2.1.5) is only unique up to a set of measure zero.

1. Show that the Heaviside function is locally integrable.
2. Write the sign function in terms of the Heaviside function.
3. Compute the derivative of the Heaviside function.

♢

The Dirac distribution is the limit of the functions

$$f_n(x) = \begin{cases} n - n^2|x|, & x \in \left[-\frac{1}{n}, \frac{1}{n}\right], \\ 0, & \text{otherwise,} \end{cases} \qquad x \in \mathbb{R}, \qquad n \in \mathbb{N}, \tag{2.1.6}$$

in the sense that[10]

$$\lim_{n\to\infty} T_{f_n}(\phi) = \delta\phi = \phi(0), \qquad \phi \in D(\Omega),$$

which is known as WEAK* CONVERGENCE. It is often interpreted as the Dirac "function" being infinite and perfectly localized at 0, which is a reasonable intuition but mathematically incorrect. However, it confirms our preceding intuition on edges: The sign function has a jump at 0 and hence its derivative there is even infinite. Mathematically: it can be *approximated* by a sequence of functions such that $\|f_n\|_1 = 1$ and $\|f_n\|_\infty \to \infty$.

Remark 2.1.8. Working with distributions as generalized functions, one has to be a bit careful. Many results extend to distributions, Fourier Analysis can be done almost completely with *tempered distributions*, but proofs become more tricky. And there are plenty of proofs around that work with a "Dirac function" and are plainly wrong.

2.2 Total and bounded variation

To represent images by means of function spaces also means that the associated norm measures the complexity of the image. The most classical one is the L_2–norm, often called ENERGY NORM in the signal processing context. It measures how "bright" the pixels are in average. This, however, has nothing to do with the content of an image or the shapes it contains. For that purpose, another norm is more suitable.

Let us begin with the univariate case here to get the geometric idea of the concept.

Definition 2.2.1 (Variation)**.** Let $f : I = [a, b] \to \mathbb{R}$ and $a = x_0 < \cdots < x_n = b$.

1. The VARIATION of f with respect to $x_0, \ldots, x_n$ is defined as

$$V(f; x_0, \ldots, x_n) := \sum_{j=1}^{n} \left| f\left(x_j\right) - f\left(x_{j-1}\right) \right|. \tag{2.2.1}$$

2. The TOTAL VARIATION of f is

$$V(f) := \lim_{\delta\to 0} \sup_{x_{j+1}-x_j<\delta} V(f; x_0, \ldots, x_n) \tag{2.2.2}$$

3. The function f is said to be of BOUNDED VARIATION if $V(f) < \infty$.

[10] Of course this requires that $0 \in \Omega$.

The concept of variation samples the function at $n+1$ points and tries to detect where it changes most. For that purpose we only need points x_j where two successive summands $f(x_{j+1})-f(x_j)$ and $f(x_j)-f(x_{j-1})$ are of different sign. If both expressions were positive or, equivalently, $f(x_{j-1}) \le f(x_j) \le f(x_{j+1})$, then the sum contains a term of the form

$$\begin{aligned} &\big|f(x_{j+1})-f(x_j)\big| + \big|f(x_j)-f(x_{j-1})\big| \\ &\quad = \quad f(x_{j+1})-f(x_j)+f(x_j)-f(x_{j-1}) = f(x_{j+1})-f(x_{j-1}) = \big|f(x_{j+1})-f(x_{j-1})\big|, \end{aligned}$$

and x_j could have been left out as well. An analogous result holds if the function is decreasing, i.e., $f(x_{j-1}) \le f(x_j) \le f(x_{j+1})$.

To understand the meaning of this property, we can assume that $f \in C^1(I)$ has a continuous derivative and note that $f(x_j)-f(x_{j-1}) = f'(\xi_j)\,(x_j-x_{j-1})$ for some $\xi_j \in [x_{j-1},x_j]$, which yields

$$\begin{aligned} V(f;x_0,\dots,x_n) \quad &= \quad \sum_{j=1}^{n} |f'(\xi_j)\,\underbrace{(x_j-x_{j-1})}_{>0}| = \sum_{j=1}^{n} \big|f'(\xi_j)\big|\,(x_j-x_{j-1}) \\ &\to \quad \int_I |f'(x)|\,dx, \end{aligned}$$

hence

$$V(f) = \|f'\|_1, \tag{2.2.3}$$

at least for C^1 functions. In the general case, we can use the distributional derivative, with the norm defined as the OPERATOR NORM

$$\|T_f\|_1 := \sup_{\phi\in D(K)} \frac{|T_f\phi|}{\|\phi\|_{K,\infty}}, \qquad K = \operatorname{supp}(\phi).$$

Indeed, we have

$$\int_I |f'(x)|\phi(x)\,dx \le \underbrace{\max_{x\in I}|\phi(x)|}_{\le \max_{x\in K}|\phi(x)|} \int_I |f'(x)|\,dx = \|\phi\|_{K,\infty}\,\|f'\|_1$$

and, with $\phi \to 1$,

$$\int_I |f'(x)|\phi(x)\,dx \to \|\phi\|_{K,\infty}\,\|f'\|_1,$$

so that

$$\|f'\|_1 = \|T_f\|_1, \qquad f' \in L_1(I). \tag{2.2.4}$$

Therefore, we can always define the total variation in the sense of generalized functions.

Remark 2.2.2. The variation $V(f) = \|f'\|_1$ is zero if and only if f is constant: if there were two points x, x' such that $f(x) \ne f(x')$, then $V(f;a,x,x',b) \ne 0$. A proof based on distributions would be significantly more complicated. The idea would be to decompose $f = f_1 + f_2$ into a *regular* part $f_1 \in L_1$ which has to be the constant and a *singular* part which may consist of a sum of shifted Dirac distributions. Since

$$\delta'(\phi) = -\phi'(0)$$

by Definition 2.1.6, these components have to vanish as well and so f has to be the constant function.

Definition 2.2.3 (TV norm I)**.** The TV NORM of a function f is defined as

$$\|f\|_{TV} := |f(a)| + V(f) = |f(a)| + \|f'\|_1. \tag{2.2.5}$$

Proposition 2.2.4. *The TV norm is a norm and the space of functions of bounded variation is a Banach space with respect to the TV norm.*

In higher dimensions, the TV norm is similar, but with a slight difference: the first derivative, $\nabla f = Df$, is *vector valued* and, in order to get a functional, we have to apply a norm on $\mathbb{R}^s$ to it.

Definition 2.2.5. By $|\cdot| : \mathbb{R}^s \to \mathbb{R}$ we denote the EUCLIDEAN NORM $|x| = \sqrt{x^T x}$, by $|\cdot|_1$ the 1–norm $\sum |x_j|$.

Definition 2.2.6 (Total variation)**.** The TOTAL VARIATION $V(f)$ of a function $f : \Omega \to \mathbb{R}$, $\Omega \subseteq \mathbb{R}^s$, is defined as

$$V(f) = \left\|\nabla f\right\|_1 := \int_\Omega \left|\nabla f(x)\right| dx, \tag{2.2.6}$$

if necessary in the distributional sense. It is also frequently called the TV NORM[11] and written as

$$\|f\|_{TV} := V(f) = \int_\Omega \left|\nabla f(x)\right| dx. \tag{2.2.7}$$

Remark 2.2.7. The total variation from Definition 2.2.6 is zero if and only if f is a constant function. Since there is no simple definition like (2.2.2), the proof is more complicated and technical, see for example [Yosida, 1965], I.3.Lemma 2.

There is a DUAL version of the Definition (2.2.7) that will be useful in the future and is compatible with the distributional meaning. To formulate it, let us recall the DIVERGENCE $\operatorname{div} f$ of a function $f : \mathbb{R}^s \to \mathbb{R}^s$, defined as

$$\operatorname{div} f := \sum_{j=1}^{s} \frac{\partial f_j}{\partial x_j}. \tag{2.2.8}$$

Proposition 2.2.8. *For any locally integrable $f : \Omega \to \mathbb{R}$ we have that*

$$V(f) = \sup\left\{-T_f(\operatorname{div}\phi) = -\int_\Omega f(x)\left(\operatorname{div}\phi(x)\right) dx : \phi \in D(\Omega)^s, |\phi| \le 1\right\}. \tag{2.2.9}$$

Proof: If f is differentiable, then, by the CAUCHY SCHWARTZ INEQUALITY,

$$\begin{aligned}
-\int_\Omega f(x)\left(\operatorname{div}\phi(x)\right) dx &= -\sum_{j=1}^{s}\int_\Omega f(x)\frac{\partial \phi_j}{\partial x_j}(x)\,dx \\
&= \sum_{j=1}^{s}\int_\Omega \phi_j(x)\frac{\partial f}{\partial x_j}(x)\,dx = \int_\Omega \phi^T(x)\nabla f(x)\,dx \le \int_\Omega \underbrace{|\phi(x)|}_{\le 1}|\nabla f(x)|\,dx \\
&\le \int_\Omega |\nabla f(x)|\,dx
\end{aligned}$$

with the supremum being assumed by the choice $\phi \to \frac{\nabla f}{|\nabla f|}$. The rest is obtained by approximating f by differentiable functions which are DENSE among distributions, cf. [Yosida, 1965]. □

The use of the euclidean norm is no accident, it is very reasonable from a geometric point of view. We will get back to that later in the context of image reconstruction.

[11] Even if, strictly speaking, it is only a seminorm since $V(f) = 0$ for constant functions; in an $L_p(\mathbb{R}^s)$ space it may thus be a norm.

Proposition 2.2.9 (Rotation invariance of total variation)**.** *If $R \in \mathbb{R}^{s\times s}$ is a* ROTATION *matrix and $R\Omega = \Omega$, then*

$$V\big(f(R\cdot)\big) = V(f). \tag{2.2.10}$$

Proof: Since $\big(\nabla f(R\cdot)\big)(x) = R\nabla f(Rx)$, we get

$$V\big(f(R\cdot)\big) = \int_\Omega \Big(\nabla^T f(Rx)\underbrace{R^T R}_{=I}\nabla f(Rx)\Big)^{1/2} dx = \underbrace{\frac{1}{|\det R|}}_{=1}\int_{R\Omega} \big|\nabla f(x)\big|\, dx = V(f),$$

which is (2.2.10). □

Discretizing the gradient as

$$\nabla f \sim \nabla_h f := \frac{1}{h}\Delta_h f = \frac{1}{h}\begin{pmatrix} f(\cdot + he_1) - f \\ \vdots \\ f(\cdot + he_s) - f \end{pmatrix}, \qquad h > 0, \tag{2.2.11}$$

where $\nabla_h \to \nabla f$ whenever f is differentiable[12], we have that

$$(\nabla_h f)_j = \frac{1}{h}\int_0^h (\nabla f)_j(\cdot + te_j)\, dt, \qquad h > 0, \qquad j = 1,\dots,s,$$

and so, by the classical norm inequalities, cf. [Golub and van Loan, 1996],

$$|x|_2 \le |x|_1 \le \sqrt{s}|x|_2, \qquad x \in \mathbb{R}^s,$$

we have that

$$\begin{aligned}\int_{\mathbb{R}^s} \big|\nabla_h f(x)\big|\, dx &\le \int_{\mathbb{R}^s} \big|\nabla_h f(x)\big|_1\, dx = \int_{\mathbb{R}^s} \left|\frac{1}{h}\int_0^h (\nabla f)_j(\cdot + te_j)\, dt\right|_1 dx \\ &\le \frac{1}{h}\int_0^h \int_{\mathbb{R}^s} \big|\nabla f\big|_1 (x + te_j)\, dx\, dt \le \frac{1}{h}\int_0^h \int_{\mathbb{R}^s} \sqrt{s}\big|\nabla f\big|(x)\, dx\, dt = V(f).\end{aligned}$$

If, on the other hand, f is continuous and h sufficiently small, then

$$\frac{1}{h}\int_0^h |(\nabla_h f)_j(x + te_j)|dt \approx |(\nabla_h f)_j(x)|.$$

This means that the naive discretization works reasonably for the total variation. The main argument in favor of total variation comes from the following concept.

Definition 2.2.10. For $y \in \mathbb{R}$ the LEVEL SET $\Lambda_y(f)$ of a function f is defined as

$$\Lambda_y(f) := \big\{x \in \Omega : f(x) > y\big\}. \tag{2.2.12}$$

The associated LEVEL CURVE is

$$\lambda_y(f) := \partial\Lambda_y(f). \tag{2.2.13}$$

Level sets are *binary* images, that is, a pixel belongs to the level set or not. They can be simply generated by Octave:

```
>> A = imread( "Meersaugrau.png" );
>> imagesc( A > 200 );
>> figure(); imagesc( A > 100 );
```

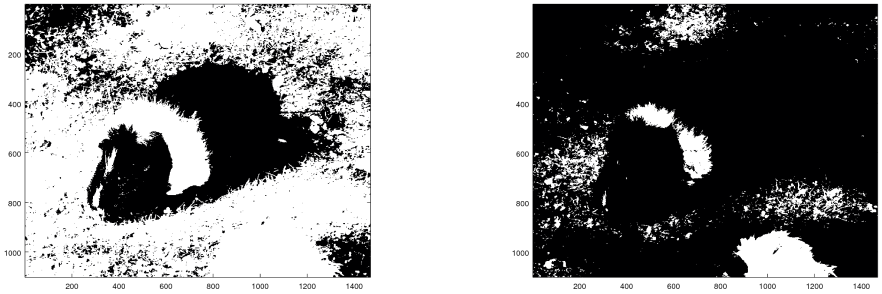

Figure 2.2.1: Two level sets for our guinea pig with the values 100 *(left)* and 200 *(right)*

Level curves are *not* curves of all pixels with a certain value, but the *boundary* of a level set. We can thus compute the level curves by determining all pixels in the boundary, i.e., all pixels that have a black and a white pixel in the neighborhood. A simple Octave routine that determines all pixels whose neighborhood does not consist of a single value is given in Fig. 2.2.2. The images can be seen in Fig. 2.2.3.

```
%% *** Advanced Imaging ***
%% LevelBound.m
%%
function B = LevelBound( F,y )
  B = F > y; % Level set

  [m,n] = size(B);
  %% padding
  BB = [ zeros(1,n+2); zeros(m,1),B,zeros(m,1); zeros(1,n+2) ];
  C = zeros( size( B ) );

  for j=0:2
    for k=0:2
      C = C + BB( j+1:m+j,k+1:n+k );
    end
  end

  B = ( C > 0 ) .* ( C < 9 );
```

Figure 2.2.2: `LevelBound.m`: Matlab code for level curve computation.

The images show that there are two types of level curves: the ones that correspond to contours, others that correspond to *texture*. The contour ones are "normal" curves while those corresponding to texture are of a more fractal nature.

Back to mathematics: if f is continuous[13], then we indeed have

$$\lambda_y(f) = \{x \in \Omega : f(x) = y\}$$

and the curve can be tracked by means of the gradient: the tangent of the level curve is orthogonal to the gradient as already mentioned before. Though it is almost impossible to give

[12] Does that also have to hold for the distributional derivative? Forget it!
[13] Which discrete images are not, but let us assume it nevertheless.

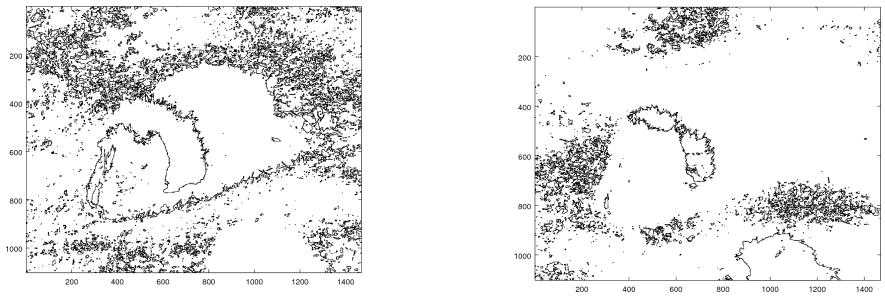

Figure 2.2.3: Level curves for the level sets from Fig. 2.2.1, drawn in black.

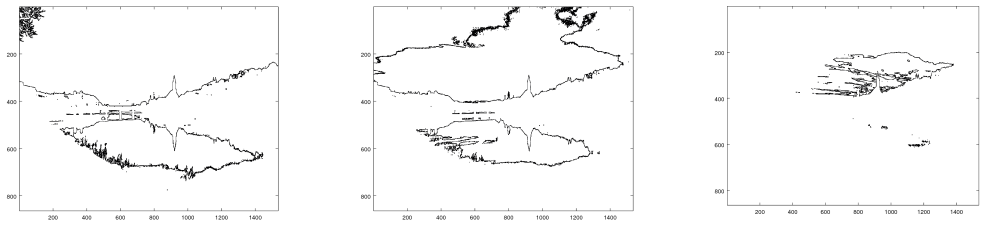

Figure 2.2.4: Level curves for the grayscale Passau image using the quite arbitrary values 50 *(left)*, 100 *middle* and 200 *(right)*.

a PARAMETRIZATION of that curve, we can measure its length in the following way. Given a number $\delta > 0$, we cover the set $\lambda_y(f)$ by balls of diameter δ:

$$\lambda_y(f) \subseteq \bigcup_{x \in X} B_{\rho_x}(x), \qquad \rho_x < \delta, \qquad B_\rho(x) := \{x' : |x - x'| \leq \rho\}.$$

Here $X \subset \Omega$ is a set of centers which can be finite or infinite. Now we set

$$H^d_\delta\left(\lambda_y(f)\right) := 4^d \inf_X \sum_{x \in X} \rho_x^d$$

as the d–dimensional volume of the coverage[14] and

$$H^d\left(\lambda_y(f)\right) := \sup_{\delta > 0} H^d_\delta\left(\lambda_y(f)\right)$$

as the limit obtained when the "fineness" of the covering goes to zero.

Definition 2.2.11. For $X \subset \Omega$, the number $H^d(X)$ is called the d–dimensional HAUSDORFF MEASURE of the set X.

It can be shown that the onedimensional or MONODIMENSIONAL Hausdorff measure of a curve coincides with its ARC LENGTH if a curve is RECTIFYABLE. After that short excursion, we can understand the following result that ties the total variation to level curves: The variation coincides with the *length* of the level curves in the sense of the monodimensional Hausdorff measure.

[14] More precisely, the volume of the bounding cubes.

Theorem 2.2.12 (CO–AREA FORMULA, see [Mallat, 1999], Theorem 2.7)**.** *If $d = 2$ and $V(f) < \infty$ then*

$$V(f) = \int_\Omega |\nabla f(x)|\, dx = \int_{-\infty}^{\infty} H^1\left(\lambda_y(f)\right) dy. \tag{2.2.14}$$

Proof: A complete proof based on distributional derivatives is substantially technical and beyond the scope of this lecture. Here, we repeat an intuitive reasoning from [Mallat, 1999] for the case when f is continuously differentiable. In this case, we can write $\lambda_y(f)$, $y \in \mathbb{R}$, as a continuous curve $\varphi(y,\cdot) : [0, \ell_y] \to \mathbb{R}^2$ which we can assume to be paramtrized according to the arc lenght, i.e., $|\varphi'(y,\cdot)| \equiv 1$ for any y. Since

$$\left(\varphi'(y,t)\right)^T \nabla f\left(\varphi(y,t)\right) = 0,$$

the gradient corresponds to the direction

$$\nu(y,t) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \varphi'(y,t),$$

normal to the tangent and satisfies

$$\left|\nabla f\left(\varphi(y,t)\right)\right| = \nabla f\left(\varphi(y,t)\right)^T \nu(y,t) = \frac{dy}{d\nu}$$

or[15]

$$\left|\nabla f\left(\varphi(y,t)\right)\right| d\nu = dy.$$

Every point x can be uniquely written as $x = \varphi(y,t)$ and we can use the local coordinate system generated by $\varphi'(y,t)$ and $\nu(y,t)$, whose functional determinant is

$$\det \begin{pmatrix} \varphi_1'(y,t) & \varphi_2'(y,t) \\ -\varphi_2'(y,t) & \varphi_1'(y,t) \end{pmatrix} = |\varphi'(y,t)|^2 = 1$$

since φ is arc length parametrized. Hence, with proper extension of the curve,

$$\begin{aligned} \int_\Omega |\nabla f(x)|\, dx &= \int_{\mathbb{R}} \int_{\mathbb{R}} \underbrace{\left|\nabla f\left(\varphi(y,t)\right)\right| d\nu}_{=dy}\, dt = \int_{-\infty}^{\infty} \int_0^{\ell_y} \left|\varphi'(y,t)\right| dt\, dy \\ &= \int_{-\infty}^{\infty} H^1\left(\lambda_y(f)\right) dy, \end{aligned}$$

since for rectifiable curves the Hausdorff measure and the arc length coincide. □

To implement the computation of the total variation or TV is very simple when using the gradient discretization (2.2.11) and replacing the integral by a RIEMANN SUM:

$$V(F) = \sum_{j,k} \left|\nabla f_{jk}\right|. \tag{2.2.15}$$

Applying the TV to 8×8 blocks of the image, we already get a method to estimate the "fractality" of image regions and to distinguish between conturs and regions with rich texture, see Fig. 2.2.5.

[15] To write this mathematically correct needs substantial effort.

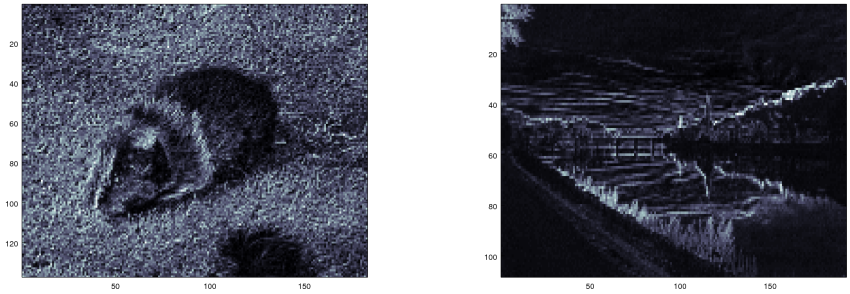

Figure 2.2.5: 8×8 Block TV of the guinea pig *(left)* and the Passau *(right)* image.

```
%% *** Advanced Imaging ***
%% ImageGradient.m
%%
function B = ImageGradient( F )
  [m,n] = size(F);

  %% x component
  B(:,:,1) = ([ zeros(m,1), F ] - [ F,zeros(m,1) ]) (1:m,1:n);
  %% y component
  B(:,:,2) = ([ zeros(1,n); F ] - [ F; zeros(1,n) ]) (1:m,1:n);
```

Figure 2.2.6: `ImageGradient.m`: Gradient of an image with simple Matlab tricks.

2.3 Denoising and the ROF functional

In many realistic situation, for example in image acquisition with digital cameras, the recorded image is not the "real image" but is affected with noise. Instead of f we know $\hat{f} = f + v$ or, in case of sampled images the values[16]

$$\hat{f}_\alpha = \hat{f}_{jk} = f(x_\alpha) + \eta_\alpha, \qquad \alpha \in \Gamma,$$

where α indexes the pixel or voxel in the image and Γ is the SAMPLING SET, usually an equidistributed grid. With our 2D–images it will be used together with jk and with the same meaning.

The simplest way to denoise images is by convolution. It makes the fundamental assump-

[16] Using notation from roundoff error analysis where the "^" usually indicates the perturbed quantity.

```
%% *** Advanced Imaging ***
%% TotalVariation.m
%%
function tv = TotalVariation( F )
  Ig = ImageGradient( F );
  tv = sum( sum( sqrt( Ig(:,:,1).^2 + Ig(:,:,2).^2 ) ) );
```

Figure 2.2.7: `TotalVariation.m`: Computation of the discrete total variation of an image.

tion that the noise has mean value zero, i.e,

$$\int_\Omega \eta(x)\,dx = 0.$$

Now one chooses a function ϕ with $0 \in \operatorname{supp}(\phi) =: \Omega_\phi$ and

$$|\phi| := \int \phi(x)\,dx \neq 0,$$

and applies the CONVOLUTION, defined as[17]

$$g * h := \int g(\cdot - x)\,h(x), \qquad g, h : \mathbb{R}^s \to \mathbb{R},$$

to consider

$$\hat{f} * \frac{\phi}{|\phi|}(x) := \frac{1}{|\phi|} \int\limits_{t\in\Omega_\phi} f(x-t)\,\phi(t)\,dt + \frac{1}{|\phi|} \int\limits_{t\in\Omega_\phi} \eta(x-t)\,\phi(t)\,dt.$$

Now we are in a slight dilemma: if we make Ω_ϕ very small but still located around the origin, then the first integral approaches $f(x)$, see [Akhieser, 1988], if, on the other hand, we make Ω_ϕ large and $\phi \approx 1$, then it will cancel the noise. Most of the classical mean value filters for denoising then try to find some compromise and end up with a filter that decimates the noise but, unfortunately, blurs the image which means that texture and, even worse, edges get lost to some extent.

To overcome that problem, we return to regularization and combine a data fidelity functional with the TV norm.

Definition 2.3.1. The RUDIN–OSHER–FATEMI FUNCTIONAL, better known as ROF FUNCTIONAL is defined for functions f, g as

$$\|f-g\|_2^2 + \lambda \|g\|_{TV}, \qquad \lambda > 0. \tag{2.3.1}$$

Remark 2.3.2. In the original paper [Rudin et al., 1992], the λ is a factor of the data fidelity functional. Here we use the formulation that is more common nowadays.

In other words: given any image f, the functional measures the distance between f and g and, at the same time, incorporates the regularity of g with respect to the TV norm.

Remark 2.3.3. In APPROXIMATION THEORY this concept is known as K–functionals and interpolation between spaces, cf. [Berens, 1968, Butzer and Berens, 1967]. The application in imaging, however, had to treat the more important point of how to solve optimization problems resulting from the ROF functional.

The task in DENOISING is now very simple: "just" solve the minimization problem

$$\min_g J(g) := \min_g \|f-g\|_2^2 + \lambda \|g\|_{TV}. \tag{2.3.2}$$

Before we have a look at the theory behind this problem, let us approach it naively and consider the discrete version

$$\min_{\boldsymbol{g}} J(\boldsymbol{g}) \qquad \text{i.e.,} \qquad \min_{g_{jk}} \frac{1}{2} \sum_{jk} \left(f_{jk} - g_{jk}\right)^2 + \lambda \sum_{jk} \left|\nabla g_{jk}\right|. \tag{2.3.3}$$

[17] For the convolution it is reasonable to have no boundary; this is usually achieved by periodizing or padding an image of finite size.

If we try to put this into a black box optimizer, we are not successful

```
>> F = imread ( "Meersaugrau.png" );lambda = .1;
>> f = @(G) sum(sum( (F-G).^2 )) + lambda*TotalVariation( G );
>> Gm = fminsearch( f,F );
error: out of memory or dimension too large for Octave's index type
```

So what about our idea of taking derivatives and setting them equal to zero? The problem is the second term which is for images of the form

$$\sum_{j=1}^{m-1}\sum_{k=1}^{n-1}\left|\nabla g_{jk}\right| = \sum_{jk}\sqrt{\left(g_{j+1,k}-g_{jk}\right)^2+\left(g_{j,k+1}-g_{jk}\right)^2}.$$

We embed it into infinite images by setting $g_{jk}=0$ if $j\le 0$ or $j>m$, and $k\le 0$ or $k>n$, respectively. The partial derivative with respect to g_{rs}, $r,s=1,\dots,m,n$ of the biinfinite sum

$$\sum_{j,k\in\mathbb{Z}}\left|\nabla g_{jk}\right| = \sum_{jk}\sqrt{\left(g_{j+1,k}-g_{jk}\right)^2+\left(g_{j,k+1}-g_{jk}\right)^2}$$

is

$$\frac{1}{2}\frac{\partial}{\partial g_{rs}}\sum_{j,k\in\mathbb{Z}}\left|\nabla g_{jk}\right| = \frac{g_{rs}-g_{r-1,s}}{|\nabla g_{r-1,s}|}+\frac{g_{rs}-g_{r,s-1}}{|\nabla g_{r,s-1}|}-\frac{g_{r+1,s}-2g_{rs}+g_{r,s+1}}{|\nabla g_{rs}|}, \tag{2.3.4}$$

which is nonzero for $r,s=1,\dots,m,n$. In the case $r=1$, the first term becomes

$$\frac{g_{1s}}{|\nabla g_{0s}|} = \frac{g_{1s}}{\sqrt{(g_{1s}-g_{0s})^2+(g_{0,s+1}-g_{0s})^2}} = \frac{g_{1s}}{|g_{1s}|},$$

and analogously for $s=1$. The function in (2.3.4) is no more linear in $\boldsymbol{g}$. Moreover, we may divide by zero if the approximating image g_{jk} is locally constant. To overcome this, the usual way is to modify $|\nabla g_{jk}|$ into

$$\left|\nabla_\varepsilon g_{jk}\right| := \sqrt{\varepsilon+\left(g_{j+1,k}-g_{jk}\right)^2+\left(g_{j,k+1}-g_{jk}\right)^2}, \qquad \varepsilon>0,$$

which explicitly avoids division by zero and gives the same results when $\varepsilon\to 0$. Moreover,

$$\frac{\partial}{\partial g_{rs}}\sum_{j,k\in\mathbb{Z}}\left|\nabla_\varepsilon g_{jk}\right| = \frac{g_{rs}-g_{r-1,s}}{|\nabla_\varepsilon g_{r-1,s}|}+\frac{g_{rs}+g_{r,s-1}}{|\nabla_\varepsilon g_{r,s-1}|}-\frac{g_{r+1,s}-2g_{rs}+g_{r,s+1}}{|\nabla_\varepsilon g_{rs}|}. \tag{2.3.5}$$

Since

$$\frac{\partial}{\partial g_{rs}}\sum_{jk}\left(f_{jk}-g_{jk}\right)^2 = 2\left(g_{rs}-f_{rs}\right),$$

the GRADIENT of the discrete ROF functional takes the form

$$\left((g_{jk}-f_{jk})+\lambda\left(\frac{g_{jk}-g_{j-1,k}}{|\nabla_\varepsilon g_{j-1,k}|}+\frac{g_{jk}-g_{j,k-1}}{|\nabla_\varepsilon g_{j,k-1}|}-\frac{g_{j+1,k}-2g_{jk}+g_{j,k+1}}{|\nabla_\varepsilon g_{jk}|}\right):j,k\right). \tag{2.3.6}$$

Solving this *nonlinear* equation is not easy but the fact that we can explicitly compute the gradient of the ROF functional suggests to minimize the *function* $J(\boldsymbol{g})$ by means of GRADIENT DESCENT. The idea is the observation that $-\nabla J(\boldsymbol{g})$ is the direction of STEEPEST DESCENT of the function J at the point $\boldsymbol{g}$ and that, if $\nabla J(\boldsymbol{g})\neq 0$, there is some $t>0$ sucht that

$$J\left(\boldsymbol{g}+t\nabla J(\boldsymbol{g})\right) < J(\boldsymbol{g}).$$

```
%% *** Advanced Imaging ***
%% ROFEvaluate.m
%%
function [t,D] = NaiveGradientStepsize( t,F,G,lambda,epslon )
  D = (-1) * ROFGradient( F,G,lambda,epslon );
  inc = 1.2; dec = .7;

  f2 = f0 = ROFEvaluate( F,G,lambda );
  f1 = ROFEvaluate( F,G+t*D,lambda );

  if ( f1 < f0 )
    %% Can we do better? Enlarge t
    f2 = ROFEvaluate( F,G+inc*t*D,lambda );
    while( f2 < f1 )
      t = t*inc;
      f1 = f2;
      f2 = ROFEvaluate( F,G+inc*t*D,lambda );
    end
  else
    %% Decrease t until we are better than preceding value
    do
      t = t*dec;
      f2 = ROFEvaluate( F,G+t*D,lambda );
    until ( ( f2 < f0 ) || ( t < 100*eps ) )
  end
```

Figure 2.3.8: `NaiveGradientStepsize.m`: Naive stepsize computation for gradient descent.

Determining t is not easy, but there exist rules to do so, see, for example, [Nocedal and Wright, 1999, Sauer, 2013b, Spellucci, 1993]. We use a very naive method here, shown in Fig. 2.3.8.

We can now experiment with a simple gradient descent method. The arguments are the reference image f, an initial value for g, values for λ and ε and an upper bound for the number of iterations. Hence, the call is like

```
>> F = (double)(imread("Meersaugrau.png") );
>> G = NaiveGradientDescent( F,F,1,10^(-8),30);
```

We can test our algorithm on the guinea pig example with the values $\lambda = 1, 10, 1000$ and at most 50 iterations. Here, we use the original image as reference image and just try to smoothen it by enforcing different levels of TV regularity. In other words, we consider the grainy texture in the image as noise. The results of the regularized image where we "denoise" the texture are shown in Fig. 2.3.9. The (discrete) TV norm of the image is *1.4943e+07*. The first two images there show little difference to the original image, the third one has much less texture but still nicely recovers the contours.

It should however be mentioned that the algorithm we used here is only heuristics and quite poor ones in addition. It converges, if at all, very slowly and we cannot guarantee that it reaches a minimum. So the art consists not so much in setting up the ROF functional but finding the minimizer in decent time and good accuracy.

Example 2.3.4. At the moment, we do not really know whether and how fast our little algorithm converges. This can be seen in Fig 2.3.10 where the procedure is once stopped after 100

Figure 2.3.9: TV minimized versions of the guinea pig with $\lambda = 1, 10, 1000$. Total variations are `1.4165e+07` *(left)*, `1.1676e+07` *(middle)* and `3.8807e+06` *(right)*.

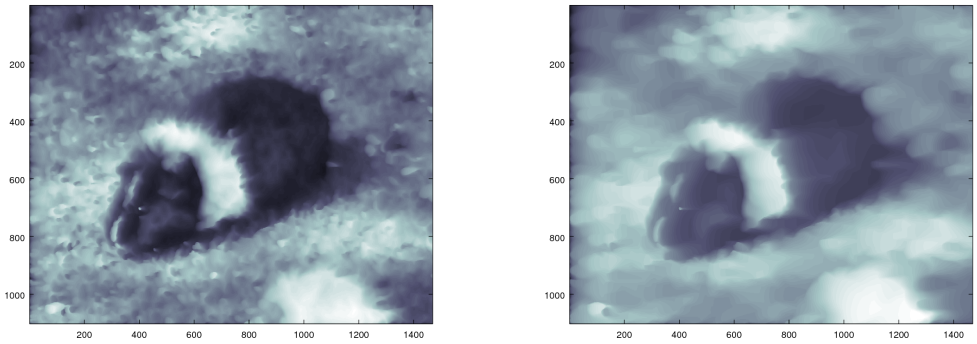

Figure 2.3.10: Highly regularized ($\lambda = 10000$) version of the image, total variation is `2.7951e+06` after 100 iterations *(left)* and `1.6061e+06` after 1000 iterations *(right)*. We do not know what the minimum is, but the contours are preserved quite well.

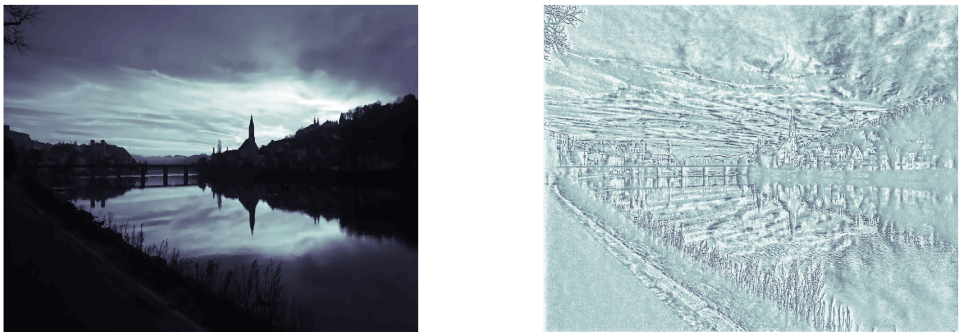

Figure 2.3.11: Regularization of the Passau image with $\lambda = 100$ *(left)* and absolute value of difference from the original image *(right)* where black corresponds to large values.

Figure 2.3.12: Original image *(left)* and perturbed by 10% noise with Octave's `rand` function.

iterations and once after 1000 iterations. This still gives *no* indication what the real optimum is as gradient descent is known to converge very slowly.

At the end of this section, let us briefly get a first idea how this method works for its original purpose, namely denoising. We use the image from Fig. 2.3.12, normalize it to $[0,1]$ and then attach 10% random noise to it before we run TV regularization on it:

```
>> F = double( imread( "giardiniereGray.png" ) );
>> F = ( F - min(min(F)) ) / ( max(max(F)) - min(min(F)) );
>> F = F + .1*( 1 - 2*rand(size(F)));
```

After that, we apply our naive gradient descent

```
>> G = NaiveGradientDescent( F,F,1,10^(-8),480);
```

The result can be seen in Fig. 2.3.13. Obviously denoising works quite well and the preservation of edges is remarkable.

In order to better understand how such optimization problems can be solved in an efficient and reliable way, we once more need additional background. Especially we will learn how the minimization problem was treated in the original paper [Rudin et al., 1992].

Figure 2.3.13: TV regularization with weight $\lambda = 1$ *(left)* and $\lambda = 10$ *(right)*. The oversmoothing is visible but the letters and symbols are still perfectly recognizable.

2.4 The basic idea of variational calculus

One way to solve the continuous minimization problem is by VARIATIONAL CALCULUS of which we will recall the basic concepts here. For more information see [Gelfand and Fomin, 1963]. Variational calculus, a classical topic in Analysis, considers the minimization of functionals of the form

$$J[f] = \int_a^b F\left(x, f, f'\right) dx \tag{2.4.1}$$

with respect to[18] $f : \mathbb{R} \to \mathbb{R}$, where $F\left(x, y, y'\right) : \mathbb{R}^3 \to \mathbb{R}$ is a function in the three *formal* variables x, y, y'.

The fundamental underlying concept is the DIFFERENTIAL of a functional, defined as follows: we fix f and consider, for functions h, the difference

$$\Delta J[h] = \Delta J[h, f] = J\left[f + h\right] - J[f].$$

We call J DIFFERENTIABLE at f if there exists a linear functional $\delta J[h]$ such that

$$\Delta J[h] = \delta J[h] + o\left(\|h\|\right) = \delta J[h] + \varepsilon_h \|h\|, \qquad \lim_{\|h\| \to 0} \varepsilon_h = 0.$$

If such a $\delta J[h]$ exists, it has to be unique. Assuming there were two differentials $\delta_1 J[h]$ and $\delta_2 J[h]$, we would obtain

$$\delta_1 J[h] - \delta_2 J[h] = \Delta J[h] - \varepsilon_{1,h} \|h\| - \Delta J[h] + \varepsilon_{2,h} \|h\| = \left(\varepsilon_{2,h} - \varepsilon_{1,h}\right) \|h\|,$$

hence

$$\lim_{\|h\| \to 0} \frac{\delta_1 J[h] - \delta_2 J[h]}{\|h\|} = \lim_{\|h\| \to 0} \frac{\left(\varepsilon_{2,h} - \varepsilon_{1,h}\right) \|h\|}{\|h\|} = 0,$$

which means that the difference is the zero functional.

Remark 2.4.1. The functional $J[f]$ is usually not differentiable but "only" convex, i.e.

$$J\left[\alpha f + (1-\alpha) g\right] \leq \alpha J\left[f\right] + (1-\alpha) J\left[g\right], \qquad \alpha \in [0,1].$$

In this case one will not consider the differential but the subdifferential, cf. [Rockafellar, 1970]; we will get to this later.

[18] We restrict ourselves to univariate functions in this exposition, but the main part of the concept is independent of the number of variables as we will see soon.

The following is the well–known "zero of derivative" condition.

Proposition 2.4.2. *If the functional $J[f]$ is minimized by f^*, then $\delta J\left[h, f^*\right] = 0$ for all h.*

Proof: If f^* is a local minimum, we get for sufficiently small $\|h\|$ that

$$0 \le J\left[f^* + h\right] - J\left[f^*\right] = \delta J[h] + \varepsilon_h \|h\|,$$

implying that $\delta J[h] \ge 0$. Now,

$$0 \le \delta J[h] = -\delta J[-h] \le 0$$

is only possible with $\delta J[h] = 0$. □

Next, we use a Taylor expansion of F to obtain

$$\begin{aligned} \Delta J[h] &= J\left[f+h\right] - J[f] = \int_a^b F\left(x, f+h, f'+h'\right) - F\left(x, f, f'\right) dx \\ &= \int_a^b h \frac{\partial F}{\partial y}\left(x, f, f'\right) + h' \frac{\partial F}{\partial y'}\left(x, f, f'\right) dx + \cdots, \end{aligned}$$

which gives

$$\delta J[h] = \int_a^b h \frac{\partial F}{\partial y}\left(x, f, f'\right) + h' \frac{\partial F}{\partial y'}\left(x, f, f'\right) dx. \tag{2.4.2}$$

Theorem 2.4.3. *If f minimizes the functional J from* (2.4.1)*, then*

$$\frac{\partial F}{\partial y}\left(x, f, f'\right) - \frac{\partial}{\partial x}\frac{\partial F}{\partial y'}\left(x, f, f'\right) = 0. \tag{2.4.3}$$

Definition 2.4.4. The differential equation (2.4.3) is called EULER–LAGRANGE EQUATION of the variational problem.

The proof of Theorem 2.4.3 relies on the following Lemma which we are not going to prove here.

Lemma 2.4.5. *If $f_0, \ldots, f_n \in C^n([a,b])$ satisfy*

$$\int_a^b \sum_{j=0}^{n} f_j(x)\, h^{(j)}(x)\, dx = 0 \tag{2.4.4}$$

for all $h \in C^n[a,b]$ such that $h^{(j)}(a) = h^{(j)}(b) = 0$, $j = 0, \ldots, n$, then

$$\sum_{j=0}^{n} (-1)^j f_j^{(j)}(x) = 0, \qquad x \in [a,b]. \tag{2.4.5}$$

Exercise 2.4.1 Prove Lemma 2.4.5. ♢

Proof of Theorem 2.4.3: Apply Lemma 2.4.5 to the identity

$$0 = \delta J[h] = \int_a^b h \frac{\partial F}{\partial y}\left(x, f, f'\right) + h' \frac{\partial F}{\partial y'}\left(x, f, f'\right) dx.$$

□

There is, of course, a multivariate version of the Euler–Lagrange equation, working with functionals of the form

$$J[f] := \int_\Omega F(x, f, \nabla f)\, dx = \int_\Omega F(x, f, f_1, \dots, f_s)\, dx, \qquad f_j := \frac{\partial f}{\partial x_j}, \tag{2.4.6}$$

for some $\Omega \subset \mathbb{R}^s$, see [Gelfand and Fomin, 1963, p. 152ff]. Though the proof is "essentially" the same, it is technically more involved, for example, the partial integration needed in the proof of Lemma 2.4.5 has to be replaced by GREEN'S THEOREM about integration along the boundary. Still, $\partial J[h] = 0$ implies the differential equation

$$0 = \frac{\partial F}{\partial f} - \sum_{j=1}^{s} \frac{\partial}{\partial x_j} \frac{\partial F}{\partial f_j} = \frac{\partial F}{\partial f} - \operatorname{div}_x \nabla_{f'} F. \tag{2.4.7}$$

Let us apply (2.4.7) to the functional

$$J[g] = \|f - g\|_2^2 + \lambda \|g\|_{TV} = \int_\Omega \big(f(x) - g(x)\big)^2 + \lambda \left|\nabla g(x)\right| dx,$$

i.e.

$$F(x, g, g_1, g_2) = \big(f(x) - g(x)\big)^2 + \lambda \sqrt{g_1^2(x) + g_2^2(x)}.$$

Thus,

$$\frac{\partial F}{\partial f} = -2\big(f(x) - g(x)\big),$$

and

$$\frac{\partial F}{\partial f_j} = \lambda \frac{g_j(x)}{\sqrt{g_1^2(x) + g_2^2(x)}},$$

leading to the nonlinear SECOND ORDER[19] partial differential equation

$$0 = 2\big(f(x) - g(x)\big) + \lambda \left(\frac{\partial}{\partial x_1} \frac{g_1(x)}{\sqrt{g_1^2(x) + g_2^2(x)}} + \frac{\partial}{\partial x_2} \frac{g_2(x)}{\sqrt{g_1^2(x) + g_2^2(x)}} \right), \tag{2.4.8}$$

with the BOUNDARY CONDITIONS

$$\frac{\partial g}{\partial n}(x) = 0, \qquad x \in \partial\Omega, \tag{2.4.9}$$

where n denotes the outward pointing normal of the boundary curve. A shorthand for (2.4.8) is[20]

$$0 = (f - g) + \lambda \operatorname{div} \frac{\nabla g}{|\nabla g|} = (f - g) + \lambda \nabla \cdot \frac{\nabla g}{|\nabla g|}. \tag{2.4.10}$$

which also generalizes to higher dimensions.

Remark 2.4.6. There is a physical background to many of these equations, essentially from classical mechanics. This is the reasons why "strange" operators like divergence were introduced: they are very useful for physicists.

[19] Recall that $g_j = \frac{\partial g}{\partial x_j}$.

[20] The factor 2 in the data fidelity term can be compensated by replacing λ by 2λ and then cancelling the factor 2

Unfortunately, such partial differential equations are not easy to solve, fortunately, there exists a lot of software for tackling such problems.

Remark 2.4.7. For the quite similar functional

$$J[f] = \int_\Omega \sqrt{1 + |\nabla f(x)|^2}\, dx$$

measuring the SURFACE ENERGY[21], the solution of the associated Euler–Lagrange equations

$$\operatorname{div} \frac{\nabla f}{\sqrt{1 + |\nabla f|^2}} = 0$$

is the respective MINIMAL SURFACE, depending only on the boundary condition, like a soap bubble.

The differential equation (2.4.10) fits into a larger and well–studied class of partial differential equations, related to second order flow problems.

Definition 2.4.8. For functions $u : \mathbb{R} \times \mathbb{R}^s \to \mathbb{R}$, $(t, x) \mapsto u(t, x)$, a differential equation of the form

$$\frac{\partial u}{\partial t} - \nabla_x \cdot (h \nabla_x u) + au = f, \tag{2.4.11}$$

with functions $a, f, h : \mathbb{R}^s \to \mathbb{R}$ is called PARABOLIC.

Example 2.4.9. The simplest example of a parabolic equation is the HEAT EQUATION

$$\frac{\partial u}{\partial t} - \nabla_x \cdot \nabla_x u = f; \tag{2.4.12}$$

taking into account that

$$\nabla \cdot \nabla = \operatorname{div} \nabla = \sum_{j=1}^{s} \frac{\partial^2}{\partial x_j^2} = \Delta,$$

this process is based on the LAPLACE OPERATOR. We will get back to that later.

One then considers the *time dependent* functions $u(t, \cdot)$, in our case a sequence of images, by solving an ORDINARY DIFFERENTIAL EQUATION and to watch how the images develop or evolve over time. The key observation is as follows.

Observation 2.4.10. *The image $u(t, \cdot)$ is a solution of the Euler–Lagrange equation* (2.4.10) *if and only if it is a* STATIONARY POINT *of the process*

$$\frac{\partial u}{\partial t} = (f - u) + \lambda \nabla \cdot \frac{\nabla u}{|\nabla u|}, \tag{2.4.13}$$

i.e., if $u_t := \frac{\partial u}{\partial t} = 0$.

Based on this observation, the simplest way to solve the Euler–Lagrange equation is to perform an EULER SCHEME, see for example [Isaacson and Keller, 1966, Stoer and Bulirsch, 1978], on the ordinary differential equation (2.4.13) and set

$$u(t + h, \cdot) = u(t, \cdot) + h \left(\big(f - u(t, \cdot)\big) + \lambda \nabla \cdot \frac{\nabla u(t, \cdot)}{|\nabla u(t, \cdot)|} \right) \tag{2.4.14}$$

[21] It could also be considered a higher dimensional arc length of the surface graph.

with a proper STEPSIZE h and an appropriate discretization of the derivatives. This process is iterated until it becomes stationary, i.e. does not change much any more. Of course, this can be improved by chosing a better ODE solver like Runge–Kutta methods and details like stopping criteria and numerical issues matter as well.

Exercise 2.4.2 Implement the iteration (2.4.14) in Matlab or Octave. ♢

This is how the denoising was performed in the original paper [Rudin et al., 1992], namely by computing the time dependent process $u(t,\cdot)$ by means of discretizing (2.4.13) and computing the image at the next time step until the process became stationary. However, since they were working on an *explicit* noise model, there was also a re–estimation of $\lambda = \lambda(t)$ according to the computed image $u(t,\cdot)$.

Optimization | 3

It was crazy. I liked it.

(R. Chandler, *The Little Sister*)

We will stick with the ROF functional from (2.3.1), but now will use methods from optimization to find the optimal balance between data fidelity and low variation.

3.1 Convex optimization and subgradients

Convex functions play a fundamental role in optimization and it is no surprise that Optimization and Convex Analysis have a lot in common, cf. [Rockafellar, 1970, Stoer and Witzgall, 1970]. In what follows, V will stand for a VECTOR SPACE and $\mathscr{H}$ for a HILBERT SPACE, i.e., a vector space with a NORM that is induced by a SCALAR PRODUCT $\langle\cdot,\cdot\rangle$.

Definition 3.1.1 (Convexity)**.** A function $f : V \to \mathbb{R}$, V some VECTOR SPACE, is called CONVEX if

$$f\big((1-\alpha)x+\alpha x'\big) \le (1-\alpha)\,f(x)+\alpha f(x'), \qquad \alpha\in[0,1], \tag{3.1.1}$$

holds for any $x, x' \in V$. A subset X of V is called CONVEX[1] if

$$x, x' \in X \qquad \Rightarrow \qquad (1-\alpha)x+\alpha x' \in X, \quad \alpha\in[0,1]. \tag{3.1.2}$$

The BOUNDARY ∂X of a convex set X consists of all points $y \in X$ that cannot be written as a convex combination $y = (1-\alpha)x+\alpha x'$, $\alpha\in[0,1]$, $x, x' \in X\setminus\{y\}$.

Convex functions form a POSITIVE CONE, i.e., if f, g are convex, then so is any function of the form $\alpha f+\beta g$, $\alpha,\beta\ge 0$. We already know two important types of convex functions on vector and Hilbert spaces.

Proposition 3.1.2 (Norms are convex)**.**

1. *The function $\|\cdot\|$ is convex on any vector space.*
2. *The function $\|\cdot\|^2$ is convex on any Hilbert space.*

Proof: For 1) we simply use the norm axioms triangle inequality and positive homogeneity:

$$\|(1-\alpha)x+\alpha x'\| \le \|(1-\alpha)x\| + \|\alpha x'\| = (1-\alpha)\|x\| + \alpha\|x'\|.$$

[1] The same word for two different concepts which are, however, closely related.

Statement 2) follows from a more general fact: if f is nondecreasing and convex and g is convex, then, since f is nondecreasing and convex, we have that

$$\begin{aligned}\big(f\circ g\big)\big((1-\alpha)x+\alpha x'\big) &= f(\underbrace{g\big((1-\alpha)x+\alpha x'\big)}_{\leq (1-\alpha)g(x)+\alpha g(x')}\\ &\leq f\big((1-\alpha)g(x)+\alpha g(x')\big)\leq (1-\alpha)f\big(g(x)\big)+\alpha f\big(g(x)\big).\end{aligned}$$

Since $\|\cdot\|$ is convex and nonnegative and $(\cdot)^2$ is convex and nonincreasing on $\mathbb{R}_+$, the claim follows. □

Exercise 3.1.1 Show that for general convex functions f,g the function $f\circ g$ need not be convex. *Hint:* $f=g=e^{-x}$ does the job. ♢

Corollary 3.1.3. *The ROF functional* (2.3.1) *is convex for any* $\lambda\geq 0$.

A useful description of convex functions is the following one.

Theorem 3.1.4. *A continuously differentiable function* $f:\mathscr{H}\to\mathbb{R}$ *is convex if and only if*

$$\big\langle \nabla f(x)-\nabla f(x'),x-x'\big\rangle \geq 0,\qquad x,x'\in\mathscr{H}. \tag{3.1.3}$$

Remark 3.1.5. Note that the gradient of $f:\mathscr{H}\to\mathbb{R}$ is a function $\nabla f:\mathscr{H}\to\mathscr{H}$ since the first derivative of a function is a linear function on the domain, hence an element from the DUAL SPACE. For a Hilbert space, this space of all continuous[2] linear functionals, is the same Hilbert space again; this observation is the RIESZ REPRESENTATION THEOREM for REFLEXIVE SPACES.

Proof of Theorem 3.1.4: We just show that convexity implies (3.1.3), for the converse see [Rockafellar, 1970]. Since, for $0<\beta_1\leq\beta_2$ and $x,y\in\mathscr{H}$ convexity implies

$$x+\beta_1 y=\frac{\beta_2-\beta_1}{\beta_2}\,x+\frac{\beta_1}{\beta_2}\,(x+\beta_2 y)\quad\Rightarrow\quad f(x+\beta_1 y)\leq\frac{\beta_2-\beta_1}{\beta_2}\,f(x)+\frac{\beta_1}{\beta_2}\,f(x+\beta_2 y)$$

and thus the MONOTONICITY PROPERTY

$$\frac{f(x+\beta_1 y)-f(x)}{\beta_1}\leq\frac{f(x+\beta_2 y)-f(x)}{\beta_2},\qquad 0<\beta_1\leq\beta_2, \tag{3.1.4}$$

and

$$D_y f(x)=\big\langle\nabla f(x),y\big\rangle=\inf_{\beta\geq 0}\frac{f\big(x+\beta y\big)-f(x)}{\beta} \tag{3.1.5}$$

as well as[3]

$$D_y f(x)=\big\langle\nabla f(x),y\big\rangle=\sup_{\beta\geq 0}\frac{f(x)-f\big(x-\beta y\big)}{\beta} \tag{3.1.6}$$

Thus, by applying (3.1.6) and (3.1.5), we get

$$\begin{aligned}&\big\langle\nabla f(x)-\nabla f(x'),x-x'\big\rangle=\big\langle\nabla f(x),x-x'\big\rangle-\big\langle\nabla f(x'),x-x'\big\rangle\\ &\quad=\sup_{\beta\geq 0}\frac{f(x)-f\big(x-\beta(x-x')\big)}{\beta}-\inf_{\beta\geq 0}\frac{f\big(x'+\beta(x-x')\big)-f(x')}{\beta}\\ &\quad\geq\frac{f(x)-f(x')}{1}-\frac{f(x)-f(x')}{1}=0,\end{aligned}$$

[2] Or bounded, which is the same.

[3] Replace y by $-y$ in (3.1.5).

which is (3.1.3). □

Interesting local extrema of convex functions have to be minima. The proof is simple and gets us acquainted to the concept.

Lemma 3.1.6. *Let $f : V \to \mathbb{R}$ be a convex function and $X \subseteq V$ a convex subset[4] of V.*

1. *If $x^* \in X$ a strict local maximum of f, then $x^* \in \partial X$.*
2. *Any local minimum of f on X is also a global minimum on X*

Proof: 1): Suppose that $x^* \notin \partial X$, then there are $x, x' \in X$ and $\alpha \in (0,1)$ such that $x^* = (1-\alpha)x + \alpha x'$. By moving x, x' closer to x^* along the line through that points, we can ensure that $f(x) < f(x^*)$ and $f(x') < f(x^*)$ since x^* is a strict local maximum. It then follows from convexity that

$$\begin{aligned} f(x^*) &= f\left((1-\alpha)x + \alpha x'\right) \leq (1-\alpha) \underbrace{f(x)}_{<f(x^*)} + \alpha \underbrace{f(x')}_{<f(x^*)} \\ &< (1-\alpha) f(x^*) + \alpha f(x^*) = f(x^*) \end{aligned}$$

which is a contradiction.

For 2) assume that $x \in X$ is a local minimum and $x' \in X$ the global minimum with $f(x') < f(x)$. Since X is convex, all the points

$$x_t := (1-t)x + tx', \qquad t \in [0,1], \qquad x_0 = x,\, x_1 = x',$$

are contained in X and we have that

$$f(x_t) = f\left((1-t)x + tx'\right) \leq (1-t) f(x) + t \underbrace{f(x')}_{<f(x)} < ((1-t)+t)\, f(x) = f(x)$$

holds for any $t \in [0,1]$ and contradicts the assumption that x is a local minimum if we let $t \to 0$. □

For convex functions on Hilbert spaces there exists a slightly more involved but very powerful concept, see [Rockafellar, 1970].

Definition 3.1.7 (Subgradient)**.** Let $f : \mathscr{H} \to \mathbb{R}$. An element $v \in \mathscr{H}$ is called a SUBGRADIENT of a convex function $f : \mathscr{H} \to \mathbb{R}$ at $x \in \mathscr{H}$ if

$$f(x') \geq f(x) + \langle v, x' - x \rangle, \qquad x' \in \mathscr{H}. \tag{3.1.7}$$

The set $\partial f(x)$ of all subgradients at x is called the SUBDIFFERENTIAL and f is said to be SUBDIFFERENTIABLE at x if $\partial f(x) \neq \emptyset$.

Remark 3.1.8. If f is differentiable at x then[5] $\partial f(x) = \{\nabla f(x)\}$, the subdifferential is a true generalization of the derivative.

[4] Since vector spaces are trivially convex, $X = V$ is well included, even if statement 1) becomes meaningless then since $\partial V = \emptyset$.

[5] The derivative of a function defined on a Hilbert space is always an element of the Hilbert space.

Remark 3.1.9. The subdifferential is additive for convex functions which have a subdifferential everywhere[6] in the sense that $A+B=\{a+b : a\in A, b\in B\}$, whenever A,B are subsets of some additive set. One inclusion is easy: if $v\in\partial f(x)$ and $w\in\partial g(x)$ then

$$\begin{aligned}(f+g)(x') &= f(x')+g(x')\geq f(x)+\langle v,x'-x\rangle+g(x)+\langle w,x'-x\rangle\\ &= (f+g)(x)+\langle v+w,x'-x\rangle,\end{aligned}$$

hence $\partial(f+g)\supseteq\partial f+\partial g$. The converse inclusion is trickier and requires the aforementioned assumptions. Moreover, the subdifferential is POSITIVE HOMOGENEOUS:

$$\partial(\lambda f)(x)=\lambda\partial f(x),\qquad \lambda>0. \tag{3.1.8}$$

Indeed, we only need to multiply (3.1.7) by $\lambda>0$ to obtain that $v\in\partial f(x)$ is equivalent to

$$\lambda f(x')\geq\lambda f(x)+\langle\lambda v,x-x'\rangle,$$

i.e., $\lambda v\in\partial(\lambda f)(x)$.

Example 3.1.10. Let us consider the subdifferential of the convex function $\|\cdot\|:\mathscr{H}\to\mathbb{R}$. In the case $x=0$ a vector v belongs to the subgradient if and only if

$$\langle v,x'\rangle=\langle v,x'-0\rangle\leq\|x'\|-\|0\|=\|x'\|,\qquad x'\in\mathscr{H},$$

which happens if and only if $\|v\|\leq 1$. This is easy to see: if $\|v\|\leq 1$ then the Cauchy–Schwarz inequality yields

$$\langle v,x'\rangle\leq\big|\langle v,x'\rangle\big|\leq\|v\|\|x'\|\leq\|x'\|,\qquad x'\in\mathscr{H},$$

while for $\|v\|>1$ the choice $x'=v$ gives

$$\langle v,x'\rangle=\langle v,v\rangle=\|v\|^2>\|v\|=\|x'\|$$

and establishes the converse.

The case $x\neq 0$ follows from a more general principle as the norm is differentiable there with derivative[7] $\|x\|^{-1}x$.

The subdifferential allows us to characterize minima of even nonsmooth convex functions like the ROF functional.

Theorem 3.1.11. *$x\in\mathscr{H}$ is a minimum of the convex function $f:\mathscr{H}\to\mathbb{R}$ if and only if $0\in\partial f(x)$.*

Proof: If $0\in\partial f(x)$, then (3.1.7) with $v=0$ immediately yields $f(x')\geq f(x)$, $x'\in\mathscr{H}$. Conversely, if x is a local, hence by Lemma 3.1.6 a global minimum, then

$$f(x')\geq f(x)=f(x)+\langle 0,x-x'\rangle,\qquad x'\in\mathscr{H},$$

tells us that $0\in\partial f(x)$. □

It is time to return to the ROF functional which now we write in the form

$$J(u)=\frac{1}{2}\big\|f-u\big\|_2^2+\lambda\|u\|_{TV}, \tag{3.1.9}$$

[6] More precisely: the interior of the set $\{x:\partial f(x)\neq\emptyset\}$ must have nonempty intersection with the respective set of the other one, see [Chambolle et al., 2009, Proposition 3.7].

[7] Keep in mind: The derivative is a linear form!

where the factor $\frac{1}{2}$ just has the purpose of making computations a bit simpler. By the Remarks 3.1.9 and 3.1.8 we then get that

$$\partial J(u) = \partial\left(\frac{1}{2}\|u-f\|_2^2 + \lambda\|u\|_{TV}\right) = \partial\frac{1}{2}\|u-f\|_2^2 + \lambda\,\partial\|\,|\nabla u|\,\|_1 = (u-f) + \lambda\,\partial\|u\|_{TV},$$

and the[8] minimum is reached if

$$0 \in \partial J(u) \qquad \Leftrightarrow \qquad \frac{f-u}{\lambda} \in \partial\|u\|_{TV}. \tag{3.1.10}$$

3.2 Duality

Definition 3.2.1 (Conjugate)**.** The LEGENDRE–FENCHEL CONJUGATE of $f : \mathscr{H} \to \mathbb{R}$ is defined as

$$f^*(x) := \sup_{v\in\mathscr{H}} \big(\langle v, x\rangle - f(v)\big). \tag{3.2.1}$$

If we fix v then the function $x \mapsto f_v(x) := \langle v, x\rangle - f(v)$ is a linear function, hence CONVEX and CONCAVE. Moreover, the pointwise supremum of convex functions is convex again:

$$f_v\left((1-\alpha)x+\alpha x'\right) \le (1-\alpha)f_v(x)+\alpha f_v(x') \le (1-\alpha)\sup_v f_v(x)+\alpha\sup_v f_v(x') = (1-\alpha)f^*(x)+\alpha f^*(x')$$

holds for all v, hence also for the supremum over v and thus we have the following observation.

Observation 3.2.2. *The function f^* is convex*[9]*.*

Consequently, also the BIDUAL function

$$f^{**} = \big(f^*\big)^* = \sup_{v\in\mathscr{H}} \big(\langle v, x\rangle - f^*(v)\big)$$

is also convex so that biduality can only be an identity for convex functions. And indeed it is.

Theorem 3.2.3. *If $f : \mathscr{H} \to \mathbb{R}$ is a* PROPER[10]*,* LOWER SEMICONTINUOUS[11] *convex function, then $f^{**} = f$.*

Proof: One half is easy: since, by definition $f^*(x) \ge \langle v, x\rangle - f(v)$ for any $v \in \mathscr{H}$, we have that

$$f(v) \ge \langle v, x\rangle - f^*(x) \qquad \Rightarrow \qquad f(v) \ge \sup_{x\in\mathscr{H}} \big(\langle v, x\rangle - f^*(x)\big) = f^{**}(v).$$

To show that $f^{**}(x) \ge f(x)$ one needs a slightly more tricky argument based on the hyperplane separation theorem, cf. [Sauer, 2017a], and the notion of the epigraph; for details see [Chambolle et al., 2009]. □

[8] Because of convexity.

[9] To be precisely, one has to be a little bit more careful. The supremum can be ∞ and the only convex function with $f(x) = \infty$ for some x is the trivially convex $f \equiv \infty$.

[10] $f \not\equiv \pm\infty$, to exclude trivialities.

[11] That means

$$f(x) \le \liminf_{x'\to x} f(x'), \qquad x \in \mathscr{H}.$$

Since

$$\langle v, x\rangle = \langle v, x\rangle - f(x) + f(x) \le \sup_{x\in\mathscr{H}} \big(\langle v, x\rangle - f(x)\big) + f(x) = f^*(v) + f(x), \qquad v, x \in \mathscr{H},$$

and therefore, for $v \in \partial f(x)$,

$$f^*(v) \ge \langle v, x\rangle - f(x) \ge \langle v, x'\rangle - f(x'), \qquad x, x' \in \mathscr{H},$$

so that

$$f^*(v) = \sup_{x'\in\mathscr{H}} \big(\langle v, x'\rangle - f(x')\big) \le \langle v, x\rangle - f(x) \le f^*(v).$$

Together with Theorem 3.2.3, this leads to the following dual description of the subdifferential.

Proposition 3.2.4 (LEGENDRE–FENCHEL IDENTITY)**.** *If f is convex then*

$$v \in \partial f(x) \qquad \Leftrightarrow \qquad \langle v, x\rangle = f(x) + f^*(v) \qquad \Leftrightarrow \qquad x \in \partial f^*(v). \tag{3.2.2}$$

Next, we compute the dual formulation of the ROF minimization. To that end, we recall that according to (3.1.10) u is solution if $\frac{f-u}{\lambda} \in \partial \|u\|_{TV}$ which is by Proposition 3.2.4 equivalent to $u \in \partial \left\| \frac{f-u}{\lambda} \right\|_{TV}^*$. In other words, the problem

$$\min_u \|u - f\|_2^2 + \lambda \|u\|_{TV}$$

becomes, setting $v := \frac{f-u}{\lambda}$, i.e., $u = f - \lambda v$, equivalent to

$$f - \lambda v = u \in \partial \|v\|_{TV}^* \qquad \Leftrightarrow \qquad 0 \in \lambda v - f + \partial \|v\|_{TV}^* \qquad \Leftrightarrow \qquad 0 \in v - \frac{f}{\lambda} + \partial \frac{1}{\lambda} \|v\|_{TV}^*,$$

see (3.1.8), so that minimizing the ROF functional is equvialent to its DUAL PROBLEM

$$\min_v \frac{1}{2} \left\| v - \frac{f}{\lambda} \right\|_2^2 + \frac{1}{\lambda} \|v\|_{TV}^*. \tag{3.2.3}$$

To explore this further, we need the convex dual of the convex functional $\|\cdot\|_{TV}$. To this extent, we recall the dual formulation of the TV norm from (2.2.9) and rewrite it as

$$\|u\|_{TV} = \sup_{|\phi|\le 1} \langle -\operatorname{div}\phi, u\rangle = -\inf_{|\phi|\le 1} \langle \operatorname{div}\phi, u\rangle, \tag{3.2.4}$$

where we use the inner product $\langle f, g\rangle = \int_\Omega fg$.

Remark 3.2.5. From now on, the presentation is **not** mathematically precise. If it were, we would have to care for proper function spaces or carefully formulate things in a distributional sense which is how we obtained the (dual) definition of the TV norm. This is doable, but very technical.

We can rewrite the CONSTRAINED formula (3.2.4) into an unconstrained one by using the simple trick of using

$$\|u\|_{TV} = \sup_v \left(\langle v, u\rangle - h(v)\right), \qquad h(v) = \frac{1}{\chi_K(v)}, \quad K = \left\{-\operatorname{div}\phi : |\phi| \le 1\right\}, \tag{3.2.5}$$

where the reciprocal of the characteristic function takes the values 1 and ∞, but is nevertheless convex since K is closed and convex. Comparing (3.2.5) with (3.2.1), we thus have that $\|u\|_{TV} = h^*(u)$ and thus

$$\|u\|_{TV}^* = h^{**}(u) = h(u).$$

Therefore, (3.2.3) becomes

$$\min_v \frac{1}{2}\left\|v - \frac{f}{\lambda}\right\|_2^2 + \frac{1}{\lambda}h(v) \qquad \Leftrightarrow \qquad \min_{v\in K} \frac{1}{2}\left\|v - \frac{f}{\lambda}\right\|_2^2$$

so that (3.2.3) is equivalent to[12]

$$\min_{|\phi|\leq 1} \frac{1}{2}\left\|\operatorname{div}\phi - \frac{f}{\lambda}\right\|_2^2, \tag{3.2.6}$$

from which we recover the desired u as $u = f - \lambda v = f + \lambda \operatorname{div}\phi$. Note that (3.2.6) can be easily discretized by associating vectors ϕ_{jk} to any pixel and we get

$$\min_{|\phi_{jk}|\leq 1} \frac{1}{2}\sum_{jk}\left(\left(\phi^1_{j+1,k} - \phi^1_{jk}\right) + \left(\phi^2_{j,k+1} - \phi^2_{jk}\right) - \frac{f_{jk}}{\lambda}\right)^2$$

and the side condition could even be expressed by Lagrange multipliers, leading to a linear system, however with a distinction whether any restriction is ACTIVE or not, i.e., whether $|\phi_{jk}| = 1$ or $|\phi_{jk}| < 1$. There are methods for this, see for example [Nocedal and Wright, 1999], but since we went so far in convex optimization, we want to learn more about methods from there.

3.3 Proximal operators and splitting

If we consider the minimization problem

$$\min_u \frac{1}{2}\|u - f\|_2^2 + \lambda F(u), \qquad \lambda > 0, \tag{3.3.1}$$

where F is a convex function(al), then this problem always has a unique solution as the function to be minimized is STRICTLY CONVEX. The solution u is characterized by

$$0 \in u - f + \lambda\,\partial F(u) \qquad \Leftrightarrow \qquad f \in (I + \lambda\,\partial F)(u) \qquad \Leftrightarrow \qquad u = (I + \lambda\,\partial F)^{-1}(f),$$

with equality since the solution is unique.

Definition 3.3.1. The mapping $f \mapsto \operatorname{prox}_F(f) := (I + \lambda\,\partial F)^{-1}(f)$ is called the PROXIMAL MAP for the function(al) F.

Example 3.3.2. A simple example is when $F = \frac{1}{\chi_K}$ is a reciprocal characteristic function of a compact set K, i.e.,

$$F(u) = \begin{cases} 1, & u \in K, \\ +\infty, & u \notin K. \end{cases}$$

Since $u = \operatorname{prox}_F(f)$ minimizes (3.3.1), i.e.,

$$\operatorname{prox}_F(f) = \operatorname*{argmin}_u \frac{1}{2}\|u - f\|_2^2 + \lambda \begin{cases} 1, & u \in K, \\ +\infty, & u \notin K, \end{cases}$$

[12] The "−" in front of the divergence can be discarded since the ϕ are symmetric around zero.

it follows that u is the ORTHOGONAL PROJECTION of f on K, i.e.

$$\operatorname{prox}_F(f) = P_K(f) := \operatorname*{argmin}_{u \in K} \|u - f\|_2. \tag{3.3.2}$$

The proximal map has another interesting property. To that end, we recall the dual of (3.3.1), namely,

$$\min_v \frac{1}{2} \left\| v - \frac{1}{\lambda} f \right\|_2^2 + \frac{1}{\lambda} F^*(v) \tag{3.3.3}$$

yielding

$$\frac{1}{\lambda} f \in \left(I + \frac{1}{\lambda} \partial F^* \right)(v) \qquad \Leftrightarrow \qquad v = \left(I + \frac{1}{\lambda} \partial F^* \right)^{-1} \left(\frac{1}{\lambda} f \right).$$

Substituting this into the duality relation $u = f - \lambda v$ used to derive (3.2.3), we thus get for the primal and dual minimizer

$$(I + \lambda \partial F)^{-1}(f) = u = f - \lambda v = f - \lambda \left(I + \frac{1}{\lambda} \partial F^* \right)^{-1} \left(\frac{1}{\lambda} f \right)$$

or

$$f = (I + \lambda \partial F)^{-1}(f) + \lambda \left(I + \frac{1}{\lambda} \partial F^* \right)^{-1} \left(\frac{1}{\lambda} f \right), \qquad \lambda > 0. \tag{3.3.4}$$

This identity, called MOREAU'S IDENTITY, decomposes f into a primal and dual component and can be seen as some analogy of an orthogonal decomposition. Setting $\lambda = 1$ in (3.3.4), we get the slightly simpler form

$$f = (I + \partial F)^{-1}(f) + \left(I + \partial F^* \right)^{-1}(f). \tag{3.3.5}$$

Example 3.3.3 (Example 3.3.2, continued)**.** If $\operatorname{prox}_F(f) = P_K(f)$, then it follows from replacing f by λf in (3.3.4) that

$$\left(I + \frac{1}{\lambda} \partial F^* \right)(f) = f - \frac{1}{\lambda} \underbrace{(I + \lambda \partial F)^{-1}}_{= \operatorname{prox}_F = P_K} (\lambda f) = f - \frac{1}{\lambda} P_K(\lambda f) =: u^*$$

Hence, the dual minimizer can be found easily by "shrinking" f to the compact set K. If P_k is not positively homogeneous, this can really depend on λ.

Now we want to use our brand new knowledge of convex optimization to derive algorithms that converge faster than gradient descent.

Remark 3.3.4. It is known from optimization [Nocedal and Wright, 1999, Sauer, 2013b] that with a proper choice of stepwidth, for example the so–called Wolfe conditions, gradient descent converges to a CRITICAL POINT where the gradient vanishes. However, it is also known that this convergence can be and often is very slow.

The first trick is called SPLITTING. Suppose we can write an minimization problem

$$\min_u F(u) + G(u)$$

in such a way that F is smooth, i.e., at least differentiable and G is SIMPLE in the sense that

$$\operatorname{prox}_G = (I + \lambda \partial G)^{-1}$$

can be computed easily.

Example 3.3.5. These requirements are met in the dual ROF problem (3.2.6) where

$$F(\phi) = \frac{1}{2}\left\|\operatorname{div}\phi - \frac{1}{\lambda}f\right\|_2^2 \qquad \text{and} \qquad G(\phi) = \begin{cases} 0, & \|\,|\phi|\,\|_\infty = \max_x |\phi(x)| \le 1 \\ +\infty, & \text{otherwise.} \end{cases}$$

Though the set of all ϕ with $|\phi| \le 1$ is convex, the functional G is not strictly convex, and its set valued subgradient takes the form

$$\partial G(\phi) = \begin{cases} 0, & |\phi| < 1, \\ \{\psi : \psi^T\phi \ge 0\}, & \left\|\,|\phi|\,\right\|_\infty = 1, \\ \mathbb{R}^s, & \left\|\,|\phi|\,\right\|_\infty > 1, \end{cases}$$

hence for any ϕ on the boundary of the unit ball $|\phi| \le 1$, the subgradient consists of all directions pointing "outwards" of the convex set or into the halfplane generated by the separating hyperplane attached to ϕ, see Fig. 3.3.1. Hence the set

$$H_\phi = (I + \lambda\partial G)(\phi)$$

coincides with this halfplane and ϕ is the preimage for any point from there. In summary,

$$(I + \lambda\partial G)(\phi) = \begin{cases} \phi, & |\phi| < 1 \\ H_\phi, & \|\,|\phi|\,\|_\infty = 1, \\ \mathbb{R}^2, & \|\,|\phi|\,\|_\infty > 1. \end{cases} \tag{3.3.6}$$

Every point outside the convex set lies in (at least) one of these hyperplanes H_ϕ and admits the respective ϕ as its preimage with respect to $I + \lambda\partial G$. The "best" preimage however, is the closest one, the ORTHOGONAL PROJECTION of ψ to the convex set and is, for balls, defined by

$$\operatorname{prox}_G(\phi) = \frac{\phi}{\max\left(\left\|\,|\phi|\,\right\|_\infty, 1\right)}. \tag{3.3.7}$$

Hence, the proximal map is indeed easy to compute.

Now the minimization is split into two parts: A gradient step for F and a projection step according to G, that is, starting with $u_0 = f$ one computes

$$v^{k+1} = u^k - \alpha_k \nabla F\left(u^k\right), \tag{3.3.8}$$

$$u^{k+1} = \operatorname{prox}_G\left(v^{k+1}\right), \tag{3.3.9}$$

or, as a single step

$$u^{k+1} = \operatorname{prox}_G\left(u^k - \alpha_k \nabla F\left(u^k\right)\right), \tag{3.3.10}$$

where α_k is the stepwidth of kth iteration. This simple form of the iteration still converges relatively slow, but there has been a lot of recent work to apply acceleration methods for convergence to this iteration.
What does all this mean for discretized ROF? Here ϕ consists of the pixel vectors $\phi_{jk} \in \mathbb{R}^2$ and

$$F(\phi) = \frac{1}{2}\sum_{jk}\left(\operatorname{div}\phi_{jk} - \frac{1}{\lambda}f_{jk}\right)^2 = \frac{1}{2}\sum_{jk}\left(\phi^1_{j+1,k} - \phi^1_{jk} + \phi^2_{j,k+1} - \phi^2_{jk} - \frac{1}{\lambda}f_{jk}\right)^2,$$

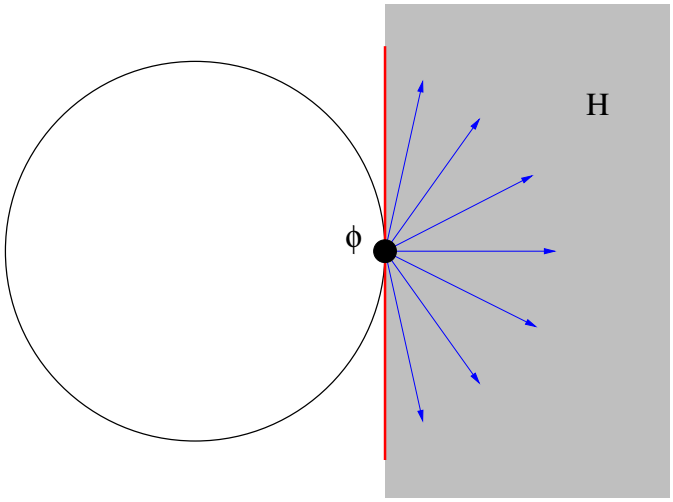

Figure 3.3.1: Subgradient of the characteristic function of a convex set. The image of a point ϕ inside is the point itself, the image of a point ϕ on the boundary is the halfspace generated by the (red) tangent hyperplane to the convex set.

such that

$$\frac{\partial F}{\partial \phi^1_{rs}} = \left(\operatorname{div}\phi_{r-1,s} - \frac{1}{\lambda} f_{r-1,s}\right) - \left(\operatorname{div}\phi_{rs} - \frac{1}{\lambda} f_{rs}\right) = \delta_1^- \left(\operatorname{div}\phi - \frac{1}{\lambda} f\right)_{rs}$$

where δ_1^- denotes the partial backwards difference operator with respect to the first variable. A similar argument works the second variable, so that we can summarize it as

$$\nabla F = \frac{\partial F}{\partial \boldsymbol{\phi}} = \nabla^- \left(\operatorname{div}\phi - \frac{1}{\lambda} f\right) = \begin{pmatrix} \delta_1^- \left(\operatorname{div}\phi - \frac{1}{\lambda} f\right) \\ \delta_2^- \left(\operatorname{div}\phi - \frac{1}{\lambda} f\right) \end{pmatrix}.$$

Exercise 3.3.1 Implement the split iteration for the ROF functional in Matlab/Octave. ♢

3.4 Split Bregman and primal–dual

The next type of methods uses so–called "augmented Lagrangian" methods. Here, we introduce an additional variable $p = \nabla u$ and solve

$$\min_{u,p} \frac{1}{2} \|u - f\|_2^2 + \lambda \| |p| \|_1 \qquad \text{subject to } p = \nabla u. \tag{3.4.1}$$

A direct Lagrange multiplier approach is difficult because the norm applied to p is still not smooth. Instead, one considers, for yet another smoothing parameter α, the functional

$$L_\alpha(u, p, \mu) := \frac{1}{2} \|u - f\|_2^2 + \| |p| \|_1 + \langle \mu, p - \nabla u \rangle + \frac{\alpha}{2} \|p - \nabla u\|_2^2$$

by alternatingly minimizing with respect to the two variables and then updating μ. The procedure is

$$\begin{aligned} u^{k+1} &= \operatorname*{argmin}_u L_\alpha(u, p^k, \mu^k), \\ p^{k+1} &= \operatorname*{argmin}_p L_\alpha(u^k, p, \mu^k), \\ \mu^{k+1} &= \mu^k + \alpha \left(p^{k+1} - \nabla u^{k+1}\right). \end{aligned}$$

Clearly, μ does not change if p^{k+1} and u^{k+1} perfectly satisfy the relationship $p^{k+1} - \nabla u^{k+1}$, otherwise one makes a step in the direction of the gradient of $\langle \cdot, p - \nabla u \rangle$ in order to make μ as "dual" as possible and to more strictly enforce this condition. This method which is based on a more heuristic approach can be shown to converge, for references see [Chambolle et al., 2009].

For the primal–dual method, we take yet another general form of the problem formulation, namely

$$\min_u F(u) + G(Au) \tag{3.4.2}$$

where F, G are convex and A is a linear operator.

Example 3.4.1. In the ROF case, this corresponds to $F = \frac{1}{2}\| \cdot - f\|_2^2$, $G = \lambda \| |\cdot| \|_1$ and $A = \nabla$.

We use the fact $G^{**} = G$ to consider $\min_u F(u) + G^{**}(Au)$ and since

$$G^{**}(Au) = \max_v \langle v, Au \rangle - G^*(v),$$

the problem (3.4.2) can be rewritten as

$$\min_u \max_v H(u, v) = \min_u \max_v \langle v, Au \rangle - G^*(v) + F(u), \tag{3.4.3}$$

which is now a MINIMAX PROBLEM. The idea is to approach it by alternativ ascending and descending steps with respect to v and u. Since the problem is convex–concave[13], we can interchange[14] min and max to obtain

$$\begin{aligned}
\min_u F(u) + G(Au) &= \min_u \max_v \langle v, Au \rangle - G^*(v) + F(u) \\
&= \max_v \min_u \langle A^* v, u \rangle - G^*(v) + F(u) = \max_v \Big(-G^*(v) + \min_u \langle A^* v, u \rangle + F(u) \Big) \\
&= \max_v \Big(-G^*(v) + \min_u \big(- \big(\langle -A^* v, u \rangle - F(u) \big) \big) \Big) \\
&= \max_v \Big(-G^*(v) - \underbrace{\max_u \big(\langle -A^* v, u \rangle - F(u) \big)}_{= F^*(-A^* v)} \Big) \\
&= \max_v - \big(G^*(v) + F^*(-A^* v) \big).
\end{aligned}$$

Since this implies for any u, v that

$$- \big(G^*(v) + F^*(-A^* v) \big) \leq F(u) + G(Au),$$

it follows that the DUALITY GAP

$$\gamma(u, v) = F(u) + G(Au) + G^*(v) + F^*(-A^* v) \tag{3.4.4}$$

is always nonnegative and becomes zero if and only if u and v are optimal solutions, respectively. This leads to a very handy stopping criterion: iterate the method until $\gamma(u, v)$ becomes small because then the (unknown) optimal values are almost reached; the duality gap even tells us to which percentage the approximate solution is already optimal as it always provides a lower bound for the function to be minimized.

[13] The linear function is both, $-G^*$ is concave and F is convex, hence we maximize a concave function and minimize a convex one.

[14] In general this is **not** permitted, cf. [Sauer, 2017b].

In the iteration, we use the idea of the Douglas–Rachford split (3.3.10) and once intepret the minimization problem with respect to u for fixed v as

$$\min_u \big(\langle v, Au\rangle - G^*(v)\big) + F(u)$$

which results in

$$0 \in \alpha\,\partial_u\big(\langle v, Au\rangle - G^*(v) + F(u)\big) = \alpha\,A^*v + \alpha\,\partial F(u) = \alpha\,A^*v - u + u + \alpha\,\partial F(u),$$

where $\alpha > 0$ is arbitrary. This can be rewritten as

$$u - \alpha\,A^*v \in (I + \alpha\,\partial F)(u),$$

yielding the iteration

$$u^{k+1} = (I + \alpha\,\partial F)^{-1}\left(u^k - A^*v^k\right) = \operatorname{prox}_F\left(u^k - \alpha\,\nabla_u\left(\langle A^*v, u\rangle - G^*(v)\right)(u^k)\right).$$

Analogously, we get for the concave maximization problem with respect to v for fixed u that

$$0 \in \beta\,\partial_u\big(\langle v, Au\rangle - G^*(v) + F(u)\big) = \beta\,Au + u - (u + \partial G^*(u)),$$

i.e.,

$$u + \beta\,Au \in (I + \beta\,\partial G^*)(u)$$

and we obtain the iteration

$$v^{k+1} = \left(I + \beta\,\partial G^*\right)^{-1}\left(u^k + \beta\,Au^k\right) = \operatorname{prox}_{G^*}\left(v^k + \beta\,\nabla_v\left(\left\langle v, Au^k\right\rangle + F(u)\right)(v^k)\right).$$

Together, this results for stepsize parameters $\alpha, \beta > 0$ in the iteration

$$v^{k+1} = \left(I + \beta\,\partial G^*\right)^{-1}\left(v^k + \beta\,Au^k\right) \tag{3.4.5}$$

$$u^{k+1} = (I + \alpha\,\partial F)^{-1}\left(u^k - \alpha\,A^*v^{k+1}\right) \tag{3.4.6}$$

that is repeated until $\gamma(u^k, v^k)$ is small enough.

Example 3.4.2 (Conjugate of gradient)**.** To make this method applicable in the setting of the ROF functional, we need to know what the CONJUGATE A^* of the gradient operator $A = \nabla$ is. Since A maps functions on $\mathbb{R}^s$ to s–valued functions, the standard inner product results in

$$\langle \phi, \nabla f\rangle = \sum_{j=1}^{s}\int_\Omega \phi_j(x)\left(\nabla f\right)_j(x)\,dx = -\sum_{j=1}^{s}\int_\Omega \frac{\partial \phi_j}{\partial x_j}(x)\,f(x)\,dx$$

for any smooth function $\phi : \Omega \to \mathbb{R}^2$ whose support is contained in the interior of Ω. Hence[15], we meet an old friend:

$$\nabla^* = -\operatorname{div}. \tag{3.4.7}$$

To formulate (3.4.5) and (3.4.6) explicitly for the ROF problem, we recall that $F = \frac{1}{2}\|\cdot - f\|_2^2$, hence

$$(I + \alpha\partial F)(u) = u + \alpha\partial F(u) = u + \alpha(u - f) = -\alpha f + (1 + \alpha)u,$$

[15] Does that surprise anyone?

so that

$$u^k - \alpha \operatorname{div} v^{k+1} = (1+\alpha)u - \alpha f \qquad \Leftrightarrow \qquad u = \frac{1}{1+\alpha}\left(u^k + \alpha\left(f - \operatorname{div} v^{k+1}\right)\right)$$

yields that (3.4.6) simplifies to the convex combination

$$u^{k+1} = \frac{1}{1+\alpha}\left(u^k + \alpha\left(f - \operatorname{div} v^{k+1}\right)\right) = \frac{1}{1+\alpha}u^k + \frac{\alpha}{1+\alpha}\left(f - \operatorname{div} v^{k+1}\right). \tag{3.4.8}$$

The dual of

$$G(v) = \lambda \int_\Omega |v(x)|\,dx = \sup_{|w|\le\lambda} \int_\Omega w^T(x)\, v(x)\,dx = \sup_w \left(\langle w, v\rangle - h_\lambda(w)\right) = h_\lambda^*(v)$$

where

$$h_\lambda(w) = \begin{cases} 0, & |w| \le \lambda, \\ +\infty, & |w| > \lambda, \end{cases}$$

is $G^* = h^{**} = h$. As in Example 3.3.5, specifically (3.3.7), we can turn (3.4.5) into the projection step

$$v^{k+1} = \frac{v^k + \alpha \nabla u^k}{\max\left(\frac{1}{\lambda}\left\| \left| v^k + \alpha \nabla u^k \right| \right\|_\infty, 1\right)}. \tag{3.4.9}$$

Remark 3.4.3. Note that points v with $|v| > \lambda$ are useless for the iteration as then $\gamma(u,v) = \infty$ for $|v| > 1$ since $G^*(v) = \infty$ in this case and all other functionals are finite. Since our goal is $\gamma(u^k, v^k) \to 0$, we always have to project to the λ–ball anyway.

For a working algorithm, we need to compute the duality gap (3.4.4). The first three parts are clear:

$$\begin{aligned} F(u) &= \frac{1}{2}\left\|f-u\right\|_2^2, \\ G(Au) &= \lambda \left\| |\nabla u| \right\|_1 = \lambda \|u\|_{TV}, \\ G^*(v) &= h(v) = 0, \end{aligned}$$

at least for the admissible values of v, otherwise the gap is $+\infty$. It remains to compute

$$F^*(-A^*v) = \max_u \left\langle -A^*v, u\right\rangle - F(u) = \max_u \left\langle -A^*v, u\right\rangle - \frac{1}{2}\|f-u\|_2^2.$$

The extremal u is found by setting

$$0 = \nabla_u \left(\left\langle -A^*v, u\right\rangle - \frac{1}{2}\|f-u\|_2^2\right) = -A^*v - (u-f) \qquad \Rightarrow \qquad u = f - A^*v.$$

Resubstituting this yields that

$$\begin{aligned} F^*(-A^*v) &= \left\langle -A^*v, f - A^*v\right\rangle - \frac{1}{2}\left\|f - (f - A^*v)\right\|_2^2 = \|A^*v\|_2^2 - \left\langle A^*v, f\right\rangle - \frac{1}{2}\left\|A^*v\right\|_2^2 \\ &= \frac{1}{2}\left\|A^*v\right\|_2^2 - \left\langle A^*v, f\right\rangle, \end{aligned} \tag{3.4.10}$$

which sums up into

$$\gamma(u,v) = \frac{1}{2}\|f-u\|_2^2 + \lambda\|u\|_{TV} + \frac{1}{2}\|\operatorname{div} v\|_2^2 - \left\langle \operatorname{div} v, f\right\rangle + G^*(v) \tag{3.4.11}$$

and only makes sense if $|v| \leq \lambda$.

It is illustrative to see what this means for $\lambda \to 0$ and $\lambda \to \infty$. In fact, if $\lambda = 0$, then, v has no choice but being identically zero and $u = f$ is the optimal choice wich also makes (3.4.11) equal to zero. If $\lambda \to \infty$, then $u = 0$ and $v = 0$ is the overregularized solution and the smoothest function imaginable.

There is only one item of bad news: it is not so simple. There is no guarantee that naive methods based on alternating optimization for separated variables converge and therefore stepsize selection and smoothing of the steps becomes very important. For details see [Chambolle et al., 2009] and [Zhu and Chan, 2008].

Imaging applications | 4

It always does seem to me that I am doing more work than I should do. It is not that I object to the work, mind you; I like work: it fascinates me. I can sit and look at it for hours. I love to keep it by me: the idea of getting rid of it nearly breaks my heart.

(J. K. Jerome, *Three Men in a Boat*)

We finally try to find out how we can apply the ideas for ROF regularization to classical imaging problems.

4.1 Denoising

As already mentioned before, the original purpose of ROF regularization was the *denoising* of images. The usual model is that of additive, independent noise at the pixels, i.e. the MEASURED QUANTITY is

$$\hat{f}_{jk} = f_{jk} + \eta_{jk}, \qquad j = 1,\dots,m,\ k = 1,\dots,n. \tag{4.1.1}$$

For some reasons that will become clear soon, we now restrict ourselves to the discrete situation. Next, one makes assumptions on the distribution of the error, based on some poor man's probability.

Definition 4.1.1. A PROBABILITY DENSITY $p : \mathbb{R} \to \mathbb{R}$ is a function such that

$$p(t) \geq 0, \quad t \in \mathbb{R}, \qquad \int_{\mathbb{R}} p(t)\,dt = 1. \tag{4.1.2}$$

A density function is the GAUSSIAN DISTRIBUTION with MEAN μ and VARIANCE σ, defined as

$$p(t|\mu,\sigma^2) := \left(2\pi\sigma^2\right)^{-1/2} e^{-\frac{1}{2\sigma^2}(t-\mu)^2}. \tag{4.1.3}$$

Exercise 4.1.1 Verify that the Gaussian distribution is a probability density. ♢

Now the probabilistic *model* of an image is that each pixel is a realization of a random process and has the distribution

$$P\left(\hat{f}_{jk} = y\right) = p\left(y|f_{jk},\beta\right); \tag{4.1.4}$$

assuming independence of the pixel values, the LIKELIHOOD of a certain measurement is

$$P(\hat{f},\beta) = \prod_{jk} p\left(\hat{f}_{jk}|f_{jk},\beta\right).$$

The goal is now to determine the distribution parameters f_{jk} in such a way that the likelihood of the measurement $\hat{f}$ is maximized. Since the log is monotonic, we can also maximize $\log P(\hat{f},\beta)$ or minimize the function

$$\begin{aligned} -\log P(\hat{f},\beta) &= -\log\left(\prod_{jk} p\left(\hat{f}_{jk}|f_{jk},\beta\right)\right) = -\sum_{jk}\log p\left(\hat{f}_{jk}|f_{jk},\beta\right) \\ &= \sum_{jk}\left(\frac{1}{2\beta}\left(\hat{f}_{jk}-f_{jk}\right)^2+\frac{\log 2\pi+\log\beta}{2}\right) = \frac{1}{2\beta}\left\|\hat{f}-f\right\|_2^2+mn\frac{\log 2\pi+\log\beta}{2}. \end{aligned}$$

Noting that the additive term does not depend on f and thus does not affect the minimization problem, we can add a REGULARIZATION TERM $\|f\|_{TV}$ and end up with the minimization problem

$$\min_f \frac{1}{2\beta}\|\hat{f}-f\|_2^2+\|f\|_{TV} \qquad \Leftrightarrow \qquad \min_f \frac{1}{2}\|\hat{f}-f\|_2^2+\beta\|f\|_{TV}, \tag{4.1.5}$$

where now the (unknown) variance of the noise becomes the smoothing parameter. However, provided that we know an approximant f for the mean values, we can determine the β that gives the best explanation of the observation by considering

$$\frac{d}{d\beta}\left(-\log P(\hat{f},\beta)\right) = -\frac{1}{2\beta^2}\|\hat{f}-f\|_2^2+\frac{mn}{2\beta}$$

and setting it equal to zero, yielding the average error

$$\beta = \frac{\|\hat{f}-f\|_2^2}{mn}. \tag{4.1.6}$$

This leads to the generic denoising algorithm that repeats

1. $f_{k+1} = \operatorname{argmin}_f \frac{1}{2}\|\hat{f}-f\|_2^2+\beta_k\|f\|_{TV}$,
2. $\beta_{k+1} = \frac{1}{mn}\|\hat{f}-f_k\|_2^2$,

until the result is "good enough". This is also the basic idea behind [Rudin et al., 1992].

Remark 4.1.2.

1. It is important to keep in mind that the validity of this approach depends on the validity of the NOISE MODEL which has to be independent and Gaussian. There are indeed examples like POISSON NOISE[1] or SALT AND PEPPER NOISE[2]. These error models lead to different optimization problems like

   $$\min_f \|\hat{f}-f\|_1+\lambda\|f\|_{TV} \tag{4.1.7}$$

   for salt and pepper noise.

2. Moreover, in contrast to the ROF functional, the one in (4.1.7) is CONTRAST INVARIANT which means that if we replace $\hat{f}$ by $c\hat{f}$, $c>0$, then the optimal f is also just multiplied by a constant. This is simply the positive homogeneity of (4.1.7).

3. Regularizing with the TV norm has no real statistical meaning and just asks for a "nice" explanation of the measurements. The "statistics" only affect the data fidelity functional.

[1] Noise distributed with respect to a Poisson distribution that acts in a *multiplicative* way.

[2] A pixel is either correct or totally wrong which happens with defect pixels in camera detectors or transmission errors.

4.2 Zooming and deblurring

Another application of ROF regularization is in digital zooming and deblurring, more generally, in solving an INVERSE PROBLEM for the image. To that end, let A be a linear operator between images which may also change the size of the image.

Example 4.2.1. Standard examples for such operators are

1. DOWNSAMPLING:

$$A : \mathbb{R}^{2m\times 2n} \to \mathbb{R}^{m\times n}, \qquad (Af)_{jk} = f_{2j,2k}, \qquad \begin{matrix} j = 1,\dots,m, \\ k = 1,\dots,n. \end{matrix}$$

   Of course, any other factor is possible and for continuous images we can even consider $Af = f(X\cdot)$ where $X \in \mathbb{R}^{s\times s}$ is an arbitrary scaling matrix.

2. BLURRING: the original image is convolved with a signal g which usually acts as a low pass filter, i.e., $\hat{g}(0) \neq 0$. The operation is then

$$(Af)_{jk} = \big(g * f\big)_{jk} = \sum_{rs} g_{rs} f_{j-r,k-s},$$

   and A is a so-called TOEPLITZ MATRIX with a lot of special structural properties.

3. INTERPOLATION/INPAINTING: the operator extracts certain pixels

$$(Af)_r = f_{j(r),k(r)}, \qquad r = 1,\dots,N.$$

   This could also be seen as an unstructured downsampling operator.

The way to apply our techniques from the preceding chapters is now simple: just integrate A into the data fidelity term and solve

$$\min_u \frac{1}{2}\|f - Au\|_2^2 + \lambda \|u\|_{TV}. \tag{4.2.1}$$

Since

$$\partial\left(\frac{1}{2}\|f - Au\|_2^2\right) = A^*\big(f - Au\big) = A^* f - A^* Au, \tag{4.2.2}$$

the subgradient of the data fidelity functional is still handled quite easily.

Remark 4.2.2. In terms of numerical linear algebra, (4.2.2) is just the least squares solution for the linear system $Au = f$ and the regularization could be interpreted as a TIKHONOV REGULARIZATION of the linear system, cf. [Golub and van Loan, 1996].

Note that A^*A is usually *not* an invertible operator, but the linear system

$$A^* Au = A^* f$$

is always solvable for u. This is standard numerical linear algebra, see, for example [Golub and van Loan, 1996].

Remark 4.2.3. For successful deblurring, the nature of the blurring operator must be available, in other words, the coefficients c of the filter have to be known. Otherwise there is no chance of feeding the operator A into (4.2.1).

4.3 Segmentation

Finally, we want to decompose an image $f : \Omega \to \mathbb{R}$ into segments $S_1, \dots, S_n \subset \Omega$ of almost constant value, say $c_1, \dots, c_n$. Of course, in order to form a PARTITION, the segments should be disjoint, i.e.,

$$S_j^\circ \cap S_k^\circ = \emptyset, \qquad j \neq k,$$

where S° denotes the interior of the region S, and cover the region,

$$\Omega = \bigcup_{j=1}^{n} S_j.$$

The underlying assumption for segmentation is that the image f is (approximately) *constant* on theses regions.

Example 4.3.1. In COMPUTERIZED TOMOGRAPHY the VOXEL values correspond to a material property, essentially related to the *density* of the material. Thus, segmentation separates different material components.

The trick in segmentation is to approximate the CHARACTERISTIC FUNCTIONS

$$\chi_j := \chi_{S_j} : \Omega \to \{0, 1\}$$

which now have to form a PARTITION OF UNITY

$$\sum_{j=1}^{n} \chi_j(x) = 1, \qquad x \in \Omega. \tag{4.3.1}$$

Hence, segmentation is to approximate f as good as possible by the piecewise constant functions

$$\sum_{j=1}^{n} c_j \chi_j(x),$$

for example,

$$\min_{S_1,\dots,S_n,\, c_1,\dots,c_n} \left\| f - \sum_{j=1}^{n} c_j \chi_j \right\|_2^2. \tag{4.3.2}$$

Since

$$\begin{aligned}
\left\| f - \sum_{j=1}^{n} c_j \chi_j \right\|_2^2 &= \int_\Omega \left(f(x) - \sum_{j=1}^{n} c_j \chi_j(x) \right)^2 dx = \sum_{k=1}^{n} \int_{S_k} \left(f(x) - \sum_{j=1}^{n} c_j \chi_j(x) \right)^2 dx \\
&= \sum_{k=1}^{n} \int_{S_k} \big(f(x) - c_k \big)^2 \, dx = \sum_{k=1}^{n} \int_\Omega \chi_k(x) \big(f(x) - c_k \big)^2 \, dx \\
&= \sum_{k=1}^{n} \big\| \chi_k \big(f - c_k \big) \big\|_2^2,
\end{aligned}$$

the minimization problem can be rewritten as

$$\min_{S_1,\dots,S_n,\, c_1,\dots,c_n} \sum_{k=1}^{n} \big\| \chi_k \big(f - c_k \big) \big\|_2^2. \tag{4.3.3}$$

Moreover, the partition should be simple which leads to the MUMFORD–SHAH problem

$$\min_{S_1,\dots,S_n,\,c_1,\dots,c_n} \lambda \sum_{k=1}^{n} \|\chi_k\|_{TV} + \frac{1}{2} \sum_{k=1}^{n} \left\|\chi_k (f - c_k)\right\|_2^2. \tag{4.3.4}$$

The total variation of the characteristic function, hence the length of the level curve for the jump function, is called the PERIMETER of the set S_k. For a careful definition and properties of this object, see [Chambolle et al., 2009].

Once the partition is known, the best c_k are easily computed.

Lemma 4.3.2. *For fixed $S_1,\dots,S_n$, we have that*

$$\operatorname*{argmin}_c \sum_{k=1}^{n} \left\|\chi_k (f - c_k)\right\|_2^2 = \left(\frac{1}{|S_k|} \int_{S_k} f(x)\,dx : k = 1,\dots,n \right), \qquad |S_k| := \int_{S_k} 1\,dx, \tag{4.3.5}$$

where $c = (c_1,\dots,c_n)$.

Proof: To minimize the functional

$$F : c \mapsto \frac{1}{2} \int_{S_k} \left(f(x) - c\right)^2 dx,$$

we take the (sub)gradient with respect to c and obtain the requirement

$$0 = \int_{S_k} (f(x) - c)\,dx = \int_{S_k} f(x)\,dx - |S_k|\,c,$$

which immediately yields (4.3.5). □

Remark 4.3.3. To solve (4.3.3) alone, one could use the following simple greedy algorithm known as K–MEANS CLUSTERING. Starting with an arbitrary partition $S_1,\dots,S_n$ first set

$$c_k = \frac{1}{|S_k|} \int_{S_k} f(x)\,dx$$

and then

$$S_k := \left\{x : \left|f(x) - c_k\right| < \left|f(x) - c_j\right|,\ j \neq k\right\}.$$

This is quite simple for discrete images, just classifying the pixels with respect to the closest constant[3], but it will lead to arbitrarily complex and disconnected decompositions which is the reason why a regularization like in (4.3.4) has to be present.

With the 2–norm as data fidelity term we thus get the AVERAGE of f on the segements of the partitions as best constants, the 1–norm would give the MEDIAN over the domain.

Due to Lemma 4.3.2, the minimization problem is a minimization problem entirely in the partition, the respective constants are uniquely determined by the partition. Nevertheless, this "simplification" does not make the problem easy: the discrete counterpart of the partition problem, the so–called POTT'S MODEL is known to be NP–hard, so the only thing that we can hope for is a CONVEX APPROXIMATION of this problem.

The trick here is that of a CONVEX ENVELOPE, i.e., trying to find, for a general, probably nonconvex function F a convex functional G such that $G \geq F$, and then to minimize G. In addition, we want that $G = F$ whenever F is convex, not losing quality on good functionals.

[3] Yes, it also needs a rule to decide if $f(x)$ has the same distance to two of the c_k, but this is easy to solve.

Lemma 4.3.4. *For any F, the* BICONJUGATE *functional F^{**} is a convex envelope.*

Proof: We just have to recall the part of the proof that we gave in Theorem 3.2.3, showing that $F^{**} \geq F$ without any assumptions on F. □

Definition 4.3.5. By $BV(\Omega, X)$, $X \subset \mathbb{R}$, we denote the functions of BOUNDED VARIATION on Ω with values in X, i.e.,

$$f \in BV(\Omega, X) \qquad \Leftrightarrow \qquad f : \Omega \to X, \qquad \|f\|_{TV} < \infty. \tag{4.3.6}$$

The characteristic functions we are considering, do now belong to $BV(\Omega, \{0,1\})$ and must sum to 1. We can thus define the functional

$$H(v) := \begin{cases} \sum_{k=1}^{n} \|v_k\|_{TV}, & v_k \in BV(\Omega, \{0,1\}), \sum v_k = 1, \\ +\infty, & \text{otherwise,} \end{cases} \qquad v = (v_k : k = 1, \ldots, n). \tag{4.3.7}$$

Based on this one can consider the convex optimization problem

$$\min_v \lambda H^{**}(v) + \sum_{k=1}^{n} \|v_k(f - c_k)\|_2^2 \tag{4.3.8}$$

for fixed $c_1, \ldots, c_n$ which can then be iterated just like in the K–means case. It turns out that the DOMAIN of H^{**}, i.e., the set of all v such that $\partial H^{**}(v) \neq \emptyset$ consists of $BV(\Omega, [0,1]^n)$, hence the envelope given by the biconjugate problem just yields the continuous version of the problem.

Since $K = BV(\Omega, [0,1]^n)$ is again a convex set, the functional[4]

$$G(v) = \begin{cases} 0, & 0 \leq v \leq 1, \sum v_k = 1, \\ +\infty, & \text{otherwise,} \end{cases}$$

can be used to complete the minimization problem as

$$\min_v \lambda \left(\sum_{k=1}^{n} \|v_k\|_{TV} + G(v) \right) + \sum_{k=1}^{n} \left\| v_k(f - c_k) \right\|_2^2, \tag{4.3.9}$$

which is now a convex problem that can be handled with the methods of the preceding chapter. Again, the "characteristic function" G is handled by an orthogonal projection on the convex set K.

[4] For the *vector valued* $v = (v_k : k = 1, \ldots, n)$ the inequality $0 \leq v \leq 1$ has to be understood coordinatewise, hence as $0 \leq v_k \leq 1$, $k = 1, \ldots, n$.

Bibliography

[Akhieser, 1988] Akhieser, N. I. (1988). *Lectures on Integral Transforms*, volume 70 of *Translations of Mathematical Monographs*. AMS.

[Berens, 1968] Berens, H. (1968). *Interpolationsmethoden zur Behandlung von Approximationsprozessen auf Banachräumen*, volume 64 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin, Heidelberg, New York.

[Boor, 1978] Boor, C. d. (1978). *A practical guide to splines*. Springer–Verlag, New York.

[Butzer and Berens, 1967] Butzer, P. L. and Berens, H. (1967). *Semi–Groups of Operators and Approximation*. Grundlehren der mathematischen Wissenschaften. Springer–Verlag.

[Chambolle et al., 2009] Chambolle, A., Caselles, V., Novaga, M., Cremers, D., and Pock, T. (2009). An introduction to total variation for image analysis. *HAL, archives-ouverts*. <hal-00437581>.

[Craven and Wahba, 1979] Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377–403.

[Foley, 1993] Foley, J. (1993). *Introduction to Computer Graphics*. Pearson Academic.

[Foley et al., 1990] Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990). *Computer Graphics*. Addison Wesley, 2nd edition.

[Gelfand and Fomin, 1963] Gelfand, I. M. and Fomin, S. V. (1963). *Calculus of Variations*. Prentice–Hall. Dover reprint, 2000.

[Golub and van Loan, 1996] Golub, G. and van Loan, C. F. (1996). *Matrix Computations*. The Johns Hopkins University Press, 3rd edition.

[Halmos, 1988] Halmos, P. (1988). *I want to be a mathematician. An automathography*. MAA Spectrum Series. Mathematical Association of America.

[Hamming, 1989] Hamming, R. W. (1989). *Digital Filters*. Prentice–Hall. Republished by Dover Publications, 1998.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.

[Isaacson and Keller, 1966] Isaacson, E. and Keller, H. B. (1966). *Analysis of Numerical Methods*. John Wiley & Sons.

[Mallat, 1999] Mallat, S. (1999). *A Wavelet Tour of Signal Processing*. Academic Press, 2. edition.

[Nocedal and Wright, 1999] Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer Series in Operations Research. Springer.

[Peña and Sauer, 2019] Peña, J. M. and Sauer, T. (2019). Update methods for the thin SVD and applications. *Lin. Alg. Appl.*, 561:41–62. arXiv:1809.03285.

[Rockafellar, 1970] Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press.

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, S. (1992). Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268.

[Sauer, 2013a] Sauer, T. (2013a). Einführung in die Numerische Mathematik. Vorlesungsskript, Universität Passau.

[Sauer, 2013b] Sauer, T. (2013b). Optimierung. Vorlesungsskript, Universität Passau.

[Sauer, 2015] Sauer, T. (2015). Learning Theory. Lecture notes, University of Passau.

[Sauer, 2017a] Sauer, T. (2017a). Constructive Approximation. Lecture notes, University of Passau.

[Sauer, 2017b] Sauer, T. (2017b). *Spieltheorie*. Logos–Verlag, Berlin.

[Schüßler, 1992] Schüßler, H. W. (1992). *Digitale Signalverarbeitung*. Springer, 3. edition.

[Spellucci, 1993] Spellucci, P. (1993). *Numerische Verfahren der nichtlinearen Optimierung*. Internationale Schriftenreihe zu Numerischen Mathematik. Birkhäuser.

[Stoer and Bulirsch, 1978] Stoer, J. and Bulirsch, R. (1978). *Einführung in die Numerische Mathematik II*. Heidelberger Taschenbücher. Springer Verlag, 2 edition.

[Stoer and Witzgall, 1970] Stoer, J. and Witzgall, C. (1970). *Convexity and Optimization in Finite Dimensions I*. Springer Verlag.

[Yosida, 1965] Yosida, K. (1965). *Functional Analysis*. Grundlehren der mathematischen Wissenschaften. Springer–Verlag.

[Zhu and Chan, 2008] Zhu, M. and Chan, T. (2008). An efficient primal-dual hybrid gradient algorithm for total variation image restoration. Technical report, CAM Reports 08–34, UCLA, Center for Applied Math.

Index