Image Inpainting
using Nonlinear Partial Differential Equations

Cand. Scient. Thesis in Applied Mathematics

Randi Holm

Department of Mathematics
University of Bergen

May 23, 2005

Preface

The work on this thesis has been an inspiring process. Sometimes it has been frustrating, but all together I have enjoyed it.

First I want to thank my supervisor Xue-Cheng Tai for proposing this interesting topic to me and for supervising me these two years. Special thanks also to Johan Lie for reading the manuscript. He has always been willing to discuss problems with me, and without his help the writing process would have been a lot more difficult.

These two years at the Department of Mathematics has been a pleasant time. I want to thank my fellow students and colleagues for a good atmosphere at work as well as friendly gatherings in the spare time.

Randi Holm
Bergen, May 2005

Contents

Preface iii

Contents v

1 Introduction 1

2 Introduction to image processing 3
- 2.1 Digital image 3
- 2.2 Image inpainting 5
- 2.3 Other fields 8

3 Calculus 13
- 3.1 Level set 13
- 3.2 The gradient vector 14
  - 3.2.1 Geometric properties of the gradient vector 15

4 Functional analysis 17
- 4.1 Definitions 17
- 4.2 Functional derivative 19
- 4.3 Function spaces 19
  - 4.3.1 Inner product spaces 19
  - 4.3.2 Space of all measurable functions 21
  - 4.3.3 Space of continuously differentiable functions 21
  - 4.3.4 Space of functions with Bounded Total Variation 21

5 Calculus of variation 23
- 5.1 Basic idea 23
- 5.2 Minimization problems 25
- 5.3 Lagrange multipliers 26
- 5.4 The method of steepest descent 29
  - 5.4.1 Unique solution 31

6 Image inpainting using a TV-Stokes equation **33**
6.1 The equation for the tangential vectors $\boldsymbol{\tau}$ 33
6.1.1 Discretization 38
6.1.2 Analogy with the Stokes equation 40
6.2 The equation for the image d 41
6.2.1 Discretization 43
6.3 The implementation 44
6.3.1 Editing in Gimp 44
6.3.2 Implementation in Matlab 45
6.3.3 Discussion 49

Bibliography **52**

Appendix **55**

Chapter 1

Introduction

Inpainting is an artistic synonym for image interpolation. The terminology “inpainting” is borrowed from museum restoration artists.

Let D denote the entire image domain. The problem of inpainting is to fill in image information in a blank domain $\Omega \subset D$ based upon the image information available outside Ω, $D\backslash\Omega$. Different techniques can be applied to solve this problem [8, 7, 9, 11, 4, 17, 3, 5, 21].

In this thesis a new method has been tried out. The idea is to find a set of tangential vectors $\boldsymbol{\tau}$ to the level curves of the image d in the missing region Ω. These tangential vectors are found such that they have the smallest Total Variation norm over Ω,

$$\min_{\boldsymbol{\tau}} \int_{\Omega} |\nabla \boldsymbol{\tau}| \,\mathrm{d}\mathbf{x} \qquad \text{subject to} \qquad \nabla \cdot \boldsymbol{\tau} = 0 \qquad \text{in } \Omega. \tag{1.1}$$

The constraint $\nabla \cdot \boldsymbol{\tau} = 0$ assures us that $\boldsymbol{\tau}$ are tangential vectors to the level curves of d. From the tangential vector field we find a surface d that fits $\boldsymbol{\tau}$ in Ω. This surface is found by

$$\min_{d} \int_{\Omega} \left(|\nabla d| - \nabla d \cdot \frac{\mathbf{n}}{|\mathbf{n}|} \right) \mathrm{d}\mathbf{x} \qquad \text{in } \Omega, \tag{1.2}$$

where $\mathbf{n}$ are the normal vectors found from the tangential vectors $\boldsymbol{\tau}$.

Two different boundary conditions are used. In most of our experiments we have used Dirichlet conditions, i.e. $\boldsymbol{\tau} = \boldsymbol{\tau}_0$ in (1.1) and $d = d_0$ in (1.2) on the boundary around Ω, $\partial\Omega$. In the other experiments we have decomposed the boundary into two parts, $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, and used different boundary conditions on $\partial\Omega_N$. To avoid information from propagating through $\partial\Omega_N$, we have used the Neumann boundary conditions $\frac{\partial d}{\partial \boldsymbol{\nu}} = \mathbf{0}$ on $\partial\Omega_N$ in (1.2). The corresponding boundary condition for (1.1) is $\boldsymbol{\tau} = \mathbf{0}$ on $\partial\Omega_N$.

First order optimality of the problems (1.1) and (1.2) leads to two nonlinear partial differential equations that we solve using the method of steepest descent.

This thesis will be organized in the following way: The next chapter gives an overview over different fields in image processing. The three following chapters introduce the theory that is necessary to solve our problem. Level sets and gradients are defined, some important theory from the functional analysis is introduced and a procedure to solve minimization problems is presented. In the last chapter we use this theory to solve (1.1) and (1.2) with the given boundary conditions. The appendix contains an article we have written on this method. The article includes the results and a discussion of these.

Chapter 2

Introduction to image processing

One basic issue in image processing is to restore degraded images. Since the popularity of digital images has increased, this field is of great interest, both for people in general and as a research area.

A general image processing problem can be modelled as

$$d_0 \quad \rightarrow \quad \boxed{\text{Image Processor}} \quad \rightarrow \quad d.$$

There are various approaches to image processing, such as stochastic modelling, wavelets and partial differential equation (PDE) approaches [2]. The last two decades or so PDE methods have been very popular.

Image processing includes a vast number of fields. Before going deeper into some of the fields, we need a definition of a digital image.

2.1 Digital image

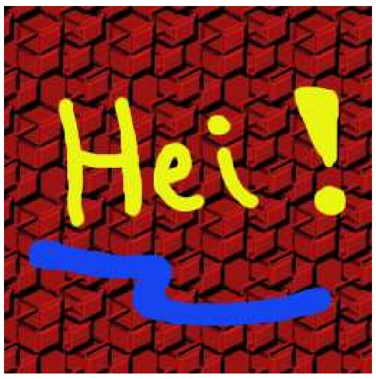
A two-dimensional digital image is a representation of an image as a finite set of digital values, called pixels. Each pixel is associated to a specific position x_{ij} in the image and has an intensity value $d(x_{ij})$ with information about the colour and intensity at the point x_{ij}. An image is usually stored as a matrix with integer entries between 0 and 255, where 0 is the weakest intensity and 255 is the strongest.

Gray-scale images are composed of shades of gray. Each pixel has a single value that tells us the intensity, varying from 0 corresponding to black and 255 corresponding to white.

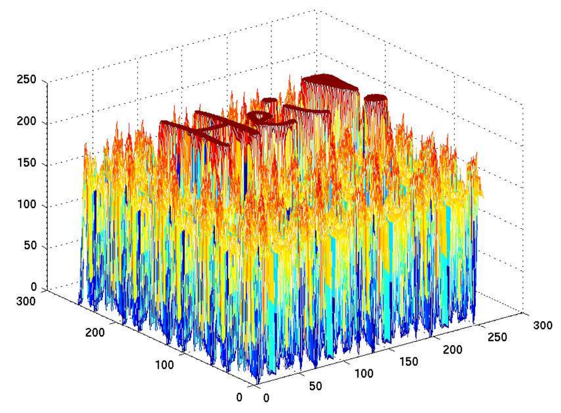
The RGB colour model is one possible model in which to represent a colour image. Shades of red, green and blue are combined to create other colours. A pixel in a color image then consists of three integer values between 0 and 255. The values define the intensity of red, green and blue, respectively. It is possible to represent $256^3 = 16777216$ different colours in this model.

The quality of an image depends on the resolution. The resolution is the number of rows and columns in the image. In an image with high resolution the pixels are so small that they are not visible one by one, but together they form a smooth image. An image with low resolution has few pixels and does not contain as much details as an image with high resolution. The closer together the pixels are, the better the image will represent the real world.

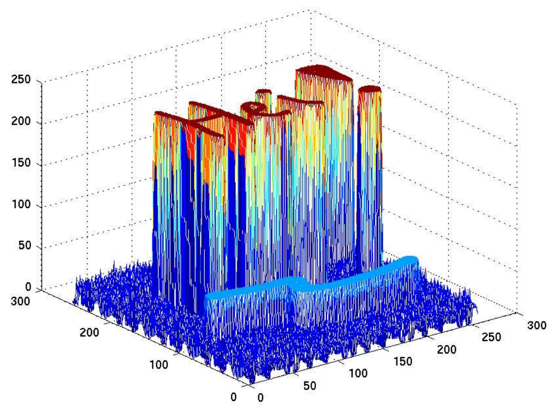
An image can be viewed as a landscape with valleys and peaks. High intensities correspond to peaks and low intensities to valleys. This is illustrated in Figure 2.1. Images are generally not smooth, but contain edges and discontinuities.



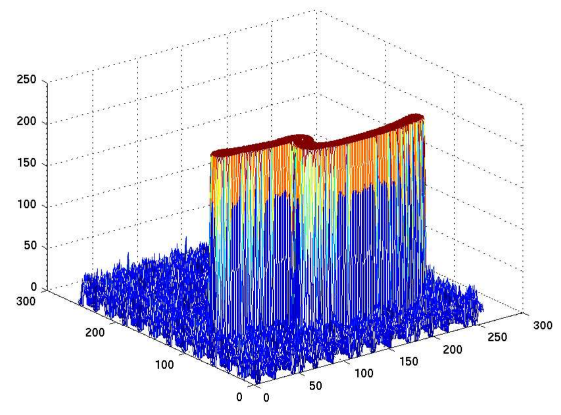
a) A colour image.



b) The matrix for the colour red.



c) The matrix for the colour green.



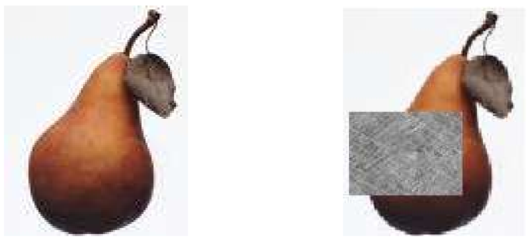
b) The matrix for the colour blue.

Figure 2.1: An example image illustrating the RGB colour model.

2.2 Image inpainting

If a part of an image is damaged or missing, a desirable task can be reconstructing it in a way that looks natural for the eye. This can be done by using information from surrounding areas and merge the inpainted region into the image so seamlessly that a typical viewer is not aware of it. The quality of the result will depend on what is missing. If the inpainting region is small and the surrounding area is without much texture, the result will be good. Texture is a measure of image coarseness, smoothness and regularity. Images with texture contain regions characterized more by variation in the intensity values than by one value fo intensity. The background of the image in Figure 2.1 is an example of texture. Large areas with lots of information lost are harder to reconstruct, because information in other parts of the image is not enough to get an impression of what is missing. If the human brain is not able to imagine what is missing, equations will not make it either.

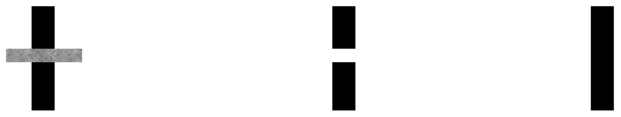
When we look at the real world we see many objects, but we do not see each object one by one. Parts of them are occluded by others all the time. Still we have an intuitive understanding of what is covered. This is because we have seen similar objects before and know what they look like. In Figure 2.2 b) a part of the image is occluded. Because we know the shape and the colour of the pear, we can imagine what is missing.



a) The original image. b) A part of the image is occluded.

Figure 2.2: Humans have an intuitive understanding of what is missing. Computers do not.

The human brain often prefers the connected structures. If we show the occluded object in Figure 2.3 a) to someone, they would most likely guess that the object in Figure 2.3 c) is the original object.



a) A narrow occlusion. b) Alternative 1. c) Alternative 2.

Figure 2.3: Illustration of the Connectivity Principle with a narrow occlusion.

Even though the remaining parts are separated far apart by the occlusion, like in Figure 2.4 a), most of us would prefer the result in Figure 2.4 c). This is referred to as the Connectivity Principle in human perception and is discussed in [9].

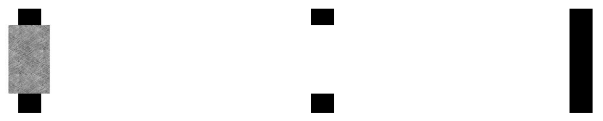

a) A broad occlusion. b) Alternative 1. c) Alternative 2.

Figure 2.4: Illustration of the Connectivity Principle with a broad occlusion.

Removing occlusions is analogous to inpainting, since the region to be inpainted can be considered as an occluding object. It is a challenge to make an algorithm that can do the inpainting in the same way as humans.

Let

$$d_0 : \mathbb{R}^2 \to \mathbb{R} \tag{2.1}$$

be a given observed image on the domain D. In $\Omega \subset D$ the image data is missing or inaccessible. We want to recover the region $\Omega \subset D$ based on information from d_0 on $\partial\Omega$ or in the surrounding band B. This is illustrated in Figure 2.5.

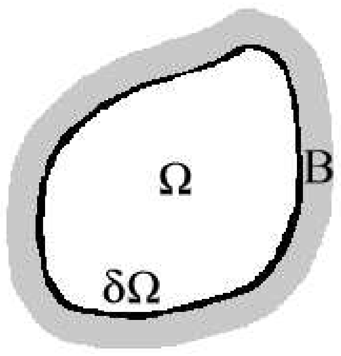


Figure 2.5: The inpainting domain Ω with the boundary $\partial\Omega$ and a surrounding band B.

A common approach to image restoration is the Bayesian framework [21, 13]. We want to find the best guess d based on the image d_0. This starts out with Bayes' formula

$$p(d|d_0) = \frac{p(d_0|d)p(d)}{p(d_0)}, \tag{2.2}$$

where $p(a|b)$ is the probability of a given b, and $p(a)$ is the probability of a. Since d_0 is given, $p(d_0)$ is a constant. $p(d)$ is found from a prior model, and $p(d_0|d)$ is found from a data model.

The goal is to imitate the way professional artists would restore a painting. In [4] the underlying methodology is gathered in four points:

1. The global picture determines how to fill in the gap, the purpose of inpainting being to restore the unity of the work.

2. The structure of the area surrounding Ω is continued into the gap, contour lines are drawn via the prolongation of those arriving at $\partial\Omega$.

3. The different regions inside Ω, as defined by the contour lines, are filled with colours, matching those of $\partial\Omega$.

4. The small details are painted, in other words, texture is added.

There are several approaches to this. Which method is preferable, depends on the information missing and how large the inpainting domain Ω is. The methods can mainly be classified into two classes.

- Texture synthesis algorithms sample textures from the region outside Ω, $D\backslash\Omega$, and use this to fill in the missing region Ω. These algorithms focus on reconstruction of large regions Ω that contain much pattern.

- Inpainting algorithms try to recreate the structures like lines and object contours in Ω. Delicate small scale features like texture will not be recreated, but if the region Ω is small, this may not be noticeable at first sight.

Many interesting articles have been written on this subject.

- In [17, 5] the two classes mentioned above are combined. The image d_0 is decomposed into a sum of two parts, $d_0 = f + g$, where f is the cartoon part capturing the basic image structures and g is the texture and noise part. Then an inpainting algorithm can be used on the first term, and a texture synthesis algorithm can be used on the second term.

- In [3] Naviers-Stokes equations for incompressible, viscous fluids are used.

$$\begin{aligned} \mathbf{u}_t + \mathbf{u} \cdot \nabla \cdot \mathbf{u} - \Delta \mathbf{u} &= -\nabla p \qquad \text{in } \Omega, \\ \nabla \cdot \mathbf{u} &= 0 \qquad\quad\ \text{in } \Omega. \end{aligned} \tag{2.3}$$

These equations describe the relationship between the velocity $\mathbf{u}$ and pressure p in a fluid. The stream function can be compared to the image intensity function d and the fluid velocity to the isophote direction $\nabla^{\perp} d$. This approach has the advantage of well-developed theoretical as well as numerical results. The solution of this equation will have continuous isophotes and produces an image intensity function d that is continuous across $\partial\Omega$.

- In [7, 9] the Total Variation norm of the image d is minimized inside the inpainting region Ω. This leads to the partial differential equation

$$\begin{aligned} \nabla \cdot \left(\frac{\nabla d}{|\nabla d|} \right) &= 0 \qquad \text{in } \Omega, \\ d &= d_0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{2.4}$$

  The drawback of this method is that the interpolation is limited to creating straight isophotes, not necessarily smoothly continued from the boundary $\partial\Omega$. This may look natural if Ω is small, but for large missing regions better algorithms are needed. Another drawback is that it does not always follow the Connectivity Principle illustrated in Figure 2.3 and Figure 2.4.

- In [9] Curvature-Driven Diffusion is used to overcome the problem with the Connectivity Principle. The model for this is

$$\begin{aligned} \nabla \cdot \left(\frac{g(\kappa)}{|\nabla d|} \nabla d \right) &= 0 \qquad \text{in } \Omega, \\ d &= d_0 \qquad \text{on } \partial\Omega \end{aligned} \tag{2.5}$$

  where $g : \mathbb{R} \to [0, \infty)$ is a continuous function satisfying $g(0) = 0$ and $g(\pm\infty) = \infty$ and $\kappa = \nabla \cdot \left(\frac{\nabla d}{|\nabla d|} \right)$ is the curvature of the level curves. Since $\frac{g(\kappa)}{|\nabla d|}$ denotes the diffusivity coefficient, this model will penalize large curvatures and encourage small ones.

Image inpainting can be used to remove text and make objects disappear from an image. It can also be used to restore old photos by removing cracks. Later in this work we will show some examples of image inpainting.

2.3 Other fields

In this section I have used [1, 2, 16].

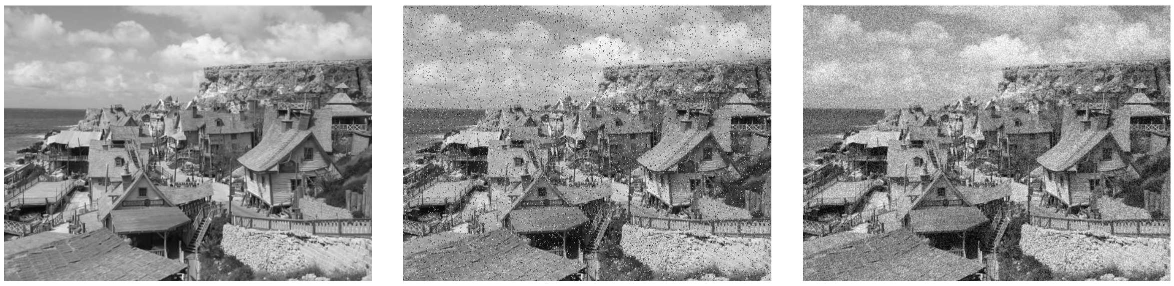
Noise reduction: We say that an image contains noise if the pixel values in the image do not reflect the true intensities of the real scene. Denoising is different from inpainting in the way that we do not have large regions of missing data.

We can write the observed image d_0 as a sum of the true image d and the additive noise n,

$$d_0(x_{ij}) = d(x_{ij}) + n(x_{ij}) \qquad \text{for } x_{ij} \in \Omega, \tag{2.6}$$

where Ω is the domain with noise. Noise can be introduced into an image in several ways depending on how the image is created. There are mainly two models of noise:

- If a few pixels in an image are meaningless, the image contains random noise. These pixels have colours that are totally different from the neighbouring pixels. This can be a result of dust on the lens. Figure 2.6 b) shows an image with a type of random noise called salt and pepper noise.
- In the second model each pixel contains a small amount of noise. This is called Gaussian noise. A plot of the distortion of a pixel against the frequency of which it occurs will look like a Gaussian distribution. Then the smallest amount of noise will occur most often. This is shown in Figure 2.6 c).



a) The original image d. b) Salt and pepper noise. c) Gaussian noise.

Figure 2.6: Different types of noise introduced into the original image d.

There are several methods for removing noise, that is recovering d from d_0. A good algorithm must preserve edges and details in the picture during the denoising.

- Linear filters are advantageous in the way that they do not require much computational time. The intensity values of the reconstructed pixels $d(x_{ij})$ are a linear combination of the values of the the observed pixels $d_0(x_{ij})$ in the neighbourhood of x_{ij}. Averaging, Gaussian and Wiener filters belong to this category. In averaging filters each pixel is replaced by the average of the pixels in the neighbourhood. This is useful for removing grain noise from a photograph. The drawback of this method is that edges will be smoothed. Gaussian filters use the standard deviation to decide the degree of smoothing and use a weighted average to calculate the pixels. A Gaussian filter preserves edges better than the averaging filter. Wiener filters use statistical methods to reduce the noise.
- Median filters are similar to averaging filters. But instead of replacing each pixel with the average of its neighbours, this filter replaces it with the median.

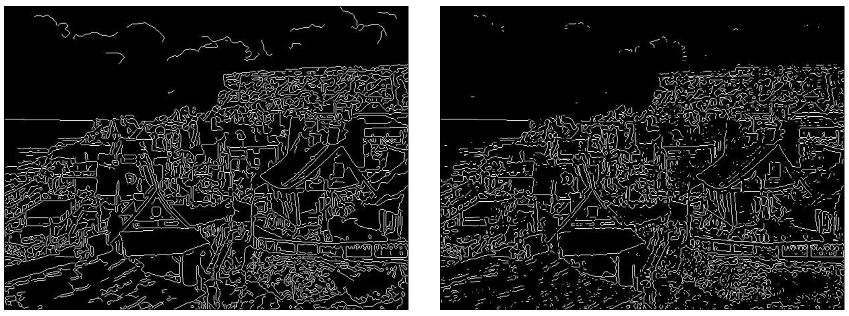
It is better than the averaging filter to remove outliers without reducing the sharpness of the image.

- Adaptive filters are filters that improve the linear methods. If the local variance is found everywhere in an image, the Wiener filter can be improved adaptively. It will perform much smoothing where the variance is small and less smoothing where the variance is large. This method will preserve the edges and parts of the image with high-frequency better than the original Wiener-filter. The drawback of this method is the requirement for more computational time.

Edge detection: The curve that follows a path of rapid change in image intensity is referred to as an edge. An edge detector tries to find the edges in an image. One way of doing this is by locating places in the image where the first derivatives of the intesity values are larger in magnitude than some threshold. If the threshold is small, the edge detector will be sensitive to blurred edges and if the threshold is large, it will only find the sharp edges. A noise reduction algorithm is often applied to the image before the edge detector. The sharp edges will not disappear even if the image is blurred.

- The Canny method looks for edges in places where the gradient of the intensities has a local maximum. It uses two thresholds, one large and one small. The large threshold locates the strong edges and the small locates the weak edges as well. This method is one of the best edge detectors, because it includes a weak edge only if it is connected to a strong edge. Therefore the method will not be sensitive to edges introduced by noise.
- The zero-cross method finds edges by looking for places where the second derivative of the intensity has a zero crossing. This will be in places where the first derivative of the intensity has an extremum.

Figure 2.7 shows these two edge detectors applied to the image in Figure 2.6 a). The result from the Canny-method seems to be the best one.

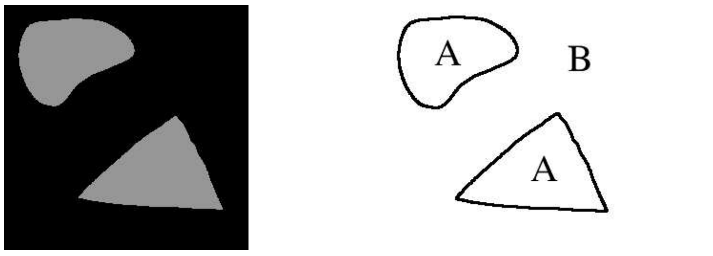


a) The Canny method b) The zero-cross method

Figure 2.7: Two different methods in edge detection.

Segmentation: Segmentation is the partition of an image into different regions. The goal is often to locate a special object or region with common properties as colour or texture. These regions need not to be connected.

Figure 2.8 shows a simple example of segmentation of a gray-scale image. Each pixel is assigned to the class A if $d(x_{ij}) \geq t$ and the class B if $d(x_{ij}) < t$, where t is the chosen threshold.



a) A gray-scale image. b) Segmentation of an image into two classes.

Figure 2.8: Segmentation of a gray-scale image.

Deblurring: An image that is out of focus is referred to as blurred. If it is known how the image blurred, it can be possible to deblur it. The blurring of an image can be caused by movement during the image capture process, i.e. either the camera or the subject was moving. It also occurs if the camera is out of focus.

A blurred image d_0 can be described by the equation

$$d_0(x_{ij}) = Hd(x_{ij}) + n(x_{ij}), \tag{2.7}$$

where H is the distortion operator, d is the original true image and n is additive noise. Note that recovering d from d_0 can be an ill-posed inverse problem [10, 14].

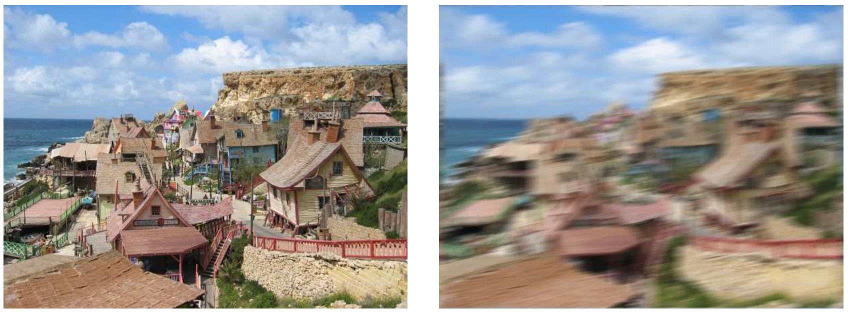
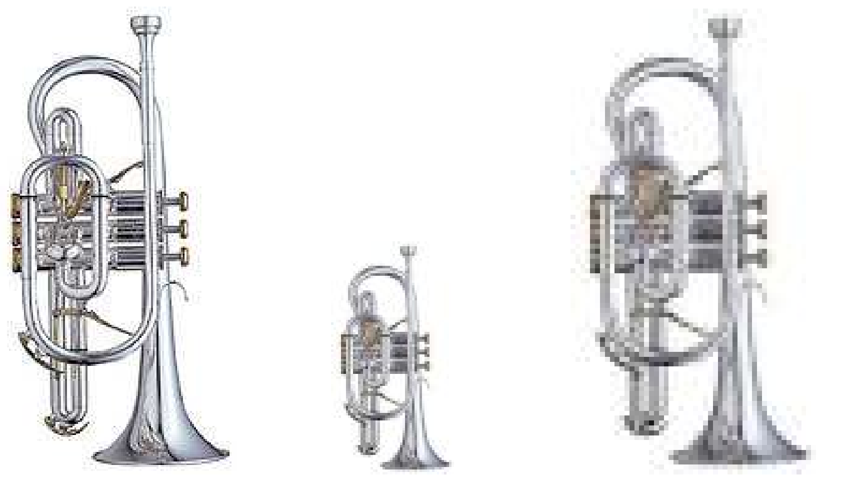

a) The original image d. b) The blurred image d_0.

Figure 2.9: Blurring of an image.

Zoom-in: When an image with low resolution is magnified, the pixels may be so large that they can be seen as squares. To improve the result it is possible to interpolate between the pixels to make the crossings between them smoother. This is called zoom-in.

In Figure 2.10 the resolution of the original image is reduced. If the image with low resolution is smaller than the original it is not visible that the crossings between the pixels are not smooth. But when it is shown in the original size, we see that there is a need for zoom-in methods.



a) My cornet (131 x 250 pixels). b) A small cornet (50 x 95 pixels). c) A magnified cornet (50 x 95 pixels).

Figure 2.10: Images with different resolutions and sizes.

Chapter 3

Calculus

3.1 Level set

A level set of a differentiable function $f : R^n \rightarrow R$ corresponding to the real value c is the set of points

$$\{(x_1, ..., x_n) \in \mathbb{R}^n \ : \ f(x_1, ...x_n) = c\}. \tag{3.1}$$

That is, the set where the function f has a given constant value c. If $n = 1$, the level set is one or more points, depending on how many solutions the function $f(x) = c$ has for the given c. If $n = 2$, the level set is a plane curve and is called a level curve. If $n = 3$, the level set is a surface and is called a level surface. For $n \geq 4$ the level set is a hyper surface, which is a generalization of a surface.

Figure 3.1 shows a surface f together with its level curves projected down to the xy-plane.

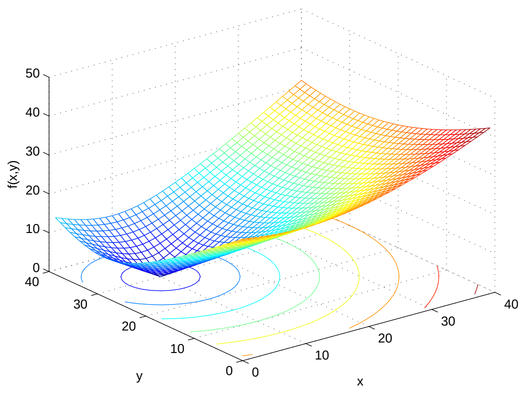


Figure 3.1: A plot of the function $f(x, y) = \sqrt{(x - 10)^2 + (y - 30)^2}$ together with its level curves $\sqrt{(x - 10)^2 + (y - 30)^2} = c$ projected down to the xy-plane.

3.2 The gradient vector

In this work we only need the following definitions and theorems for functions of two variables. Generalizations to functions of more variables are straight forward and can be found in [20].

Definition 1 (Differentiable) *We say that the function $f(x,y)$ is differentiable at the point (x_0,y_0) if*

$$\lim_{(h,k)\to(0,0)} \frac{f(x_0+h,y_0+k)-f(x_0,y_0)-h\frac{\partial f}{\partial x}(x_0,y_0)-k\frac{\partial f}{\partial y}(x_0,y_0)}{\sqrt{h^2+k^2}} = 0. \tag{3.2}$$

The function $f(x,y)$ is differentiable at the point (x_0,y_0) if and only if the surface $z=f(x,y)$ has a non-vertical tangent plane at (x_0,y_0). This implies that the first partial derivatives of f exist and that f is continuous at (x_0,y_0).

Definition 2 *At any point where the first partial derivatives of the function $f(x,y)$ exist, we define the gradient vector $\nabla f(x,y)$ by*

$$\nabla f(x,y) = \frac{\partial f}{\partial x}(x,y)\mathbf{i} + \frac{\partial f}{\partial y}(x,y)\mathbf{j} \tag{3.3}$$

and the perpendicular gradient vector $\nabla^{\perp} f$ by

$$\nabla^{\perp} f(x,y) = -\frac{\partial f}{\partial y}(x,y)\mathbf{i} + \frac{\partial f}{\partial x}(x,y)\mathbf{j} \tag{3.4}$$

The relationship between the level curves of f and the gradient vector ∇f is stated in the following theorem.

Theorem 1 *If $f(x,y)$ is differentiable at the point (x_0,y_0) and $\nabla f(x_0,y_0) \neq \mathbf{0}$, then $\nabla f(x_0,y_0)$ is a normal vector to the level curve of f that passes through (x_0,y_0).*

The proof of this theorem can be found in [18].

3.2.1 Geometric properties of the gradient vector

- At (x_0, y_0), $f(x, y)$ increases most rapidly in the direction of the gradient vector $\nabla f(x_0, y_0)$. The maximum rate of increase is $|\nabla f(x_0, y_0)|$.
- At (x_0, y_0), $f(x, y)$ decreases most rapidly in the direction of $-\nabla f(x_0, y_0)$. The maximum rate of decrease is $|\nabla f(x_0, y_0)|$.
- The rate of change of $f(x, y)$ at (x_0, y_0) is zero in directions tangent to the level curve of f that passes through (x_0, y_0).

In Figure 3.2 the level curves of the function f in Figure 3.1 are plotted together with the gradient vectors ∇f. We see that the gradients are perpendicular to the level curves and point in the directions where f increases the most.

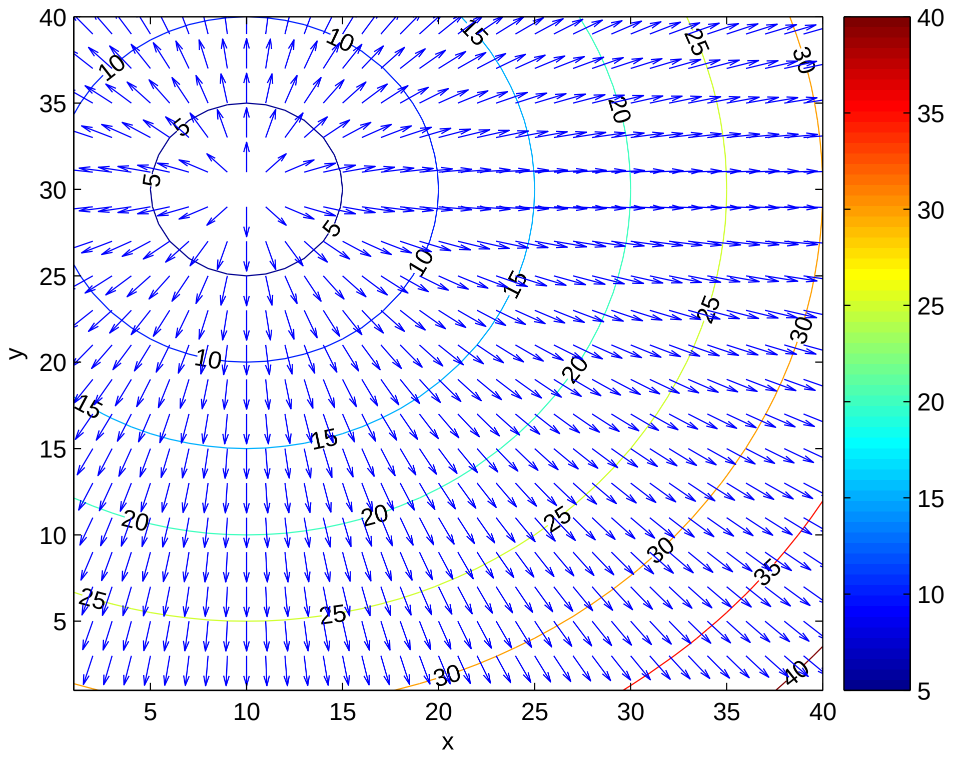


Figure 3.2: The level curves $\sqrt{(x-10)^2 + (y-30)^2} = c$ and the normal vectors $\nabla f = \left(\frac{x-10}{\sqrt{(x-10)^2+(y-30)^2}}, \frac{y-30}{\sqrt{(x-10)^2+(y-30)^2}}\right)$.

Chapter 4

Functional analysis

In order to study the minimization problems, theory from the functional analysis must be used. We will use a definition of a functional derivative where the functionals are defined on a Banach space. To define what a Banach space is, some other definitions are required.

4.1 Definitions

Definition 3 (Metric space) *A metric space is a pair (M, d), where M is a set and d is a metric on M (or distance function on M), that is, a function defined on $M \times M$ such that for all x, y, $z \in M$ we have:*

1. *d is real-valued, finite and nonnegative,*
2. *$d(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$,*
3. *$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$,*
4. *$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ (The triangle inequality).*

The most common metric is the Euclidean metric. In n-space this is defined as $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$. For $n \leq 3$ this metric resembles the intuitive understanding of distance.

Definition 4 (Cauchy sequence) *A Cauchy sequence is a sequence in a metric space $(M,\ d)$ such that for every positive real number $\epsilon > 0$, there exists an integer N such that for all integers $m, n > N$, the distance $d(\mathbf{x}^m,\ \mathbf{x}^n) < \epsilon$.*

Thus a Cauchy sequence is a sequence whose terms become arbitrarily close to each other as m, $n \to \infty$. This does not imply that the sequence has a limit in M.

Definition 5 (Complete metric space) *$(M,\ d)$ is a complete metric space if every Cauchy sequence in M converges to a limit in M.*

This implies that every convergent sequence in a metric space is a Cauchy sequence.

An example of a complete metric space is the space of the real numbers $\mathbb{R}$ with respect to its natural metric $d(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \mathbf{y}$.

The rational numbers are not complete. π is the limit of a sequence of rational numbers, but π itself is irrational. Leibniz' expressed π as an infinite sum

$$\pi = 4 \sum_{n=0}^{\infty} (-1)^n \frac{1}{2n+1}. \tag{4.1}$$

Figure 4.1 shows how this sum converges to π.

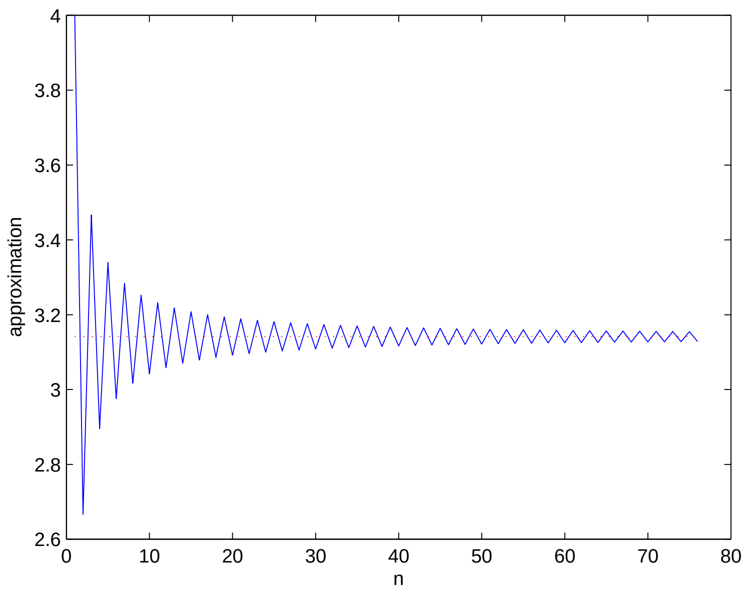


Figure 4.1: The sum in (4.1) (blue line) and the exact value of π (red dotted line).

Definition 6 *A nonempty set $X \subset \mathbb{R}^n$ is a vector space if $\mathbf{x} + \mathbf{y} \in X$ and $c\mathbf{x} \in X$ for all $\mathbf{x} \in X$, $\mathbf{y} \in X$, and for all scalars c.*

Definition 7 *Let E be a real vector space. A norm on E is a mapping $\|\cdot\| : E \to [0, \infty)$ which satisfies the following conditions:*

- $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ *for all* $\mathbf{x}, \mathbf{y} \in E$,
- $\|c\mathbf{x}\| = |c| \; \|\mathbf{x}\|$ *for all* $\mathbf{x} \in E$, $c \in \mathbb{R}$,
- $\|\mathbf{x}\| = 0$ *iff* $\mathbf{x} = \mathbf{0}$.

A normed space is a pair $(E, \|\cdot\|)$ where E is a real vector space and $\|\cdot\|$ is a norm on E.

The norm induces a distance function $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$. This means that E is a metric space.

A norm provides us with a notion of convergence: We say a sequence $\{\mathbf{x}^k\}_{k=1}^{\infty} \subset X$ converges to $\mathbf{x} \in X$, written $\mathbf{x}^k \to \mathbf{x}$, if $\lim\limits_{k\to\infty} \|\mathbf{x}^k - \mathbf{x}\| = 0$.

Definition 8 (Banach space) *A Banach space is a normed space which is a complete metric space with respect to the metric induced by its norm.*

Then a Banach space is a normed vector space which is complete, that is, within which each Cauchy sequence converges.

4.2 Functional derivative

A functional is a real-valued function on a vector space V, usually of functions. The functional derivative is the derivative of a functional with respect to a function and is a generalization of the function derivative. It tells us how the functional changes when the function changes by a small amount.

In this work the space V can be restricted to be a Banach space. A functional on a Banach space is a a scalar-valued mapping which is continuous, but not necessarily linear.

Let V be a Banach space and let $F : V \to \mathbb{R}$ be a functional. The definition of the functional derivative at $\mathbf{x}$ in the direction of $\mathbf{y}$ is then

$$F'(\mathbf{x})\mathbf{y} = \lim_{\epsilon\to 0} \frac{F(\mathbf{x} + \epsilon\mathbf{y}) - F(\mathbf{x})}{\epsilon}. \tag{4.2}$$

F is (Frechet-)differentiable at $\mathbf{x}$ if

$$F(\mathbf{x} + \mathbf{y}) = F(\mathbf{x}) + F'(\mathbf{x})\mathbf{y} + o(\|\mathbf{y}\|_V) \qquad \text{as } \|\mathbf{y}\|_V \to 0. \tag{4.3}$$

$F'(\mathbf{x})$ will be a bounded linear functional.

The intuition from Calculus carries over to functional derivatives. $\mathbf{x}$ is a critical point of F if $F'(\mathbf{x}) = 0$, that means

$$F'(\mathbf{x})\mathbf{y} = 0 \quad \forall \quad \mathbf{y} \in X. \tag{4.4}$$

This critical point condition is called the Euler-Lagrange equation for the functional F.

4.3 Function spaces

4.3.1 Inner product spaces

Definition 9 *An inner product on a real vector space V is a mapping*

$$(\cdot, \cdot) : V \times V \to \mathbb{R} \tag{4.5}$$

such that, for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and all $\lambda \in \mathbb{R}$, *1.* $(\mathbf{x},\, \mathbf{y}) = (\mathbf{y},\, \mathbf{x})$,

2. $(\mathbf{x},\, \mathbf{y} + \mathbf{z}) = (\mathbf{x},\, \mathbf{y}) + (\mathbf{x},\, \mathbf{z})$,

3. $\lambda(\mathbf{x},\, \mathbf{y}) = (\lambda\mathbf{x},\, \mathbf{y})$,

4. $(\mathbf{x},\, \mathbf{x}) > 0$ *when* $\mathbf{x} \neq \mathbf{0}$,

An inner product space is a pair $(V,\, (\cdot,\, \cdot))$ *where* V *is a real vector space and* $(\cdot,\, \cdot)$ *is an inner product on* V*.*

It follows that $(\mathbf{x},\, \mathbf{x}) = 0$ iff $\mathbf{x} = 0$.

Definition 10 *If* $(\cdot,\, \cdot)$ *is an inner product, the associated norm is*

$$|\mathbf{x}| = (\mathbf{x},\, \mathbf{x})^{\frac{1}{2}}. \tag{4.6}$$

Definition 11 (Hilbert space) *A Hilbert space is an inner product space which is a complete metric space with respect to the metric induced by its inner product.*

We can say that a Hilbert space is a Banach space endowed with an inner product which generates the norm. Every Hilbert space is a Banach space, but there exist plenty of Banach spaces which are not Hilbert spaces.

Euclidean spaces

For each positive integer n, let $\mathbb{R}^n$ be the set of all ordered n-tuples of real numbers, that is

$$\mathbf{x} = (x_1,\, x_2,\, ...,\, x_n), \tag{4.7}$$

where $(x_1,\, x_2,\, ...,\, x_n)$ is a point in n-space. We define the inner product of $\mathbf{x}$ and $\mathbf{y}$ by

$$(\mathbf{x},\, \mathbf{y}) = \sum_{i=1}^{n} x_i y_i, \tag{4.8}$$

and the norm by

$$|\mathbf{x}| = (\mathbf{x},\, \mathbf{x})^{\frac{1}{2}} = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}. \tag{4.9}$$

The vector space $\mathbb{R}^n$ with the above inner product and norm is called Euclidean n-space.

4.3.2 Space of all measurable functions

Assume Ω is an open subset of $\mathbb{R}^n$. We define $L^p(\Omega)$ to be the space of all measurable functions $u : \Omega \to \mathbb{R}$ for which $\|u\|_{L^p(\Omega)} < \infty$,

$$L^p(\Omega) = \{u : \Omega \to \mathbb{R} \ : \ \text{u is Lebesgue measurable, } \|u\|_{L^p(\Omega)} < \infty\}, \tag{4.10}$$

where

$$\|u\|_{L^p(\Omega)} = \begin{cases} \left(\int_\Omega |u|^p \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}} & \text{if } 1 \le p < \infty, \\ \operatorname{ess} \sup\limits_{\Omega} \ |u| & \text{if } p = \infty. \end{cases} \tag{4.11}$$

4.3.3 Space of continuously differentiable functions

Assume Ω is an open subset of $\mathbb{R}^n$. We define $C^k(\Omega)$ to be the space of k-times continuously differentiable functions,

$$C^k(\Omega) = \{u : \Omega \to \mathbb{R} \ : \ u \text{ is } k\text{-times continuously differentiable}\}. \tag{4.12}$$

The support of a continuous function $u(x)$ defined on $\mathbb{R}^n$ is the closure of the set of points where $u(x)$ is non-zero, that is

$$\operatorname{supp} u = \overline{\{x \in \mathbb{R}^n \ : \ u(x) \neq 0\}}. \tag{4.13}$$

A set is bounded if it is contained in a ball $B_R(0)$ with R sufficiently large. If supp u is bounded, we say that u has compact support. $C_0^k(\Omega)$ denotes the functions in $C^k(\Omega)$ with compact support.

4.3.4 Space of functions with Bounded Total Variation

In the image processing community the Total Variation norm was introduced in order to be able to preserve edges in an image during the regularization. Let $\Omega \subseteq \mathbb{R}^n$ be a bounded open set. The Total Variation norm is the L^1-norm of the gradient and can be defined as

$$\|u\|_{TV(\Omega)} = \int_\Omega |\nabla u| \, \mathrm{d}\mathbf{x} \qquad \text{for } u \in C^1(\Omega). \tag{4.14}$$

This norm is a measure of the amount of oscillation found in the function u. A more rigorous definition is based on the dual form

$$\|u\|_{TV(\Omega)} = \sup \left\{ \int_\Omega u \cdot \nabla \mathbf{g} \, \mathrm{d}\mathbf{x} \ : \ \mathbf{g} \in C_0^1(\Omega, \mathbb{R}^n), \ |\mathbf{g}(\mathbf{x})| \le 1 \ \forall \ \mathbf{x} \in \Omega \right\}, \tag{4.15}$$

where $u \in L^1(\Omega)$. This is basically the space of functions with Bounded Total Variation, $BV(\Omega)$. The definition of this space is

$$BV(\Omega) = \{u \in L^1(\Omega) \ : \ \|u\|_{TV} < \infty\}. \tag{4.16}$$

$BV(\Omega)$ is a Banach space endowed with the norm

$$\|u\|_{BV(\Omega)} = \|u\|_{L^1(\Omega)} + \|u\|_{TV(\Omega)}. \tag{4.17}$$

Thus BV-functions are the L^1 functions with bounded TV-norm, and discontinuous functions are included in this space. The advantage of the TV-norm is that it allows for discontinuities, but at the same time disfavours oscillations, such as noise.

$BV_0(\Omega)$ denotes the functions belonging to $BV(\Omega)$ whose support is a compact subset of Ω.

Chapter 5

Calculus of variation

Calculus of variations is essentially a generalization of ordinary calculus [12]. It is a field which deals with finding extremas of functionals.

5.1 Basic idea

The idea is to solve a partial differential equation by finding the minimum of a functional. A partial differential equation is an equation involving partial derivatives of an unknown function with respect to more than one independent variable.

We want to solve the partial differential equation

$$A[u] = 0, \tag{5.1}$$

where $A[\cdot]$ denotes a given, possibly nonlinear, partial differential operator and u is the unknown. If the operator $A[\cdot]$ is the derivative of an energy functional $I[\cdot]$ we can write

$$A[\cdot] = I'[\cdot]. \tag{5.2}$$

By solving equation (5.1) we find the critical points of $I[\cdot]$,

$$I'[u] = 0. \tag{5.3}$$

It may be easier to find a critical point of $I[u]$ than solving the partial differential equation (5.1). $I[u]$ is typically formed as an integral involving the unknown function u and some of its derivatives.

Suppose that $\Omega \subset \mathbb{R}$ is a bounded, open set with smooth boundary $\partial\Omega$ and $L : \mathbb{R}^n \times \mathbb{R} \times \overline{\Omega} \to \mathbb{R}$ is a smooth function. We will write

$$L = L(\mathbf{p},\, z,\, \mathbf{x}) = L(p_1,\, ...,\, p_n,\, z,\, x_1,\, ...,\, x_n) \tag{5.4}$$

for $\mathbf{p} \in \mathbb{R}^n$, $z \in \mathbb{R}$ and $\mathbf{x} \in \Omega$ and use the notation

$$\begin{aligned} D_{\mathbf{p}}L &= (L_{p_1}, ..., L_{p_n}), \\ D_z L &= L_z, \\ D_{\mathbf{x}}L &= (L_{x_1}, ..., L_{x_n}). \end{aligned} \tag{5.5}$$

Assume that $I[\cdot]$ has the explicit form

$$I[w] = \int_\Omega L(\nabla w(\mathbf{x}), w(\mathbf{x}), \mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{5.6}$$

for smooth functions $w : \overline{\Omega} \to \mathbb{R}$ satisfying a boundary condition

$$w = g \qquad \text{on } \partial\Omega. \tag{5.7}$$

We suppose that among all w satisfying the boundary condition (5.7) there exists a function u that is a minimizer of $I[\cdot]$. This function u will be the solution of a certain nonlinear partial differential equation.

To show this we consider the real-valued function

$$i(h) = I[u + hv], \tag{5.8}$$

for $h \in \mathbb{R}$ and $v \in C_0^\infty$.

Since u is a minimizer of $I[\cdot]$ and $u + hv = u = g$ on $\partial\Omega$, $i(\cdot)$ has a minimum at $h = 0$, therefore

$$i'(0) = I'[u]v = 0. \tag{5.9}$$

As stated in Section 4.2, $I'[u]v$ is the derivative of I in the direction of v. u is a critical point of I and (5.9) is called the Euler-Lagrange equation associated with the functional I.

Now we want to compute the derivative of

$$i(h) = \int_\Omega L(\nabla u + h\nabla v,\, u + hv,\, \mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{5.10}$$

Using the Chain Rule we get

$$i'(h) = \int_\Omega \sum_{i=0}^{n} L_{p_i}(\nabla u + h\nabla v, u + hv, \mathbf{x})v_{x_i} + L_z(\nabla u + h\nabla v, u + hv, \mathbf{x})v\,\mathrm{d}\mathbf{x}. \tag{5.11}$$

Since $h = 0$ at a minimum, we deduce that

$$0 = i'(0) = \int_\Omega \sum_{i=0}^{n} L_{p_i}(\nabla u, u, \mathbf{x})v_{x_i} + L_z(\nabla u, u, \mathbf{x})v\,\mathrm{d}\mathbf{x}. \tag{5.12}$$

Theorem 2 (Integration-by-parts formula) *Assume that Ω is a bounded, open subset of $\mathbb{R}^n$ and $\partial\Omega$ is C^1. Let u, $v \in C^1(\overline{\Omega})$. Then*

$$\int_\Omega u_{x_i} v\,\mathrm{d}\mathbf{x} = -\int_\Omega u v_{x_i}\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} u v \nu^i\,\mathrm{d}\mathbf{S} \qquad (i = 1, ..., n), \tag{5.13}$$

where $\boldsymbol{\nu} = (\nu^1, ..., \nu^n)$ is the outward pointing unit normal vector field along $\partial\Omega$.

By Theorem 2 we obtain

$$0 = \int_\Omega \left[-\sum_{i=0}^{n} (L_{p_i}(\nabla u, u, \mathbf{x}))_{x_i} + L_z(\nabla u, u, \mathbf{x}) \right] v \, \mathrm{d}\mathbf{x}. \tag{5.14}$$

The boundary integral vanishes since v has compact support. Since (5.14) holds for all test functions v, u solves the nonlinear partial differential equation

$$-\sum_{i=0}^{n} (L_{p_i}(\nabla u, u, \mathbf{x}))_{x_i} + L_z(\nabla u, u, \mathbf{x}) = 0. \tag{5.15}$$

This is a quasi-linear second order partial differential equation in divergence form. Solving this equation is equivalent to searching for a minimum of (5.6).

5.2 Minimization problems

Before going deeper into the minimization problems, we need a few more definitions.

Definition 12 (Local minimum values) *A function f has a local minimum value $f(\mathbf{x}_1)$ at the point $\mathbf{x}_1$ in its domain provided there exists a number $h > 0$ such that $f(\mathbf{x}) \geq f(\mathbf{x}_1)$ whenever $\mathbf{x}$ is in the domain of f and $|\mathbf{x} - \mathbf{x}_1| < h$.*

A function f can have many local minima. This is illustrated in Figure 5.1.

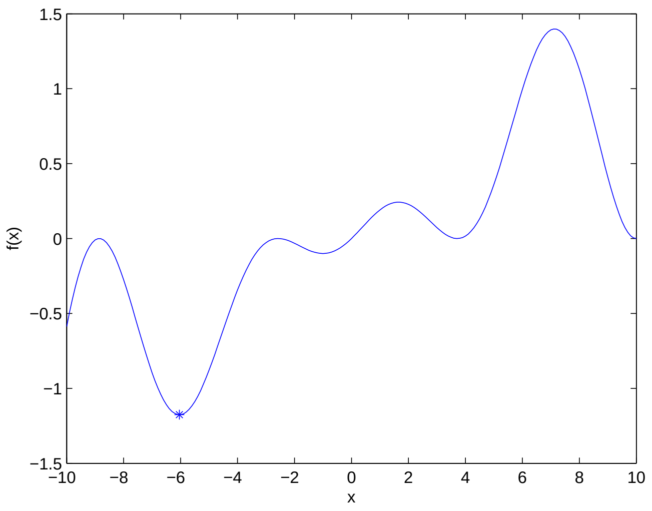


Figure 5.1: A function $f(x)$ with several local minima. The global minimum is shown as $*$.

Definition 13 (Absolute minimum values) *A function f has an absolute minimum value $f(\mathbf{x}_1)$ at the point $\mathbf{x}_1$ in its domain if $f(\mathbf{x}) \geq f(\mathbf{x}_1)$ holds for every $\mathbf{x}$ in the domain of f.*

This means that a function can have at most one absolute maximum or minimum value, though this value can be assumed at many points.

When the global minimum of a function $f(\mathbf{x})$ is to be found, we want to find $\mathbf{x}$ such that $f(\mathbf{x})$ is the smallest possible. A function $f(\mathbf{x})$ can attain its minimum value only at points $\mathbf{x}$ that belong to one of the following classes:

- Critical points of f, that is points $\mathbf{x}$ in the domain of f where $f'(\mathbf{x}) = 0$.
- Singular points of f, that is points $\mathbf{x}$ in the domain of f where $f'(\mathbf{x})$ is not defined.
- Endpoints of the domain of f, that is points $\mathbf{x}$ in the domain of f that do not belong to any open interval contained in the domain of f.

The unconstrained minimization problem is formulated as

$$\min_{\mathbf{x}} f(\mathbf{x}), \tag{5.16}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

Sometimes a constraint must be fulfilled at the minimum. We can for example look for the minimum only at points where $g(\mathbf{x}) = 0$. The constrained minimization problem is formulated as

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad \text{subject to} \qquad g(\mathbf{x}) = 0, \tag{5.17}$$

where $\mathbf{x}$ and f are defined as above and $g : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

5.3 Lagrange multipliers

Lagrange multipliers is a method for finding local extrema of a multivariate function $f(x_1, ..., x_n)$ subject to a constraint $g(x_1, ..., x_n) = 0$. f and g must be functions with continuous first partial derivatives in the open set containing $g(x_1, ..., x_n) = 0$ and $\nabla g \neq 0$ at any point in the open set. The method will not work if the extreme value occurs at an endpoint of the constraint $g(x_1, ..., x_n) = 0$.

The Lagrange function is defined by

$$L(x_1, ..., x_n, \lambda) = f(x_1, ..., x_n) + \lambda g(x_1, ..., x_n), \tag{5.18}$$

where λ is a constant called the Lagrange multiplier. The value of λ tells us something about how hard $L(x_1, ..., x_n, \lambda)$ is pushing or pulling against the constraint $g(x_1, ..., x_n)$. For an extremum to exist

$$df = \frac{\partial f}{\partial x_1} dx_1 + ... + \frac{\partial f}{\partial x_n} dx_n = 0. \tag{5.19}$$

Since g is held constant

$$dg = \frac{\partial g}{\partial x_1}dx_1 + ... + \frac{\partial g}{\partial x_n}dx_n = 0. \tag{5.20}$$

If (5.20) is multiplied by a constant λ and added to (5.19), we get the equation

$$(\frac{\partial f}{\partial x_1} + \lambda\frac{\partial g}{\partial x_1})dx_1 + ... + (\frac{\partial f}{\partial x_1} + \lambda\frac{\partial g}{\partial x_n})dx_n = 0. \tag{5.21}$$

Since all the differentials are independent, it follows that

$$\frac{\partial f}{\partial x_k} + \lambda\frac{\partial g}{\partial x_k} = 0 \quad \forall\, k = 1, ..., n, \tag{5.22}$$

and

$$\nabla f = -\lambda\nabla g. \tag{5.23}$$

This implies that at an extremum ∇f and ∇g are parallel.

To explain why this is correct we take a look at the two-dimensional case. We want to minimize the function $f(x_1, x_2)$ subject to $g(x_1, x_2) = 0$. When we walk along the contour $g(x_1, x_2) = 0$, we must cross the level curves of $f(x_1, x_2)$ in the direction that f decreases most. In a point where f does not decrease in any direction, the level curve of f through that point is parallel to the level curve of the constraint. Thus the gradients are parallel and we have found the minimum.

We illustrate this method by an example. We want to find the minimum of $f(x, y) = 2(x-1)x-y^2-4$ on the line $y = -\frac{3}{2}$. This can be formulated as a constrained minimization problem

$$\min_{x,\,y} 2(x-1)x - y^2 - 4 \qquad \text{subject to} \qquad y + \frac{3}{2} = 0, \tag{5.24}$$

and can be solved by the Lagrange multipliers method. The Lagrange function will be

$$L(x, y, \lambda) = 2(x-1)x - y^2 - 4 + \lambda(y + \frac{3}{2}). \tag{5.25}$$

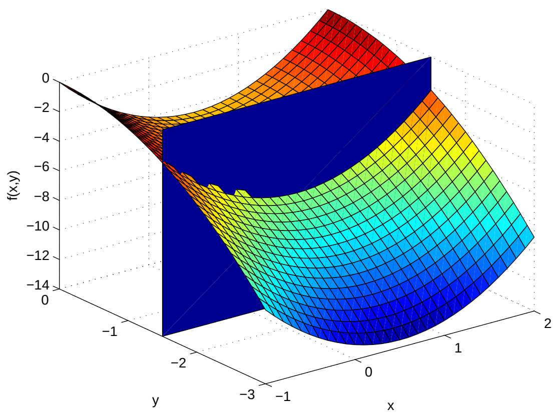
For critical points of L we want

$$\begin{aligned} 0 &= \frac{\partial L}{\partial x} = 4x - 2, \\ 0 &= \frac{\partial L}{\partial y} = -2y + \lambda, \\ 0 &= \frac{\partial L}{\partial \lambda} = y + \frac{3}{2}. \end{aligned} \tag{5.26}$$

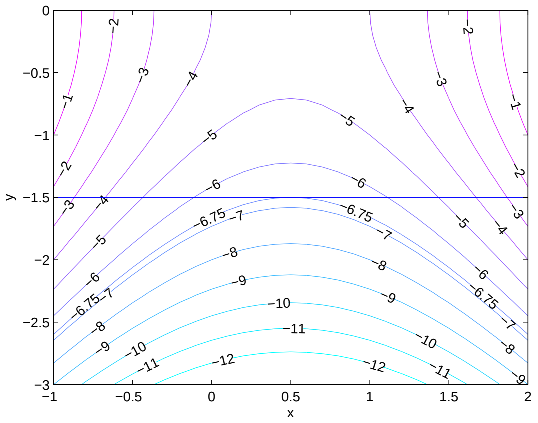
From these equations we find that the critical point must be $(\frac{1}{2}, -\frac{3}{2})$ where $f(\frac{1}{2}, -\frac{3}{2}) = -6\frac{3}{4}$. This is the minimum we searched for.

The Lagrange multipliers can be generalized to deal with multiple constraints $g_1(x_1, ..., x_n) = 0$, ..., $g_m(x_1, ..., x_n) = 0$ for $m < n$. Thus

$$L(x_1, ..., x_n) = f(x_1, ..., x_n) + \lambda_1 g_1(x_1, ..., x_n) + ... + \lambda_m g_m(x_1, ..., x_n), \tag{5.27}$$



a) A plot of the function $f(x,y)$ and the plane $y = -\frac{3}{2}$.



b) The level curves of $f(x,y)$ and the line $y = -\frac{3}{2}$.

Figure 5.2: Finding the minimum of the function $f(x,y) = 2(x-1)x - y^2 - 4$ subject to the constraint $g(x,y) = y + \frac{3}{2} = 0$.

and

$$\nabla f = \lambda_1 \nabla g_1 + ... + \lambda_m \nabla g_m \tag{5.28}$$

is required at the extrema.

Similarly, if $F(u) : V \longrightarrow \mathbb{R}$ and $G(u) : V \longrightarrow \mathbb{R}$ are C^1-functionals on a Banach space V, we can minimize $F(u)$ on the constraint set $C = \{u \in V \ : \ G(u) = 0\}$. The Lagrange functional is defined by

$$L(u, \lambda) = F(u) + \int_\Omega \lambda G(u) \, \mathrm{d}\mathbf{x}, \tag{5.29}$$

where $u : \Omega \longrightarrow \mathbb{R}$ and $\lambda : \Omega \longrightarrow \mathbb{R}$ are functions.

5.4 The method of steepest descent

To find the solution of a minimization problem iterative algorithms can be used. These algorithms start with a given initial guess $\mathbf{x}^0$ and generate a sequence of iterates $\{\mathbf{x}^k\}_{k=0}^{\infty}$ that hopefully will converge to the solution $\mathbf{x}^*$. Which algorithm to use depends on the problem, but all good algorithms should have the following properties:

- Robustness means that the solution is insensitive to the inputs in the algorithm. The algorithm must perform well on a wide variety of problems and for all choices of initial variables. Stability is also necessary. A stable algorithm gives nearly the right answer to nearly the right question.
- Efficiency is important, especially for large problems. If the algorithm requires too much computer time or storage, it may be impossible to solve the problem.
- We say that an algorithm is accurate if the relative error between the mathematical problem $f(\mathbf{x})$ and the computed problem $g(\mathbf{x})$ is small. The relative error is defined as
$$\frac{\|g(\mathbf{x}) - f(\mathbf{x})\|}{\|f(\mathbf{x})\|}. \tag{5.30}$$

These goals may be at the expense of each others. An efficient algorithm is not necessarily the most accurate one.

The method of steepest descent can be used for calculating a local maximum or minimum of a real-valued function $f(\mathbf{x})$. If $f(\mathbf{x})$ is continuous and differentiable in a neighbourhood of a point $\mathbf{x}^0$, $f(\mathbf{x})$ increases fastest from the point $\mathbf{x}^0$ in the direction of $\nabla f(\mathbf{x}^0)$ and decreases fastest in the direction of $-\nabla f(\mathbf{x}^0)$. To find a minimum one can start at a point $\mathbf{x}^0$ and take a small step in the direction of $-\nabla f(\mathbf{x}^0)$ to a new point $\mathbf{x}^1$. This can be done several times by the algorithm

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \nabla f(\mathbf{x}^n), \quad n = 0, 1, 2, ..., \tag{5.31}$$

where $\mathbf{x}^0$ is an initial guess. If Δt is small enough, $\mathbf{x}^n$ gets closer to the minimum as n increases. At a minimum $\mathbf{x}^*$ the gradient $\nabla f(\mathbf{x}^*)$ equals zero and the iterative algorithm has converged.

This method does not need the second derivatives and works in spaces of any dimension. A drawback of the method is that many iterations may be needed before convergence. In general the convergence rate is only linear, that is

$$\lim_{k\to\infty} \frac{\|\mathbf{e}^{k+1}\|}{\|\mathbf{e}^k\|} < 1, \tag{5.32}$$

where $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}^*$. The method is also very sensitive to poor scaling, that is if changes to $\mathbf{x}$ in one direction produce much larger variations in f than changes to $\mathbf{x}$ in other directions.

If the minimizer $\mathbf{x}^*$ lies in a long narrow valley, the convergence is poor. This is because the contours of f near the minimum $\mathbf{x}^*$ tend to highly eccentric ellipses. If we follow the direction of the steepest descent from the initial guess $\mathbf{x}^0$ in Figure 5.3, this will not lead us much closer to the minimum. The convergence can be improved by calculating the optimal Δt in each step. This will take more computational time and conjugate gradients method is often a better alternative.

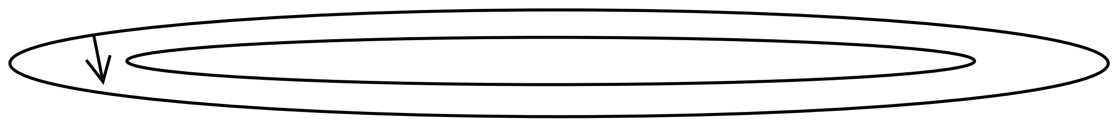


Figure 5.3: The level curves of a poorly scaled problem. The vector points in the steepest descent direction.

5.4.1 Unique solution

A minimization problem can have several local minima. We are interested in finding the best of all such minima, the global minima. The problem is that even if the iterations have converged, we do not know that the solution is a global minimizer.

In some cases the minimizers are simple to characterize.

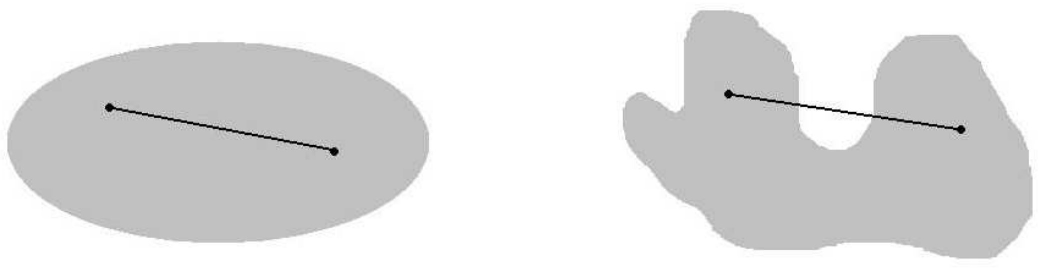
Theorem 3 *When f is convex, any local minimizer $\mathbf{x}^*$ is a global minimizer of f. If in addition f is differentiable, then any stationary point $\mathbf{x}^*$ is a global minimizer.*

The term convex need to be defined.

Definition 14 (Convex set) *$S \in \mathbb{R}^n$ is a convex set if the straight line segment connecting any two points in S lies entirely inside S. Formally, for any two points $\mathbf{x} \in S$ and $\mathbf{y} \in S$, we have*

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in S \qquad \textit{for all } \alpha \in [0,1]. \tag{5.33}$$

This definition is illustrated in Figure 5.4.



a) A convex set. b) A non-convex set.

Figure 5.4: A straight line segment connecting two points in a set.

Definition 15 (Convex function) *f is a convex function if its domain is a convex set and if for any two points $\mathbf{x}$ and $\mathbf{y}$ in this domain, the graph of f lies below the straight line connecting $(\mathbf{x}, f(\mathbf{x}))$ to $(\mathbf{y}, f(\mathbf{y}))$ in the space $\mathbb{R}^{n+1}$. That is, we have*

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}), \qquad \textit{for all } \alpha \in [0,1]. \tag{5.34}$$

This definition is illustrated in Figure 5.5.

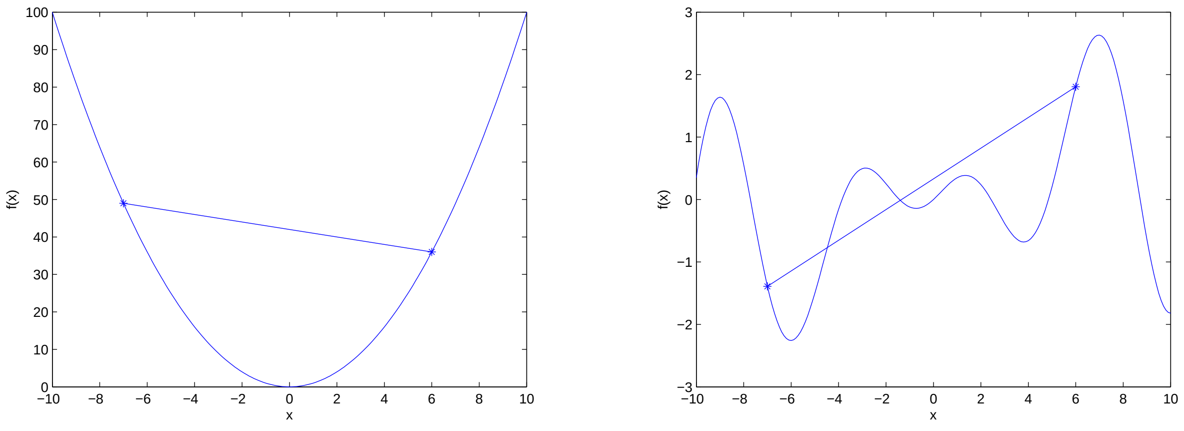

a) A convex function. b) A non-convex function.

Figure 5.5: A straight line segment connecting two points at a function.

Chapter 6

Image inpainting using a TV-Stokes equation

As in Section 2.2 we have an observed image d_0 with a region Ω where the image data is missing. We define B to be a surrounding band of Ω and $\tilde{\Omega} = \Omega \cup B$. This was illustrated in Figure 2.5. Based on information from the surrounding band B we want to reconstruct the image in the region Ω. The reconstructed image is called d.

Our approach is to solve two minimization problems. The first one will find a set of tangential vectors $\boldsymbol{\tau}$ to the level curves of the image d in the missing region Ω. The second will find a surface d that fits these tangential vectors.

A similar approach is used in noise removal [15].

6.1 The equation for the tangential vectors $\boldsymbol{\tau}$

For a given image d, $\boldsymbol{\tau} = \nabla^{\perp} d$ are the tangential vectors to the level curves of d,

$$\boldsymbol{\tau} = \left(-\frac{\partial}{\partial y} d, \frac{\partial}{\partial x} d \right) = (v, u) \,. \tag{6.1}$$

We let $\boldsymbol{\tau}$ belong to the space of bounded variation, $\boldsymbol{\tau} \in BV(\Omega)$. As explained in Section 4.3.4, $\boldsymbol{\tau}$ are then allowed to be a discontinuous function.

To solve the inpainting problem we find the tangential vectors $\boldsymbol{\tau}$ that have the smallest TV-norm over Ω.

We have solved the minimization problem

$$\min_{\boldsymbol{\tau}} \int_{\tilde{\Omega}} |\nabla \boldsymbol{\tau}| \, \mathrm{d}\mathbf{x} + \frac{1}{\epsilon} \int_{B} |\boldsymbol{\tau} - \boldsymbol{\tau_0}|^2 \, \mathrm{d}\mathbf{x} \quad \text{subject to} \qquad \nabla \cdot \boldsymbol{\tau} = 0 \qquad \text{in } \tilde{\Omega}, \tag{6.2}$$

where $| \cdot |$ is the Euclidian norm.

Since

$$\nabla \boldsymbol{\tau} = \begin{pmatrix} \frac{\partial v}{\partial x} & \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} & \frac{\partial u}{\partial y} \end{pmatrix}, \tag{6.3}$$

the Euclidean norm of $\nabla\boldsymbol{\tau}$ is

$$|\nabla\boldsymbol{\tau}| = \sqrt{\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2}. \tag{6.4}$$

The Euclidean norm of $\boldsymbol{\tau} - \boldsymbol{\tau_0}$ is $|\boldsymbol{\tau} - \boldsymbol{\tau_0}| = \sqrt{(v - v_0)^2 + (u - u_0)^2}$.

Each term of (6.2) has a physical meaning. The first term smoothes the tangential vectors in the missing region Ω and the surrounding band B. The second term makes sure that $\boldsymbol{\tau}$ is close to $\boldsymbol{\tau_0}$ in the surrounding band B, where $\boldsymbol{\tau}_0$ are the tangential vectors from the observed image d_0. ϵ is a weight parameter that decides how close $\boldsymbol{\tau}$ will be to $\boldsymbol{\tau}_0$ in the surrounding band B. For a small value of ϵ the tangential vectors $\boldsymbol{\tau}$ must be closer to $\boldsymbol{\tau}_0$ in B than for a large value of ϵ. The constraint $\nabla \cdot \boldsymbol{\tau} = 0$ ensures us that $\boldsymbol{\tau}$ are tangential vectors to the level curves of a given function d. This can be explained from the Divergence Theorem.

Theorem 4 (Divergence theorem) . *Let R be a regular, closed region in the xy-plane whose boundary, C, consists of one or more piecewise smooth, non-self-intersecting, closed curves. Let $\mathbf{n}$ denote the unit outward (from R) normal field on C.*
If $\mathbf{F} = F_1(x,y)\mathbf{i} + F_2(x,y)\mathbf{j}$ is a smooth vector field on R, then

$$\int_R \nabla \cdot \mathbf{F}\, \mathrm{d}\mathbf{x} = \int_C \mathbf{F} \cdot \mathbf{n}\, \mathrm{d}s. \tag{6.5}$$

From this theorem we deduce that if $\nabla \cdot \mathbf{F} = 0$ in the region R, then $\mathbf{F} \cdot \mathbf{n} = 0$ at the boundary C. The identity $\mathbf{F} \cdot \mathbf{n} = |\mathbf{F}||\mathbf{n}|\cos\theta$, where θ is the angle between $\mathbf{F}$ and $\mathbf{n}$, tells us that $\mathbf{F}$ must be perpendicular to $\mathbf{n}$ if $\mathbf{F} \neq 0$ and $\mathbf{n} \neq 0$.

Let R be the region inside the level curve C. If $\nabla \cdot \boldsymbol{\tau} = 0$ in R, we know that $\boldsymbol{\tau}$ are perpendicular to $\mathbf{n}$ and thus the tangential vectors to this level curve.

In this work we have chosen the surrounding band B to be only one pixel wide. If we let $\epsilon \to 0$, the minimization problem reduces to

$$\min_{\boldsymbol{\tau}} \int_\Omega |\nabla\boldsymbol{\tau}|\, \mathrm{d}\mathbf{x} \quad \text{subject to} \quad \begin{array}{rcll} \nabla \cdot \boldsymbol{\tau} &=& 0 & \text{in } \Omega, \\ \boldsymbol{\tau} &=& \boldsymbol{\tau_0} & \text{on } \partial\Omega. \end{array} \tag{6.6}$$

To deal with the constraint $\nabla \cdot \boldsymbol{\tau} = 0$ a Lagrange multiplier is used. We then get the Lagrange functional

$$L(\boldsymbol{\tau}, \lambda) = \int_\Omega |\nabla\boldsymbol{\tau}|\, \mathrm{d}\mathbf{x} + \int_\Omega \lambda \nabla \cdot \boldsymbol{\tau}\, \mathrm{d}\mathbf{x}. \tag{6.7}$$

To achieve the boundary condition $\boldsymbol{\tau} = \boldsymbol{\tau_0}$ on $\partial\Omega$ we assume that $\boldsymbol{\tau_0}$ is defined on all of $\overline{\Omega}$ with $\boldsymbol{\tau_0} \in BV(\Omega)$. A natural way to express the boundary condition is to require $\boldsymbol{\tau} - \boldsymbol{\tau_0} \in BV_0(\Omega)$. $\boldsymbol{\tau}$ are not allowed to range freely over $BV(\Omega)$, but only over the admissible set

$$A = \{\boldsymbol{\tau} \in BV(\Omega) \,:\, \boldsymbol{\tau} - \boldsymbol{\tau_0} \in BV_0(\Omega)\} = \{\boldsymbol{\tau} = \boldsymbol{\tau_0} + \boldsymbol{\mu} \,:\, \boldsymbol{\mu} \in BV_0(\Omega)\}. \tag{6.8}$$

To obtain the corresponding Euler-Lagrange equation we calculate the derivatives $\frac{\partial L}{\partial \boldsymbol{\tau}}$ and $\frac{\partial L}{\partial \lambda}$ of the Lagrange functional and set them equal to zero. From the definition of the functional derivative in Section 4.2, we get

$$\begin{aligned} 0 = \frac{\partial L}{\partial \boldsymbol{\tau}} \cdot \boldsymbol{\mu} &= \lim_{h\to 0} \frac{L(\boldsymbol{\tau} + h\boldsymbol{\mu}, \lambda) - L(\boldsymbol{\tau}, \lambda)}{h} \\ &= \lim_{h\to 0} \frac{1}{h} \int_\Omega |\nabla(\boldsymbol{\tau} + h\boldsymbol{\mu})| - |\nabla \boldsymbol{\tau}| \,\mathrm{d}\mathbf{x} + \lim_{h\to 0} \frac{1}{h} \int_\Omega \lambda \nabla \cdot (\boldsymbol{\tau} + h\boldsymbol{\mu}) - \lambda \nabla \cdot \boldsymbol{\tau} \,\mathrm{d}\mathbf{x}, \end{aligned} \tag{6.9}$$

where $\boldsymbol{\mu} = (w, z)$ are allowed to vary in $BV_0(\Omega)$. Using the difference of two squares, the first term of the sum simplifies to

$$|\nabla(\boldsymbol{\tau} + h\boldsymbol{\mu})| - |\nabla \boldsymbol{\tau}| = \frac{(\nabla(\boldsymbol{\tau} + h\boldsymbol{\mu}))^2 - (\nabla \boldsymbol{\tau})^2}{|\nabla(\boldsymbol{\tau} + h\boldsymbol{\mu})| + |\nabla \boldsymbol{\tau}|} = h\frac{2\nabla \boldsymbol{\tau} \cdot \nabla \boldsymbol{\mu} + h\boldsymbol{\mu} \cdot \boldsymbol{\mu}}{|\nabla(\boldsymbol{\tau} + h\boldsymbol{\mu})| + |\nabla \boldsymbol{\tau}|}. \tag{6.10}$$

Theorem 5 (The Dominated Convergence theorem) . *Assume that the functions* $\{f_k\}_{k=1}^\infty$ *are Lebesgue integrable and*

$$f_k \to f \qquad a.e. \tag{6.11}$$

Suppose also that

$$|f_k| \le g \qquad a.e., \tag{6.12}$$

for some integrable function g. Then

$$\int_{\mathbb{R}^n} f_k \,\mathrm{d}\mathbf{x} \to \int_{\mathbb{R}^n} f \,\mathrm{d}\mathbf{x}. \tag{6.13}$$

Using the Dominated Convergence theorem, it is possible to move the limit inside the integral and let $h \to 0$,

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\tau}} \cdot \boldsymbol{\mu} &= \int_\Omega \lim_{h\to 0} \frac{2\nabla \boldsymbol{\tau} \cdot \nabla \boldsymbol{\mu} + h\boldsymbol{\mu} \cdot \boldsymbol{\mu}}{|\nabla(\boldsymbol{\tau} + h\boldsymbol{\mu})| + |\nabla \boldsymbol{\tau}|} \,\mathrm{d}\mathbf{x} + \int_\Omega \lim_{h\to 0} \lambda \nabla \cdot \boldsymbol{\mu} \,\mathrm{d}\mathbf{x} \\ &= \int_\Omega \frac{\nabla \boldsymbol{\tau} \cdot \nabla \boldsymbol{\mu}}{|\nabla \boldsymbol{\tau}|} \,\mathrm{d}\mathbf{x} + \int_\Omega \lambda \nabla \cdot \boldsymbol{\mu} \,\mathrm{d}\mathbf{x}, \end{aligned} \tag{6.14}$$

where the dot product between $\nabla \boldsymbol{\tau}$ and $\nabla \boldsymbol{\mu}$ is

$$\nabla \boldsymbol{\tau} \cdot \nabla \boldsymbol{\mu} = \frac{\partial v}{\partial x}\frac{\partial w}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial w}{\partial y} + \frac{\partial u}{\partial x}\frac{\partial z}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial z}{\partial y}, \tag{6.15}$$

and the dot product between ∇ and $\boldsymbol{\mu}$ is

$$\nabla \cdot \boldsymbol{\mu} = \frac{\partial w}{\partial x} + \frac{\partial z}{\partial y}. \tag{6.16}$$

Theorem 6 (Green's formulas) *Assume that Ω is a bounded, open subset of $\mathbb{R}^n$ and $\partial\Omega$ is C^1. Let u, $v \in C^2\left(\overline{\Omega}\right)$. Then*

$$\int_\Omega Dv \cdot Du \, \mathrm{d}\mathbf{x} = -\int_\Omega u\Delta v \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega} \frac{\partial v}{\partial \vec{\nu}} u \, \mathrm{d}s, \tag{6.17}$$

$$\int_\Omega u_{x_i} v \, \mathrm{d}\mathbf{x} = -\int_\Omega u v_{x_i} \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega} uv\nu^i \, \mathrm{d}s, \tag{6.18}$$

where $\boldsymbol{\nu} = (\nu^1, ..., \nu^n)$ is the outward pointing unit normal vector field along $\partial\Omega$.

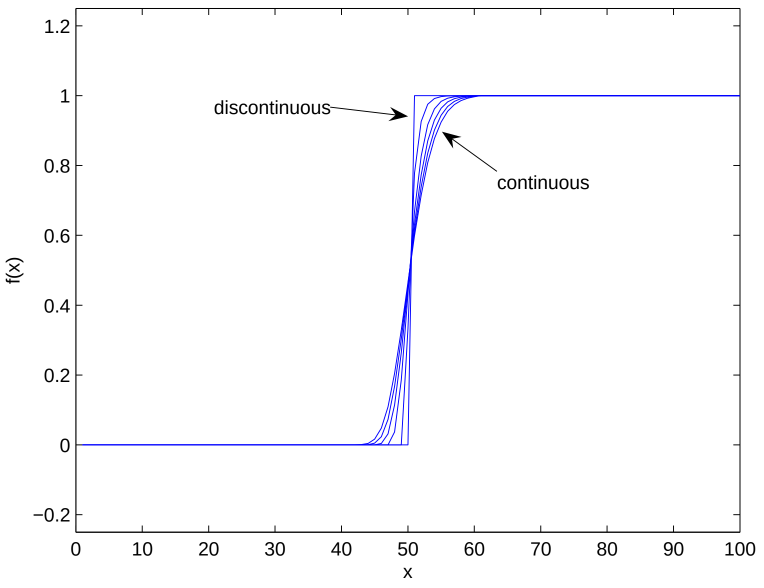


Figure 6.1: A step function can be approximated by smooth functions.

Since $\boldsymbol{\tau} \in BV(\Omega)$, $\boldsymbol{\tau}$ do not necessarily fulfill the assumptions for Theorem 6. But the way we represent an image, makes it possible to approximate the functions in $BV(\Omega)$ with functions in $C^2\left(\overline{\Omega}\right)$. Therefor we can regard the functions we use as smooth functions and use Theorem 6. Using Greens formulas on the first part and the last part gives

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\tau}} \cdot \boldsymbol{\mu} = &-\int_\Omega \nabla \cdot \left(\frac{\nabla \boldsymbol{\tau}}{|\nabla \boldsymbol{\tau}|}\right) \cdot \boldsymbol{\mu} \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega} \frac{\nabla \boldsymbol{\tau}}{|\nabla \boldsymbol{\tau}|} \cdot \boldsymbol{\nu} \cdot \boldsymbol{\mu} \, \mathrm{d}s \\ &- \int_\Omega \nabla\lambda \cdot \boldsymbol{\mu} \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega} \lambda \boldsymbol{\nu} \cdot \boldsymbol{\mu} \, \mathrm{d}s, \end{aligned} \tag{6.19}$$

where $\boldsymbol{\nu}$ is the outer normal vector to the boundary $\partial\Omega$. Since $\boldsymbol{\mu} \in BV_0(\Omega)$, $\boldsymbol{\mu} = \mathbf{0}$ on the boundary $\partial\Omega$ and the two boundary integrals vanish. The derivative of L with respect to

λ is easier to calculate,

$$0 = \frac{\partial L}{\partial \lambda} = \int_\Omega \nabla \cdot \boldsymbol{\tau} \, \mathrm{d}\mathbf{x}. \tag{6.20}$$

As (6.19) holds for all functions $\boldsymbol{\mu}$, we conclude that $\boldsymbol{\tau}$ solves the nonlinear PDE

$$\begin{aligned} \nabla \cdot \left(\frac{\nabla \boldsymbol{\tau}}{|\nabla \boldsymbol{\tau}|} \right) + \nabla \lambda &= \mathbf{0} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{\tau} &= 0 \qquad \text{in } \Omega, \\ \boldsymbol{\tau} &= \boldsymbol{\tau_0} \qquad \text{on } \partial\Omega. \end{aligned} \tag{6.21}$$

The denominator $|\nabla \boldsymbol{\tau}|$ will be large for large $\nabla \boldsymbol{\tau}$, such that significant diffusion only takes place in regions of small changes in the image.

When $\nabla \boldsymbol{\tau}$ vanishes this equation will be highly non-regular. To regain some regularity we modify the variational problem. $|\nabla \boldsymbol{\tau}|$ in (6.6) is regularized by

$$|\nabla \boldsymbol{\tau}|_\epsilon = \sqrt{|\nabla \boldsymbol{\tau}|^2 + \epsilon}, \tag{6.22}$$

where $\epsilon > 0$ is a small constant. The minimization problem is then

$$\min_{\boldsymbol{\tau}} \int_\Omega \sqrt{|\nabla \boldsymbol{\tau}|^2 + \epsilon} \, \mathrm{d}\mathbf{x} \quad \text{subject to } \begin{aligned} \nabla \cdot \boldsymbol{\tau} &= 0 \qquad \text{in } \Omega, \\ \boldsymbol{\tau} &= \boldsymbol{\tau_0} \qquad \text{on } \partial\Omega, \end{aligned} \tag{6.23}$$

and the modified Lagrange functional is

$$L_\epsilon(\boldsymbol{\tau}, \lambda) = \int_\Omega \sqrt{|\nabla \boldsymbol{\tau}|^2 + \epsilon} \, \mathrm{d}\mathbf{x} + \int_\Omega \lambda \nabla \cdot \boldsymbol{\tau} \, \mathrm{d}\mathbf{x}. \tag{6.24}$$

This leads to the regularized version of (6.21),

$$\begin{aligned} \nabla \cdot \left(\frac{\nabla \boldsymbol{\tau}}{\sqrt{|\nabla \boldsymbol{\tau}| + \epsilon}} \right) + \nabla \lambda &= \mathbf{0} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{\tau} &= 0 \qquad \text{in } \Omega, \\ \boldsymbol{\tau} &= \boldsymbol{\tau_0} \qquad \text{on } \partial\Omega. \end{aligned} \tag{6.25}$$

If we do not want the information to float into the region Ω from a part of the boundary $\partial\Omega_N$, we decompose the boundary into two parts, $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ and let $\boldsymbol{\tau} = \mathbf{0}$ on $\partial\Omega$. (6.25) will then be replaced by

$$\begin{aligned} \nabla \cdot \left(\frac{\nabla \boldsymbol{\tau}}{\sqrt{|\nabla \boldsymbol{\tau}| + \epsilon}} \right) + \nabla \lambda &= \mathbf{0} \qquad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{\tau} &= 0 \qquad \text{in } \Omega, \\ \boldsymbol{\tau} &= \boldsymbol{\tau_0} \qquad \text{on } \partial\Omega_D, \\ \boldsymbol{\tau} &= \mathbf{0} \qquad \text{on } \partial\Omega_N. \end{aligned} \tag{6.26}$$

6.1.1 Discretization

To find the minimum of $L(\boldsymbol{\tau}, \lambda)$ numerically we use the method of steepest descent, Section 5.4. The algorithm is

$$\begin{aligned} \boldsymbol{\tau}^{n+1} &= \boldsymbol{\tau}^n + \Delta t_1 \nabla \cdot \left(\frac{\nabla \boldsymbol{\tau}^n}{\sqrt{|\nabla \boldsymbol{\tau}^n| + \epsilon}} \right), \\ \lambda^{n+1} &= \lambda^n + \Delta t_1 \nabla \cdot \boldsymbol{\tau}^n, \end{aligned} \tag{6.27}$$

where Δt_1 is the step length.

The variables are discretized as shown in the Figure 6.2. It is important that the approximation of the derivatives lies on the same grid as the variable we are computing.

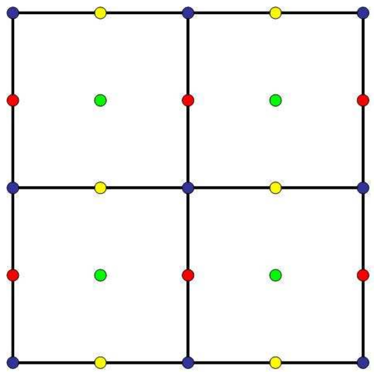


Figure 6.2: The discretization of an image d, where blue points correspond to the pixels $d_{i,j}$, green points correspond to $\lambda_{i+\frac{1}{2},j+\frac{1}{2}}$, yellow points correspond to $u_{i+\frac{1}{2},j}$ and red points correspond to $v_{i,j+\frac{1}{2}}$.

We define

$$u_{i,j} = u(x_i, y_j), \tag{6.28}$$

such that

$$u_{i+1,j} = u(x_{i+1}, y_j) \quad \text{and} \quad u_{i,j+1} = u(x_i, y_{j+1}). \tag{6.29}$$

To approximate the derivatives we have used forward, backward and centered difference formulas.

Definition 16 (Forward difference formula)

$$F_x^h u_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{h} \tag{6.30}$$

$$F_y^h u_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{h} \tag{6.31}$$

Definition 17 (Backward difference formula)

$$B_x^h u_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h} \tag{6.32}$$

$$B_y^h u_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{h} \tag{6.33}$$

Definition 18 (Centered difference formula)

$$C_x^h u_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} \tag{6.34}$$

$$C_y^h u_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2h} \tag{6.35}$$

In this work we have set the distance between the pixels to be 1, i.e. $h = 1$.

The updating formulas for u, v and λ are

$$\begin{aligned}
u^{n+1}_{i+\frac{1}{2},j} &= u^n_{i+\frac{1}{2},j} + \Delta t_1 \left[B_x \left(\frac{F_x u^n_{i+\frac{1}{2},j}}{\sqrt{G^n_{i+1,j} + \epsilon}} \right) + B_y \left(\frac{F_y u^n_{i+\frac{1}{2},j}}{\sqrt{H^n_{i+\frac{1}{2},j+\frac{1}{2}} + \epsilon}} \right) + B_y \lambda^n_{i+\frac{1}{2},j+\frac{1}{2}} \right], \\
v^{n+1}_{i,j+\frac{1}{2}} &= v^n_{i,j+\frac{1}{2}} + \Delta t_1 \left[B_x \left(\frac{F_x v^n_{i,j+\frac{1}{2}}}{\sqrt{H^n_{i+\frac{1}{2},j+\frac{1}{2}} + \epsilon}} \right) + B_y \left(\frac{F_y v^n_{i,j+\frac{1}{2}}}{\sqrt{G^n_{i,j+1} + \epsilon}} \right) + B_x \lambda^n_{i+\frac{1}{2},j+\frac{1}{2}} \right], \\
\lambda^{n+1}_{i+\frac{1}{2},j+\frac{1}{2}} &= \lambda^n_{i+\frac{1}{2},j+\frac{1}{2}} + \Delta t_1 \left(F_x v^n_{i,j+\frac{1}{2}} + F_y u^n_{i+\frac{1}{2},j} \right),
\end{aligned} \tag{6.36}$$

where

$$\begin{aligned}
G^n_{i,j} = &\left(\frac{1}{2} C_x v^n_{i,j+\frac{1}{2}} + \frac{1}{2} C_x v^n_{i,j-\frac{1}{2}} \right)^2 + \left(F_y v^n_{i,j-\frac{1}{2}} \right)^2 \\
&+ \left(F_x u^n_{i-\frac{1}{2},j} \right)^2 + \left(\frac{1}{2} C_y u^n_{i-\frac{1}{2},j} + \frac{1}{2} C_y u^n_{i+\frac{1}{2},j} \right)^2,
\end{aligned} \tag{6.37}$$

is located at the blue points in Figure 6.2 and

$$\begin{aligned}
H^n_{i+\frac{1}{2},j+\frac{1}{2}} = &\left(\frac{1}{2} C_y v^n_{i,j+\frac{1}{2}} + \frac{1}{2} C_y v^n_{i+1,j+\frac{1}{2}} \right)^2 + \left(F_x v^n_{i,j+\frac{1}{2}} \right)^2 \\
&+ \left(F_y u^n_{i+\frac{1}{2},j} \right)^2 + \left(\frac{1}{2} C_x u^n_{i+\frac{1}{2},j} + \frac{1}{2} C_x u^n_{i+\frac{1}{2},j+1} \right)^2,
\end{aligned} \tag{6.38}$$

is located at the green points in Figure 6.2.

Alternative approach

It is possible to solve the equations in another way. Instead of solving

$$\nabla \cdot \boldsymbol{\tau} = 0 \quad \text{in } \Omega, \tag{6.39}$$

we solve

$$\begin{aligned} -\Delta\lambda &= \nabla \cdot \left[\nabla \left(\frac{\nabla \boldsymbol{\tau}}{|\nabla \boldsymbol{\tau}|} \right) \right] && \text{in } \Omega, \\ \frac{\partial \lambda}{\partial \mathbf{n}} &= 0 && \text{on } \partial\Omega. \end{aligned} \tag{6.40}$$

Since a solution of (6.40) can only be determined up to a constant, the discretization of Δ is not invertible. If we modify this system by replacing the first row of the discretization by $(1, 0, ..., 0)$ and the first entry on the right hand side by 0, this system will have a unique solution. Instead of the updating formula for λ in (6.36) this equation is solved in each iteration.

6.1.2 Analogy with the Stokes equation

The Stokes equation with zero boundary condition is

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega, \\ \nabla \cdot \mathbf{u} &= 0 && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{0} && \text{on } \partial\Omega, \end{aligned} \tag{6.41}$$

where $\mathbf{u}$ is the velocity field of a steady fluid flow within the region Ω subject to the external force $\mathbf{f}$ and the pressure p. $\nabla \cdot \mathbf{u} = \mathbf{0}$ is the incompressibility condition.

If we replace $\frac{\nabla \boldsymbol{\tau}}{|\nabla \boldsymbol{\tau}|_\epsilon}$ in equation (6.25) with $\nabla \boldsymbol{\tau}$, we solve the Stokes equation with zero body force $\mathbf{f}$. The Lagrange multiplier λ, corresponding to the incompressibility condition, arises as the pressure p, but with reversed sign.

6.2 The equation for the image d

In the second minimization problem we want to find a surface d which fits the tangential vectors $\boldsymbol{\tau}$ found in the first part (6.26). To allow d to be a discontinuous function we let it belong to $BV(\Omega)$. d is found from the minimization problem

$$\min_d \int_{\tilde{\Omega}} \left(|\nabla d| - \nabla d \cdot \frac{\mathbf{n}}{|\mathbf{n}|} \right) \mathrm{d}\mathbf{x} + \frac{1}{\epsilon} \int_B (d - d_0)^2 \, \mathrm{d}\mathbf{x}. \tag{6.42}$$

The first term makes sure that the level curves of d will be orthogonal to $\mathbf{n}$ in the region $\tilde{\Omega}$. Let α be the angle between ∇d and $\mathbf{n}$, where $\mathbf{n}$ are the normal vector found from the tangential vectors,

$$\mathbf{n} = (v, -u). \tag{6.43}$$

Then

$$|\nabla d| - \nabla d \cdot \frac{\mathbf{n}}{|\mathbf{n}|} = |\nabla d| (1 - \cos\alpha) \tag{6.44}$$

is always non-negative. If α equals zero, ∇d and $\mathbf{n}$ is oriented in the same direction. Therefor d is found such that the tangent vectors to the level curves are the tangent vectors $\boldsymbol{\tau}$ calculated in the first part. The second term makes sure that the image d is not far from the initial image d_0 in the surrounding band B. As in the first part we choose B to be only one pixel wide and let $\epsilon \to 0$. We get the minimization problem

$$\min_d \int_{\Omega} \left(|\nabla d| - \nabla d \cdot \frac{\mathbf{n}}{|\mathbf{n}|} \right) \mathrm{d}\mathbf{x} \qquad \text{and} \qquad d = d_0 \text{ on } \partial\Omega. \tag{6.45}$$

The functional to be minimized is then

$$F(d) = \int_{\Omega} \left(|\nabla d| - \nabla d \cdot \frac{\mathbf{n}}{|\mathbf{n}|} \right) \mathrm{d}\mathbf{x}. \tag{6.46}$$

Similar to the first part, we assume that d_0 is defined in all of $\overline{\Omega}$ with $d_0 \in BV(\Omega)$ and require $d - d_0 \in BV_0(\Omega)$. d is allowed to range over the admissible set

$$A = \{d \in BV(\Omega) : d - d_0 \in BV_0(\Omega)\} = \{d = d_0 + \mu : \mu \in BV_0(\Omega)\}. \tag{6.47}$$

As in the first part, the derivative of the functional need to be calculated,

$$\begin{aligned} \frac{\partial F}{\partial d}\mu &= \lim_{h \to 0} \frac{F(d + h\mu) - F(d)}{h} \\ &= \lim_{h \to 0} \frac{1}{h} \int_{\Omega} \left(|\nabla d + h\nabla\mu| - |\nabla d| - \left(\nabla(d + h\mu) \cdot \frac{\mathbf{n}}{|\mathbf{n}|} - \nabla d \cdot \frac{\mathbf{n}}{|\mathbf{n}|} \right) \right) \mathrm{d}\mathbf{x}, \end{aligned} \tag{6.48}$$

where μ is a function belonging to $BV_0(\Omega)$. By using the difference of two squares and factorization we get

$$|\nabla d + h\nabla\mu| - |\nabla d| = \frac{(\nabla(d + h\mu))^2 - (\nabla d)^2}{|\nabla(d + h\mu)| + |\nabla d|} = h \frac{2\nabla\mu \cdot \nabla d + h\nabla\mu \cdot \nabla\mu}{|\nabla(d + h\nabla\mu)| + |\nabla d|}. \tag{6.49}$$

By Theorem 5, the Dominated Convergence theorem, it is possible to take the limit inside the integral sign. The equation simplifies to

$$\frac{\partial F}{\partial d}\mu = \int_\Omega \lim_{h\to 0}\left(\frac{2\nabla d\cdot\nabla\mu + h\nabla\mu\cdot\nabla\mu}{|\nabla\left(d+h\mu\right)| + |\nabla d|} - \nabla\mu\cdot\frac{\mathbf{n}}{|\mathbf{n}|}\right)\mathrm{d}\mathbf{x}. \tag{6.50}$$

When $h \to 0$ the equation becomes

$$\frac{\partial F}{\partial d}\mu = \int_\Omega \frac{\nabla d\cdot\nabla\mu}{|\nabla d|} - \nabla\mu\cdot\frac{\mathbf{n}}{|\mathbf{n}|}\,\mathrm{d}\mathbf{x}. \tag{6.51}$$

Using Greens formulas in Theorem 6, we get

$$\int_\Omega\left(\frac{\nabla d}{|\nabla d|} - \frac{\mathbf{n}}{|\mathbf{n}|}\right)\cdot\nabla\mu\,\mathrm{d}\mathbf{x} = -\int_\Omega \nabla\cdot\left(\frac{\nabla d}{|\nabla d|} - \frac{\mathbf{n}}{|\mathbf{n}|}\right)\mu\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega}\left(\frac{\nabla d}{|\nabla d|} - \frac{\mathbf{n}}{|\mathbf{n}|}\right)\cdot\boldsymbol{\nu}\mu\,\mathrm{d}s, \tag{6.52}$$

where $\boldsymbol{\nu}$ is the outer normal vector to the boundary $\partial\Omega$. Since $\mu \in BV_0(\Omega)$, the boundary integral vanishes and the Euler-Lagrange condition is

$$0 = \frac{\partial F}{\partial d}\mu = -\int_\Omega \nabla\cdot\left(\frac{\nabla d}{|\nabla d|} - \frac{\mathbf{n}}{|\mathbf{n}|}\right)\mu\,\mathrm{d}\mathbf{x}. \tag{6.53}$$

As this holds for all functions $\mu \in BV_0\left(\Omega\right)$, the equation to solve is

$$\begin{aligned}\nabla\cdot\left(\frac{\nabla d}{|\nabla d|} - \frac{\mathbf{n}}{|\mathbf{n}|}\right) &= 0 \qquad \text{in } \Omega,\\ d &= d_0 \qquad \text{on } \partial\Omega.\end{aligned} \tag{6.54}$$

To avoid singularities when $|\nabla d| = 0$, we use the same regularization as in the previous section,

$$|\nabla d|_\epsilon = \sqrt{|\nabla d|^2 + \epsilon}. \tag{6.55}$$

When normalizing the normal vectors $\mathbf{n}$, the same regularization is used. This leads to the minimization problem

$$\min_d \int_\Omega\left(\sqrt{|\nabla d|^2+\epsilon} - \nabla d\cdot\frac{\mathbf{n}}{\sqrt{|\mathbf{n}|^2+\epsilon}}\right)\mathrm{d}\mathbf{x} \qquad \text{and} \qquad d = d_0 \text{ on } \partial\Omega. \tag{6.56}$$

The functional to minimize is

$$F_\epsilon\left(d\right) = \int_\Omega\left(\sqrt{|\nabla d|^2+\epsilon} - \nabla d\cdot\frac{\mathbf{n}}{\sqrt{|\mathbf{n}|^2+\epsilon}}\right)\mathrm{d}\mathbf{x}, \tag{6.57}$$

and the regularized Euler-Lagrange equation is

$$\begin{aligned}\nabla\cdot\left(\frac{\nabla d}{\sqrt{|\nabla d|^2+\epsilon}} - \frac{\mathbf{n}}{\sqrt{|n|^2+\epsilon}}\right) &= 0 \qquad \text{in } \Omega,\\ d &= d_0 \qquad \text{on } \partial\Omega.\end{aligned} \tag{6.58}$$

Along edges where $|\nabla d| \gg \epsilon$, the solution of (6.58) will be nearly the same as the solution of (6.54). In smooth regions $|\nabla d| \ll \epsilon$ and the denominator will be close to a constant, $\sqrt{|\nabla d| + \epsilon} \approx \sqrt{\epsilon}$.

If the boundary is decomposed into two parts, $\partial\Omega_D$ and $\partial\Omega_N$, we use zero Neumann boundary conditions on $\partial\Omega_N$.

$$\begin{aligned} \nabla \cdot \left(\frac{\nabla d}{\sqrt{|\nabla d|^2 + \epsilon}} - \frac{\mathbf{n}}{\sqrt{|\mathbf{n}|^2 + \epsilon}} \right) &= 0 && \text{in } \Omega, \\ d &= d_0 && \text{on } \partial\Omega_D, \\ \frac{\partial d}{\partial \boldsymbol{\nu}} &= \mathbf{0} && \text{on } \partial\Omega_N. \end{aligned} \tag{6.59}$$

If we choose $\mathbf{n} = \mathbf{0}$ in (6.59), this PDE reduces to the TV-method in [19].

6.2.1 Discretization

We have again used the method of steepest descent to find the minimum of $F(d)$. The algorithm is

$$d^{n+1} = d^n + \Delta t_2 \left[\frac{\nabla d^n}{\sqrt{|\nabla d^n|^2 + \epsilon}} - \frac{\mathbf{n}}{\sqrt{|\mathbf{n}|^2 + \epsilon}} \right], \tag{6.60}$$

where Δt_2 is the step length.

After discretizing we get

$$\begin{aligned} d_{i,j}^{n+1} = d_{i,j}^n + \Delta t_2 \Bigg(& B_x \left(\frac{F_x d_{i,j}^n}{\sqrt{\left(F_x d_{i,j}^n\right)^2 + \left(\frac{1}{2} C_y d_{i,j}^n + \frac{1}{2} C_y d_{i+1,j}^n\right)^2 + \epsilon}} - n_{i+\frac{1}{2},j}^1 \right) \\ & + B_y \left(\frac{F_y d_{i,j}^n}{\sqrt{\left(F_y d_{i,j}^n\right)^2 + \left(\frac{1}{2} C_x d_{i,j}^n + \frac{1}{2} C_x d_{i,j+1}^n\right)^2 + \epsilon}} - n_{i,j+\frac{1}{2}}^2 \right) \Bigg), \end{aligned} \tag{6.61}$$

where $\mathbf{n} = (n^1, n^2)$ is discretized as

$$n_{i+\frac{1}{2},j}^1 = \frac{u_{i+\frac{1}{2},j}}{\sqrt{u_{i+\frac{1}{2},j}^2 + \left(\frac{1}{4}\left(v_{i,j-\frac{1}{2}} + v_{i,j+\frac{1}{2}} + v_{i+1,j+\frac{1}{2}} + v_{i+1,j-\frac{1}{2}}\right)\right)^2 + \epsilon}}, \tag{6.62}$$

and

$$n_{i,j+\frac{1}{2}}^2 = -\frac{v_{i,j+\frac{1}{2}}}{\sqrt{v_{i,j+\frac{1}{2}}^2 + \left(\frac{1}{4}\left(u_{i-\frac{1}{2},j} + u_{i+\frac{1}{2},j} + v_{i+\frac{1}{2},j+1} + v_{i-\frac{1}{2},j+1}\right)\right)^2 + \epsilon}}. \tag{6.63}$$

6.3 The implementation

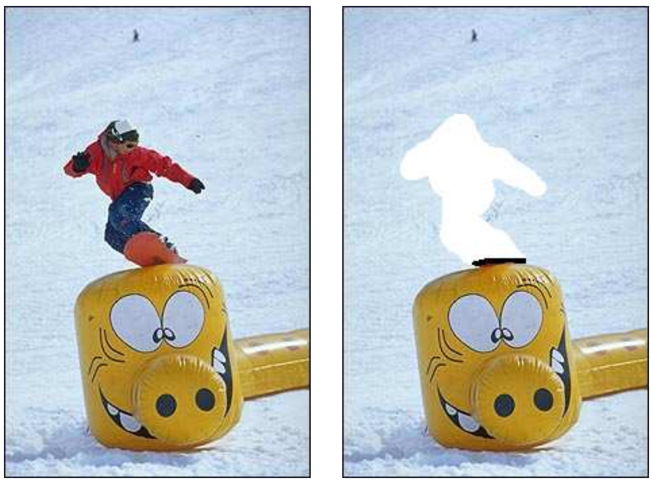
The code for solving the problem consists of several parts. I will give an outline of the main parts of the implementation.

6.3.1 Editing in Gimp

The region Ω to be inpainted must be marked manually in such a way that it can be recognized in the code. To do this I have used the image manipulation program Gimp. Gimp is a freely distributed piece of software and is available on the web, http://www.gimp.org/.

Since the boundary is decomposed into two parts with different boundary conditions, the code must be able to separate these two parts from each other. I have solved this by filling the parts of Ω verging towards $\partial\Omega_D$ with the colour white and the parts of Ω verging towards $\partial\Omega_N$ with the colour black. If one of these colours is present in the rest of the image, I have to choose another colour instead. It is important that all the damaged information around $\partial\Omega_D$ is removed. Otherwise it will propagate into the inpainting region and destroy the result. Outside $\partial\Omega_N$ it is not necessary to be that accurate since no information will propagate through this part of the boundary.

In the image in Figure 6.3 a) I want to remove the man. The inpainting region Ω is marked as explained above.



a) The original image.   b) The image with the inpainting region Ω marked.

Figure 6.3: An example of how the inpainting region is marked in Gimp.

It is important that the image is stored in a format that saves all the information in the image. The JPEG-format is not well suited, because it compresses the image to save storage space. When the code later tries to recognize the region Ω it will not be able to find it exactly as I specified it. I have chosen to use the PNM-format.

6.3.2 Implementation in Matlab

The code for this problem is made in Matlab. After specifying the parameters Δt_1, Δt_2 and ϵ this code automatically fills in the missing region Ω in the chosen image d_0.

The image d_0 is converted to a double precision matrix. In the computations of the tangential vectors $\boldsymbol{\tau}$ and the image d, information in an area around the inpainting region is used. To make the code work even when the inpainting region is on the boundary of the image, I extend the image with two pixels around the whole boundary. The values of the added pixels are the same as those on the boundary of the image.

To recognize the inpainting region Ω the code finds all the indices with the chosen colours. The indices of the tangential vectors $\boldsymbol{\tau}$ and the image d_0 on the boundary $\partial\Omega_N$ must also be found.

It is easiest to implement the code to calculate everything on a rectangle. The maximum and minimum indices of Ω in each direction are used to create the smallest possible rectangle.

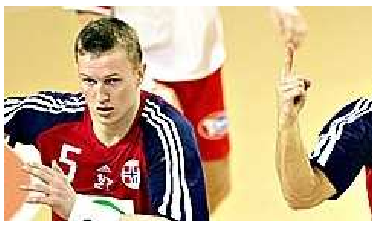
Computation of the tangential vectors $\boldsymbol{\tau}$

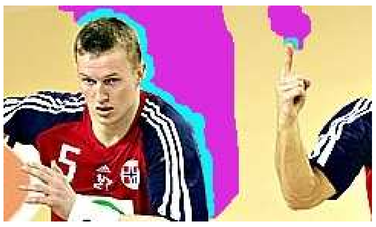
To improve the convergence the inpainting region is filled with random values from the interval $[0, 255]$. From this image the new tangential vectors $\boldsymbol{\tau}$ are calculated on the rectangle. After each iteration the values outside the inpainting region are replaced by their initial values and the tangential vectors $\boldsymbol{\tau}$ are set to zero on the boundary $\partial\Omega_N$. The zero boundary condition will then propagate into the region.

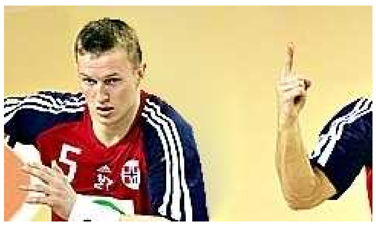
This step has two parameters, Δt_1 and ϵ which must be chosen properly for the algorithm to converge. During experiments I have found that $\Delta t_1 = 0.03$ is a good choice. With this value all the experiments converged. With a smaller value more iterations are needed. In the alternative approach (6.40) it is possible to choose a larger Δt_1. The advantage of this approach is that it converges after fewer iterations. The parameter ϵ is supposed to be small, and I have used $\epsilon = 0.0001$ in most occasions. If it is chosen to be larger the tangential vectors will be smoother, and the calculations will converge faster. It will also be stable even if a large Δt_1 is used.

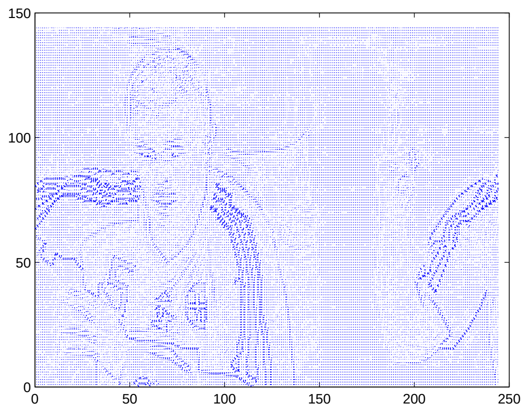
To check that this part of the code is correct I look at the boundary $\partial\Omega$. The tangential vectors $\boldsymbol{\tau}$ should propagate smoothly from the boundary $\partial\Omega_D$ and the zero boundary condition should propagate from the boundary $\partial\Omega_N$.

In the image in Figure 6.4 a) I have tried to remove one of the players. Since both white and black is present in the rest of the image, I have used violet and cyan to mark the region Ω instead. Figure 6.4 d) shows the original tangential vectors $\boldsymbol{\tau}_0$ and Figure 6.4 e) shows the tangential vectors after I have filled the inpainting region Ω with random values from 0 to 255. As expected, the tangential vectors have propagated smoothly into Ω from the boundary of the violet region and the zero boundary condition has propagated from the cyan region. This is shown in Figure 6.4 f).

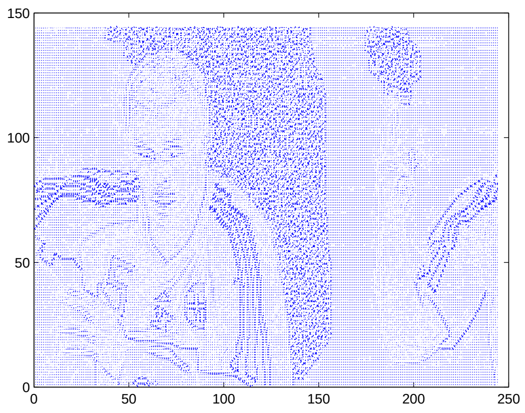

a) The original image d_0.


b) The image with the inpainting region marked.

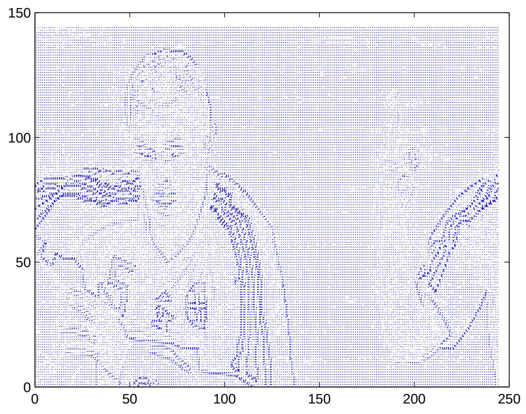

c) The restored image d.



d) The tangential vectors $\boldsymbol{\tau}_0$.



e) The tangential vectors $\boldsymbol{\tau}$ before any iterations.



f) The tangential vectors $\boldsymbol{\tau}$ of the restored image d.

Figure 6.4: Example of an image inpainting problem.

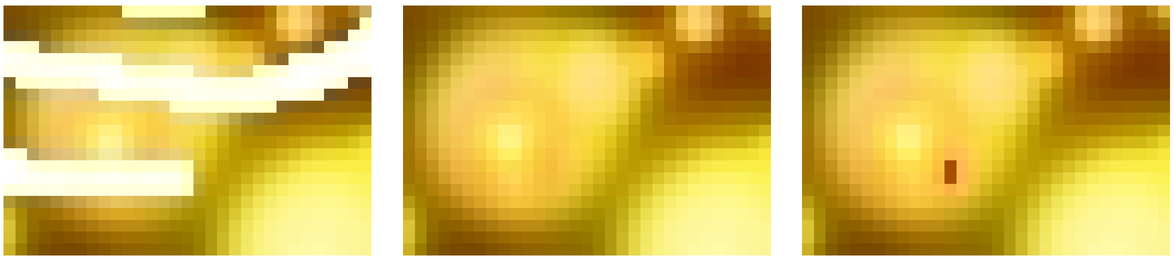
Computation of the image d

In the second part I use the normalized normal vectors found from the tangential vectors calculated in the first part. All the pixels in the rectangular region are computed and the indices of the inpainting region are used to update d only in Ω. The normal vectors $\mathbf{n}$ are set to zero on the boundary $\partial\Omega_N$ and d is updated on $\partial\Omega_N$ in each iteration. The iterations are repeated until convergence.

In this part the parameter Δt_2 must be chosen in such a way that d converges as fast as possible. If the normal vectors are very smooth, Δt_2 can be large, approximately 1. $\Delta t_2 = 0.1$ will in general be a good choice. A normalization is necessary in this part as well, and ϵ plays the same role as in the first part.

It is not uncommon that some pixels get values outside the interval $[0, 255]$. During the iterations I let the pixels oscillate in the interval $[-20, 300]$. Often it will converge inside the correct interval, but to be sure I always set all pixels to be in the interval $[0, 255]$ when the iterations have reached a steady state.

To make sure that the second part of the code is correct, I try to reconstruct a known image from its true normal vectors. Having all the level curves in an image will determine the image itself. Since I have approximations to the isophote directions in discrete points inside Ω this will only be an approximation. The true normal vectors can be very oscillating and some pixels do not converge unless epsilon is approximately 10. This is shown in Figure 6.5. In Figure 6.5 b) the image is reconstructed perfectly, but in Figure 6.5 c) there is a red spot which does not disappear.



a) The inpainting region marked white.

b) $\epsilon = 10$

c) $\epsilon = 0.0001$

Figure 6.5: Reconstruction of an image from its true normals.

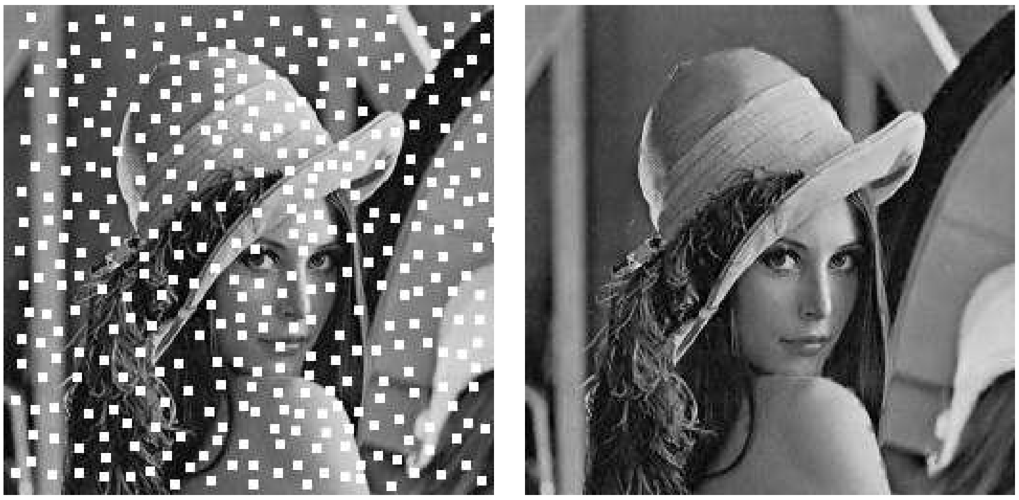
If the image is a colour image, the two steps must be performed three times, one for each matrix in the RGB colour model.

6.3.3 Discussion

The method has been tested on different types of problems. These are included in the appendix.

In the first examples I remove a part of a gray-scale image and try to restore it. This is a good way of testing the code, because I can choose the inpainting region as I want. From these tests I find that the method produces good results if the inpainting region consists of small disconnected areas.

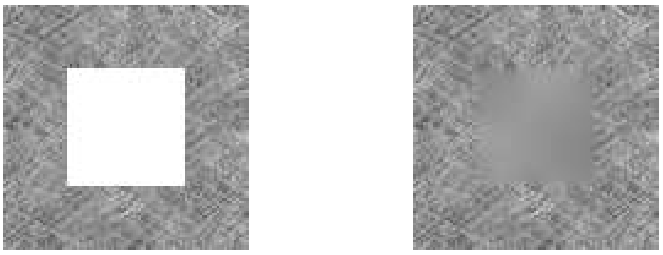
An example of this is shown in Figure 6.6. In Figure 6.6 a) many small areas are missing. The result in Figure 6.6 b) is nearly perfect. The only problem is that the edge of her shoulder has some dark spots.



a) The original image d_0. b) The reconstructed image d.

Figure 6.6: Reconstruction of an image with several small regions missing.

The method has some weaknesses. One weakness is that it does not recreate texture very well. This is illustrated in Figure 6.7.

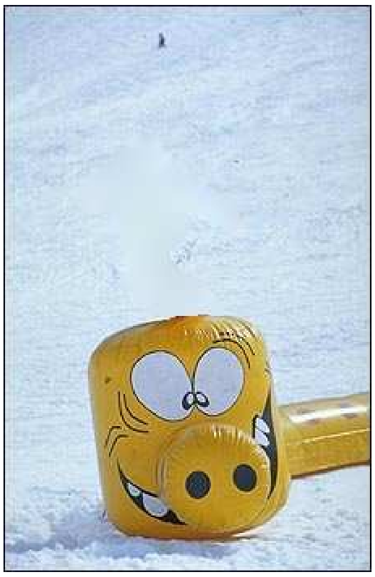


a) The original image d_0. b) The reconstructed image d.

Figure 6.7: Reconstruction of an image with texture.

To improve this we could combine the method with a texture synthesis technique as explained in Section 2.2.

I have also tried to make objects disappear. The image in Figure 6.3 is an example of this. Figure 6.8 shows the restored image d.

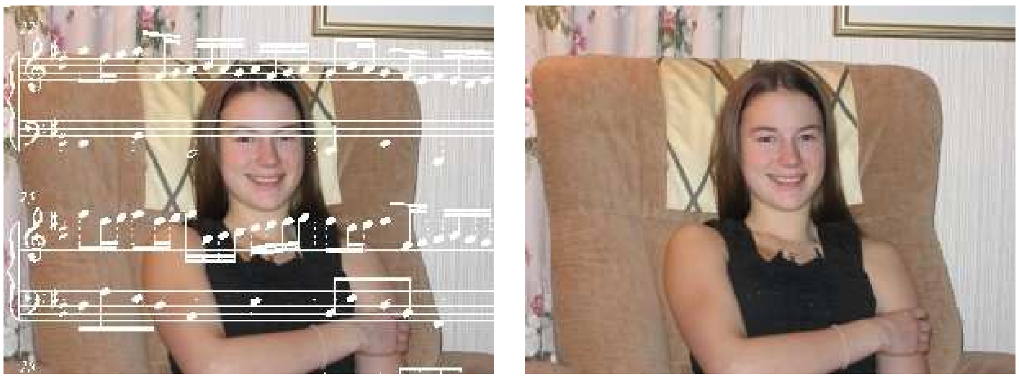


The restored image d.

Figure 6.8: Removal of a man from an image.

Another example of this is shown in Figure 6.4. Neither of these two examples contain much texture and the results are good.

The method makes good results when I try to remove text written on an image. In Figure 6.9 a similar problem is shown.



a) The original image d_0. b) The reconstructed image d.

Figure 6.9: Reconstruction of an image with musical notes written on it.

In general, the method does not restore objects that have parts of their boundaries missing. Neither does it connect lines and edges very well, especially not across wide regions.

The convergence varies from image to image. If the regions to be inpainted are small, few iterations are necessary. This is also the case if the missing regions are long and narrow as in Figure 6.9. In the alternative approach (6.40) fewer iterations are needed, but if the inpainting region Ω is large, this will require considerably more computational time.

The implementation is not optimal. An example of this is that the computations are done on a rectangle containing the missing regions. In images like in Figure 6.6 there are done a lot more computations than necessary.

A further discussion of the TV-Stokes method and our results are presented in the appendix.

Bibliography

[1] The mathworks. http://www.mathworks.com/.

[2] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing*. Springer, 2002.

[3] A. L. Bertalmio, M.and Bertozzi and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition*, volume 1, 2001.

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000. ISBN:1-58113-208-5.

[5] M. Bertalmio, L. Vese, G. Sapiro, and S Osher. Simultaneous structure and texture image inpainting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, June 2003. ISBN: 0-7695-1900-8.

[6] F. C. Chan, S. Osher, and J. Shen. The digital tv filter and nonlinear denoising. *IEEE Transactions*, 10, February 2001.

[7] T. Chan and J. Shen. Mathematical models for local non-texture inpaintings. *Siam Journal on Applied Mathematics*, 62(3), 2002.

[8] T. F. Chan, S. H. Kang, and J. Shen. Euler's elastica and curvature-based image inpainting. *SIAM Journal on Applied Mathematics*, 63(2), 2002.

[9] T. F. Chan and J. Shen. Non-texture inpainting by curvature-driven diffusions (CDD). *J. Visual Comm. Image Rep.*, 12(4), 2001.

[10] T. F. Chan and C. K. Wong. Total variation blind deconvolution. *IEEE Transactions on Image Processing*, 7(3), 1998.

[11] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9), September 2004.

[12] L. C. Evans. *Partial Differential Equations.* American Mathematical Society, 2002. ISBN 0-8218-0772-2.

[13] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 1984.

[14] P. C. Hansen. Deconvolution and regularization with toepliz matrices. *Numerical Algorithms*, 29, 2002.

[15] M. Lysaker, S. Osher, and X.C. Tai. Noise removal using smoothed normals and surface fitting. *IEEE Transaction on Image Processing*, 13(10), 2004.

[16] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer, 2003.

[17] S. Osher, A. Solè, and L. Vese. Image decomposition, image restoration and texture modeling using total variation minimization and the H^{-1} norm. *IEEE*, 2003.

[18] A.A. Robert. *Calculus.* Addison Wesley, 1999. ISBN 0-201-39607-6.

[19] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 1992.

[20] W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill International Editions, 1976. ISBN 0-07-085613-3.

[21] J. Shen. Inpainting and the fundamental problem of image processing. *SIAM News*, 36(5), 2003.

Appendix

Image inpainting using a TV-stokes equation[*]

Xue-Cheng Tai[†], Stanley Osher[‡] and Randi Holm[§]

May 23, 2005

Abstract

Based on some geometrical considerations, we propose a two-step method to do digital image inpainting. In the first step, we try to propagate the isophote directions into the inpainting domain. An energy minimization model combined with the zero divergence condition is used to get a nonlinear Stokes equation. Once the isophote directions are constructed, an image is restored to fit the constructed directions. Both steps reduce to the solving of some nonlinear partial differential equations. Details about the discretization and implementation are explained. The algorithms have been intensively tested on synthetic and real images. The advantages of the proposed methods are demonstrated by these experiments.

1 Introduction

For a digital image, inpainting refers to the process of filling-in missing data. It ranges from removing objects from an image to repairing damaged images and photographs.

The term of "digital inpainting" seems to be introduced into image processing by Bertalmi, Sapiro, Casseles and Ballester [2]. In the past few years, several different approaches have been proposed to tackle this complicated image processing task. The basic idea for most of the inpainting techniques is to do a smooth propagation of the information in the region surrounding the inpainting area and interpolating level curves in a proper way [2, 21, 5]. However, there are different strategies to achieve these goals. In [2], they proposed to minimize some energy to compute the restored image and this results in the solving of some coupled nonlinear differential equations. In a related work [4], this idea is further extended to guarantee that the level curves are propagated into the inpainting domain. In [3], a connection between the isophote direction of the image and the Navier-Stokes equation is observed and they propose to solve some transport equations to fill in the inpainting domain. This is related to our methods. Another related work is [10] where a minimization of the divergence is done to construct optical flow functions.

[*]We acknowledge support from the Norwegian Research Council.

[†]Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5007 Bergen, Norway. Email: Tai@mi.uib.no. URL: http://www.mi.uib.no/~tai.

[‡]Department of Mathematics UCLA, California, USA (e-mail: sjo@math.ucla.edu)

[§]Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5007 Bergen, Norway.

The work of [8, 6] is trying to minimize the TV-norm of the reconstructed image to fill in the missing data. In some later work [7, 9], energy involving the curvature of the level curves is used and this is in some sense trying to guarantee that the level curves are connected in a smooth fashion. The equations obtained from such models are highly nonlinear and of higher order.

Recently, texture inpainting has attracted more attentions. In [20], the image in the surrounding area is first decomposed into texture and structure and then propagated into the inpainting domain in different ways. This idea to decompose texture and structure is also utilized in [11]. Some statistical approaches are used in [1] to do texture synthesis and structure propagation.

We shall also mention some recent works which related the phase-field model and Ginzburg-Landau equation to image processing, [14, 15, 12, 11]. These ideas were used in [14, 15, 12] for image segmentation. In [11] it has been used for image inpainting.

The idea used in this work has been motivated by [18, 19, 2, 3]. We still follow the basic ideas for image inpainting, i.e. we are trying to propagate the information into the inpainting domain along the isophote directions. However, we choose a two-step method to carry out this task as in [19]. The first step is trying to reconstruct the isophote directions for the missing data. The second step is trying to construct an image fitting to the restored directions. This is the same idea used in [19] to remove noise from digital images. One new idea which is essential to the present method is that we impose the zero divergence condition on the constructed directions. This guarantees that there exists an image such that its isophote directions are the restored vectors. This is important when the inpainting regions is relatively large. Different from [3], we obtain our TV-Stokes equation from this consideration which implies that the obtained vectors have the smallest TV-norm. The solution of the Navier-Stokes equation may not have such a property. Inherited from the flexibility of our approach, we propose some novel ideas to alternate the boundary condition for the inpainting domain to select the information that shall be propagated into the region. We have only tested our algorithms on propagated structure information. It is possible to combine it with texture inpainting as in [20].

This work is organized in the following way. In section 2, we explain the detailed mathematical principles for our methods. First, some geometrical motivations are explain for our methods. These geometrical observations are then combined with some energy minimization models to get the nonlinear equations for our inpainting methods. Discretization and implementation details are then supplied which are needed for solving the equations numerically. When solving the equations, it is rather easy to change the boundary conditions. Due to this flexibility, we show that it is rather easy to discriminate against some information and block it from propagating into the inpainting region. Intensive numerical experiments on real and synthetic images are supplied in Section 3. With these experiments, we are trying to summarize the convergence behavior of the algorithms and compare our method with some literature results.

2 The Mathematical principles

Suppose that an image $u_0 : R \mapsto [a, b]$ is defined on a rectangle domain R and $a < b$. We shall assume that $\Omega \subset R$ is the domain where the data is missing. We want to fill in the information on Ω based in the geometrical and photometric information surrounding the region Ω. Similar to [2], we shall use information in a band B around the domain Ω. We shall use $\tilde{\Omega} = \Omega \cup B$ in the following.

2.1 Connection between digital images and flow fields

In [3], the connection between image inpainting and fluid dynamics is done by observing that the isophote directions of an image are corresponding to an incompressible velocity field. This same observation will be used here in our work. However, the equation we shall use for the inpainting is different and is more inspired by the work of [19]. We give a brief outline of the idea of [19] in the following.

Given scalar functions u and v, denote:

$$\nabla u = (u_x, u_y),\ \nabla^{\perp} u = (-u_y, u_x),\ \nabla \times (u,v) = u_y - v_x,\ \nabla \cdot (u,v) = u_x + v_y.$$

Given an image d_0, the level curves:

$$\Gamma(c) = \{x : \ d_0(x) = c, \quad \forall c \in (-\infty, \infty)\}.$$

has the normal vectors $\vec{n}(x)$ and tangential vectors $\vec{\tau}(x)$ given by

$$\vec{n}(x) = \nabla d_0(x) \quad \vec{\tau}(x) = \nabla^{\perp} d_0(x).$$

The vector fields $\vec{n}$ and $\vec{\tau}$ satisfy

$$\nabla \times \vec{n}(x) = 0, \quad \nabla \cdot \vec{\tau}(x) = 0. \tag{1}$$

Suppose that the surface $d_0(x)$ is exposed to rain, then the rain will flow down the surface along the directions $-\vec{n}(x)$. One observation is that the surface d_0 can be constructed from the vector fields $\vec{n}(x)$ or $\vec{\tau}(x)$ using some boundary condition or some other information along the curve.

For image inpainting, the information of d_0 in the surrounding band B is known. Thus, we also know the normal and tangential vectors of d_0 in B. The principle idea to fill in the information in Ω is to propagate the vector field $\vec{n}$ or $\vec{\tau}$ into the interior region Ω. Afterwards, we construct an image in region Ω to fit the computed vectors in Ω.

Define $\vec{\tau}_0 = \nabla^{\perp} d_0$. There are different ways to propagate the vectors from B into Ω. In [3], the Navier-Stokes equations are used. Here, we shall use an energy minimization model to propagate the vector fields, i.e. we shall solve

$$\min_{\nabla \cdot \vec{\tau} = 0} \int_{\tilde{\Omega}} |\nabla \vec{\tau}| dx + \frac{1}{\epsilon} \int_B |\vec{\tau} - \vec{\tau}_0|^2 dx \tag{2}$$

Above, $\int_{\tilde{\Omega}} |\nabla \vec{\tau}| dx$ is the total variation for vector field $\vec{\tau}$. We require $\nabla \cdot \vec{\tau} = 0$ to guarantee that the reconstructed vector field $\vec{\tau}$ is a tangential vector for the level curves of a scalar function in the region Ω. The penalization parameter ϵ is chosen to be very small to guarantee that $\vec{\tau} \approx \vec{\tau}_0$ in B. For most of the cases we have tested, it is enough to take B to be just one pixel wide around Ω. For such a case, we can take $\epsilon \to 0$ and thus the minimization problem reduces to find a $\vec{\tau}$ such that $\vec{\tau} = \vec{\tau}_0$ on $\partial\Omega$ which solves:

$$\min_{\nabla \cdot \vec{\tau} = 0} \int_{\Omega} |\nabla \vec{\tau}| dx. \tag{3}$$

The reason why we use the total variation norm of $\vec{\tau}$ is due to the fact that the boundary value $\vec{\tau}_0$ may have discontinuities. In order to propagate such a discontinuity into the region Ω, we need to allow $\vec{\tau}$ to have discontinuities and thus the TV-norm is preferred to the H^1-norm.

We use χ_B to denote the characteristic function over the domain B, i.e. $\chi_B = 1$ in B and $\chi_B = 0$ elsewhere. If we use a Lagrange multiplier λ to deal with the divergence constraint $\nabla \cdot \vec{\tau} = 0$, the Euler-Lagrangian equation of (2) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla \vec{\tau}}{|\nabla \vec{\tau}|} \right) + \dfrac{\chi_B}{\epsilon}(\vec{\tau} - \vec{\tau}_0) - \nabla \lambda &= 0 \text{ in } \tilde{\Omega}, \\ \nabla \cdot \vec{\tau} &= 0 \text{ in } \tilde{\Omega}, \\ \nabla \vec{\tau} \cdot \vec{\nu} &= \vec{0} \text{ on } \partial \tilde{\Omega}. \end{cases} \tag{4}$$

Here, $\vec{\nu}$ denotes the outer unit normal vector of $\partial\tilde{\Omega}$. Similarly, the Euler-Lagrangian equation of (3) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla \vec{\tau}}{|\nabla \vec{\tau}|} \right) - \nabla \lambda &= 0 \text{ in } \Omega, \\ \nabla \cdot \vec{\tau} &= 0 \text{ in } \Omega, \\ \vec{\tau} &= \vec{\tau}_0 \text{ on } \partial \Omega. \end{cases} \tag{5}$$

Once the tangential vector field $\vec{\tau}$ is available in $\tilde{\Omega}$, it is easy to get the normal vector field $\vec{n}$. Let u and v be the two components of the vector field $\vec{\tau}$, i.e. $\vec{\tau} = (u, v)$. Then, we have

$$\vec{n}(x) = \vec{\tau}^{\perp}(x) = (-v, u). \tag{6}$$

From the vector field $\vec{n}(x)$, we use the same idea as in [19, 2] to construct an image d whose normal vectors shall fit the computed vectors $\vec{n}(x)$. This is achieved by solving the following minimization problem:

$$\min \int_{\tilde{\Omega}} |\nabla d| - \nabla d \cdot \frac{\vec{n}}{|\vec{n}|} dx + \frac{1}{\epsilon} \int_B |d - d_0|^2 dx. \tag{7}$$

The penalization parameter ϵ can be chosen to be same as in (2). It can also be chosen to be different. In case that B is only one pixel wide around Ω, the above minimization problem reduces to the following problem if we take $\epsilon \to 0$:

$$\min_d \int_{\Omega} |\nabla d| - \nabla d \cdot \frac{\vec{n}}{|\vec{n}|} dx \quad \text{ and } d = d_0 \text{ on } \partial\Omega. \tag{8}$$

The Euler-Lagrangian equation of (7) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla d}{|\nabla d|} - \dfrac{\vec{n}}{|\vec{n}|} \right) + \dfrac{\chi_B}{\epsilon}(d - d_0) &= 0 \text{ in } \tilde{\Omega}, \\ (\dfrac{\nabla \vec{\tau}}{|\nabla \vec{\tau}|} - \dfrac{\vec{n}}{|\vec{n}|}) \cdot \vec{\nu} &= \vec{0} \text{ on } \partial \tilde{\Omega}. \end{cases} \tag{9}$$

Similarly, the Euler-Lagrangian equation of (8) is:

$$\begin{cases} -\nabla \cdot \left(\dfrac{\nabla d}{|\nabla d|} - \dfrac{\vec{n}}{|\vec{n}|} \right) &= 0 \text{ in } \Omega, \\ d &= d_0 \text{ on } \partial \Omega. \end{cases} \tag{10}$$

2.2 Discretization

We shall try to explain some of the details in discretizing the equations derived in the last section for numerical simulations. For clarity, we shall only outline the details for algorithms (5) and (10). The discretization for (4) and (9) can be done in a similar way.

For simplicity, gradient decent method will be used in our simulations. The gradient flow equation for $\vec{\tau}$ is:

$$\frac{\partial \vec{\tau}}{\partial t} - \nabla \cdot (\frac{\nabla \vec{\tau}}{\|\nabla \vec{\tau}\|}) - \nabla \lambda = 0 \qquad \text{in } \Omega, \tag{11}$$

$$\nabla \cdot \vec{\tau} = 0, \text{ in } \Omega, \quad \vec{\tau} = \vec{\tau}_0 \qquad \text{on } \partial\Omega. \tag{12}$$

where $\|\nabla \vec{\tau}\| = \sqrt{|u_x|^2 + |u_y|^2 + |v_x|^2 + |v_y|^2}$. We have tried two algorithms to solve (11)-(12). The first algorithm uses the following iterative procedure to update $\vec{\tau}$ and λ with the time step Δt_1 and initial values properly chosen:

$$\vec{\tau}^{n+1} = \vec{\tau}^n + \Delta t_1 \left[\nabla \cdot \left(\frac{\nabla \vec{\tau}^n}{\|\nabla \vec{\tau}^n\|} \right) + \nabla \lambda^n \right], \tag{13}$$

$$\lambda^{n+1} = \lambda^n + \Delta t_1 \nabla \cdot \vec{\tau}^n \tag{14}$$

The second algorithm is trying to update $\vec{\tau}$ and λ by:

$$\vec{\tau}^{n+1} = \vec{\tau}^n + \Delta t_1 \left[\nabla \cdot \left(\frac{\nabla \vec{\tau}^n}{\|\nabla \vec{\tau}^n\|} \right) + \nabla \lambda^n \right], \tag{15}$$

$$-\Delta \lambda^{n+1} = \nabla \cdot \left[\nabla \cdot \left(\frac{\nabla \vec{\tau}^n}{\|\nabla \vec{\tau}^n\|} \right) \right]. \tag{16}$$

In (16), Δ denotes the Laplace operator and we impose a zero Neumann boundary condition for λ^{n+1}. If $\nabla \cdot \tau^0 = 0$ and (16) is satisfied by all λ^n, then we get from (15) that

$$\nabla \cdot \tau^{n+1} = 0, \quad \forall n.$$

We use a stagged grid to approximate u, v and λ. Note that $\vec{\tau} = (u, v)$ is used to construct d. When we try to compute d from (9) or (10), we are trying to enforce the following relation approximately: $u = -d_y, \quad v = d_x$. Due to this relation, the grid points used in the approximation for u are chosen to be the points marked with $*$. The approximation points for v are marked with $\circ$. The centers of the rectangle elements marked with $\star$ are used as the approximation points for λ. The vertices of the rectangular mesh are used as the approximation points for d. The horizontal axis represents the x-variable and the vertical axis represents the y-variable, c.f Figure 1.

For a given domain Ω, we use $U_h(\Omega)$ to denote all the approximation points $*$ for u inside Ω, $V_h(\Omega)$ to denote all the approximation points $\circ$ for v inside Ω, $\Lambda_h(\Omega)$ to denote all the approximation points $\star$ for λ inside Ω and $D_h(\Omega)$ to denote all the approximation points for d inside Ω. The updating formula for

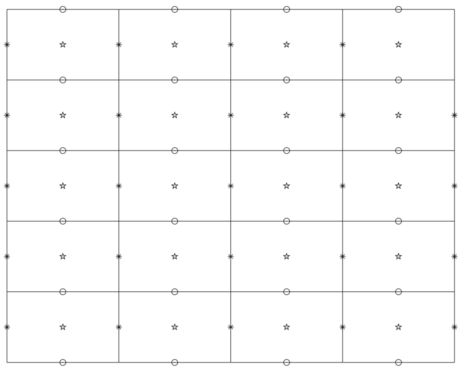


Figure 1: The pixels and the approximation points for u, v, λ and d. The approximation points are: $*$ for u, $\circ$ for v, $\star$ for λ.

(u, v) and λ for (13)-(14) are:

$$u^{n+1} = u^n + \Delta t_1 \left(D_x^- \left(\frac{D_x^+ u^n}{T_1^n} \right) + D_y^- \left(\frac{D_y^+ u^n}{T_2^n} \right) + C_x^{h/2} \lambda^n \right) \quad \text{on } U_h(\Omega), \tag{17}$$

$$v^{n+1} = v^n + \Delta t_1 \left(D_x^- \left(\frac{D_x^+ v^n}{T_2^n} \right) + D_y^- \left(\frac{D_y^+ v^n}{T_1^n} \right) + C_y^{h/2} \lambda^n \right) \quad \text{on } V_h(\Omega), \tag{18}$$

$$\lambda^{n+1} = \lambda^n + \Delta t_1 (C_x^{h/2} u^{n+1} + C_y^{h/2} v^{n+1}) \quad \text{on } \Lambda_h(\Omega). \tag{19}$$

Above, $D_x^\pm, D_y^\pm$ are the standard forward/backward finite difference operators and $C_x^{h/2}, C_x^{h/2}$ are the central finite difference operators with mesh size $h/2$. h denotes the mesh size for the approximations and is taken to be one. The terms T_1^n and T_2^n are evaluated as in the following:

$$T_1^n = \sqrt{|D_x^+ u|^2 + |C_y^h u|^2 + |C_y^h v|^2 + |D_y^+ v|^2 + \epsilon} \text{ on } \Lambda_h(\Omega), \tag{20}$$

$$T_2^n = \sqrt{|C_x^h u|^2 + |D_y^+ u|^2 + |D_y^+ v|^2 + |C_y^h v|^2 + \epsilon} \text{ on } D_h(\Omega). \tag{21}$$

If we use the second algorithm to compute (u, v) and λ from (15)-(16), the solution of (16) is not unique due to the use of the Neumann boundary condition. We fix the value of λ to be zero at one point on the boundary to overcome this problem, which is standard for this kind of problems. Fast methods, like FFT (Fast Fourier Transformation), can be used to solve (16).

Once the iterations for u and v have converged to a steady state, we use them to obtain d. Note that the relation between (u, v) and $\vec{n}$ is as in (6). Similar as in [19], the following gradient flow scheme is used to update d of (10):

$$d^{n+1} = d^n + \Delta t_2 \Bigg(\left(D_x^- \left(\frac{D_x^+ d^n}{D_1^n} + \frac{v}{\sqrt{\hat{u}^2 + v^2 + \epsilon}} \right) \right. \\ \left. + \left(D_y^- \left(\frac{D_y^+ d^n}{D_2^n} - \frac{u}{\sqrt{u^2 + \hat{v}^2 + \epsilon}} \right) \right) \Bigg) \text{ on } D_h(\Omega). \tag{22}$$

In the above, $\hat{u}$, $\hat{v}$ are the average values of the four nearest approximation points and

$$D_1^n = \sqrt{|D_x^+ d^n|^2 + |C_y^h d^n|^2 + \epsilon} \text{ on } D_h(\Omega), \tag{23}$$

$$D_2^n = \sqrt{|C_x^h d^n|^2 + |D_y^+ d^n|^2 + \epsilon} \text{ on } D_h(\Omega). \tag{24}$$

This iteration is the standard gradient updating for d. We could use the AOS scheme of [16, 17] to accelerate the convergence. The AOS scheme was first proposed in [16, 17]. It was later rediscovered in [22, 13] and used for image processing problems.

Up to now we have only explained the approximation details for (5) and (10). It is easy to see that the discretization for (4) and (9) can be done in a similar way. The Dirichlet or Neumann boundary conditions for the different equations shall be implemented in the standard way and we will omit the details.

2.3 Other kind of boundary conditions

We have proposed two alternatives to deal with the information which is in the surrounding area of Ω, i.e.

- Using information in a narrow band around the inpainting region Ω and trying to propagate this information into the region Ω using equations (4) and (9).
- Using information of the two nearest pixels around the inpainting region Ω and using equations (5) and (10) to propagate the information into the region Ω.

There is no strong argument about which of these two alternatives are the better. In fact, numerical experiments show that one produces better result for some images, while the other one produces better for some other images. In most of the tests given in this work, we have used the boundary conditions (5) and (10).

In the following, we shall even propose another boundary condition to treat some special situations. For some images, we may want some of the information from the surrounding area to be propagated into Ω, while some other information from the surrounding area is not welcome to be propagated into the region Ω, see Figures 10, 12, 13. In order to deal with this kind of situation, we propose the following alternative:

- Decompose the boundary $\partial\Omega$ into two parts, i.e. $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$. For equation (5), replace the boundary condition by

  $$a)\ \vec{\tau} = \vec{\tau}_0 \text{ on } \partial\Omega_D, \quad b)\ \vec{\tau} = \vec{0} \text{ on } \partial\Omega_N, \tag{25}$$

  and replace the boundary condition of (10) by

  $$a)\ d = d_0 \text{ on } \partial\Omega_D \quad b)\ \frac{\partial d}{\partial \vec{\nu}} = 0 \text{ on } \partial\Omega_N. \tag{26}$$

Condition (26.b) means that we do not want to propagate any information through $\partial\Omega_N$. Due to the reason that $\nabla d^\perp \approx \vec{\tau}$, condition (26.b) implies that we must have condition (25.b) for $\vec{\tau}$ on $\partial\Omega_N$. A similar procedure can be performed for equations (4) and (9).

3 Numerical Experiments

First, we explain how to choose ϵ, Δt_1 and Δt_2 in numerical implementations. We add ϵ to the denominator to avoid dividing by zero in (20)-(21) and (23)-(24). If ϵ is chosen to be large, the computed image will be smoother. If ϵ is chosen to be too small, it may slow down the convergence. We have chosen ϵ to be the same in (20)-(21) and (23)-(24), but it will be different from example to example.

With large Δt_1 and Δt_2, the iterations will converge faster, but if they are too large, the scheme is unstable. For most occurrences $\Delta t_1 \approx 0.03$ will lead to convergence of the normal vectors. A smaller Δt_1 will also work, but more iterations might be necessary. If the normal vectors are smooth, Δt_2 is less sensitive and can be chosen to be large. If vector field is less smooth, Δt_2 must be smaller.

3.1 Example 1

In this example we test out our method on an image from a Norwegian newspaper. The image shows a man jumping from Jin Mao Tower, a 421 meters tall building in the city of Shanghai. We want to remove the man and restore the city in the background.

The first part of the code computes the normal vectors in the missing region. From Figure 3 we see that the vectors are propagating into the inpainting region in a smooth fashion. When $\Delta t_1 = 0.03$ and $\epsilon = 10$ are used, a steady state is reached after 3000 iterations using (13)-(14). If we use (15)-(16), less than 1000 iterations are needed to reach a steady state, see Figure 3 e) and Figure 3 f).

The second part reconstructs the image using the computed normal vectors. Figure 4 shows how the man gradually is disappearing during the iterations. With $\Delta t_2 = 0.15$ it takes 30000 iterations before a steady state is reached. In the resulting image the man has disappeared completely and the background is restored in a natural way. There are no discontinuities in the sky, and the skyline is almost a straight line. It is nearly impossible to detect that the sky and the skyline contains the missing region.

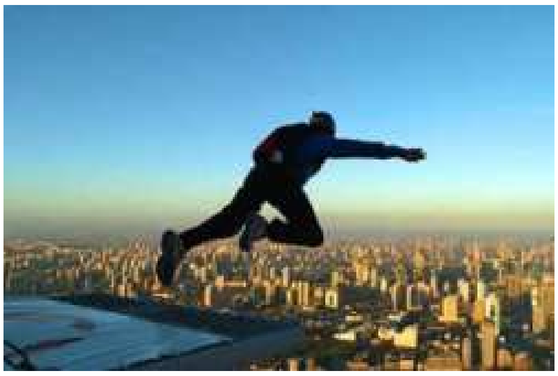
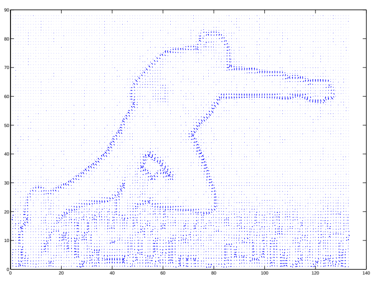
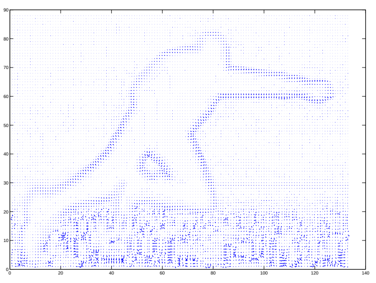


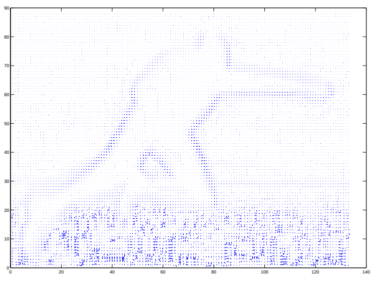
Figure 2: The original image.



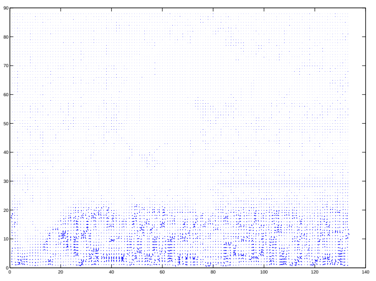
a)



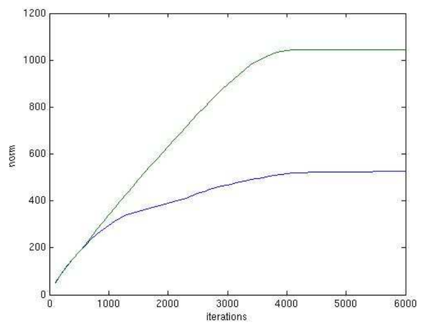
b)



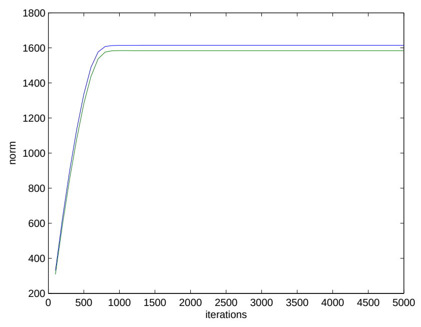
c)



d)

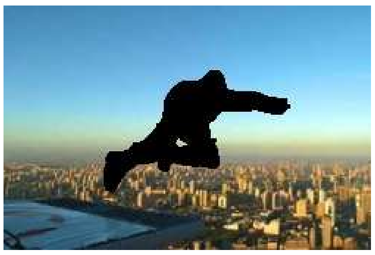


e)

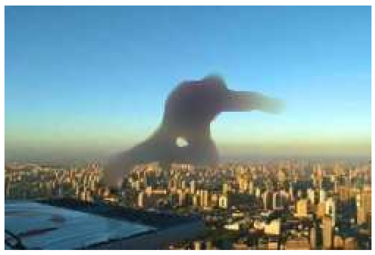


f)

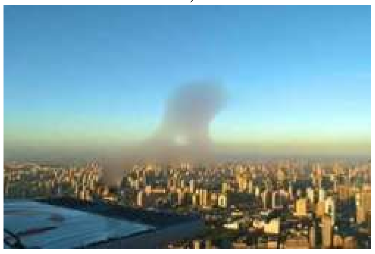
Figure 3: The restored flow vector $\vec{\tau}$ using (13)-(14) at different iterations. a) at iteration 0; b) at iteration 1000; c) at iteration 2000; d) at iteration 3000; e) The plot for $\|u\|$ and $\|v\|$ which shows that the equations (13)-(14) reach a steady state, i.e. at iteration 3000. f) In this plot, we show the convergence for $\|u\|$ and $\|v\|$ using equations (15)-(16). They reach steady states quicker than (13)-(14), i.e. at iteration 1000.



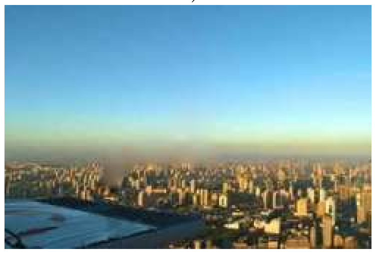
a)



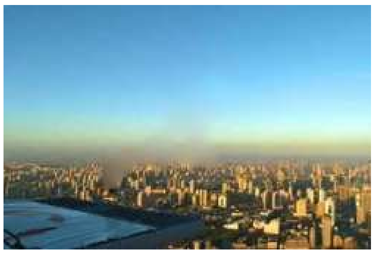
b)



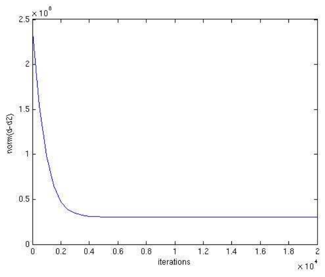
c)



d)



e)

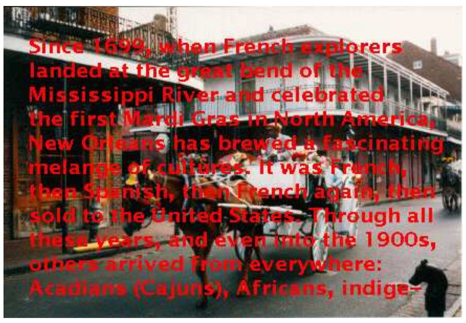


f)

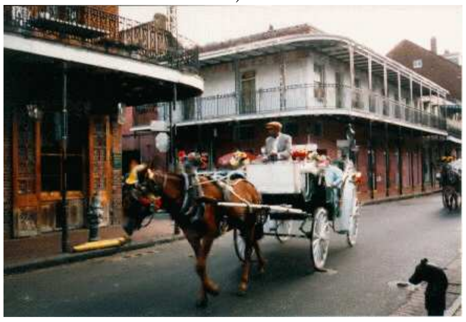
Figure 4: The restored image d using equation (10) at different iterations. a) at iteration 0; b) at iteration 10000; c) at iteration 20000; d) at iteration 30000; e) The restored image using the new method (15)-(16) to find $\vec{\tau}$. f) The plot for $\|d - d_0\|$ which shows that the equation (5) reaches a steady state, i.e. at iteration 30000.

3.2 Example 2

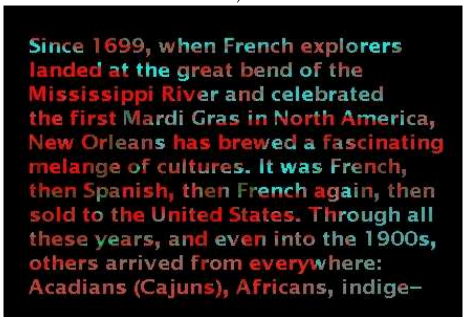
We test our method on some well-know examples which have been tested by others using different methods [2]. We use these results to show the quality of the restored images compared with other methods.


Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

a)



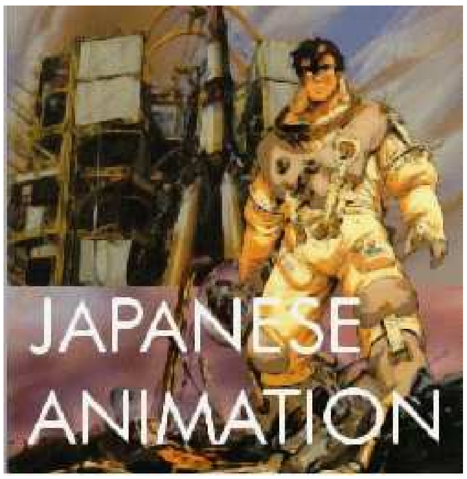
b)


Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

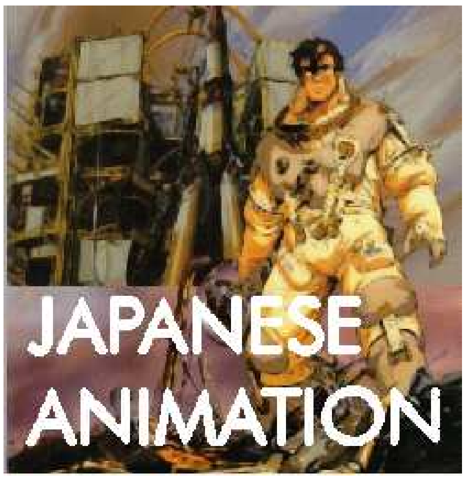
c)

Figure 5: a) The original image. b) The restored image using equations (5) and (10). c) The difference image.

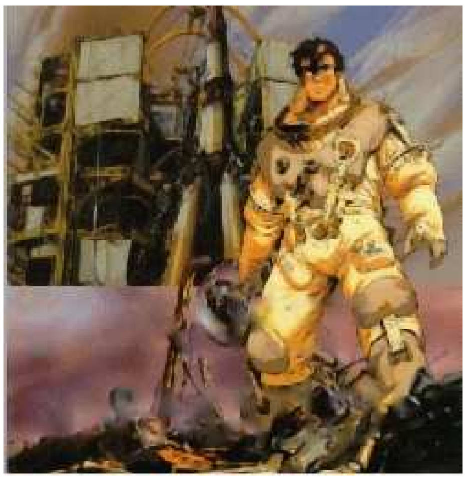
In this example as shown in Figure 5, red text is written over the picture. The text is the inpainting area, and we want to fill it with information from the image. With $\epsilon = 1$ and $\Delta t_1 = 0.03$ the normal vectors converge after 7000 iterations for (13)-(14). The second part of the code converged after only 3000 iterations with $\Delta t_2 = 0.5$.



a)



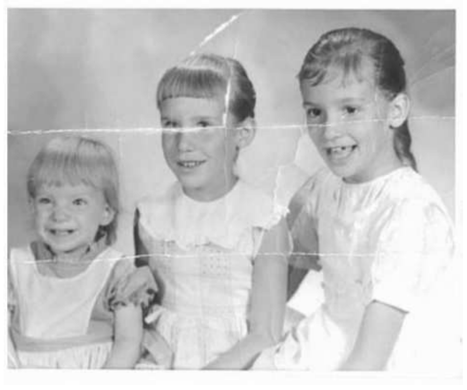
b)



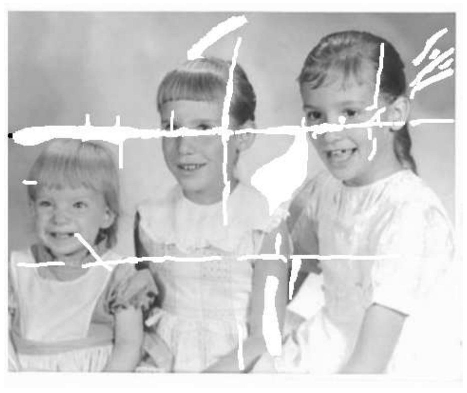
c)

Figure 6: a) The original image. b) The image with the inpainting region obtained manually. c) The restored image using equations (5) and (10).

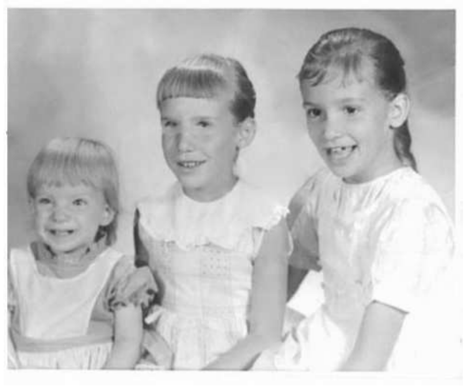
In Figure 6, another image which has been tested in the literature, is used here to compare our method with the others, [2, 1]. The image has the white text 'Japanese animation', and we want to remove this. An area around the text is lighter than the background and has to be restored as well. Figure 6 b) shows the manually obtained inpainting region. Figure 6 c) shows restored image. The values for Δt_1 and Δt_2 are chosen to be the same as in the previous example, and the convergence is nearly the same.



a)



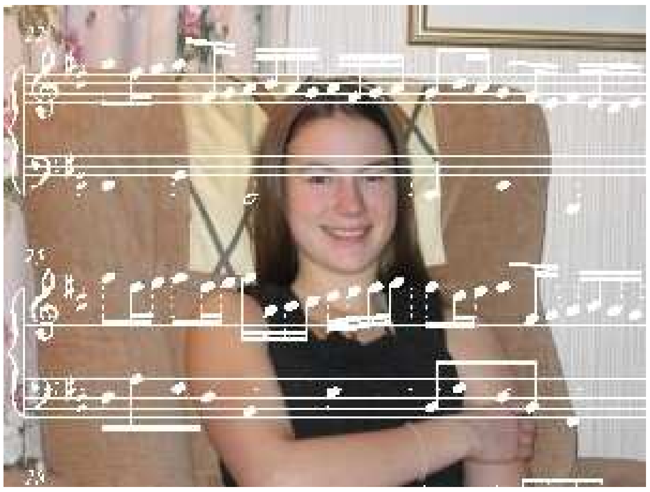
b)



c)

Figure 7: a) The original image d_0. b) The image with the inpainting region white. c) The restored image d.

Figure 7 a) shows an old photo which has been damaged. We mark the inpainting region in white colour, as shown in Figure 7 b) and try to restore it. The result is shown in Figure 7 c).


a)

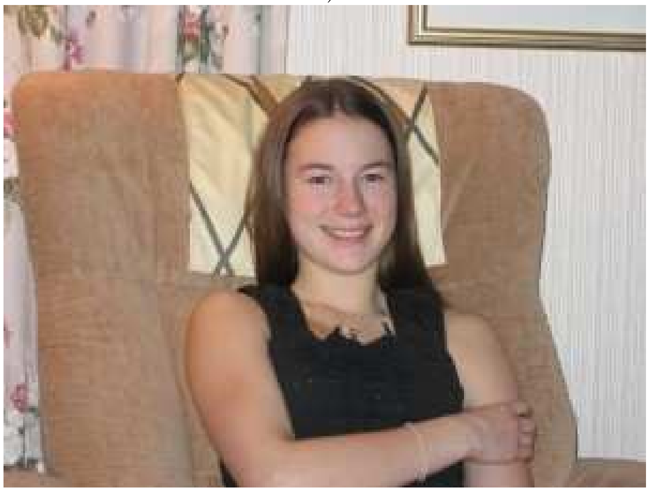

b)

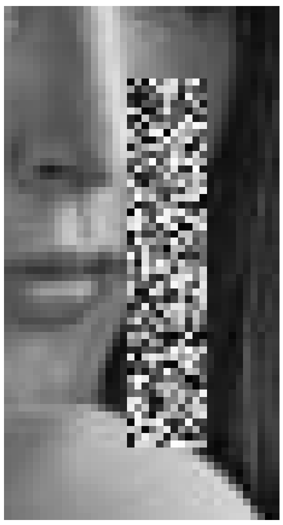
Figure 8: a) The image with the inpainting region white. b) The restored image using equations (5) and (10).

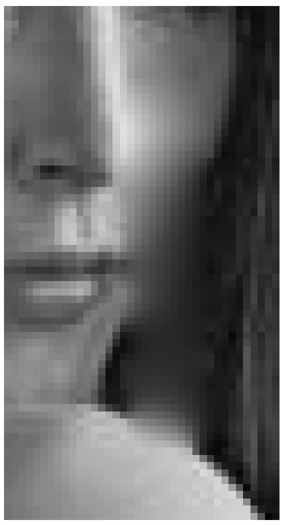
The image in Figure 8 a) shows another situation where our algorithm can be applied. The image has a piece of musical notes written on it. A large amount of information is lost, but it is scattered on the image in narrow areas. The first part converges after 2500 iterations and the second part converges after 1000 iterations when using our algorithm for this image. The restored image in Figure 8 b) looks rather good.

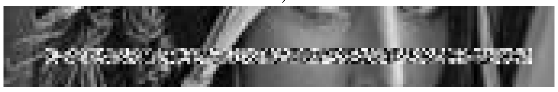
3.3 Example 3

Figure 9 a) shows a picture of Lena. We have removed a rectangle containing a part of her face and shoulder, Figure 9 a), and restored it in 9 b). The image has converged, but it is still possible to see where the image has been restored. The line connecting the shoulder has diffused to the background. In Figure 9 c), a long narrow rectangle is removed. Here the result is better. In Figure 9 d) it is only possible to see a discontinuity on her hat and on her nose.

In both examples $\epsilon = 1$, $\Delta t_1 = 0.03$ and $\Delta t_2 = 0.3$. The first part converged after about 7000 iterations, while the second part converged after 70000 iterations in Figure 9 b) and 10000 iterations in Figure 9 d).


a)


b)


c)

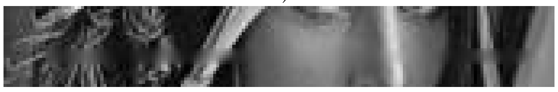
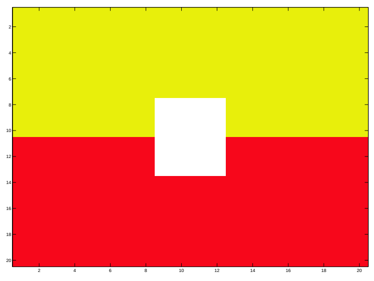

d)

Figure 9: a) The image with the inpainting region removed. b) The restored image using only Dirichlet boundary condition. c) The image with the inpainting region removed. d) The restored image using only Dirichlet boundary condition.

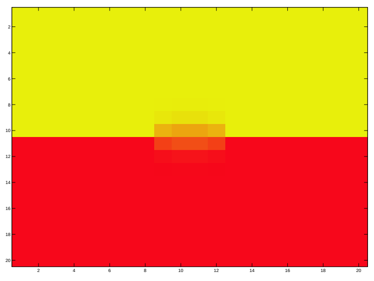
3.4 Example 4

To test the code for the new boundary condition (25)-(26), we made a simple image, see Figure 10. Information is missing in a rectangle in the middle of the image which only has two intensity values. If we use Dirichlet boundary conditions (5)-(10), all information from the surrounding area will be transported into the inpainting region. If the Neumann boundary is used (25)-(26), it is possible to choose which intensity value to be selected to propagate into the inpainting region. The result is shown in Figure 10. The result using Dirichlet boundary conditions is displayed in Figure 10 b). With ϵ=0.0001, $\Delta t_1 = 0.01$, the normal vectors converged after 12000 iterations and with $\Delta t_2 = 0.2$ the second part converged after 25000 iterations. With a larger ϵ, the corners and the boundary close to the corners may be smeared.

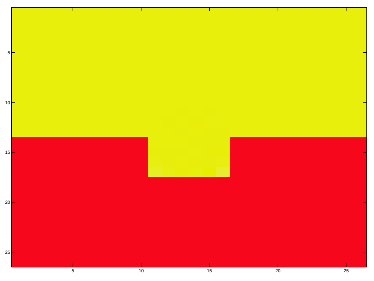
Figure 10 c) shows a similar test with Dirichlet conditions on the upper half and with Neumann boundary conditions on the lower half of the boundary of the inpainting region. From Figure 10 c) we see that both colours have propagated to the interior.



a)



b)



c)

Figure 10: a) The image with the inpainting region marked. b) The image obtained with Dirichlet boundary. c) The image obtained using Dirichlet and Neumann boundary conditions.

3.5 Example 5

In this example, we process an image from the match between Norway and Croatia in the XIX Men's World Championship. We want to remove the croatian player in Figure 11. When a Dirichlet condition is used around the whole boundary, Figure 12 a), colours from the Norwegian players propagate into the background. To make it look natural, it is necessary to use Neumann boundary conditions around the two Norwegian players. The inpainting region and the Neumann boundary are marked in Figure 12 b). Figure 12 c) shows the restored image using this new boundary condition. When Neumann boundary condition is used, the colour on the Neumann boundary does not influence the interior.

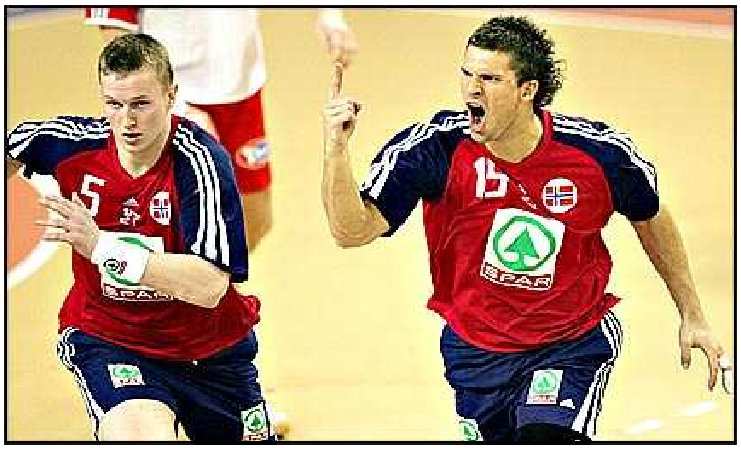
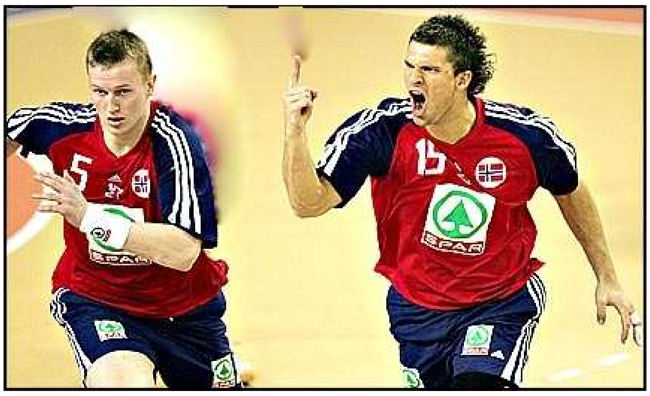
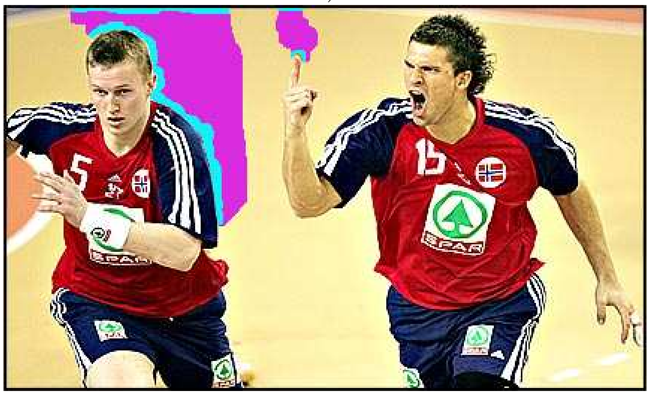


Figure 11: An image from the match between Norway and Croatia in the XIX Men's World Championship.


a)


b)

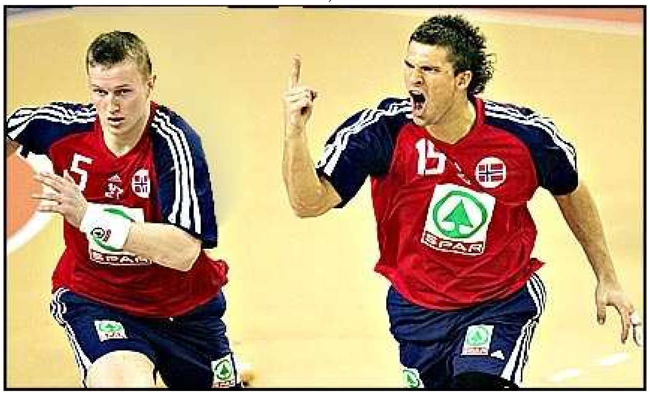

c)

Figure 12: a) The restored image using Dirichlet boundary conditions. b) The image with the inpainting region violet. c) The restored image using Dirichlet and Neumann boundary conditions.

3.6 Example 6

This example has more texture in the background. We want to remove the snowboarder and fill in the missing region. It is not desirable that the yellow object in the front propagates into the inpainting region. Figure 13 d) shows that the best result is obtained with Neumann conditions on part of the boundary.

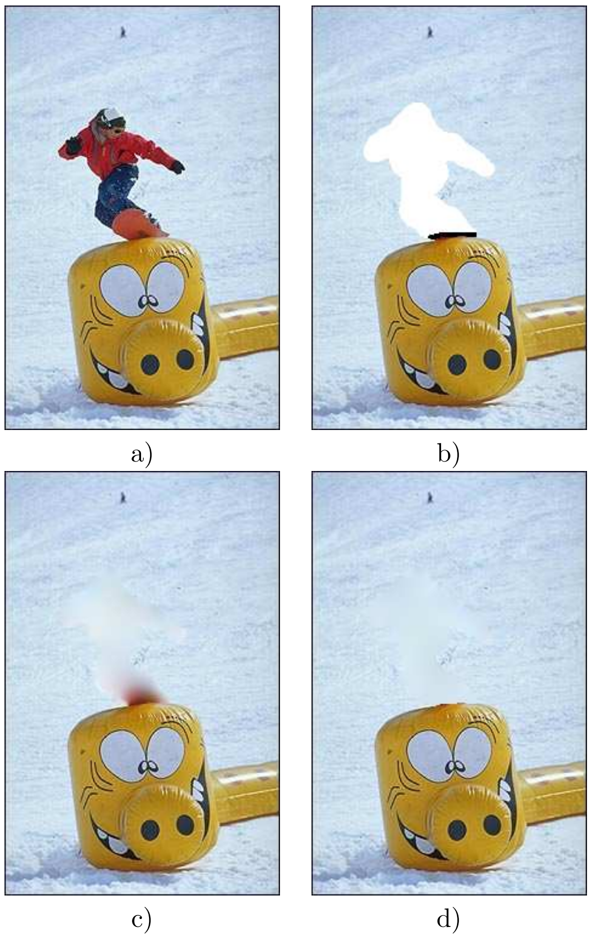


Figure 13: a) A photo taken by Espen Lystad, a well-known snowboard photographer in Norway. b) The image with the inpainting region marked. The Neumann boundary is black. c) The restored image only using Dirichlet boundary condition. d) The restored image using Dirichlet and Neumann boundary conditions.

3.7 Example 7

In several of the proceeding examples, we have tried to show the advantages of our algorithms. In this example, we shall give a more qualified comparison between our method and other methods proposed in the literature. Figure 14 a) shows the image with the inpainting region. The restored image using the method of [8] and our method are shown in Figure 14 b) and c) respectively. In the computation, we have used $\Delta t_1 = 0.03$, $\Delta t_2 = 0.2$ and $\epsilon = 0.0001$. To show more details, we display the zoomed images in Figure 15. When the two methods are compared, we see that the spiral is most visible using the TV-inpainting method [8], see Figure 14 b).

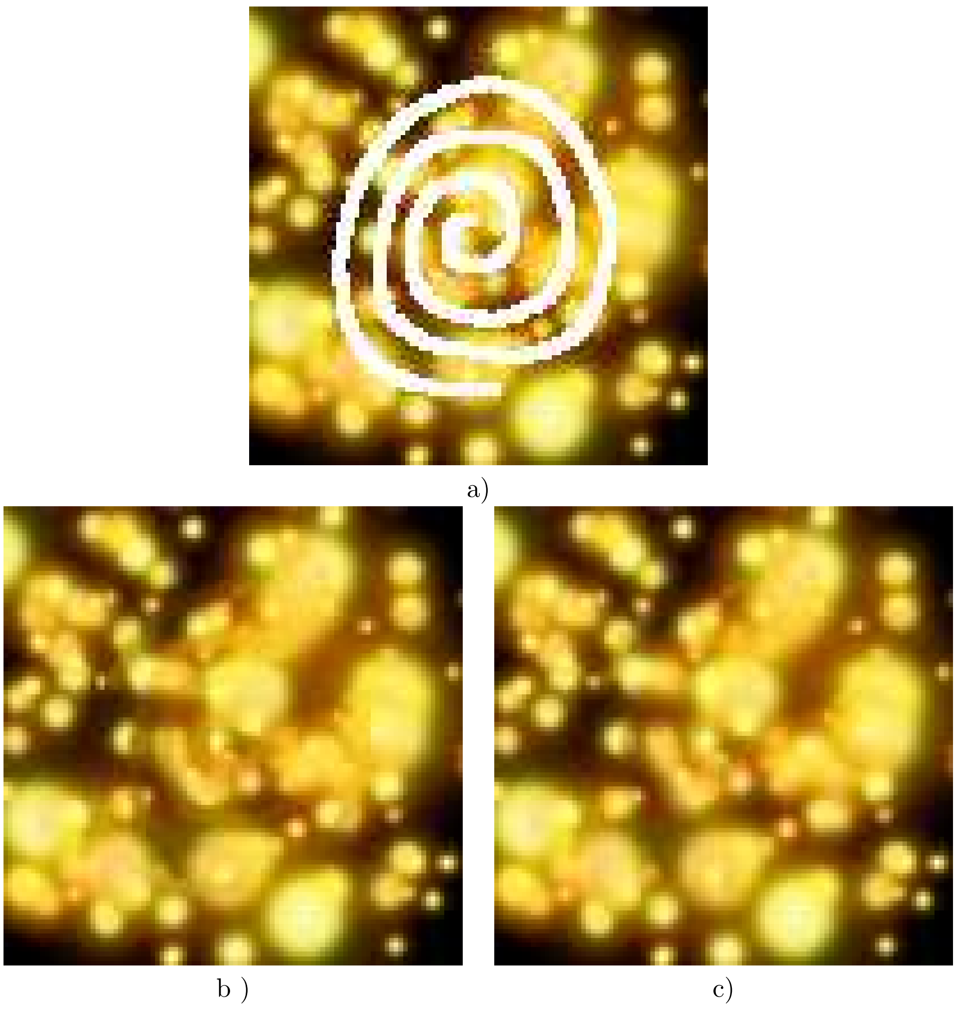
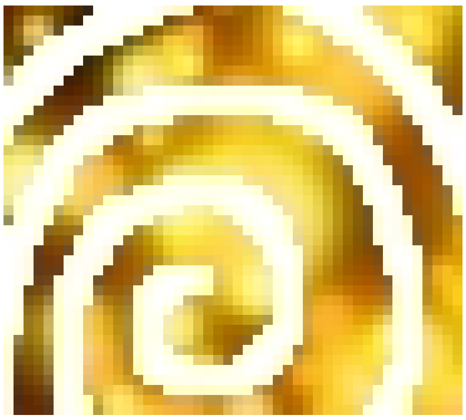
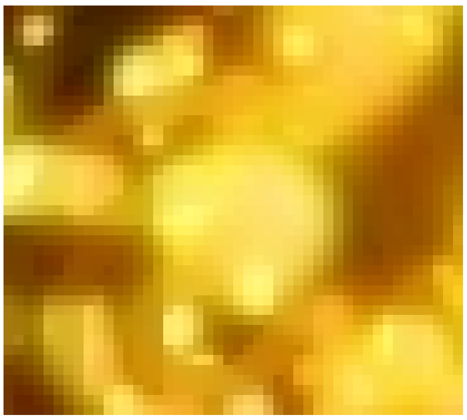


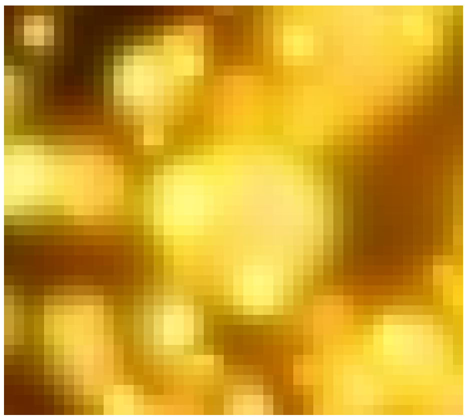
Figure 14: a) The original image d_0. b)The restored image using the method of [8]. c) The restored image using our method.



a)



b)



c)

Figure 15: a) The original image d_0 zoomed in. b)The restored image using the method of [8] zoomed in. c) The restored image using our method zoomed in.

References

[1] Criminisi A., Perez P., and Toyama K. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Process.*, 13(9), 2004.

[2] C. Ballaster, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Trans. Image Processing*, (10):1200–1211, 2000.

[3] M. Bertalmio, A. L. Bertozzi, and J. Sapiro. Navier-stokes, fluid dynamics and image and video inpainting. In *Proc. Conf. Comp. Vision Pattern Rec.*, pages 355–362, 2001.

[4] M. Bertalmio, G. Sapiro, C. Ballaster, and V. Caselles. Image inpainting. *Computer Graphics, SIGGRAPH*, 2000.

[5] Vicent Caselles, Simon Masnou, Jean-Michel Morel, and Catalina Sbert. Image interpolation. In *Séminaire sur les Équations aux Dérivées Par-*

tielles, 1997–1998, pages Exp. No. XII, 15. École Polytech., Palaiseau, 1998.

[6] Tony Chan and Jianhong Shen. Variational restoration of nonflat image features: Models and algorithms. *SIAM J. Appl. Math.*, 61(4):1338–1361, 2000.

[7] Tony F. Chan, Sung Ha Kang, and Jianhong Shen. Euler's elastica and curvature-based inpainting. *SIAM J. Appl. Math.*, 63(2):564–592, 2002.

[8] Tony F. Chan and Jianhong Shen. Mathematical models for local nontexture inpaintings. *SIAM J. Appl. Math.*, 62(3):1019–1043, 2002.

[9] Tony F. Chan, Jianhong Shen, and Luminita Vese. Variational PDE models in image processing. *Notices Am. Math. Soc.*, 50(1):14–26, 2003.

[10] F. Guichard and L. Rudin. Accurate estimation of discontinuous optical flow by minimizing divergence related functionals. *Proceedings of International Conference on Image Processing, Lausanne, September, 1996*, page 497, 1996.

[11] Grossauer H. and Scherzer O. Using the complex G-inzburg-Landau equation for digital inpainting in 2D and 3D. In *Sacle space methods in computer vision*, Lectures notes in Computer Sciences 2695. Springer, 2003.

[12] Shen J. L gamma-convergence approximation to piecewise constant mumford-shah segmentation. Tech. rep. (05-16), 2005.

[13] Weickert J. *Anisotropic Diffusion in Image Processing.* Stutgart, B. G. Teubner, 1998.

[14] Johan Lie, Marius Lysaker, and Xue-Cheng Tai. A binary level set model and some applications to image processing. *IEEE Trans. Image Processing*, to appear.

[15] Johan Lie, Marius Lysaker, and Xue.-Cheng. Tai. A variant of the levelset method and applications to image segmentation. *Math. COmp*, to appear.

[16] T. Lu, P. Neittaanmaki, and X-C Tai. A parallel splitting up method and its application to navier-stoke equations. *Applied Mathematics Letters*, 4:25–29, 1991.

[17] T. Lu, P. Neittaanmaki, and X-C Tai. A parallel splitting up method for partial differential equations and its application to navier-stokes equations. *RAIRO Math. Model. and Numer. Anal.*, 26:673–708, 1992.

[18] M. Lysaker, A. Lundervold, and X. C. Tai. Noise Removal Using Fourth-Order Partial Differential Equation with Applications to Medical Magnetic Resonance Images is Space and Time. *IEEE Transactions on Image Processing*, 12(12):1579–1590, 2003.

[19] M. Lysaker, Stanley Osher, and Xue-Cheng Tai. Noise removal using smoothed normals and surface fitting. *IEEE Trans. Image Processing*, 13(10):1345–1457, 2004.

[20] M. Bertalmio M., Vese L., Sapiro G., and OSher O. Simultaneous texture and structure image inpainting. *IEEE Trans. Image Process.*, 10(8), 2003.

[21] Simon Masnou. Disocclusion: a variational approach using level lines. *IEEE Trans. Image Process.*, 11(2):68–76, 2002.

[22] J. Weickert, B. H. Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE TRans. Image Process.*, 7:398–409, 1998.