Universität Leipzig
Fakultät für Mathematik und Informatik
(Institut für Informatik)

Projection Technique For Vortex-Free Image Registration

Diplomarbeit

Leipzig, Februar 2007

Vorgelegt von: Scheibe, Patrick
geb. am: 5. November 1979
Studiengang: Informatik

Betreuender Hochschullehrer: Dr. Jens-Peer Kuska

ABSTRACT

One important application of image processing in medicine is to register tissue samples onto another. Registering these high textured images with non-parametric methods leads sometimes to solutions which are known to be suboptimal.

This thesis is concerned with a novel approach for image registration. We present a projection technique for a curvature based non-parametric registration method which suppresses unwanted vortices in the displacement field. This new strategy does not change the registration method itself but it continuously *leads* the process of registration to a vortex-free solution.

The grounding method was introduced in [Ami94], used in [BK05] and extended in [BK06] with a vortex suppression term. Our new method calculates a *Helmoltz decomposition* on the intermediate steps and projects out unwanted vortices.

This thesis describes the whole process of the image registration. Starting from the mathematical description of the Helmholtz decomposition, its variational presentation and the consequential partial differential equation, we will go on by looking at the discrete approximation, parts of the implementation and the application on images.

Finally the results in comparison with the two other methods of Kuska and Braumann [BK05, BK06] are presented. For this purpose, samples are given and deformed with an artificial transformation. We will use these results and discover general properties, advantages and drawbacks of the different approaches.

ACKNOWLEDGEMENTS

Various people supported me on the way to this thesis. I want to express my gratitude by dedicating these lines to them.

First of all my thanks to all of my family who gave me support in everything I did. Especially my mother who supported me unconditionally and gave me backing for my ideas. I am so happy of having them around me every day.

Many discussions with my friends helped me to revise and improve things. Especially Georg Martius who shared the years of study with me had more than often a sympathetic ear for my problems. The opportunity of discussing with him was a big enrichment for me.

I would like to give a very big thank to the reviewers too. Sophia Schumann, Allan D. Clark and Georg Martius found many mistakes and helped to bring this thesis in an appropriate form.

Finally I wish to thank Jens-Peer Kuska. He supervised me in an inimitable good way but with the words of Umberto Eco:

> 'It is unnecessary to thank the supervisor. In helping you he has only complied with his responsibility.'[Eco05, p. 228]

That is the reason why my concern is to thank him for being my teacher for more than four years. His advice and his ability to explain let me see things in a different light. If I came forward in understanding incomprehensible topics then it was grounded in his constant effort to share his knowledge and to explain it in a simple way.

CONTENTS

CHAPTER 1

INTRODUCTION

PUBLIC NOTICE AS REQUIRED BY LAW:
Any Use of This Product, in Any Manner Whatsoever, Will Increase the Amount of Disorder in the Universe. Although No Liability Is Implied Herein, the Consumer Is Warned That This Process Will Ultimately Lead to the Heat Death of the Universe. [HS91]

One of the biggest difficulties in software engineering for medical image applications is to enable a computer to perform tasks which a physician can do in a fraction of a second. A surgeon for example is able to identify a fracture on a radiograph in a very short time. A pathologist would not need more than a few moments to mark and rate cancer when doing a biopsy. All of these processes are very important in medicine and at least some of them can be done automatically by a program.

Ulf-Dietrich Braumann et al, published an article in 2005 in which they describe an algorithm to calculate a three-dimensional reconstruction of a tumor invasion front:

> ... the intention of this paper is to get an objective quantification of tumor invasion based on three-dimensionally reconstructed tumoral tissue data. The image processing chain introduced here is capable to reconstruct selected parts of tumor invasion fronts from histological serial sections of remarkable extent (90-500 slices)[BKE+05a]

The procedure involves taking a paraffin-embedded uterine cervix sample and slicing it. The resulting slices are photographed and the digital images run through steps of registration and segmentation. The goal of the registration steps is to find a transformation for two adjacent images so that they fit upon one another. By putting the registered and segmented images "digital-onto-another", a three-dimensional reconstruction of the included carcinoma can be developed. The detailed description of the whole process can be found in [BKE+05a].

In this thesis we will deal with one particular step. The non-parametric curvature-based registration of the tissue images. This is the last step in a chain of registration runs containing a first rigid registration and a polynomial non-linear registration. A main aim of this thesis is firstly to explain why the curvature-based registration needs improvement and secondly to show the solution procedure and to give examples showing the advance.

In the first part the reader will be introduced to the mathematics used. Furthermore a classification of registration methods is given and in this context some popular image registrations are explained.

The second part of the thesis contains the motivation and the solution procedure of our novel approach. We will give an insight into the mathematical depiction, the discretization for digital images and parts of the implementation and conclude with the results which contain a comparison to the already existing methods.

CHAPTER 2

MATHEMATICAL SETTING

ADVISORY:
There is an Extremely Small but Nonzero Chance That, Through a Process Know as 'Tunneling', This Product May Spontaneously Disappear from Its Present Location and Reappear at Any Random Place in the Universe, Including Your Neighbor's Domicile. The Manufacturer Will Not Be Responsible for Any Damages or Inconvenience That May Result. [HS91]

To understand the details of our new registration method, a basic insight into mathematics is required. Especially vector and variational calculus will play an important role. Since one concern is to address this thesis to a wide audience most of the mathematical methods will be explained detailed. Thereby as much hints and illustrations as possible are given to prevent wrong assumptions and conclusions.

We try, while recognising it is a difficult task, to give a closed, step by step representation of the mathematics chapter ordered by difficulty. Especially in the section *Image Registration* this is not always possible because it needs the calculus of variations. Furthermore we make an effort to use a uniform notation so that the reader can gather the meaning of formulas and symbols easily. The summary of these notations can be looked up in the appendix.

2.1 DIGITAL IMAGES

From the mathematical point of view an image is a function that maps a coordinate from a set $\Omega \subset \mathbb{R}^d$ to color, gray-scale or other physical values where $d \in \mathbb{N}$ is dimension of the domain. Since we only consider gray value images of the dimension two, the *image function* or *image* R maps points to gray values:

$$R\colon\ \mathbb{R}^2 \to \mathbb{R}. \tag{2.1}$$

The domain of images is in general not limited to dimension two. Data from a MRI for instance are often volume data sets. Therefore one gets a value for every point in the scanned body. That is the reason why this image function would have a domain in the three-dimensional space. In contrast to that, normal x-ray pictures show only two-dimensional versions of three-dimensional body parts.

The definition of an image $R(\boldsymbol{x})$ states the following properties:

- The image $R(\boldsymbol{x})$ exists only in a compact area $\Omega \subset \mathbb{R}^d$.
- An image has an appropriate value on every point. More exactly, the image data must be in the interval $0 \leq R(\boldsymbol{x}) < \infty$ for all $\boldsymbol{x} \in \mathbb{R}^d$.
- The integral $\int_{\mathbb{R}^d} R(\boldsymbol{x})^k d\boldsymbol{x}$ where $k > 0$ must be finite.

In contrast to purely mathematical definition most pictures have no continuous representation of their image function. Image data from a CCD camera, magnetic resonance imaging or computer tomography are *sampled*, that is only a set of values from the image

function is available. To get a closed form of the image function again, we could interpolate the values. Hence we get a closed representation, for instance as polynomial, and therefore we can treat every image as a continuous function.

The last paragraph described how to get closed form from sampled data. Sometimes it is necessary to have only a few values from a continuous picture. This may be the case for instance when it is printed on a screen. We will call this *sampling* and consider it as the evaluation of the image function on a defined set of points. In figure 2.1 for instance an image is given which is sampled 15 times in every direction. The value on every grid point is called *pixel* . Here we have 15 pixels in every direction and so the size of this sampled image is 15×15 pixels. Note that the expression *dimension of an image* defines the size and not the domain.

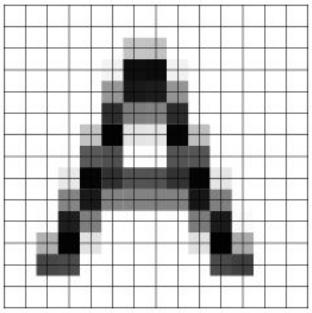


Figure 2.1: An example of a digital image. Every grid point has a gray-level value.

As one can see the terms *digital image* and *image function* are manifold and have more applications than the common word *image* would suggest. We will use these terms interchangeably but in almost every case we mean a function $R : \mathbb{R}^2 \supset \Omega \to \mathbb{R}$ that represents the digital image.

2.2 IMAGE REGISTRATION

Image registration is a very important discipline in the field of image processing. The name is commonly used to describe the process of making two or more images *similar* in a certain sense. Let us introduce a more formal definition by giving first an example of a real-world registration problem.

Assume you have two air photos of a town. One of the photos is up to date, but the other one was taken years ago. For the pilot who took the newer one it was impossible to have exactly the same position and direction as the former pilot.

Now the task is to compare these two photos in an automated process. This seems to

be difficult when the photos are not similar because even if they show the same place, the computer cannot recognize this. It is first necessary to bring *one photo upon the other*. In the simplest case this is a rotation or a translation of the newer photo so that same streets, buildings, etc. have the same coordinates in both pictures.

The process of bringing two pictures into one coordinate system by applying a transformation to one is called *image registration*. The image that is used as basis for the registration is called *reference* R whereas the other one is called *template* T.

To choose the best transformation $\mathfrak{T}$ that maps the template onto the reference, a criterion D is required that describes the quality of $\mathfrak{T}$. With the criterion two different transformations can be compared. An often used criterion is the *distance* of the reference R and the transformed template $\mathfrak{T}(T)$. The task is then to find the transformation $\mathfrak{T}_{\min}$ in a set Θ of possible functions that minimizes the criterion D between the sample and the template.

$$\mathfrak{T}_{\min} = \min_{\mathfrak{T}\in\Theta} D(R, \mathfrak{T}(T)) \tag{2.2}$$

The transformation $\mathfrak{T}$ takes effect on the image by transforming the coordinates. It corrects or displaces the points of the image. This is given in a displacement field $\boldsymbol{u} : \mathbb{R}^2 \to \mathbb{R}^2$ which provides the displacement vectors for every point of the image. The registered template is then given by

$$T'(\boldsymbol{x}) = T(\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x})) \tag{2.3}$$

The two images in figure 2.2 are an example for an easy registration problem. To register the sample onto the template there is nothing more to do than to move the 'A' along the arrow. After this translation the two images are congruent and their *distance* vanishes.

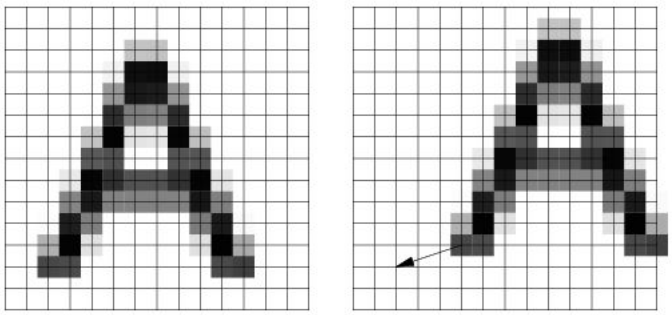


Figure 2.2: An example of two images which can easily be registered onto another.

The choice of those distance measures can be manifold. One commonly used function

for it is the sum of squared differences.

$$\mathcal{D}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega}(R(\boldsymbol{x}) - T(\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x})))^2 d\boldsymbol{x} \tag{2.4}$$

Equation 2.4 is often used as at least one part of the registration criterion. In many cases the registration has to minimize a sum of criteria, where every part represents another property. Notice that $\mathcal{D}$ has the signature $\mathcal{D}\colon\ \mathbb{R}^d \to \mathbb{R}$. In the variational calculus this is called a *functional* and can be interpreted as a measure. In this case it measures the similarity of two images when a transformation $\boldsymbol{u}$ is applied.

One important thing to understand is that there is not necessarily just one transformation which fulfills the criterion $\mathcal{D}$. Potentially there exists more than one, or even infinitely many of them. With respect to our last example it is not necessary to move *all* pixels of the template. It would not change anything in the result when some of the surrounding white pixels are not displaced. At this point the reader may figured out another dilemma. When a registration uses the difference $\mathcal{D}$ as the only criterion, then a transformation would minimize $\mathcal{D}$ if it does the following for all points $\boldsymbol{x}$ of the reference R:

- Find a point $\boldsymbol{x}_s$ in the template T having the same or a similar *value*. From this it follows that $R(\boldsymbol{x}) = T(\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x}))$ with $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}_s$.
- Returning this displacement vector for the point, ensuring that now this point is excellent registered.

This strategy produces probably a displacement field where the vectors are strewn at large. But it definitely minimizes the *distance* of the two images. According to the definition of image registration on page 6 this transformation would be of *outstanding quality*.

This problem is grounded in the fact, that our transformation is only restricted by the registration criterion that represents the distance. If the transformation would be searched only in a subset of all conceivable functions, it can force the solutions to be *good*. In the case of this method, where the solution function is only restricted by fulfilling the criterion $\mathcal{D}$, it seems there is something more to do. The used criterion lacks an important detail that suppresses clutter in the transformation.

Even in the case of the two air photos humans know much more about how the transformation must look. Maybe one map is a bit rotated, shifted or stretched, but we know the pixels of the transformed map must kind of stay together. We do not want displacement rules which *destroy* the template too much.

For this purpose we consider another functional to measure the *smoothness* of a displacement field $\boldsymbol{u}$. Many different definitions of such a term are possible but all of them have in common that they restrict the result in a certain sense.

One important smoother we will use in this thesis was firstly introduced by [Ami94]. It makes use of the directional derivatives of the transformation which basically described how fast $\boldsymbol{u}$ changes. If we restrict the change of $\boldsymbol{u}$, we force the displacement to transform points which are at close quarters similarly.

The smoother of Amit is given in equation 2.5. It uses the not yet introduced *Laplace operator* Δ which covers the used directional derivatives. Hence readers which are not familiar with the notation may return to this point again after checking the next section where a detailed introduction to vector calculus is given.

$$\mathcal{S}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega}(\Delta u_1)^2 d\boldsymbol{x} + \frac{1}{2}\int_{\Omega}(\Delta u_2)^2 d\boldsymbol{x} \tag{2.5}$$

2.3 Displacement Fields and Vector Calculus

For most registration methods it is not necessary to use displacement fields to represent the transformation. In the case of a rigid registration for example it is far more effective to use matrices. The method used in this thesis is a non-linear one and displacement fields are a suggestive way to represent the transformation. Therefore they were introduced in the last section, even for the given easy example.

A solution *displacement field* for an image registration is a vector field which assigns every point on the image plane to a vector. It is the displacement rule for the assigned point. To apply a transformation to an image one has to move every pixel of it along its displacement vector.

Since images here are two-dimensional the type of such a field $\boldsymbol{u}$ is in general

$$\boldsymbol{u}: \quad \mathbb{R}^2 \supset \Omega \rightarrow \mathbb{R}^2 \tag{2.6}$$

So $\boldsymbol{u}$ is the short form of the vector-valued function $\boldsymbol{u}(\boldsymbol{x})$ that maps a point $\boldsymbol{x} \in P$ to a vector $\boldsymbol{u}(\boldsymbol{x})$ [MV90, p. 421].

There are various ways to visualize vector fields. Two methods are used in this thesis to show features and compare fields. To enable the reader to interpret them, these techniques will be explained in detail with examples in the following pages.

The first method to display vector fields is to draw an arrow of the direction $\boldsymbol{u}(\boldsymbol{x})$ at selected points $\boldsymbol{x} \in \Omega$. These arrows represent the field $\boldsymbol{u}(\boldsymbol{x})$. The length of the arrow is related to the length of the vector. In some cases it is better to use color to represent the length of the vectors and leave the arrows at the same size.

We will call this visualization method simply *vector field plot*. It is used for fields where only a few vectors are drawn, i.e. where the number of arrows one can draw before

the whole plot becomes too confusing is limited.

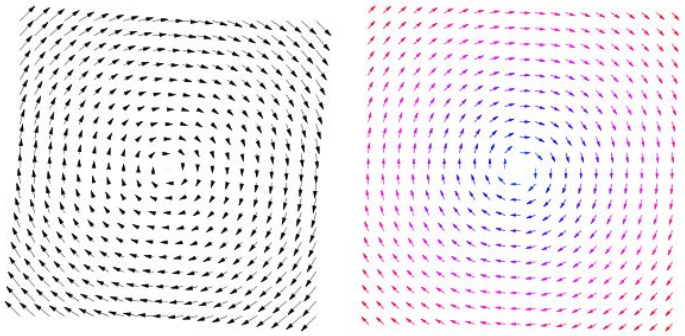


Figure 2.3: The vector field $\boldsymbol{u}((x_1, x_2)^{\mathrm{T}}) = (-x_2, x_1)^{\mathrm{T}}$ plotted in two different ways. The left picture shows the lengths of the vectors as lengths of the drawn arrows while the right one uses colors.

Another often used visualization method for vector fields is the *line integral convolution* (LIC). First introduced 1993 [CL93] it became a very popular type of visualization, especially for high density vector fields. The LIC has some advantages over the normal vector field plot. Very complex situations of the field cannot be displayed with all details using arrows. Especially in parts of the field like in the upper middle of the pictures in figure 2.4 the LIC gives a much better impression of what is going on.

In a line integral convolution the field is displayed in a way a painter would do it with a brush by following the direction of the vectors. To display the different vector lengths it is possible to use different colors. In the sample figure 2.4 and in many other LICs in this thesis the color spectrum goes from blue to red. Short vectors are represented with a blue color in the LIC. The longer the vectors are the more red the color of this area becomes.

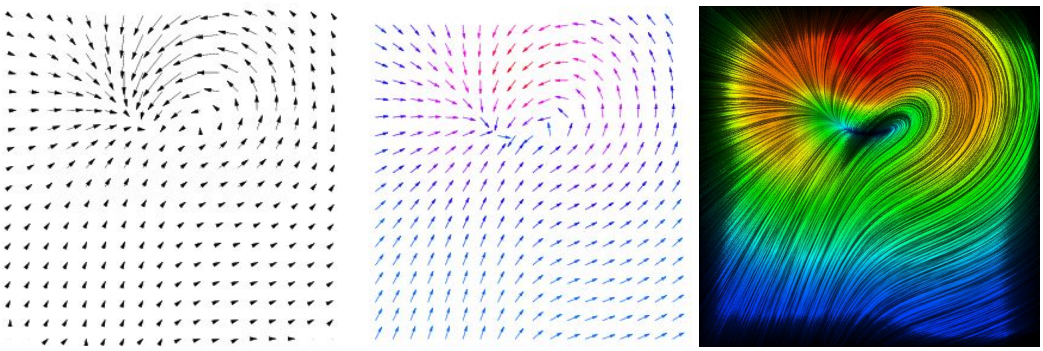


Figure 2.4: Two basic methods of vector field visualization. On the left two pictures the *field plot*. In the first case the length of the vectors is represented in the length of the arrows. The second case shows this property with different colors. The right picture shows the LIC of the same field.

There are two disadvantages of the line integral convolution; firstly the algorithm is computationally expensive, secondly the direction of the field is lost. Two different

charges are a good example for a field with a *source* and a *sink*[1]. Figure 2.5 demonstrates that it is not possible to distinguish the negative from the positive pole in the LIC.

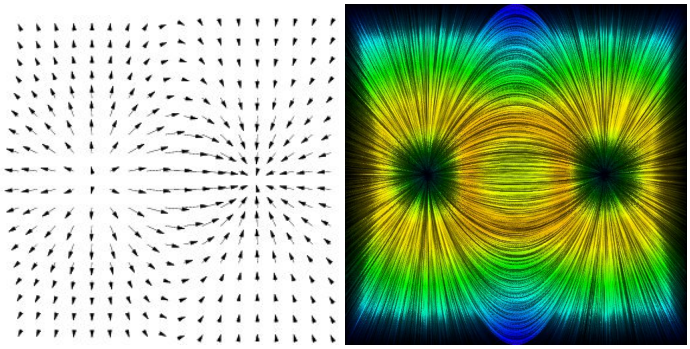


Figure 2.5: Example for the loss of the flow-direction in a line integral convolution.

The vector field plot contains this information, because one can easily tell where the arrows come from and where they end, but in a LIC both types of *singularities* would look equally.

Because the images used have always a size equal or greater than 256 pixels in every direction and the displacement field of the registration has the same dimension we are reliant on the LIC. It is possible to revert to a normal field plot but only in some very special cases or for demonstration purposes. However later we will see that the LIC of a displacement field is fully sufficient to examine the properties we are interested in.

In the next section we will introduce some properties of fields and we will develop how features can be expressed by operators. Most of those features are known by intuition. In a water stream for instance everyone knows what a swirl will look like. We will connect this knowledge with a mathematical description to use it in the later sections.

2.3.1 Spatial Differential Operations

Several basic operations on fields exist. In the next section we will introduce them. To simplify the notation of many formulas containing partial derivatives, we will make use of the *nabla operator* which is given by

$$\nabla := \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{pmatrix} \tag{2.7}$$

[1] Sinks and sources are discussed later when the *divergence* of a vector field is introduced.

THE GRADIENT

The gradient of a scalar-valued field is the vector field of its partial derivatives.

$$\text{grad } f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_2} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_3} f(\boldsymbol{x}) \end{pmatrix} \tag{2.8}$$

An example is given in the two-dimensional *scalar-valued* sample function $f(\boldsymbol{x}) = \exp(-(x_1^2 + x_2^2))$.

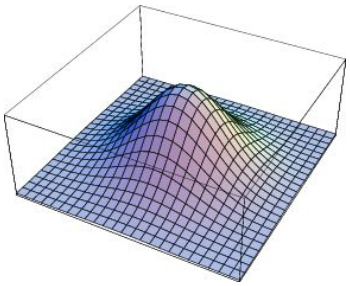


Figure 2.6: A three-dimensional plot of the scalar-valued function $f(\boldsymbol{x}) = \exp(-(x_1^2 + x_2^2))$.

For the gradient of f the partial derivatives with respect to x_1 and x_2 are required. The gradient of this field is then given by

$$\text{grad } f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) = \begin{pmatrix} -2x_1 \cdot \exp(-(x_1^2 + x_2^2)) \\ -2x_2 \cdot \exp(-(x_1^2 + x_2^2)) \end{pmatrix} \tag{2.9}$$

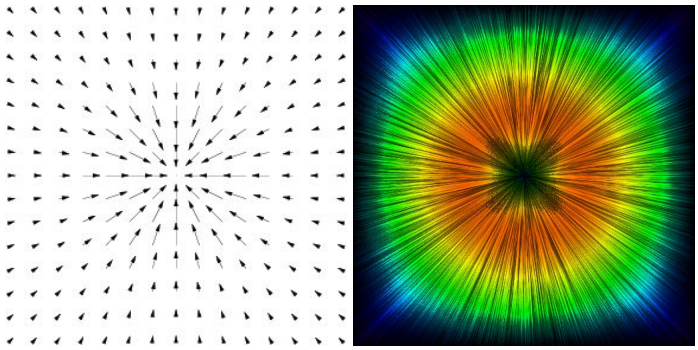


Figure 2.7: Field plot and line integral convolution of ∇f.

Figure 2.7 shows the gradient field and one can observe that it is the field where every vector points in the direction of the greatest alteration of f.

THE DIVERGENCE

For every vector field $\boldsymbol{f}(\boldsymbol{x})$ with the components

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} u_1(\boldsymbol{x}) \\ u_2(\boldsymbol{x}) \\ u_3(\boldsymbol{x}) \end{pmatrix} \tag{2.10}$$

we can define the *divergence* by

$$\operatorname{div} \boldsymbol{f}(\boldsymbol{x}) = \nabla \cdot \boldsymbol{f}(\boldsymbol{x}) = \frac{\partial}{\partial x_1} u_1(\boldsymbol{x}) + \frac{\partial}{\partial x_2} u_2(\boldsymbol{x}) + \frac{\partial}{\partial x_3} u_3(\boldsymbol{x}) \tag{2.11}$$

Let us point out the effect of this operator with a given example. Assume the field of equation 2.12 and its field plot which is shown in figure 2.8.

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} 2\, e^{-(1+x)^2-y^2}\ (1 - e^{4\,x}\ (-1 + x) + x) \\ -2\, e^{-(1+x)^2-y^2}\ (-1 + e^{4\,x})\ y \end{pmatrix} \tag{2.12}$$

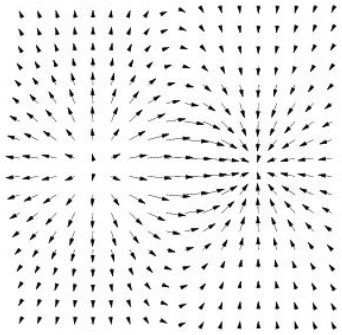


Figure 2.8: The vector field plot of equation 2.12.

The result of this operation on the field $\boldsymbol{f}$ is given in figure 2.8. By assuming that this field is a flowing liquid, one could come to the conclusion that there is some kind of *source* on the left and some sort of sink on the right side, but what exactly is a source or a sink?

Observing these special points closer one could deduce that a source in the field is a point which consists only of vectors that point out of it. More precisely, a source has more *outgoing* vectors than *incoming* vectors. A sink would then be the opposite. In the divergence plot 2.9 these special points of the vector field are special too because they are a maximum and a minimum. It seems that the divergence represents how much a point is a sink or a source. A divergence of zero would then indicate that we have as much incoming flow as outgoing.

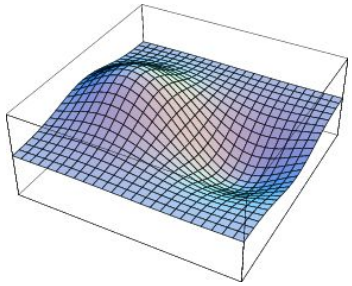


Figure 2.9: The divergence field of equation 2.12.

And of course the exact definition is: The divergence of a vector field is a measure of the existence of sinks and sources. If the divergence of a field is zero than it is called *solenoidal* or *source-free*[LP05].

Now after the gradient and the divergence were introduced let us point out a third differential operator which is a combination of these two.

THE LAPLACE OPERATOR

The divergence of a gradient field is called *Laplace operator* or *Laplacian*. Its name comes from the French mathematician Pierre-Simon Laplace and it is denoted by the Greek uppercase letter delta. In cartesian coordinates it is given by

$$\text{div grad } f = \nabla \cdot \nabla f = \Delta f = \frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2} + \frac{\partial^2 f}{\partial x_3^2} \tag{2.13}$$

It is often used in partial differential equations in physics and it will appear in later sections of this thesis.

THE CURL

For every vector field $\boldsymbol{f}$ we can find another field that measures the rotaion of it. The notation of this operator differs from book to book. In German literature it is often called *rotation* with the operator *rot*. We will use *curl*, so keep in mind that these two are equivalent.

The *curl* can be expressed with the nabla operator and the cross product of vectors.

$$\text{curl } \boldsymbol{f} = \nabla \times \boldsymbol{f} = \begin{pmatrix} \frac{\partial}{\partial x_2} f_3 & - & \frac{\partial}{\partial x_3} f_2 \\ \frac{\partial}{\partial x_3} f_1 & - & \frac{\partial}{\partial x_1} f_3 \\ \frac{\partial}{\partial x_1} f_2 & - & \frac{\partial}{\partial x_2} f_1 \end{pmatrix} \tag{2.14}$$

Notice that the *curl* of a vector field is a field of vectors, where the length of each describes

the strength of the rotation and the direction of each vector is normal to the plane of rotation.

Assume one is interested in the curl of the field of equation 2.15. It is possible to imagine one would put a little piece of cork on the x_1-x_2-plane. What happens is that this piece would start to rotate counterclockwise because the field on the right side is faster.

$$\boldsymbol{f}_1(\boldsymbol{x}) = \begin{pmatrix} 0 \\ x_1 \\ 0 \end{pmatrix} \tag{2.15}$$

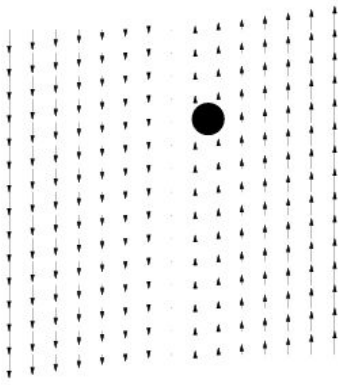


Figure 2.10: A simple field with non-zero curl. Taking the point as a little piece of cork on a flowing water surface, it would start to rotate. That is grounded in the faster flow on the right side.

But what would the curl field look like? As described above the vectors of the curl are normal to the rotation plane. This sample field engenders only rotations in the x_1-x_2-plane. So all vectors of the curl have to be vertical to it.

The lengths of them should be related to the strength of the rotation. But the piece of cork would rotate with the same speed everywhere in the field because the *difference* of the flow on the left and on the right side is the same at every point.

Calculating the curl of the sample field $\boldsymbol{f}_1$ confirms this deliberation.

$$\text{curl } \boldsymbol{f}_1 = \begin{pmatrix} \frac{\partial}{\partial x_2} f_3 & - & \frac{\partial}{\partial x_3} f_2 \\ \frac{\partial}{\partial x_3} f_1 & - & \frac{\partial}{\partial x_1} f_3 \\ \frac{\partial}{\partial x_1} f_2 & - & \frac{\partial}{\partial x_2} f_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.16}$$

REMARKS

The last section gave an overview of differential operators. One condition to calculate those operators is that the used functions are appropriate differentiable. Since we work

on images we must ensure that we are able to calculate the derivatives.

Therefore the task of the following section is to introduce an approximation for derivating digital images. The way of representing images mathematically was introduced in section 2.1 on page 4. Now this definition is used for introducing a method called *finite differences*.

2.4 FINITE DIFFERENCES METHOD

By working with digital images we will often be confronted with the situation of having discrete image values on a regular grid. There a calculation of derivatives as we know it is not possible.

Finite differences are the easiest way to define approximations for the derivatives on such discrete functions. For the required derivative of a point, the method interpolates this value together with some surrounding sample points with a polynomial. This polynomial can now be derivated and evaluated at this point.

One drawback of an interpolation polynomial is that it starts to oscillate when too many grid points are used. This means that between these points the interpolation is incredibly bad but the grid points match perfectly. A sample is given in figure 2.11. As one can see the four values of the left image are interpolated very well, even in the space between them. The higher order polynomial for the 15 values does not interpolate the border area in an acceptable way.

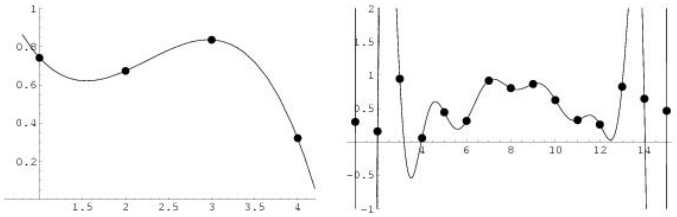


Figure 2.11: Comparison of the interpolation of 4 and 15 points with polynomials.

The consequence of this behavior is that it makes no sense to interpolate all values of a discrete image at once. The result would be incredibly bad. Therefore, every image value is interpolated separately using only a few surrounding grid points.

Assume we have a set of values $y_0, y_1, \ldots, y_n$ which represent the height values in figure 2.12. Since the pixels in the image are distributed on a regular grid the values in the right figure have a distance h from each other.

Taking now y_3 and two surrounding values on each side and interpolating them with a polynomial, results in a function $f(x)$ that goes through each of the five points and

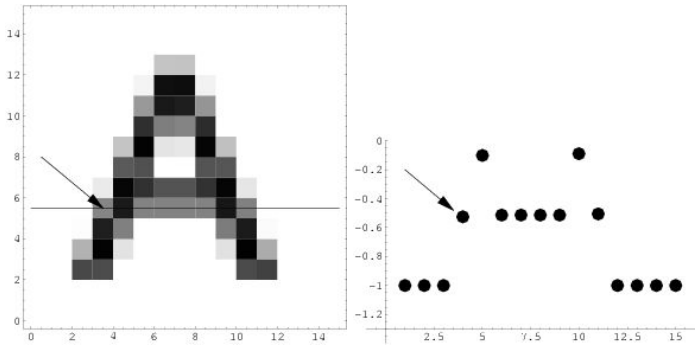


Figure 2.12: The values of the pixels in the marked line are plotted as heights in the right image.

$f(0) = y_3$. One can get this polynomial by using the Newton interpolation algorithm. This calculation is not presented here but it is straight forward and can be found in many mathematical books, for instance in [BSM05, MV90].

$$\begin{aligned} f(x) = \frac{1}{24h^4}(-2h+x)(-h+x)(x(h+x)y_1 - \\ 4x(2h+x)y_2 + 6(h+x)(2h+x)y_3) \\ +4(2h-x)x(h+x)(2h+x)y_4 + \\ x(-h+x)(h+x)(2h+x)y_5 \end{aligned} \tag{2.17}$$

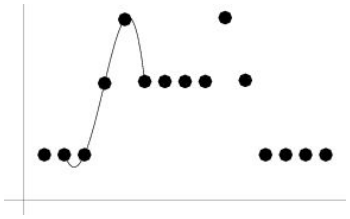


Figure 2.13: The plot of the function 2.17. It interpolates five values of the set.

When $f(x)$ is now derivated w.r.t. x and then evaluated at $x = 0$ we get an approximation of the derivative for y_3.

$$\begin{aligned} \frac{d}{dx}f(x) = \frac{1}{12h^4}(h^3(y_1 - 8y_2 + 8y_4 - y_5) - \\ h^2x(y_1 - 16y_2 + 30y_3 - 16y_4 + y_5) + \\ 2x^3(y_1 - 4y_2 + 6y_3 - 4y_4 + y_5) + \\ 3hx^2(-y_1 + 2y_2 - 2y_4 + y_5)) \end{aligned} \tag{2.18}$$

$$f'(0) = \frac{1}{12h}(y_1 - 8y_2 + 8y_4 - y_5) \tag{2.19}$$

This result can be used to define a partial derivative for images, because by derivating in only one direction there is just one dependent variable. The other one is assumed to be constant. Hence the approximation of the first order derivative of an image $T(i, j)$ w.r.t. i is given by:

$$\frac{\partial}{\partial i}T(i,j) \approx \frac{1}{12h}(T(i-2h,j) - 8T(i-h,j) + 8T(i+h,j) - T(i+2h,j)) \tag{2.20}$$

In the same way one can find the approximation for the second order derivative. The only difference is, that one has to derivate the interpolating polynomial two times. This will result in following term:

$$\frac{\partial^2}{\partial i^2}T(i,j) \approx \frac{1}{12h^2}(-T(i-2h,j)+16T(i-h,j)-30T(i,j)+16T(i+h,j)-T(i+2h,j)) \tag{2.21}$$

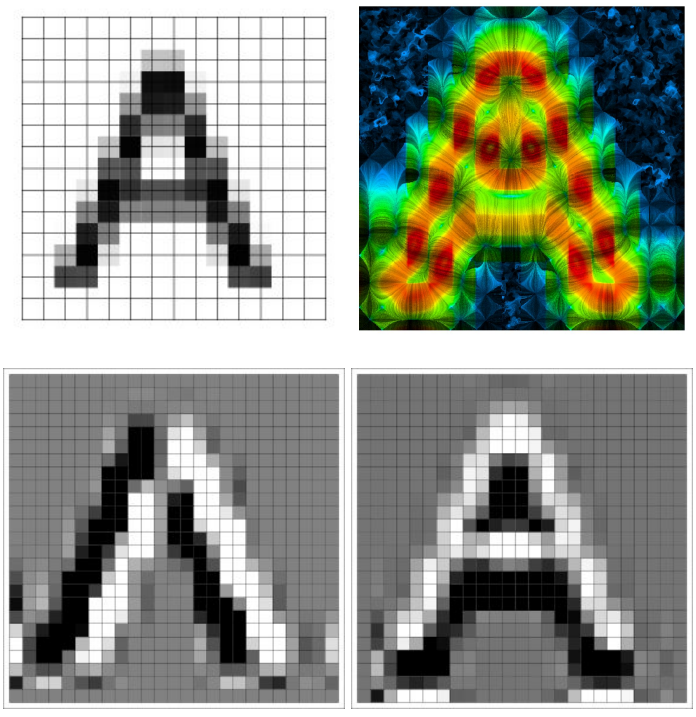


Figure 2.14: The already known image of the A (upper left) together with the LIC of its gradient field (upper right) and the derivative in horizontal direction (lower left) and in vertical direction (lower right).

As one can see the finite differences are an easy way to approximate derivatives. Now it is possible to calculate operators like the gradient for digital images.

One special case was not considered up to now and this is what happens at the border of an image. At any point where $i = 0$ for instance it seems not possible to calculate the derivative w.r.t. i because terms like $T(i - 2h, j)$ would lead to a nonexistent point of the image. Several solutions are possible for that case. One common way is to take the images as *periodic* and use the pixels from the opposite border for the missing ones.

In the sections of this thesis where finite differences are used, borders are handled by taking one more *inner* point for one missing border point. That means we break the symmetry of taking the same number of points from the left and the right for the interpolation. A case discrimination is required. For all *inner points*, where enough neighbors exist the normal formula is used. The *border points* are handled with another formula that takes for instance one more inner point when not enough border points exist. In the case of all $T(0, j)$ for instance we would interpolate the points $T(0, j), T(h, j), T(2h, j)$ and $T(3h, j)$ with a polynomial, derivate it and evaluate it at $x = 0$.

CHAPTER 3

TYPES OF IMAGE REGISTRATION

THIS IS A 100% MATTER PRODUCT:
In the Unlikely Event That This Merchandise Should Contact Antimatter in Any Form, a Catastrophic Explosion Will Result. [HS91]

In this chapter a basic overview about the different types of digital image registration is given. The chapter is divided into three parts. The first one explains how methods can be distinguished by the transformation they produce.

The second part describes image registrations which belong to parametric methods and the third part deals with non-parametric image registrations.

The purpose of the chapter is to give a basic classification over image registration methods and to explain some of them in more detail. What an image registration is and how it can be defined was explained in section 2.2 and is not a topic of this chapter.

Notice that there are many different classification criteria and we picked out only two of them. Registration methods can be distinguished for instance by the nature of the transformation, domain of the transformation, optimization procedure and so on. A very detailed description of how registrations can be classified is given by Maintz and Viergever [MV98].

We divide them firstly by the nature of the transformation and secondly by a classification into parametric and non-parametric methods. Furthermore some concrete methods are explained to provide the reader a little deeper insight to what is possible.

3.1 Transformation Types

By saying *types*, we basically mean the class of functions the transformation is in. Since some registrations are for very basic mappings it is possible to represent them with easier methods than displacement fields. A transformation which only includes rotations and translations for example is called *rigid* and can be represented with a matrix and a vector.

With this approach every transformation can be categorized into four types. They are called *rigid*, *affine*, *projective* and *curved* transformation. A function can also be a composition of more than one of them and if so, is named after the more general type, e.g. a composition of a rigid and an affine transformation is again an affine transformation.

The following sections will introduce the four categories and show which coordinate transformations are possible with them.

3.1.1 Rigid Transformation

A transformation is *rigid* when it performs only rotations and translations. This can be expressed with a rotation matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and a translation vector $\boldsymbol{b} \in \mathbb{R}^n$.

Rotations matrices are matrices where

$$\mathbf{Q}\mathbf{Q}^\top = \mathbb{1} \quad \text{and} \quad \det \mathbf{Q} = 1 \tag{3.1}$$

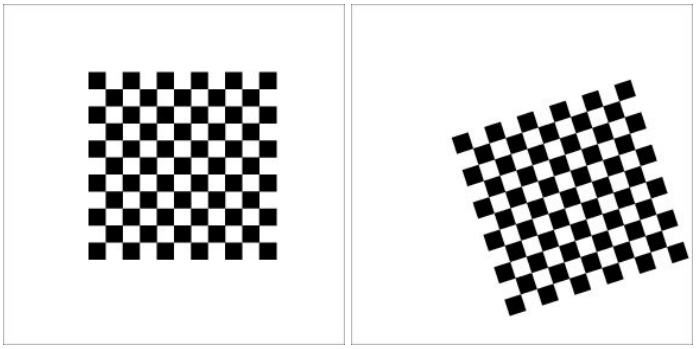


Figure 3.1: Rigid transformation sample.

A function $\boldsymbol{f}_R$ performing a rigid transformation by rotating and translating vectors $\boldsymbol{x}$ is then given by

$$\boldsymbol{f}_R(\boldsymbol{x}) = \mathbf{Q}\boldsymbol{x} + \boldsymbol{b} \tag{3.2}$$

3.1.2 AFFINE TRANSFORMATION

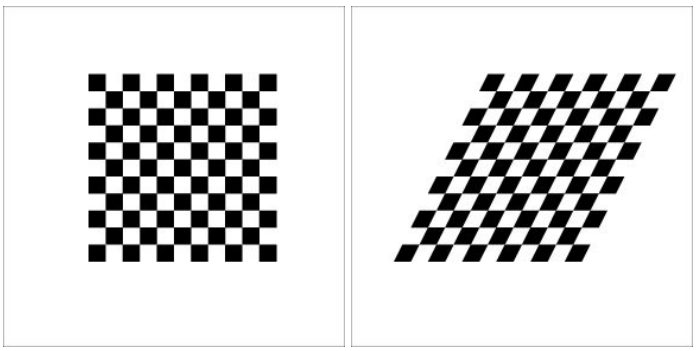


Figure 3.2: Affine transformation sample.

In [MV98] an affine transformation is defined to be a function which maps parallel lines onto parallel lines. By the usage of a more general matrix $\mathbf{A}$ with $\det \mathbf{A} > 0$ for the rotation matrix of the rigid transformation we get a representation of this type.

$$\boldsymbol{f}_A(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \quad \text{with} \quad \det \mathbf{A} > 0 \tag{3.3}$$

3.1.3 PROJECTIVE TRANSFORMATION

The significant property which states that a transformation is in the projective type class is that *lines are mapped onto lines*. Figure 3.3 shows this behavior.

For the matrix representation of these transformations a $(n+1) \times (n+1)$ matrix is required when we are in the n-dimensional space. These *homogeneous coordinates* are

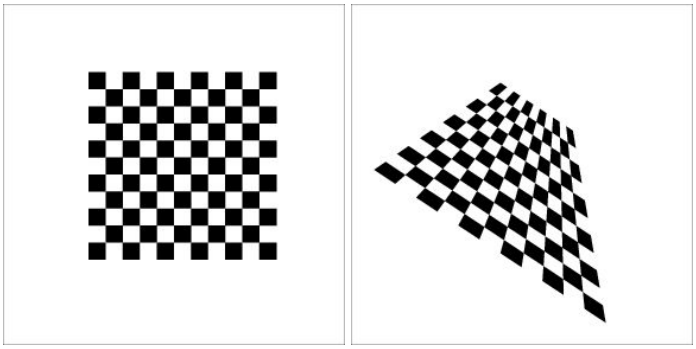


Figure 3.3: Projective transformation sample.

often used in computer graphics when projecting three-dimensional scenes onto a two-dimensional screen [Str03].

3.1.4 CURVED TRANSFORMATION

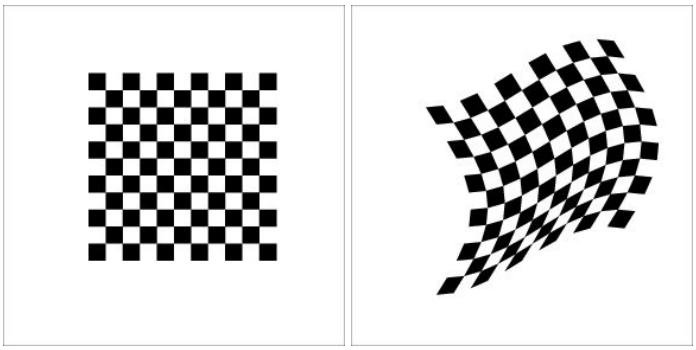


Figure 3.4: Curved transformation sample.

In all previous types it was possible to give a matrix representation for the transformation. In general a curved transformation does not have such a closed form. For those kinds, *displacement fields* are a very common representation. A displacement field assigns every point to a vector, the *displacement vector*. If $\boldsymbol{u}(\boldsymbol{x})$ is the displacement vector for the position $\boldsymbol{x}$, then the resulting point $\boldsymbol{x}'$ is given by

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x}). \tag{3.4}$$

Displacement fields, their usage and how they take effect on images is described more detailed in section 2.3.

Figure 3.4 shows a sample of a curved transformation. Functions of this class map *curves onto curves*. This property defines a smoothness criterion of the function because mapping curves onto curves means basically that neighbouring points remain neighbour-

ing to some extent. It may be useful to divide this class into subclasses. One property which is important to know is whether or not the function is bijective.

In many applications of image registration, solutions which are not bijective can be disgarded. Bijective functions avoid to displace two different points to the same place which results in folds in the transformed image.

3.1.5 REMARKS

This short classification which was given in [MV98] provides only a coarse graduation. As mentioned briefly, especially the curved transformation type contains a wide range of image registration methods. Many of the methods given in [Mod04], for instance the elastic, fluid and diffusion registration, are in this class.

Our registration method described in the later sections calculate transformations which are of the curved type too. This is grounded in the way in which it will compute its solution. Like the fluid registration for instance, we do not find a transformation which matches a given form. We only give general conditions the solution has to fulfill and so it is very unlikely and cannot be guaranteed that the found transformation is of a lower type than the curved one.

3.2 PARAMETRIC IMAGE REGISTRATION

Parametric registrations are working with a finite set of parameters which have to be found. We will see that these parameters are mostly the coefficients of basis functions of the registration transformation.

Furthermore parametric methods work with a set of *features*. Those features exist in the reference image and in the template. They are used to find a transformation, mapping every feature in the template onto its corresponding feature in the reference image.

With this approach we can give a definition for the registration problem: Having m different features in the template and in the reference where the ith feature in the reference is denoted by $\mathcal{F}(R(\boldsymbol{x}), i)$ then the registration problem is to find a transformation $\mathfrak{T}$ with

$$\mathcal{F}(R(\boldsymbol{x}), i) = \mathfrak{T}(\mathcal{F}(T(\boldsymbol{x}), i)), \quad i = 1, \ldots, m \tag{3.5}$$

The transformation $\mathfrak{T}$ maps the ith feature in the template onto the ith one in the reference. This brings us to the question what are typical features of an image. So called landmarks in the images can play the role of a feature. These landmarks may be artificial labels inside the image, marking special positions. When they were set up before imaging they

are called *hard* whereas *soft* landmarks are those which can be extracted from the image data itself.

A popular example of hard landmarks is for instance little marks on the fingertips in the x-ray image of a human hand. It is obvious that artificial marks are easy to recognize in the image. In contrast to that the position of implicitly given landmarks may be very hard to find.

The next section will go deeper into the procedure of a registration with explicitly given landmarks.

3.2.1 LANDMARK-BASED METHODS

The landmarks of an image $T(\boldsymbol{x})$ are on spatial positions in the image and since the registration maps points in the template onto points in the reference we will treat the ith landmark as a position

$$\mathcal{F}(T(\boldsymbol{x}), i) = \boldsymbol{x}_{T,i}, \quad i = 1, \ldots, m. \tag{3.6}$$

Corresponding to the difference of images that was introduced in 2.2 we can measure the quality of a transformation $\boldsymbol{\varphi}(\boldsymbol{x})$ by observing how well it maps the features onto each other. With the Euclidean norm $\|\cdot\|$ the added square differences of the landmarks are given by

$$\mathcal{D}^{\mathrm{LM}}[\boldsymbol{\varphi}] = \sum_{i=1}^{m} \|\boldsymbol{x}_{R,i} - \boldsymbol{\varphi}(\boldsymbol{x}_{T,i})\|^2. \tag{3.7}$$

Note that there are no restrictions for the transformation T except that it has to map the landmarks onto each other.

When we assume that $\boldsymbol{\varphi}$ can be expressed as a sum of basis functions then $\boldsymbol{\varphi}$ is characterized by the coefficients of the basis functions.

$$\boldsymbol{\varphi} = \sum_{k=1}^{n} \alpha_k \boldsymbol{\psi}_k, \quad \alpha_k \in \mathbb{R} \tag{3.8}$$

The task is then to find the coefficients of $\boldsymbol{\varphi}$, so that $\boldsymbol{\varphi}$ minimizes $\mathcal{D}^{\mathrm{LM}}$ for two given images. For more details of the computation we refer to [Mod04, pp. 28-30]

One thing is really important: Since the above method has to find a transformation which maps the landmarks onto another and since it has not to meet any other requirements the result of the image registration is highly dependent on the choice of the basis functions $\boldsymbol{\psi}_k$. Even when a given solution maps the landmarks perfectly, it does not say anything about the quality of the registration. Modersitzki showed a sample where the

template was completely deformed but landmarks were registered fine. They said in their explanation:

> Although the quadratic polynomial is optimal with respect to the data, it is not preferable for registration. This is because the quadratic is not bijective, manifests oscillation, and does not reflect the monotonicity of the data.[Mod04]

At this point a restriction for the transformation is necessary which forces the method to choose a *smooth* solution. With such a smoothing term the landmark-based method can be improved to the *landmark-based smooth registration*. Since the smoothing term is used in our method too and its behavior is similar to the one here we refer for a deeper insight to [Mod04] and in this thesis to pages 7 and 35.

3.2.2 Principal Axes-Based Method

In contrast to the previous method where the landmarks were explicitly given, another approach is imaginable: The calculation of the landmarks out of the image. Detecting them automatically can be a difficult matter and in many cases humans are still required. A sample for a fully automated method is the *principal axes transformation* (PAT). The PAT registers images by searching them for the principal axes and using them as *features* for the landmark-based registration. The principal axes of a body or a shape are physical properties which are related to the mass distribution within the body. These principal axes are invariant with respect to rotations and translations. Therefore two identical images which only differ in the position can be registered precisely with this method. Even for other deformations the PAT can be reasonably good.

A drawback of the method is that there are situations possible where different shapes share the same principal axes. Therefore these different images cannot be distinguished by the feature and it is likely that a registration would fail.

3.2.3 Remarks

The previous section discussed the parametric image registration whose methods share the property that they try to find appropriate parameters. Additionally most of them are working with external features which can either be artificial or be extracted from the image.

A general drawback of those methods is that the detection of the so called landmarks needs expert knowledge. If the landmarks are artificial and placed before imaging then this is done by a human with the knowledge of for instance the anatomy of the specimen.

Extracting the landmarks with the computer needs an expert too for at least adjusting the parameters of the extraction algorithm. For our purpose which is the automated registration of maybe hundreds of tissue images this is not suitable.

3.3 NON-PARAMETRIC IMAGE REGISTRATION

All non-parametric image registration techniques explained in this section follow the same approach. They base on a variational problem which defines the registration criterion. One part of this criterion is always a distance $\mathcal{D}(\boldsymbol{u})$ or more exact $\mathcal{D}(\boldsymbol{u}, R, T)$ which measures the similarity of the reference R and the template T under the displacement $\boldsymbol{u}$.

> A direct minimization of the distance measure has some drawbacks: the problem is ill-posed since small changes in the input data may lead to large changes of the output data, the solution is not unique since the problem is not convex, and the deformation may not even be continuous. [Mod04]

This behavior was already described on page 7. The answer is an additional *smoothing term* $\mathcal{S}$ which forces the method to run into solutions that are more likely than others. The choice of this smoother is the significant criterion that distinguishes the different approaches of the non-parametric methods.

Nevertheless they share the same variational approach: If two images, reference R and template T, and also a parameter $\alpha > 0$ are given, then the registration problem can be expressed in the functional

$$\min_{\boldsymbol{u}} \mathcal{T}(\boldsymbol{u}) = \min_{\boldsymbol{u}} \left(\mathcal{D}(\boldsymbol{u}) + \alpha \mathcal{S}(\boldsymbol{u})\right). \tag{3.9}$$

Four well known methods follow this approach. Some are strongly physically motivated. The *elastic registration* is related to the behavior of an elastic body whereas the *fluid registration* mimics the behavior of liquids. The optical flow was the grounding for the *curvature registration* and the *diffusion registration* was based on some deliberations about the smoothness of the displacement field.

In the following sections we will describe the basic idea of the elastic and the fluid registration. We will not go into detail but give properties and drawbacks of them. For the curvature registration a more detailed description is given because we will need this knowledge through the rest of this thesis.

3.3.1 ELASTIC REGISTRATION

As the name suggests the elastic registration mimics the behavior of an elastic body and was first introduced by [Bro81]. Looking back to the rigid transformation, the volume elements of the image plane were fixed to their neighbours. Therefore their position regarding to their neighbours was still the same after transforming the image.

The approach of an elastic body is different. The volume elements can change their position with respect to the properties of the elastic material. When an external force is applied to such a body, it can be deformed. At the same time when a force is applied a property of the body, called the *inner stress* or *tension* changes. This influences the shape of the body too, so that it poises in an appearance where inner stress and external forces are in an equilibrium.

Therefore the registration uses a smoothing term which represents the elastic potential of the displacement field $\boldsymbol{u}$.

$$\mathcal{S}^{\text{el}}(\boldsymbol{u}) = \int_{\Omega} \frac{\mu}{4} \sum_{j,k=1}^{d} \left(\partial_{x_j} u_k + \partial_{x_k} u_j\right)^2 + \frac{\lambda}{2} \left(\operatorname{div} \boldsymbol{u}\right) d\boldsymbol{x} \tag{3.10}$$

The two parameters μ and λ denote the Lamé constants which are used to define what elastic properties the transformation should have. When $\mathcal{S}^{\text{el}}$ is used as the smoother of the registration, a solution $\boldsymbol{u}$ has to minimize a sum where one part is

$$\boldsymbol{f} = \mu \Delta \boldsymbol{u} + (\lambda + \mu) \nabla \operatorname{div} \boldsymbol{u} \tag{3.11}$$

By expanding $\boldsymbol{f}$ to its components one sees that the parts are coupled in the last cross derivative term.

$$\begin{aligned} \boldsymbol{f}_{x_1} &= \mu \left(\frac{\partial^2}{\partial x_1^2} u_{x_1} + \frac{\partial^2}{\partial x_2^2} u_{x_1} \right) + (\lambda + \mu) \left(\frac{\partial^2}{\partial x_1^2} u_{x_1} + \frac{\partial^2}{\partial x_1 \partial x_2} u_{x_2} \right) \\ \boldsymbol{f}_{x_2} &= \mu \left(\frac{\partial^2}{\partial x_1^2} u_{x_2} + \frac{\partial^2}{\partial x_2^2} u_{x_2} \right) + (\lambda + \mu) \left(\frac{\partial^2}{\partial x_2^2} u_{x_2} + \frac{\partial^2}{\partial x_1 \partial x_2} u_{x_1} \right) \end{aligned} \tag{3.12}$$

The variational calculus states that a minimizer of equation 3.10 is a solution of the Euler-Lagrange equation. Therefore the Navier-Lamé equation 3.11 is a result of 3.10. The registration problem for the elastic method is to find a transformation $\boldsymbol{u}$ which minimizes the functional

$$\min_{\boldsymbol{u}} \mathcal{T}^{\text{el}}(\boldsymbol{u}) = \min_{\boldsymbol{u}} \left(\mathcal{D}(\boldsymbol{u}) + \mathcal{S}^{\text{el}}(\boldsymbol{u}, \mu, \lambda) \right) \tag{3.13}$$

with given Lamé constants μ and λ. It is a method whose impelling force is the requirement to minimize the distance between the images and it is restricted or *regularized* by the laws of an elastic body.

Details about the method can be found in [Bro81] and [Mod04]. We will not follow these calculations but give some general remarks about the method.

Following Modersitzki who said: "The regularizer is very restricted, [...] For very dissimilar objects, elastic registration is not the method of choice. The distance measure may not be reduced sufficiently;". The method works fine for many applications but if the deformation of the images is too heavy, the registration fails. This is because on a certain level of deformation the inner stress raises the smoothing term in a way that this transformation is not a minimizer of the functional.

Therefore one typical test, mapping a filled circle onto a 'C' is not possible with the elastic registration. Nevertheless in more realistic samples the method yields good results.

3.3.2 FLUID REGISTRATION

The fluid registration has, as with the the elastic one, an underlying real world model which is adapted from the behavior of liquids. Like in the elastic registration too, the impelling force of the fluid registration is the endeavor to minimize the distance between template and reference image. The difference between them is the form of the regularizer or smoother. This smoothing term forces the registration to satisfy the assumptions stated in the model of fluid mechanics. These are

- The law of the conservation of mass which states that the mass of a closed system remains constant, regardless what happens inside.
- The law of the conservation of the momentum. This law states that in a closed system, the momentum which is the product of mass and velocity cannot be changed. Note that a closed system is one which is not affected by external forces.

With the knowledge of the elastic registration one can find easily a variational approach for the fluid registration. Regarding the images as a set of particles then for the elastic registration one can state that adjacent particles remain neighbouring in a certain manner after transformation. For the fluid registration a similar property exists: In a flow of a liquid, adjacent particles have related velocities. Since the velocity of a particle is the derivative of its position w.r.t. the time, one can simply define the regularizer for the fluid registration with the help of the elastic smoother.

$$\mathcal{S}^{\text{fl}}(\boldsymbol{u}) = \mathcal{S}^{\text{el}}\left(\frac{\partial}{\partial t}\boldsymbol{u}\right) \tag{3.14}$$

For this reason the resulting equation for the smoothing term is given by

$$\boldsymbol{f}^{\text{fl}} = \mu\Delta\boldsymbol{u}_t + (\lambda+\mu)\nabla\text{div }\boldsymbol{u}_t. \tag{3.15}$$

For relevant details about the physical derivation of the formulas we refer first of all to the thesis of Bro-Nielsen and related papers [BNG96, BN96]. Nonetheless Modersitzki provides a compact presentation from the theory up to the implementation details [Mod04].

Since the fluid registration mimics the behavior of a liquid it is possible to register almost every reference-template combination. This is an advantage in some special applications where the fluid registration works very well. In general and in many medical applications an *unlimited flowing* of the template into the reference image does not match the reality.

In our special case it means that slices of a specimen do not act like a fluid at any time. For this reason the fluid registration is not the method of choice in 3-D tumor reconstruction.

3.3.3 CURVATURE-BASED REGISTRATION

The previous two methods have some general drawbacks. They are useful for special applications. The elastic method is often too stiff and therefore unable to repair greater deformations.

On the other hand the fluid registration can handle most deformations. This is not wanted in every case because very often a set of presumptions were made to the kind of deformation. Especially in medical applications it is very unlikely that organs and tissue act like fluids. With the fluid registration it is possible to register almost every set of images which is not wanted in the most cases.

Beside this, another property came out to be a drawback. The regularizers of the elastic, the fluid and the diffusion[1] registration is very dependent on the initial positions of the images. That means, this method is error prone when the deformation contains an affine transformation.

Assume we have such an affine deformation. The displacement field that solves this problem is then of the form

$$\boldsymbol{u}(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x} + \boldsymbol{b}, \quad \text{with} \quad \mathbf{A} \in \mathbb{R}^{2\times 2}, \quad \boldsymbol{b} \in \mathbb{R}^2 \tag{3.16}$$

Taking now elastic potential from equation 3.10 and consider it has to rate such an

[1] This method will not be explained herein. Nevertheless it is sharing this drawback with the fluid and elastic registration.

affine transformation. By putting the displacement of equation 3.16 in the elastic regularization term we get

$$\begin{aligned}\mathcal{S}^{\text{el}}(\boldsymbol{u}) &= \int_\Omega \frac{\mu}{4}\left((a_{11}+a_{11})^2 + 2(a_{12}+a_{21})^2 + (a_{22}+a_{22})^2\right) + \frac{1}{2}\lambda(a_{11}+a_{22})\,d\boldsymbol{x} \\ &= \int_\Omega \mu\left(a_{11}^2 + \frac{1}{2}(a_{12}+a_{21})^2 + a_{22}^2\right) + \frac{1}{2}\lambda(a_{11}+a_{22})\,d\boldsymbol{x}\end{aligned} \tag{3.17}$$

One can see that the regularization term will penalize such deformation because an affine transformation will not necessarily zeroize equation 3.17. Only a special form of $\mathbf{A}$ will provide this.

$$\boldsymbol{u} = \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \quad \text{with} \quad \mathbf{A} = \begin{pmatrix} 0 & c_{12} \\ -c_{12} & 0 \end{pmatrix} \quad \Longleftrightarrow \quad \mathcal{S}^{\text{el}}(\boldsymbol{u}) = 0 \tag{3.18}$$

This is the reason why a pre-registration is unavoidable when images should be registered with one of these methods. The now introduced *curvature-based* registration is more robust against such initial conditions. It is not the case that pre-registration steps become unnecessary but the new form of the regularizer will not penalize affine transformations in the displacement.

The idea of the curvature regularizer comes originally from a very early work of Horn and Schunck [HS81]. Their method for determining the optical flow in a series of images was as ill-posed as the non-parametric image registration without a smoothing term. The optical flow in their work has to fulfill some smoothness criteria too because ”If every point of the brightness pattern can move independently, there is little hope of recovering the velocities” [HS81].

Optical flow is the movement a brightness pattern makes, assuming two images showing the same object on different points in time. The result is a vector field representing the motion of the brightness pattern. A movement of the object does not always cause a change in optical flow. A rotation of a uniformly colored sphere around its center for instance does not change its optical flow. Therefore they suggested a smoothness constraint which does not penalize such (affine) transformations and made usage of the squared Laplacians. This was reasonable because it goes along with a smooth, continuous result wich can include affine transformations.

> In simple situations, both Laplacians are zero. If the viewer translates parallel to a flat object, rotates about a line perpendicular to the surface or

> travels orthogonally to the surface, then the second partial derivatives of both u and v vanish [...] [HS81]

Their idea was picked up by Amit who used this approach to develop a new "non-linear variational problem for image matching" [Ami94]. Amit gave the first version of the smoothing term $\mathcal{S}^{\text{curv}}$ which was used later by Fischer and Modersitzki in many different publications (e.g. [FM03a, FM02a, FM02b]).

$$\mathcal{S}^{\text{curv}}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega} (\Delta \boldsymbol{u})^{\top} \cdot (\Delta \boldsymbol{u})\, d\boldsymbol{x} \tag{3.19}$$

The name, *curvature-based registration*, was also introduced by Fischer and Modersitzki in [FM03a]. It was chosen because the integral in 3.19 can be *interpreted* as some kind of curvature of the components of $\boldsymbol{u}$. Nevertheless it comes from the optical flow and has no direct physical motivation like the fluid or the elastic registration.

One significant advance of the new smoothing term was that it does not penalize affine transformations because the second order derivatives vanish there.

$$\forall \mathbf{A} \in \mathbb{R}^{2\times 2},\, \boldsymbol{b} \in \mathbb{R}^2 :\ \boldsymbol{u} = \mathbf{A}\boldsymbol{x} + \boldsymbol{b} \Longrightarrow \mathcal{S}^{\text{curv}}(\boldsymbol{u}) = 0 \tag{3.20}$$

Strictly speaking this should make an initial pre-registration step redundant but actually it is better to perform at least a rigid pre-registration before running the non-parametric curvature-based step. The reason is twofold. Firstly we will see that one has to choose boundary conditions for the calculation of the solution. If some presumptions on the boundary are made, for instance that we have a zero displacement there, then it is no longer possible to get a solution which includes for instance a translation. Kuska and Braumann have already pointed this out:

> None of the boundary conditions is consistent with a rigid transformation that includes a rotation around a point c within the image, i. e. $\boldsymbol{u}(\boldsymbol{x}) = \mathbf{R}(\boldsymbol{x} - \boldsymbol{c}) + \boldsymbol{t}$, with the rotation matrix $\mathbf{R}$. The only reason for this choice of boundary conditions is that a fast solution method exists for them. [BK05]

Secondly many of the solution methods that are presented by Amit [Ami94], Fischer and Modersitzki [FM02a, FM02b, FM03a] or Kuska and Braumann [BK05, BK06] use an iterative approach that starts from the zero displacement field and "moves" toward a solution. It is then better when the images are pre-registered and close to the desired solution.

> The functional $\mathcal{J}$ is non-linear and may have many global and local minima. In the sequel we will be mainly interested in finding local minima of

$\mathcal{J}$ close to the initial point $U(x) \equiv 0$, which corresponds to the identity map. [Ami94][2]

The boundary conditions that were pointed out are a requirement to solve the PDE that is defined by the joint registration criterion of the curvature-based registration.

$$\min_{\boldsymbol{u}} \mathcal{T}(\boldsymbol{u}) = \min_{\boldsymbol{u}} \left(\mathcal{D}(\boldsymbol{u}) + \mathcal{S}^{\text{curv}}(\boldsymbol{u})\right) \tag{3.21}$$

This PDE follows directly according to the variational calculus which states that the first Gâteaux derivative of a minimizer $\boldsymbol{u}$ of equation 3.21 must be zero. The calculation of this functional derivative is shown in the next chapter where we explain in more detail the solution procedure.

$$-\alpha\Delta^2\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x})\big) = 0 \tag{3.22}$$

with

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x})) = \Big(R(\boldsymbol{x}) - T\big(\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x})\big)\Big) \cdot \nabla T\big(\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x})\big) \tag{3.23}$$

and

$$\Delta^2\boldsymbol{u}(\boldsymbol{x}) = \frac{\partial^4\boldsymbol{u}}{\partial x_1^4}(\boldsymbol{x}) + 2\frac{\partial^4\boldsymbol{u}}{\partial x_1^2\partial x_2^2}(\boldsymbol{x}) + \frac{\partial^4\boldsymbol{u}}{\partial x_2^4}(\boldsymbol{x}). \tag{3.24}$$

This equation is called the biharmonic or bipotential equation and is well studied. The biharmonic equation can not be solved directly and therefore other approaches are used. One popular method is to introduce an artificial time parameter and use the new equation to iterate towards the solution. This proceeding is described in section 4.2.

Finally, the behavior of the curvature-based registration can be described to be in the middle between fluid and elastic registration. Since the fluid registration allows extremely large deformation, the elastic one can handle only small, locally bounded displacements.

The behavior of the curvature-based registration is close to the elastic one but allows larger deformations. Affine transformations are not penalized by the curvature smoother and, respecting the boundary conditions, a deformation can include such transformations.

3.4 REMARKS

The last chapter was a short introduction to the wide area of image registration. Since a great many applications need registration methods which are improved to solve a specific problem, the field of what is possible in this area is extremely large. In most cases these special solutions only change the behavior of a basic method or extend them.

[2]The functional $\mathcal{J}$ in the work of Amit is the same as the functional $\mathcal{T}$ in equation 3.9.

For a deeper insight into this area we suggest to read for instance [MV98]. Maintz and Viergever provide a very sufficient classification structure and they refer to a large amount of publications. With this overview what is possible in image registration for medicine one can go deeper in reading the details of special methods.

CHAPTER 4

THE VORTEX-FREE IMAGE REGISTRATION

NOTE:

The Most Fundamental Particles in This Product Are Held Together by a "Gluing" Force About Which Little is Currently Known and Whose Adhesive Power Can Therefore Not Be Permanently Guaranteed. [HS91]

4.1 MOTIVATION

The method presented in this thesis extends an image registration procedure described by Braumann and Kuska [BK05]. They showed a simple method for a non-parametric registration and used it for instance in the three-dimensional reconstruction of carcinoma [BKE+05a]. By providing high textured tissue images to this registration method, one significant problem arose. The images seemed not to be registered in the way they were supposed to be. The registration step seemed to have a problem with images that contain many similar structures. In those regions an anomalous high amount of vortices in the displacement field can be observed. The question is whether they are caused by the difference of the images that are registered or whether these vortices have other reasons.

To find this out we have to take a look at the accruement of the images. Details of the whole process are given on page 1287 of [BKE+05a]. Here the reader will only find a short summary, sufficient enough to understand the conclusions.

The images are photographs of a sliced human cervix which was embedded in paraffin. The image registration is used to register the pictures of the slices onto another in the same order as they were cut. In the cutting process itself a kind of a blade will slide through the object and separate one slice. This may squeeze the tissue a bit but it is really unlikely that regions are twisted.

The single slice is put under a microscope and the region to photograph is selected manually. In this step it is obviously not possible to position every slice in the same way. Thus two sequenced slices differ in the place and it may happen that they are rotated. These global rotations have nothing in common with the local vortices that arise in the displacement field because when the non-parametric registration starts the global translations and rotations were already extracted by previous registration steps.

The sequence of these procedure steps is the reasons why vortices in the registration are very unlikely and should not be part of the solution. To improve the method a first approach was chosen containing a vortex suppression term in the variational equation [BK06].

$$\min_{\boldsymbol{u}} \mathcal{T}(\boldsymbol{u}) = \min_{\boldsymbol{u}} \left(\mathcal{D}(\boldsymbol{u}) + \mathcal{V}(\boldsymbol{u}) + \mathcal{S}^{\text{curv}}(\boldsymbol{u})\right) \tag{4.1}$$

The terms $\mathcal{D}$ and $\mathcal{S}^{\text{curv}}$ are the already introduced functionals for minimal difference and smoothness of the transformation (see page 7 and 8). $\mathcal{V}$ is a measure for the total amount of vortices in the field $\boldsymbol{u}$ and it is defined by

$$\mathcal{V}(\boldsymbol{u}) = \frac{\omega}{2} \int_{\Omega} \left(\frac{\partial u_2}{\partial x_2}(\boldsymbol{x}) - \frac{\partial u_1}{\partial x_2}(\boldsymbol{x}) \right)^2 d\boldsymbol{x}. \tag{4.2}$$

In fact, to suppress vortices with a term in the variational equation has some disadvantages. The first one is the term itself. It describes not exactly what actually should be done. We found out that exact solutions of those kind of registration problems do not contain any vortices. By only suppressing them it is still possible to get a not vortex free displacement field. This depends primarily on the setting of ω. However, it is not always a good choice to take high values for ω because it is only one term in a sum. If it is chosen very high, the influences of the other two terms may disappear. It is possible that this ends in situations where a vortex-free displacement field goes at the expense of smoothness and minimal difference.

Another drawback is the additional parameter which was introduced by the term. The job of this image registration is supposed to register not only single pairs of images. It is supposed to be used in an automated process that registers many images consecutively. It would slow down the procedure when the vortex suppression term requires a manual justification in some registration steps.

These properties of the vortex suppression caused the search for a better solution. The idea was to turn back to the first equation

$$\min_{\boldsymbol{u}} \mathcal{T}(\boldsymbol{u}) = \min_{\boldsymbol{u}} \left(\mathcal{D}(\boldsymbol{u}) + \alpha \mathcal{S}^{\text{curv}}(\boldsymbol{u})\right) \tag{4.3}$$

and extend it with a vortex extraction. The following sections detail how it was possible to realize this during the solution procedure.

4.2 A Solution Procedure for the Curvature-Based Registration

As said in the previous section the method extends a solution introduced in [BK05]. It is grounded on an idea that the displacement field should fulfill the criteria of minimal difference between the registered images.

$$\mathcal{D}(\boldsymbol{u}) = \frac{1}{2} \int_{\Omega} (R(\boldsymbol{x}) - T(\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x})))^2 d\boldsymbol{x} \tag{4.4}$$

and smoothness

$$\mathcal{S}^{\text{curv}}(\boldsymbol{u}) = \frac{1}{2} \int_{\Omega} (\Delta \boldsymbol{u})^{\mathrm{T}} \cdot (\Delta \boldsymbol{u}) d\boldsymbol{x}. \tag{4.5}$$

To find the function $\boldsymbol{u}$ that solves the variational problems given in equation 4.3 the functional derivative $\delta\mathcal{T}(\boldsymbol{u}; \boldsymbol{g})$ is required. Similar to the derivative of a function w.r.t. a variable it is possible to derivate a functional w.r.t. a vector $\boldsymbol{g}$. This is called first Gâteaux

variation [MV01, page 407]. For a solution $\boldsymbol{u}^*$ of the variational problem a necessary condition is

$$\delta\mathcal{T}(\boldsymbol{u}^*;\boldsymbol{g}) = 0. \tag{4.6}$$

Similar to the procedure of finding an extreme value of a function, the first derivative of the functional is calculated and equation 4.6 is solved.

$$\begin{aligned}
&\delta\mathcal{T}(\boldsymbol{u}+\epsilon\boldsymbol{g}) = \\
&\quad \frac{1}{2}\frac{d}{d\epsilon}\int_\Omega d\boldsymbol{x}\ \ (R(\boldsymbol{x}) - T(\boldsymbol{x}-(\boldsymbol{u}(\boldsymbol{x})+\epsilon\boldsymbol{g})))^2 + \alpha(\Delta(\boldsymbol{u}+\epsilon\boldsymbol{g})^2\Big|_{\epsilon=0} \\
&\quad = \frac{1}{2}\int_\Omega d\boldsymbol{x}\ \ (R(\boldsymbol{x}) - T(\boldsymbol{x}-(\boldsymbol{u}(\boldsymbol{x})+\epsilon\boldsymbol{g})))^2\frac{\partial}{\partial\epsilon} + \alpha(\Delta(\boldsymbol{u}+\epsilon\boldsymbol{g})^2\frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} \\
&\quad = \frac{1}{2}\int_\Omega d\boldsymbol{x}\ \ 2(R(\boldsymbol{x}) - T(\boldsymbol{x}-(\boldsymbol{u}(\boldsymbol{x})+\epsilon\boldsymbol{g})))\nabla T(\boldsymbol{x}-(\boldsymbol{u}(\boldsymbol{x})+\epsilon\boldsymbol{g}))\boldsymbol{g} + \\
&\qquad\qquad 2\alpha(\Delta\boldsymbol{u}+\Delta\epsilon\boldsymbol{g})\Delta g\Big|_{\epsilon=0} \\
&\quad = \frac{1}{2}\int_\Omega d\boldsymbol{x}\ \ 2(R(\boldsymbol{x}) - T(\boldsymbol{x}-\boldsymbol{u}(\boldsymbol{x})))\nabla T(\boldsymbol{x}-\boldsymbol{u}(\boldsymbol{x}))\boldsymbol{g} - 2\alpha\Delta\boldsymbol{u}\Delta g \\
&\quad = \int_\Omega d\boldsymbol{x}\ \ \boldsymbol{g}\cdot\big((R(\boldsymbol{x}) - T(\boldsymbol{x}-\boldsymbol{u}(\boldsymbol{x})))\nabla T(\boldsymbol{x}-\boldsymbol{u}(\boldsymbol{x})) - \alpha\Delta^2\boldsymbol{u}\big)
\end{aligned} \tag{4.7}$$

The derivated functional 4.7 will become zero when one of the factors in between the integral gets zero. Therefore the registration transformation has to fulfill the following PDE:

$$-\alpha\Delta^2\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{f}\big(\boldsymbol{x},\boldsymbol{u}(\boldsymbol{x})\big) = 0 \tag{4.8}$$

with

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}(\boldsymbol{x})) = \Big(R(\boldsymbol{x}) - T\big(\boldsymbol{x}-\boldsymbol{u}(\boldsymbol{x})\big)\Big)\cdot\nabla T\big(\boldsymbol{x}-\boldsymbol{u}(\boldsymbol{x})\big) \tag{4.9}$$

and

$$\Delta^2\boldsymbol{u}(\boldsymbol{x}) = \frac{\partial^4\boldsymbol{u}}{\partial x_1^4}(\boldsymbol{x}) + 2\frac{\partial^4\boldsymbol{u}}{\partial x_1^2\partial x_2^2}(\boldsymbol{x}) + \frac{\partial^4\boldsymbol{u}}{\partial x_2^4}(\boldsymbol{x}). \tag{4.10}$$

A common approach to solve the highly non-linear equation 4.8 is to introduce an artificial time parameter t with

$$\lim_{t\to\infty}\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}(\boldsymbol{x}) \tag{4.11}$$

The time dependent equation for the displacement $\boldsymbol{u}(\boldsymbol{x}, t)$ is then given by

$$\frac{\partial \boldsymbol{u}}{\partial t}(\boldsymbol{x}, t) + \alpha\, \Delta^2 \boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}, t)) \tag{4.12}$$

At this point we have to consider that we are working on digital images. Therefore we have to determine the discrete solution $\boldsymbol{U}$ with $\boldsymbol{U}(t) = \boldsymbol{u}(\boldsymbol{x}_{i,j}, t)$ for $i = 1, \ldots N^{(\ell)}$, $j = 1, \ldots, M^{(\ell)}$. Furthermore, this method is a multi-resolutional approach and iterates different discretization levels. These levels are denoted by a superscript on the discrete field. Therefore $\boldsymbol{U}^{(\ell)}(t)$ with $\ell = 0, \ldots, L$ denotes the field at the discretization level ℓ. In this notation $\ell = 0$ is the coarsest grid an $\ell = L$ the finest grid. On the next discretization level the meshpoints are doubled. This means $N^{(\ell+1)} = 2N^{(\ell)}$ and $M^{(\ell+1)} = 2M^{(\ell)}$.

For the changing to the next level we introduce two operators. The restriction of the solution $\boldsymbol{U}$ to the next coarser grid is denoted by $\mathcal{R}\left[\boldsymbol{U}^{(\ell+1)}(t)\right] = \boldsymbol{U}^{(\ell)}(t)$ and in the other case $\mathcal{P}\left[\boldsymbol{U}^{(\ell)}(t)\right] = \boldsymbol{U}^{(\ell+1)}(t)$.

Notice that $\boldsymbol{U}$ describes displacements for the regular mesh and for this reason it is not sufficient to resample $\boldsymbol{U}$ to get another discretization level. $\mathcal{P}$ and $\mathcal{R}$ have to perform a scaling on the data.

For a time discrete version of equation 4.12 we will use two numerical methods for differential equations. For $y' = f(t, y)$, an initial value $y(t_0) = y_0$ and a time step h, the *explicit Euler* (eq. 4.13) and the *implicit mid-point* (eq. 4.14) rule is given by

$$y_{n+1} = y_n + h f(t_n, y_n) \tag{4.13}$$

$$y_{n+1} = y_n + \frac{h}{2}\big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\big) \tag{4.14}$$

On the coarsest level $\ell = 0$ the discrete version of equation 4.12 with a time step h is given by

$$\boldsymbol{V}^{(0)}(t+h) - \boldsymbol{U}^{(0)}(t) = -\frac{\alpha\, h}{2}\left(\Delta^2 \boldsymbol{V}^{(0)}(t+h) + \Delta^2 \boldsymbol{U}^{(0)}(t)\right) + h\, \boldsymbol{F}^{(0)}(\boldsymbol{U}^{(0)}(t)) \tag{4.15}$$

In this equation we used the implicit mid-point rule to approximate the term $\alpha\, \Delta^2 \boldsymbol{u}(\boldsymbol{x}, t)$ and the explicit Euler for $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}, t))$. The temporary approximation of $\boldsymbol{U}^{(\ell)}$ is denoted by $\boldsymbol{V}^{(\ell)}$ and the discrete approximation of $\boldsymbol{f}$ on the mesh points of level $\ell = 0$ is denoted by $\boldsymbol{F}^{(0)}$.

The other levels up to the finest grid are calculated by equation 4.16. The difference is that the implicit mid-point rule was used for the non-linear term $\boldsymbol{F}$ too. Notice that $\boldsymbol{V}^{(\ell)}$

is substituted by the prolongation of the already calculated $\boldsymbol{V}^{(\ell-1)}$ in the last term.

$$\begin{aligned}\boldsymbol{V}^{(\ell)}(t+h) - \boldsymbol{U}^{(\ell)}(t) = &-\frac{\alpha\, h}{2}\left(\Delta^2 \boldsymbol{V}^{(\ell)}(t+h) + \Delta^2 \boldsymbol{U}^{(\ell)}(t)\right) \\ &+h\, \boldsymbol{F}^{(\ell)}\left(\frac{1}{2}\big(\mathcal{P}[\boldsymbol{V}^{(\ell-1)}](t+1) + \boldsymbol{U}^{(\ell)}(t)\big)\right)\end{aligned} \tag{4.16}$$

When all levels up to $\boldsymbol{V}^{(L)}$ are done the new $\boldsymbol{U}^{(l)}(t+h)$ are determined by restricting the result on the finest level to all other levels below.

$$\begin{aligned}\boldsymbol{U}^{(L)} &= \boldsymbol{V}^{(L)} \\ \boldsymbol{U}^{(\ell)} &= \mathcal{R}\left[\boldsymbol{U}^{(\ell+1)}\right] \qquad \ell = L-1, \ldots, 0\end{aligned} \tag{4.17}$$

After this cycle is complete the convergence of $\boldsymbol{U}(t+h)$ is checked. The time is incremented by the step size h and the cycle is restarted until

$$\left\|\boldsymbol{U}^{(L)}(t+h) - \boldsymbol{U}^{(L)}(t)\right\| < \epsilon, \quad \text{with } \epsilon \ll 1 \tag{4.18}$$

The equations shown still contain the square of the Laplacian which has to be approximated too. For this purpose Braumann and Kuska used equation 25.3.33 of [AS84]. The stencil of this approximation is shown in figure 4.1.

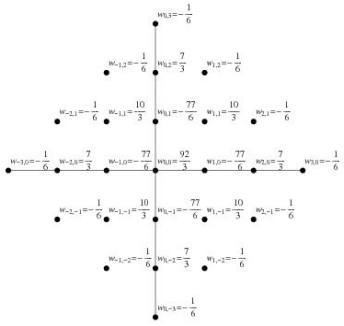


Figure 4.1: Stencil for the discrete approximation of Δ^2

For the calculation of the $\boldsymbol{V}^{(\ell)}$ in every level one has to solve a linear system of equations for the components of $\boldsymbol{V}^{(\ell)}$. For a periodic boundary this computation can be simplified by using the discrete Fourier transform. Since the components of $\boldsymbol{V}^{(\ell)}$ on different grid points become a multiplication by a frequency in the Fourier space, only one equation has to be solved per mesh point. This property is explained more detailed in section 4.3.2 where we use again this simplification.

$$\boldsymbol{V}^{(\ell)}_{m,n} = \frac{1}{N^{(\ell)}\,M^{(\ell)}} \sum_{\mu=0}^{M^{(\ell)}-1} \sum_{\nu=0}^{N^{(\ell)}-1} \hat{\boldsymbol{V}}^{(\ell)}_{\mu,\nu} \exp\left(2\,\pi\,\mathbb{i}\left(\frac{m\,\mu}{M^{(\ell)}} + \frac{n\,\nu}{N^{(\ell)}}\right)\right) \tag{4.19}$$

With the Fourier coefficients of equation 4.19, the discrete approximation of the PDE (eq. 4.16) can be rewritten. With $\beta = \alpha\,h/{\delta^{(\ell)}}^4$ where $\delta^{(\ell)}$ is the space between two mesh points in the level ℓ and $\omega^{(\ell)}_{\nu} = 2\,\pi/N^{(\ell)}$ and $\omega^{(\ell)}_{\mu} = 2\,\pi/M^{(\ell)}$ we get the following equation for $\hat{\boldsymbol{V}}^{(\ell)}$

$$\begin{aligned}
\hat{\boldsymbol{V}}^{(\ell)}_{\mu,\nu}(t+h) &= -\hat{\boldsymbol{U}}^{(\ell)}_{\mu,\nu}(t) + \frac{6\,\left(2\,\hat{\boldsymbol{U}}^{(\ell)}_{\mu,\nu}(t) + h\,\hat{\boldsymbol{F}}^{(\ell)}_{\mu,\nu}\right)}{q(\mu,\nu;\omega^{(\ell)}_{\mu},\omega^{(\ell)}_{\nu})} \\
q(\mu,\nu;\omega_{\mu},\omega_{\nu}) &= 6 - 4\,\beta\Bigg\{23 + 10\,\cos(\mu\,\omega_{\mu})\,\cos(\nu\,\omega_{\nu}) - \frac{77}{4}\,\left[\cos(\mu\,\omega_{\mu}) + \cos(\nu\,\omega_{\nu})\right] \\
&\quad + \frac{7}{2}\,\left[\cos(2\,\mu\,\omega_{\mu}) + \cos(2\,\nu\,\omega_{\nu})\right] \\
&\quad - \frac{1}{4}\big[\,\cos(3\,\mu\,\omega_{\mu}) + \cos(3\,\nu\,\omega_{\nu}) \\
&\quad\quad + \cos(\mu\,\omega_{\mu} - 2\,\nu\,\omega_{\nu}) + \cos(2\,\mu\,\omega_{\mu} - \nu\,\omega_{\nu}) \\
&\quad\quad + \cos(2\,\mu\,\omega_{\mu} + \nu\,\omega_{\nu}) + \cos(\mu\,\omega_{\mu} + 2\,\nu\,\omega_{\nu})\big]\Bigg\}
\end{aligned} \tag{4.20}$$

A periodic boundary for the result $\boldsymbol{U}$ is not the only possibility. Especially in combination with the vortex extraction we will use a boundary where the first and third order of derivatives vanish. For this purpose the discrete cos-transform can be used instead of the full Fourier.

Nonetheless, the basic form of equation 4.20 will remain. Only the denominator changes to $q(\mu + 1/2, \nu + 1/2; 2\omega_{\mu}, 2\omega_{\nu})$.

4.2.1 Drawbacks of the Method

When the curvature based image registration was tested it came out that some properties of it do not really match the requirements. As shown in the motivation section, vortices arise in the transformation where there should not be any. To verify where they exactly arise, test images and test transformations were applied.

In these tests the samples were deformed with a transformation that is vortex-free. At best the registration procedure should find the inverse transformation which is vortex-free

too.

Equation 4.21 gives such a function that is vortex-free, namely the frog-eye transformation.

$$\boldsymbol{\xi}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & r \geq 1 \\ \frac{1+\sqrt{x_1^2+x_2^2}}{2}\boldsymbol{x} & \text{otherwise.} \end{cases} \tag{4.21}$$

The advantage of using this transformation is the fact that the inverse of it is known. Thus it is easy to recognize transformation errors of the method. The inverse of the frog-eye is defined by

$$\boldsymbol{\xi}^{-1}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & r = 0 \text{ or } r \geq 1 \\ \frac{-1+\sqrt{8r+1}}{2r}\boldsymbol{x} & \text{otherwise.} \end{cases} \tag{4.22}$$

To give a visual impression of what this transformation looks like, a regular grid is taken and transformed. When the image of the regular grid is denoted with $T(\boldsymbol{x})$ then the frog-eye image is $f(\boldsymbol{\xi}(\boldsymbol{x}))$. The result is showed in figure 4.2.

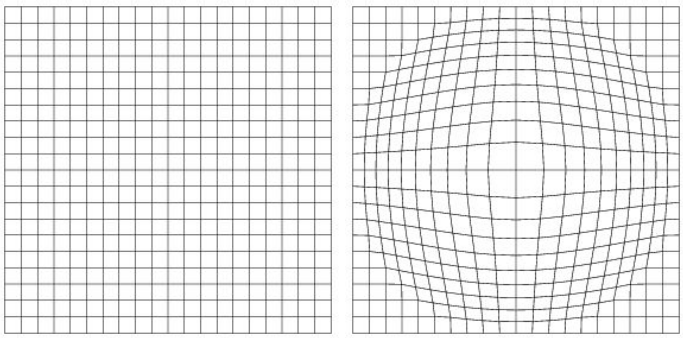


Figure 4.2: Visualization of the frog-eye transformation.

After choosing an appropriate deformation function one should look for suitable images. Although the registration has to work with images of tissue, they are not used here. The problem with those images is that an untrained human eye is not able to recognize registration errors. Far better are pictures of animals because everyone has an intuition for them. Leopards and tigers are excellent samples. Their textured coat provides those high structured regions in the image where the registration method makes mistakes.

Figure 4.3 shows such an image. Particularly the head of the leopard contains a regular pattern of many similar spots. We will see in the results that these regions will exercise errors in the method.

To verify the quality of the registration three basic visualizations are used. The first one, the line integral convolution, was already introduced in section 2.3. It is used here to visualize the displacement field $\boldsymbol{u}$ of the registration.

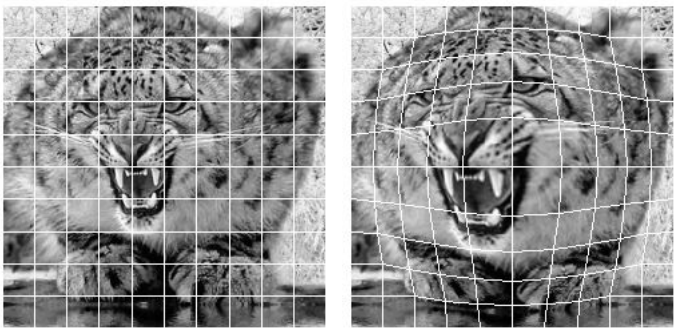


Figure 4.3: An image of a snow leopard which is transformed (right) with the frog-eye transformation

This line integral convolution can be displayed in a three-dimensional way by plotting the lengths of the vectors of the displacement field as height values. The LIC is then put on this height field as texture.

The third method is grounded on the fact that the inverse transformation of the frog-eye is known. Hence it is possible to subtract the displacement field $\boldsymbol{u}$ from the exact solution $\boldsymbol{\xi}^{-1}$ and plot the lengths of the resulting vectors as a height field. This plot allows us to locate the exact positions and size of the error-prone regions.

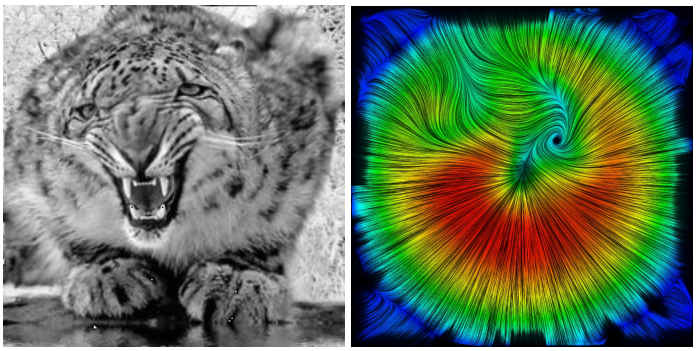


Figure 4.4: The result of the registration of the leopard with the curvature based method.

Figure 4.4 shows the registered leopard. In the upper half image, in the region of nose and eyes, the method worked incorrectly. This can be seen in the line integral convolution. Actually the inverse transformation does not contain any vortices.

The reason for these vortices in the solution is that the method does not find the transformation in one step but in many steps. It begins with a zero solution and solves consequently equation 4.12. This can be seen as a flow into the solution.

When the method works on a region with many similar structures, situations are likely where many directions of the solution flow are possible. It maybe flows into local minima which are not part of the solution and when this occurs it is very likely that the method

will not find the path to the correct solution again.

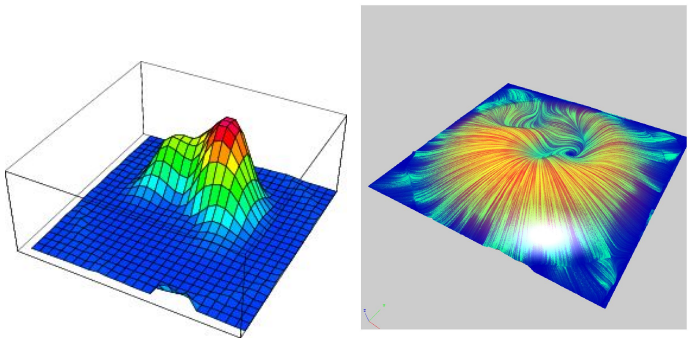


Figure 4.5: Visualizations of the results of the leopard image registration. The left picture shows the absolute errors. The right one is the LIC of the displacement mapped on the height field, representing the lengths of the displacement vectors.

This is the reason for the behavior of the registration with the image of the leopard. In the area of the deformed head several branches for the solution exist which lead to very suboptimal results. That is grounded in the fact that the spotted coat causes local minima.

The error plot in figure 4.5 shows that exactly in those regions the found transformation differs from the inverse of the frog-eye transformation. The idea was now to *lead the solution flow* on a path where vortices are impossible. In every time step vortices are projected out of the partial solution and therefore the solution cannot flow into such a bad branch.

The next section gives a projection method for vortices. We start with the mathematical description of the method and go on with the method of discretization the formula. Finally the basic algorithm is presented so that the reader has a detailed insight in the procedure.

4.3 Projection Technique for Vortices

4.3.1 Helmholtz Decomposition for Vector Fields

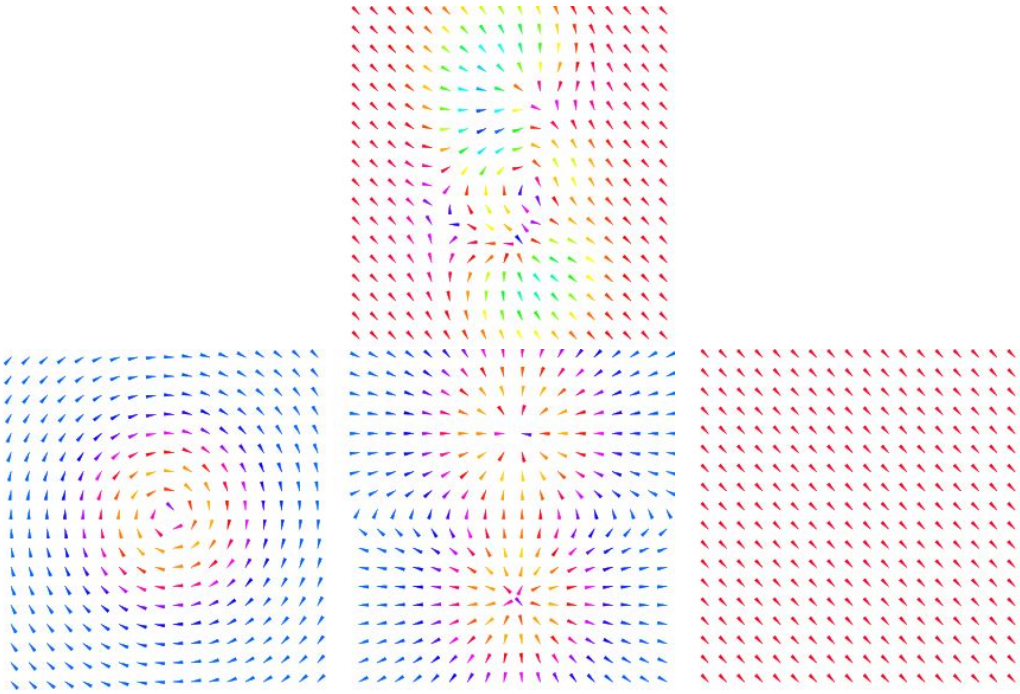


Figure 4.6: Example that shows how a field can be split in its components. The upper image is the sum of the other three. Notice that the left one is the solenoidal, the middle the irrotational and the right one is the homogeneous component of the field.

To realize the projection a theorem is required which comes from Hermann von Helmholtz. This fundamental theorem of vector calculus states that every vector field which is sufficient smooth and decays fast enough can be decomposed to its irrotational and solenoidal components.

$$\boldsymbol{v} = -\nabla\varphi + \nabla \times \boldsymbol{p} + \boldsymbol{h} \tag{4.23}$$

In equation (4.23) the term $-\nabla\varphi$ is the irrotational or the curl free component and the term $\nabla \times \boldsymbol{p}$ is the solenoidal or divergence free component of the field $\boldsymbol{v}$.

The vector field $\boldsymbol{h}$ is the homogeneous part whose divergence and curl is zero:

$$\operatorname{div} \boldsymbol{h} = \operatorname{curl} \boldsymbol{h} = 0 \tag{4.24}$$

One can give a simple proof by substituting the definitions of the differential operators.

If $\nabla \times \boldsymbol{p}$ equals the rotational part of $\boldsymbol{v}$ then it should be a result of equation (4.23) when applying the *curl* operator to it.

$$\begin{aligned}
\nabla \times \boldsymbol{v} &= \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{pmatrix} \times \begin{pmatrix} -\frac{\partial}{\partial x_1}\varphi \\ -\frac{\partial}{\partial x_2}\varphi \\ -\frac{\partial}{\partial x_3}\varphi \end{pmatrix} + \nabla \times (\nabla \times \boldsymbol{p}) + \underbrace{\nabla \times \boldsymbol{h}}_{=0} \\
\nabla \times \boldsymbol{v} &= \begin{pmatrix} -\frac{\partial^2}{\partial y \partial z}\varphi & + & \frac{\partial^2}{\partial y \partial z}\varphi \\ -\frac{\partial^2}{\partial z \partial x}\varphi & + & \frac{\partial^2}{\partial z \partial x}\varphi \\ -\frac{\partial^2}{\partial x \partial y}\varphi & + & \frac{\partial^2}{\partial x \partial y}\varphi \end{pmatrix} + \nabla \times \boldsymbol{p} \\
\nabla \times \boldsymbol{v} &= \nabla \times \boldsymbol{p}
\end{aligned} \tag{4.25}$$

In the same way one can prove that $-\nabla\varphi$ is the irrotational component of $\boldsymbol{v}$.

Since we operate on images our working space is only two dimensional. Therefore the rotational field $\nabla \times \boldsymbol{p}$ and the divergence field $-\nabla\varphi$ only have values in the first two coordinates. For this reason the vector potential $\boldsymbol{p}$ has only one entry in the third coordinate.

$$\boldsymbol{p} = \begin{pmatrix} 0 \\ 0 \\ p \end{pmatrix} \tag{4.26}$$

p is a scalar field and describes the magnitude and the direction of the rotation in every point $(x, y, 0)^\top$.

$$\nabla \times \boldsymbol{p}(x, y) = \nabla \times \begin{pmatrix} 0 \\ 0 \\ p(x, y) \end{pmatrix} = \begin{pmatrix} \partial p(x, y)/\partial y \\ -\partial p(x, y)/\partial x \\ 0 \end{pmatrix} \tag{4.27}$$

Now everything is prepared to formulate the following variational problem: We are trying to find a vector field that minimizes the distance to our original field and which is per definition a vortex-only field. This can be expressed in a variational equation that minimizes the following functional.

$$\mathcal{F}(\boldsymbol{p}) = \frac{1}{2} \int_\Omega |\boldsymbol{v} - \nabla \times \boldsymbol{p}|^2 d\boldsymbol{x} \tag{4.28}$$

Since $\boldsymbol{p}$ is a vector potential and has only one non-zero component in the third dimension it is possible to expand the expression in 4.28 and change the equation to be a

functional in p.

$$\mathcal{F}(p) = \frac{1}{2}\int_\Omega \left(v_1 - \frac{\partial}{\partial x_2}p\right)^2 + \left(v_2 + \frac{\partial}{\partial x_1}p\right)^2 d\boldsymbol{x} \tag{4.29}$$

The first Gâteaux variation is given by

$$\begin{aligned}
&\delta\mathcal{F}(p+\epsilon g)\\
&= \frac{1}{2}\int_\Omega d\boldsymbol{x}\left(\left(v_1 - \frac{\partial}{\partial x_2}(p+\epsilon g)\right)^2 + \left(v_2 + \frac{\partial}{\partial x_1}(p+\epsilon g)\right)^2\right)\frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0}\\
&= \frac{1}{2}\int_\Omega d\boldsymbol{x} 2\left(v_2 + \frac{\partial}{\partial x_1}(p+\epsilon g)\right)\frac{\partial}{\partial x_1}g - 2\left(v_1 - \frac{\partial}{\partial x_2}(p+\epsilon g)\right)\frac{\partial}{\partial x_2}g\bigg|_{\epsilon=0}\\
&= \frac{1}{2}\int_\Omega d\boldsymbol{x}\ \ 2\left(v_2 + \frac{\partial}{\partial x_1}(p+\epsilon g)\right)\frac{\partial}{\partial x_1}g - 2\left(v_1 - \frac{\partial}{\partial x_2}(p+\epsilon g)\right)\frac{\partial}{\partial x_2}g\bigg|_{\epsilon=0}\\
&= \quad\int_\Omega d\boldsymbol{x}\left(\left(\frac{\partial}{\partial x_1}v_2 + \frac{\partial^2}{\partial x_1^2}p)\right) - \left(\frac{\partial}{\partial x_2}v_1 - \frac{\partial^2}{\partial x_2^2}p\right)\right)g
\end{aligned} \tag{4.30}$$

A function p minimizing this derivative has to fulfill the equation that follows directly by setting the term in the integral to zero.

$$\begin{aligned}
0 &= \left(\frac{\partial}{\partial x_1}v_2 + \frac{\partial^2}{\partial x_1^2}p)\right) - \left(\frac{\partial}{\partial x_2}v_1 - \frac{\partial^2}{\partial x_2^2}p\right)\\
\frac{\partial^2}{\partial x_1^2}p + \frac{\partial^2}{\partial x_2^2}p &= \frac{\partial}{\partial x_1}v_2 - \frac{\partial}{\partial x_2}v_1\\
\Delta p &= \Phi(\boldsymbol{v})
\end{aligned} \tag{4.31}$$

Concluding one can say: To find the potential field p that represents the strength and the direction of the vortices in a vector field $\boldsymbol{v}$, the Poisson equation given in 4.31 has to be solved.

4.3.2 Approximation for Discrete Vector Fields

Since we do not operate with analytic functions but with discrete displacement fields it is not possible to calculate the Poisson equation directly. How to approximate the derivatives on discrete data was presented in section 2.4 of the background chapter.

The displacement field is a set of vectors on a regular mesh and is denoted as $\boldsymbol{u}[i,j]$ with $i = 1,\ldots,M$ and $j = 1,\ldots,N$. To find the discrete potential $p[i,j]$ for every point

(i, j) the following approximation for the first order derivatives in direction i is used:

$$\frac{\partial}{\partial i} v_2(i,j) \approx \frac{1}{12h}(v_2[i-2h,j] - 8v_2[i-h,j] + 8v_2[i+h,j] - v_2[i+2h,j]). \quad (4.32)$$

The second order derivative which is required for the Laplace operator is approximated by

$$\frac{\partial^2}{\partial i^2} p[i,j] \approx \frac{1}{12h^2}(-p[i-2h,j]+16p[i-h,j]-30p[i,j]+16p[i+h,j]-p[i+2h,j]). \quad (4.33)$$

These approximations are used to form a discrete version of the Poisson equation given in 4.31.

$$\begin{aligned}
&\frac{1}{12h^2}(-p[i-2h,j] + 16p[i-h,j] - 30p[i,j] + 16p[i+h,j] - p[i+2h,j] - \\
&\quad p[i,j-2h] + 16p[i,j-h] - 30p[i,j] + 16p[i,j+h] - p[i,j+2h]) = \\
&\quad \frac{1}{12h}(v_2[i-2h,j] - 8v_2[i-h,j] + 8v_2[i+h,j] - v_2[i+2h,j] - \\
&\quad\quad v_1[i,j-2h] - 8v_1[i,j-h] + 8v_1[i,j+h] - v_1[i,j+2h]) \quad (4.34)
\end{aligned}$$

As one can see this gives a linear system of equations for p. For a displacement field with 512×512 vectors, what tends to be a small dimension for images, one has to solve a LSE with 262144 equations. A good way to solve such systems is to use the two-dimensional discrete Fourier transformation.

$$p[i,j] = \frac{1}{N \cdot M} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} \hat{p}[\mu,\nu] \exp\left(2\pi \mathrm{i} \left(\frac{i\mu}{M} + \frac{j\nu}{N}\right)\right) \quad (4.35)$$

That is grounded in the shift theorem which states that a shift in the coordinates changes to a multiplication in the Fourier base. We can prove it for the two-dimensional Fourier

representation: Given a shift k to one of the coordinates, then

$$\begin{aligned} p[i+k,j] &= \frac{1}{N \cdot M} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} \hat{p}[\mu,\nu] \exp\left(2\pi\mathbb{i}\left(\frac{(i+k)\mu}{M} + \frac{j\nu}{N}\right)\right) \\ &= \frac{1}{N \cdot M} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} \hat{p}[\mu,\nu] \exp\left(2\pi\mathbb{i}\left(\frac{k\mu}{M} + \frac{i\mu}{M} + \frac{j\nu}{N}\right)\right) \\ &= \frac{1}{N \cdot M} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} \hat{p}[\mu,\nu] \cdot \omega_m^k \exp\left(2\pi\mathbb{i}\left(\frac{i\mu}{M} + \frac{j\nu}{N}\right)\right) \end{aligned}$$

with

$$\omega_m = \exp\left(2\pi\mathbb{i}\frac{\mu}{M}\right). \tag{4.36}$$

Using this theorem it is possible to rewrite equation 4.34. By changing into the Fourier base the shifts in the coordinates are replaced by multiplications with the appropriate phases. Notice that only the left side is transformed into the Fourier basis. The right side is calculated as it is, for the reason that a Fourier transform requires an assumption of the boundary conditions. This is circumvented by using a special formula for the derivatives at boundary points which was introduced on page 18.

The Fourier transformed, pre-calculated right side is denoted by $\hat{\varphi}[i,j]$ and the rewritten formula 4.34 is given by

$$\begin{aligned} &\frac{1}{12h^2}\left(-\omega_m^{-2h}\hat{p}[\mu,\nu] + 16\omega_m^{-h}\hat{p}[\mu,\nu] - 30\hat{p}[\mu,\nu] + 16\omega_m^{h}\hat{p}[\mu,\nu] - \omega_m^{2h}\hat{p}[\mu,\nu] - \right. \\ &\left.\omega_n^{-2h}\hat{p}[\mu,\nu] + 16\omega_n^{-h}\hat{p}[\mu,\nu] - 30\hat{p}[\mu,\nu] + 16\omega_n^{h}\hat{p}[\mu,\nu] - \omega_n^{2h}\hat{p}[\mu,\nu]\right) = \hat{\varphi}[\mu,\nu] \end{aligned} \tag{4.37}$$

This equation can now easily be resolved to $\hat{p}[i,j]$ and the term in braces can be simplified by using trigonometric rules.

$$\hat{p}[\mu,\nu] = -\frac{6h^2\hat{\varphi}[\mu,\nu]}{30 - 16\cos(\lambda) + \cos(2\lambda) - 16\cos(\kappa) + \cos(2\kappa)}$$

with

$$\lambda = 2\pi h\frac{\mu}{M} \qquad \text{and} \qquad \kappa = 2\pi h\frac{\nu}{N} \tag{4.38}$$

Equation 4.38 has to be calculated for every vector of a field to get the Fourier transformed potential $\hat{p}$. Since the whole procedure from an input field $\boldsymbol{v}$ to its rotational part $\nabla \times \boldsymbol{p}$ consists of several steps, we will give a summarized order of events.

1. Calculate $\varphi[i,j]$ for $i = 1,\ldots,M$ and $j = 1,\ldots,N$ which is the right side of equation 4.31. For the approximation of the derivatives, finite differences are used (eq. 4.32). When boundary points are processed where not enough surrounding points exists, another formula is used which takes more inner points (see p. 18).

2. Transform $\varphi[i,j]$ with the DFT.

3. For every point (i,j) with $i = 1,\ldots,M$ and $j = 1,\ldots,N$ calculate equation 4.38. Remember that h is the distance between the grid points. It is assumed we have a regular grid and the distance between the grid points is equal in every direction.

4. Calculate the inverse Fourier transform of $\hat{p}[i,j]$ for every $i = 1,\ldots,M$ and $j = 1,\ldots,N$. This results in the potential field p which represents the strength and the direction of all vortices in the input field $\boldsymbol{v}$. The vector potential $\boldsymbol{p}$ which is required for the next step is simply given by $\boldsymbol{p} = (0,0,p)^\top$.

5. The vortex field of $\boldsymbol{v}$ can now be calculated by $\nabla \times \boldsymbol{p}$.

4.4 Algorithm and Implementation

Since the last section gives a description which is already very close to an algorithm, the concern of this part is to give a better insight into the implementation. For that reason we will provide a ready for use pseudo code representation which can be implemented instantly.

Our implementation had to meet some special requirements. Therefore we firstly present a short list which explains design notes and the language used:

- The basic Helmholtz decomposition should be a stand-alone, extensible and reusable library written in C++.

- Since it is far easier to visualize and work with vector fields in MATHEMATICA a package and a MATHLINK program was required connecting the Helmholtz decomposition library with MATHEMATICA . How type conversion, function calls and data transfer work in those MATHLINK programs can be found for instance in [Wol03, Kus97].

- The third requirement had significant influences on the design of the classes for the C++ library. The projection algorithm should be included into the existing image registration but it was not possible to write a separate module. The reason for this is that the already existing library is an all-in-one CWEB implementation.

  The final decision was to *rebuild* the C struct types and functions for vectors, arrays and vector arrays with C++ classes in a way that function calls are very close to the notation in the CWEB file. Then it is on the one hand possible to copy the vortex projection into CWEB code and on the other hand to write and test an independent module.

4.4.1 THE PROJECTION ALGORITHM

The basic projection algorithm 1 is straight forward. For the calculation of the curls at the beginning and in the end the finite difference approximation showed in algorithm 2 was used. The Fourier transformations, forward and backward, were provided by the FFTW library. This procedure is not explained here and we refer to the manual and literature [FJ98].

Algorithm 1 The Helmholtz decomposition for a discrete vector field $\boldsymbol{v}$ with values $\boldsymbol{v}[i,j]$ in the range of $i = 1,\ldots,M$ and $j = 1,\ldots,N$.

for all i,j with $i = 1,\ldots,M$ and $j = 1,\ldots,N$ **do**
  $\varphi[i,j] \Leftarrow \frac{\partial v_2[i,j]}{\partial x_1} - \frac{\partial v_1[i,j]}{\partial x_2}$ {see algorithm 2}
end for
$\hat{\varphi} \Leftarrow DFT(\varphi)$
$\hat{p} \Leftarrow$ Process algorithm 3
$p \Leftarrow InverseDFT(\hat{p})$
for all i,j with $i = 1,\ldots,M$ and $j = 1,\ldots,N$ **do**
  $p_x \Leftarrow \frac{\partial}{\partial x_2} p[i,j]$
  $p_y \Leftarrow -\frac{\partial}{\partial x_1} p[i,j]$
  $\boldsymbol{p}[i,j] \Leftarrow (p_x, p_y)$
end for

The first calculation in the algorithm 1 is the right side of equation 4.31. For this purpose and for the derivatives at the end we used the concept of algorithm 2. As said in the previous section the derivatives boundary points are calculated with different formulas. Therefore we use a case discrimination that decides whether we are directly on, one point away or more than one point away from the boundary.

The next step in algorithm 1 is that the resulting potential field is processed with the help of the FFTW library which is a free and very fast implementation [FJ98] of the discrete Fourier transform. With this transformed potential field everything is prepared to

Algorithm 2 Calculation of the first order derivative w.r.t. x_2 of the first component of a vector field $\boldsymbol{v}$ at the position (i, j).

Given a discrete vector field $\boldsymbol{v}$ and $\boldsymbol{v}[i, j]$ the value at the position (i, j) with $i = 1, \ldots, M$ and $j = 1, \ldots, N$. Assume h is the distance between two gridpoints in the direction of j.
if $j = 1$ **then**
$d \Leftarrow -\frac{3}{2}v_1[i, 1] + 2v_1[i, 2] - \frac{1}{2}v_1[i, 3]$
end if
if $j = 2$ **then**
$d \Leftarrow -\frac{1}{3}v_1[i, 1] - \frac{1}{2}v_1[i, 2] + v_1[i, 3] - \frac{1}{6}v_1[i, 4]$
end if
if $j = (N - 1)$ **then**
$d \Leftarrow \frac{1}{3}v_1[i, j + 1] + \frac{1}{2}v_1[i, j] - v_1[i, j - 1] + \frac{1}{6}v_1[i, j - 2]$
end if
if $j = N$ **then**
$d \Leftarrow \frac{3}{2}v_1[i, j] - 2v_1[i, j - 1] + \frac{1}{2}v_1[i, j - 2]$
end if
if $j > 2 \wedge j < N - 1$ **then**
$d \Leftarrow \frac{1}{12}v_1[i, j - 2] - \frac{2}{3}v_1[i, j - 1] + \frac{2}{3}v_1[i, j + 1] - \frac{1}{12}v_1[i, j + 2]$
end if
return d/h

calculate equation 4.38 for every point in the field. This iteration over all points is shown in algorithm 3.

After all points are calculated, the resulting field $\hat{p}$ is processed through the inverse Fourier transform. The last step is the calculation of $\nabla \times \boldsymbol{p}$ which can again be performed with the help of algorithm 2.

4.4.2 AN INTERFACE TO MATHEMATICA

To test the Helmholtz decomposition and to see how it works in different circumstances an interface to MATHEMATICA was implemented. Basically this interface is for converting the MATHEMATICA types down to C types and for providing a better user interface. To call the C++ library function from MATHEMATICA three stages have to be developed. The last stage which is the Helmholtz decomposition in this case was already explained.

When using MATHEMATICA most of the additional features come from so called *packages* which are basically a set of normal function definitions. Such a package is the interface for the user who accesses your low-level library methods. In combination with the Helmholtz decomposition library such a package was developed too. It provides not only access to the library functions but some more tools for working with vector fields. This package is called *FieldTools* .

Algorithm 3 The basic iteration to calculate the Fourier transformed vortex potential of the input field.

for $l = 0, \ldots, N-1$ **do**
  $\lambda \Leftarrow \frac{2\pi hl}{N}$
  $c_{2l} \Leftarrow \cos(\lambda)$
  $c_{4l} \Leftarrow \cos(2\lambda)$
  for $k = 0, \ldots, M-1$ **do**
    $\kappa \Leftarrow \frac{2\pi hk}{M}$
    $c_{2k} \Leftarrow \cos(\kappa)$
    $c_{4k} \Leftarrow \cos(2\kappa)$
    $\hat{p}[i,j] \Leftarrow \frac{6h^2\hat{\varphi}[i,j]}{30-16c_{2l}+c_4l-16c_{2k}+c_{4k}}$
  end for
end for

Listing 4.1: A part of the *FieldTools* package showing the definition of a function.

```
0  Options[ExtractRotationalField]={
     TransformType->FullFourier
   }

   ExtractRotationalField::parm="Wrong␣type␣or␣form␣of␣parameter.";
5  ExtractRotationalField[m_,dx_:NumericQ,dy_:NumericQ,opts___]:=
   Module[{dim,xData,yData,ret,tt,ttype},
     dim = Dimensions[m];
     (* Extract the option settings *)
     {tt}={TransformType}/.{opts}/.Options[ExtractRotationalField];
10   (* Check if the array has the right
     dimensions and is bigger than 5x5 *)
     If[ArrayQ[m,3,NumericQ] && dim[[3]]==2 && dim[[1]]>5 && dim[[2]]>5,
       (*then*)
       {xData,yData}=Transpose[Flatten[m,1]];
15     ttype=Switch[tt,WithSin,0,WithCos,1,_,2];
       (* Call the MathLink function *)
       ret=Global`rawExtractRotationalField[ N[xData],
                                             N[yData],
                                             dim[[1]],
20                                           dim[[2]],
                                             N[dx],
                                             N[dy],
                                             ttype];
       If[ret=!=$Failed,
25       (*then*)
         (*Transpose the result to have a list in the form of
         {{{x11,y11},{x12,y12},...},{{x21,y21},...}} *)
         Return[Transpose[ret, {3, 1, 2}]];,
         (*else*)
30       Return[$Failed];],
       (*else*)
       Message[ExtractRotationalField::parm];
       Return[$Failed];
       ]
35 ]
```

To see how a function call in MATHEMATICA down to the C++ library works we will follow the way of one provided *FieldTools* function. As an example, listing 4.1 gives the definition of the `ExtractRotationalField` function.

The definition consists broadly of three parts. The first one is to process options and check the validity of the parameters. In the second part at line 17 a call to the raw function is done. This call leads to the next deeper level which is the MATHLINK layer. The third and last part handles return values and errors and gives the results back to the MATHEMATICA front-end where it was called.

The raw function of the extraction in line 17 is of interest because it sends the data to a linked C program. MATHEMATICA provides header files and a preprocessor for creation of those programs. One has to write a combination file consisting of MATHEMATICA and C code. This template file is processed with the *mprep* tool which is this preprocessor. The output is an extended C file so that the compiled program can be called by MATHEMATICA

.

Listing 4.2: A part of the MATHLINK template file.

```
0  :Begin:
   :Function:      extractRotationalField
   :Pattern:       rawExtractRotationalField[dataX_List,dataY_List,
                     nx_Integer,ny_Integer,dx_Real,
                     dy_Real,transformType_Integer]
5  :Arguments:     {dataX,dataY,nx,ny,dx,dy,transformType}
   :ArgumentTypes: {RealList,RealList,Integer,Integer,Real,Real,Integer}
   :ReturnType:    Manual
   :End:

10 void extractRotationalField(
      double *dataX,
      long lx,
      double *dataY,
      long ly, int nx, int ny, double dx, double dy, int t){
15
        if(lx!=ly || nx!=ny || nx<5){
          MLPutSymbol(stdlink,"$failed");
          return;
        }
20
        VectorArray2D *in = new VectorArray2D();
        int addr;
        in->Init(nx,ny,1.0,1.0);
        for(int i=0; i<nx; ++i){
25        for(int j=0; j<ny; ++j){
            addr=j*nx+i;
            in->set(i,j,dataX[addr],dataY[addr]);
          }
        }
30
        VectorArray2D *out = new VectorArray2D();
        switch(t){
          case 0:
            HelmholtzHodge2D::ExtractRotationalFieldWithSinTransform(
35            in,
              in->nx,
              in->ny,
              in->dx,
              in->dy,
40            out);
            break;
          case 1:
          // The rest is omitted
```

As one can see in listing 4.2 the file consists of parts that are dedicated to MATHEMATICA and parts that are pure C or C++. The first 8 lines are for the pattern matching of the function call. Exacting, when the MATHEMATICA package calls a function matching the `:Pattern:` section then this call is forwarded to the C function matching `:Function:`.

Inside the `extractRotationalField` function the last layer is reached by calling the Helmholtz decomposition library. Now the three steps, namely the MATHEMATICA package, the MATHLINK program and the decomposition library, are complete. The decomposition algorithm returns its result to the higher level until it reaches the user in the MATHEMATICA front-end.

With this interface to MATHEMATICA it is now possible to validate whether the Helmholtz decomposition works correctly. In the next section we will apply some test vector fields to the algorithm. We will show how exactly the method is working and what properties this decomposition has.

4.4.3 TEST SITUATIONS FOR THE PROJECTION

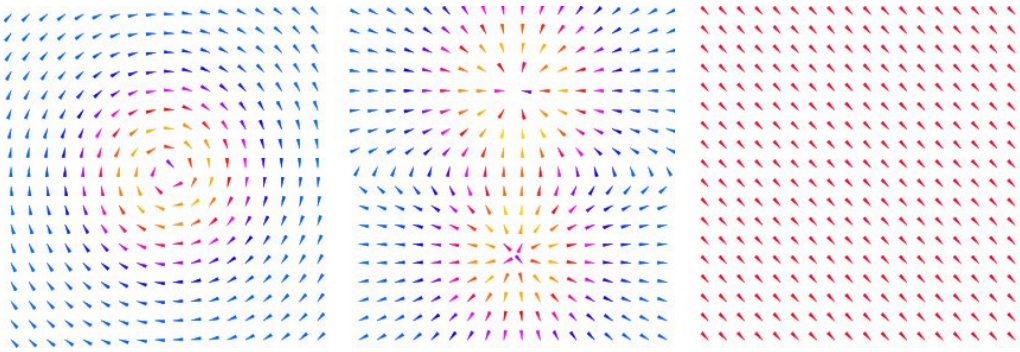


Figure 4.7: Three different vector fields. The left is a rotational or divergence-free field. The middle one is a solenoidal field and the right field has only homogeneous parts.

To show how the vortex projection algorithm works three fields were created, given in figure 4.7. Every single field has special properties which the other two do not have. Particularly the left field is the only one with a vortex, because the middle one is a divergence-only field and the left one is homogeneous.

The plan is to add them together and let the algorithm find the rotational part. Since we know how the real rotational part must look like we can verify whether the method works correctly or not.

By adding them together one can see in figure 4.8 that it is not really possible to recognize the parts of the field. Notice that these plots of the fields show *all* vectors of the

discrete data. This means the algorithm does not have more information about the field than the viewer has.

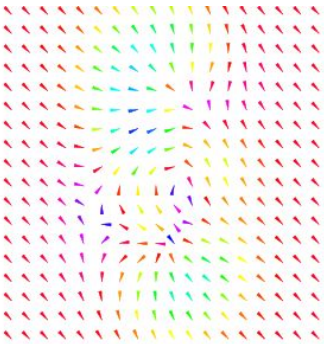


Figure 4.8: The sum of the three single fields of figure 4.7.

In figure 4.9 the result of the decomposition algorithm is shown together with the original rotational part. The method extracted the vortex in the middle very well.

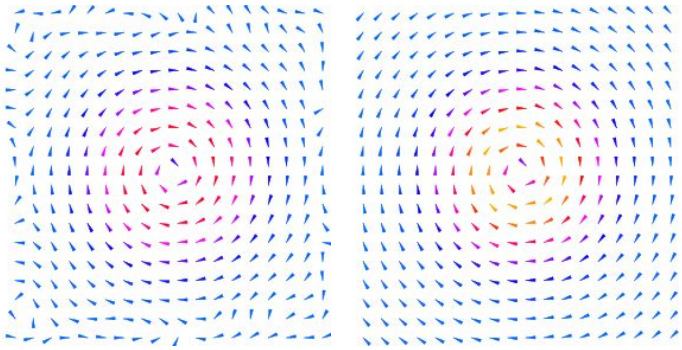


Figure 4.9: The result of the decomposition algorithm together with the original rotational part (right).

To give an exact measure for the quality of the extraction the difference of the original field, denoted by $\boldsymbol{v}_o[i,j]$, and the extracted one, denoted by $\boldsymbol{v}_e[i,j]$ is calculated. The summed norm of the difference field is then given by

$$\sum_{i=1}^{M}\sum_{j=1}^{N}\|\boldsymbol{v}_o[i,j]-\boldsymbol{v}_e[i,j]\|. \tag{4.39}$$

For better comparisons table 4.1 gives not only the summed norm of the error but also the values for the single fields and the sum of them. Since it is far more interesting where the errors were made, a plot of the difference field is very useful. To create it the norms of the difference vectors are not added together but ploted as a hight field.

In this plot (figure 4.10) one can see that the error is very high in middle region. This is not unexpected because the center is not very smooth. At the boundary mistakes arose

Vector field	Norm
Original added fields	561.068
Original homogeneous part	458.205
Original divergence part	226.138
Original rotational part	118.797
Extracted rotational part	118.455
Difference to the original	0.342

Table 4.1: The norm of the fields.

too. As we will see very soon this is caused by the boundary itself. The problem is that the theorem of Helmholtz is given for smooth and fast decaying vector fields. With discrete data it is not always possible to have a smooth enough field. Furthermore at the boundary the field ends immediately. What if the vortices in the field do not decay fast enough and we have a vortex on the boundary?

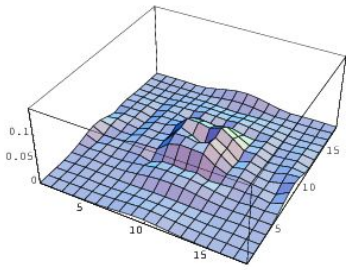


Figure 4.10: The error of the vortex projection.

We will see that vortices near the boundary are going to be a problem for the method. The second sample will show this. A vortex was placed directly onto the boundary so that only one half of it is visible. In such a situation the choice of the boundary conditions will become very important for the success of the method. By choosing different transformations for the decomposition it is possible to force the extracted rotational field to fulfill different boundary conditions.

1. If the Fourier transform is used, the extracted field will have a periodic boundary.
2. If the sine transform is used, the extracted field is zero in the direction vertical to the boundary.
3. If the cosine transform is used, the extracted field is zero in the direction of the boundary.

Figure 4.11 shows the results of different boundary conditions. With a periodic boundary a part of the vortex in the top is reflected into the bottom of the field. The cosine transform

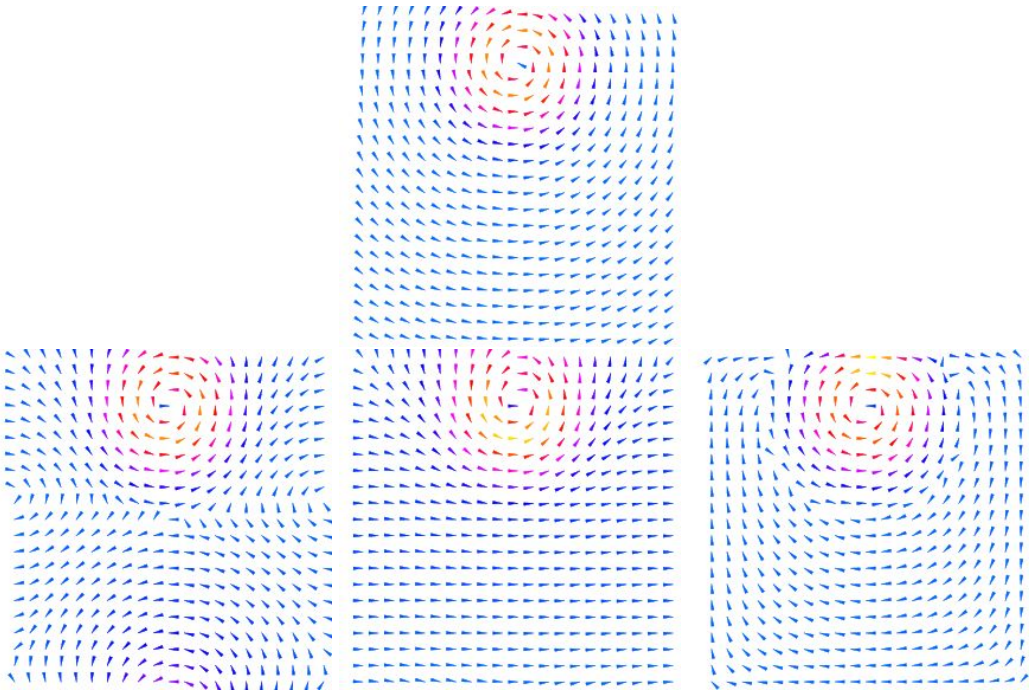


Figure 4.11: Sample for a big vortex very near to the boundary. In the second row the results of the decomposition with different boundary conditions are shown. The left picture shows Fourier transform with periodic boundary, the middle one shows cosine transform with normal boundary and the right one shows sine transform with zero displacement to the boundary.

with its normal displacement to the boundary extracts the vortices worst. That is caused by the fact that a vortex *near* the boundary does not have many vertical components. Hence for most vortex containing fields the sine transform method yields the best results.

Vector field	Norm	Percentage
Original field	1449.32	100%
Difference with Fourier transform	570.571	39%
Difference with cosine transform	962.193	66%
Difference with sine transform	276.154	19%

Table 4.2: A comparison of the made errors of different methods.

4.5 Integration into the Registration Process

In the last sections we have shown how the two parts, namely the curvature registration and the vortex extraction, work seperately. Now we want to describe how to combine these two things.

The curvature registration solution method iterates over the time and the discretization level. Since we start with a zero displacement, the only place where vortices can arise is the $\boldsymbol{F}^{(\ell)}$ term which is Fourier transformed and used in the following equation. This equation was already described on page 40.

$$\hat{\boldsymbol{V}}^{(\ell)}_{\mu,\nu}(t+h) = -\hat{\boldsymbol{U}}^{(\ell)}_{\mu,\nu}(t) + \frac{6\left(2\,\hat{\boldsymbol{U}}^{(\ell)}_{\mu,\nu}(t) + h\,\hat{\boldsymbol{F}}^{(\ell)}_{\mu,\nu}\right)}{q(\mu,\nu;\omega^{(\ell)}_{\mu},\omega^{(\ell)}_{\nu})} \tag{4.40}$$

Therefore, the place where we will put in the vortex extraction directly after the force for a single iteration step was calculated. Algorithm 4 gives a summarized view of the sequence of steps which were explained explicitly through this chapter.

Algorithm 4 Given two images, namely the reference R and the deformable template T, a number of discretization levels L, a time step $h > 0$ and an initial displacement field at zero level with $\boldsymbol{U}^{(0)} = 0$. Furthermore the images were sampled for every discretization level so we have $R^{(\ell)}$ and $T^{(\ell)}$ for every level $\ell = 0, \ldots, L$. The following algorithm describes the complete curvature-based registration procedure and shows where the vortex extraction takes place.

$t \Leftarrow 0$
repeat
  for $\ell = 0, \ldots, L$ **do**
    for all $m = 1, \ldots, M^{(\ell)}$ and $n = 1, \ldots, N^{(\ell)}$ **do**
      $\boldsymbol{F}^{(\ell)} \Leftarrow \left(R^{(\ell)}(\boldsymbol{x}_{m,n}) - T^{(\ell)}(\boldsymbol{x}_{m,n} - \boldsymbol{U}^{(\ell)}_{m,n})\right) \cdot \text{Grad}(T^{(\ell)}(\boldsymbol{x}_{m,n} - \boldsymbol{U}^{(\ell)}_{m,n}))$
    end for
    $\boldsymbol{F}^{(\ell)} \Leftarrow \text{VortexExtraction}(\boldsymbol{F}^{(\ell)})$
    $\hat{\boldsymbol{U}}^{(\ell)} \Leftarrow \text{DFT}(\boldsymbol{U}^{(\ell)})$
    $\hat{\boldsymbol{F}}^{(\ell)} \Leftarrow \text{DFT}(\boldsymbol{F}^{(\ell)})$
    for all $\mu = 1, \ldots, M^{(\ell)}$ and $\nu = 1, \ldots, N^{(\ell)}$ **do**
      $\hat{\boldsymbol{V}}^{(\ell)}_{\mu,\nu} \Leftarrow -\hat{\boldsymbol{U}}^{(\ell)}_{\mu,\nu} + \frac{6\left(2\,\hat{\boldsymbol{U}}^{(\ell)}_{\mu,\nu} + h\,\hat{\boldsymbol{F}}^{(\ell)}_{\mu,\nu}\right)}{q(\mu,\nu;\omega^{(\ell)}_{\mu},\omega^{(\ell)}_{\nu})}$
    end for
  end for
  $\boldsymbol{V}^{(\ell)} \Leftarrow \text{InverseDFT}(\hat{\boldsymbol{V}}^{(\ell)})$
  $error \Leftarrow \left\|\boldsymbol{V}^{(L)} - \boldsymbol{U}^{(L)}\right\|$
  $\boldsymbol{U}^{(L)} \Leftarrow \boldsymbol{V}^{(L)}$
  for $\ell = L-1, \ldots, 0$ **do**
    $\boldsymbol{U}^{(\ell)} \Leftarrow \mathcal{R}\left[\boldsymbol{U}^{(\ell+1)}\right]$
  end for
  $t \Leftarrow t + h$
until $error > \epsilon$

4.6 REMARKS

With the last chapter we showed that the introduced curvature-based method has some basic drawbacks when ill-suited image data is used. The question is now whether we can improve the method significantly by applying our projection algorithm.

As we saw the projection method performs unreasonably in some cases. Therefore chapter 5 will discuss whether or not the error in the Helmholtz decomposition deteriorates the method completely.

CHAPTER 5

RESULTS

PLEASE NOTE:

Some Quantum Physics Theories Suggest That When the Consumer Is Not Directly Observing This Product, It May Cease to Exist or Will Exist Only in a Vague and Undetermined State. [HS91]

In this part we will compare two existing image registration methods with our new vortex extraction procedure. The most interesting part is, how much it improves the underlying curvature based method. The second comparison is to confront our method with the vortex suppression registration of Kuska and Braumann [BK06].

We will show three different examples where the first and the second one is an artificially created problem and the third is a realistic application. We have chosen the images of a leopard and a tiger for the artificial samples. The realistic application example is given in two slices that were consecutively cut from a specimen of a squamous cell carcinoma of the uterine cervix.

The images of the animals were deformed using the frog-eye transformation transformation already introduced. The advantage of this transformation is that it is a non-linear deformation which is bijective. Additionally its inverse is known. Therefore we can easily provide the correct displacement vector for each point.

5.1 Leopard with Frog-eye Deformation

5.1.1 Input Images

Figure 5.1: Leopard input images. The left serves as reference and the right as deformable template.

For the first example we use the front view of a leopard. The position of the animal in the image was chosen in this way because the frog-eye takes effect in a radius around the middle of the picture. With this choice we have the head containing the most fine structures in exactly this region. Therefore it is very likely that if errors occur they are caused by the dark spots of the head.

5.1.2 Registration Results

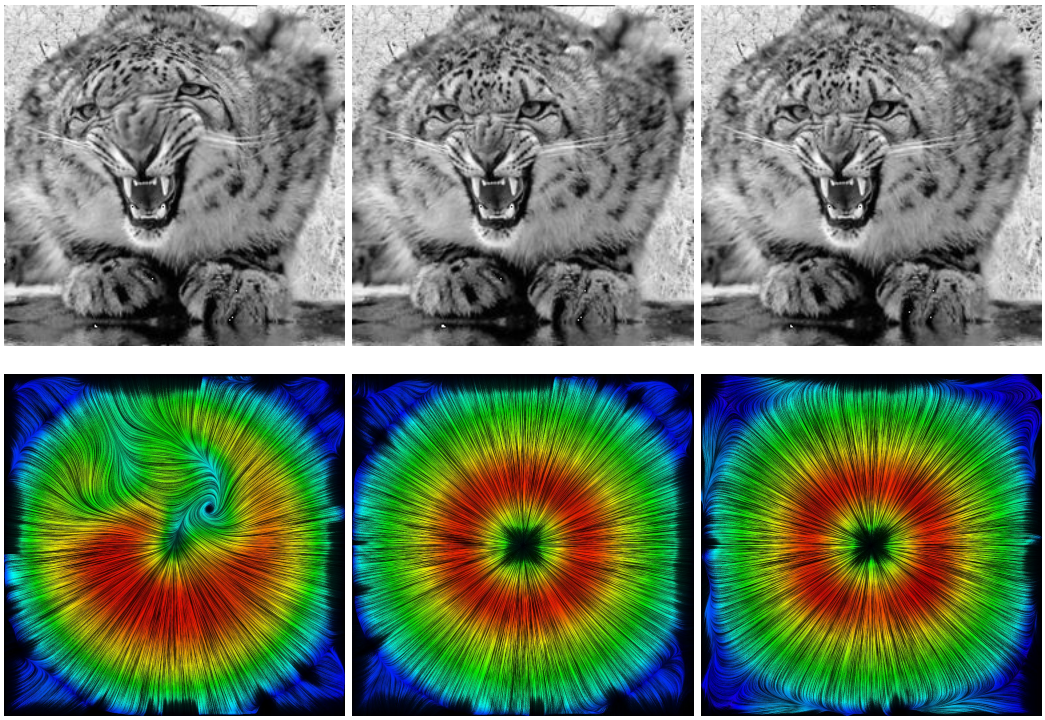


Figure 5.2: The upper row shows the results of the registration methods. Below every leopard one can see the LIC of the belonging displacement field. From left to right following methods are shown: Curvature-based, curvature-based with vortex suppression and curvature-based with vortex extraction.

The results of the registrations are shown in the upper row of figure 5.2. One can observe that without any vortex handling the leopard cannot be registered well. It still contains an intense deformation of the head.

This behavior is easier to observe in the LIC because the displacement field of the inverse frog-eye is very regular. The difference between the first and the other two examples is apparent.

The vortex suppression and the vortex extraction method seem to work similarly. Especially it is not possible to find any registration errors when one looks at the images of the leopards only. Even the LIC of the two vortex handling methods look quite similar. Differences are notable at the borders but it is not possible to say whether errors occur or not.

Very interesting is why the irregularity in the first method does not look like a real vortex. Our new method extracts only vortices and is not able to correct any other properties.

The answer is that this wrong displacement field is the result of many small calculation steps. During the whole process the method *flowed* into a wrong result because each step was allowed to use vortices to find the next step. Maybe this leads to better intermediate solutions but the result is not the global minimum of the registration problem.

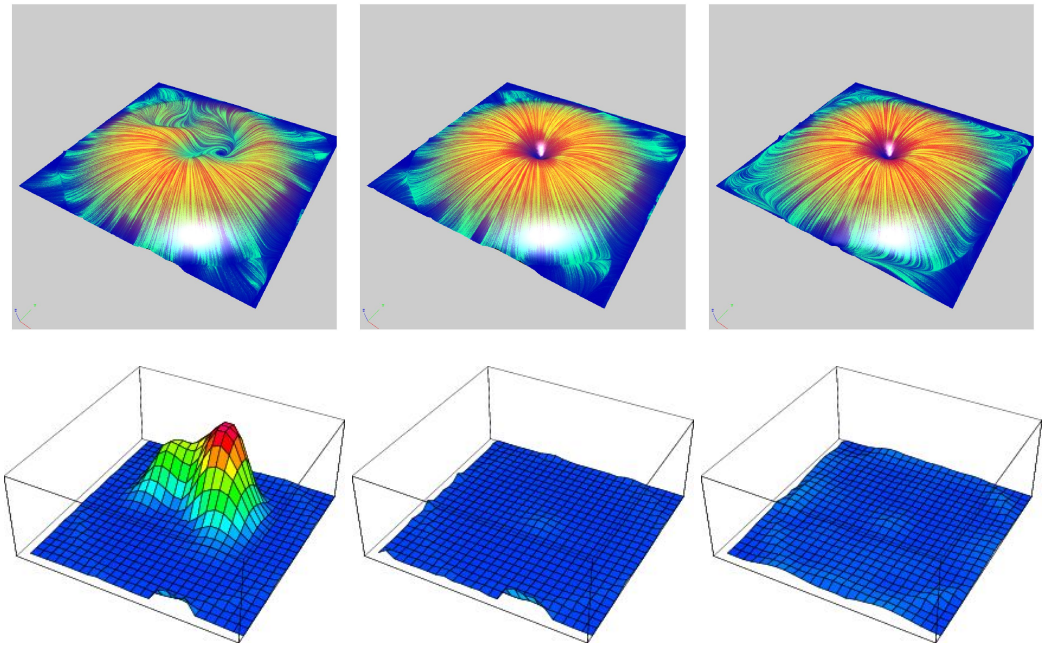


Figure 5.3: First row: LIC on a bump map representing the strength of the displacement. Second row: Error plots of the obtained displacement vector fields.

In the illustrations in figure 5.3 two different properties are shown. The first row contains the displacement fields plotted onto a bump map of the displacement vector lengths. The second row shows the absolute error of the displacement field. Notice that all plots share the same scale. Therefore we can verify the extent of the error made by the first method and we see that the two vortex handling registrations are quite similar.

In the most right error plot (which is again the vortex extraction method) one can see that the amount of the error on the circle defining the boundary of the frog-eye is higher than in the middle registration (which is the vortex suppression method).

Finally, we can say that our new vortex extraction method is a big improvement to the basing image registration. The continuous projection of the vortices leads the registration process over the local minima in the intermediate steps. Therefore a much better solution is found.

Compared to the vortex suppression method we make a few more errors near the borders of the image. Whether this is significant for realistic applications or only a side effect of the frog-eye deformation cannot be said at this point. It will be discussed later.

5.2 TIGER WITH FROG-EYE DEFORMATION

5.2.1 INPUT IMAGES

Figure 5.4: Tiger input images. The left serves as reference and the right as deformable template.

The second example is similar to the first one. Equivalent to the leopard, the tiger has a highly structured coat too. It is obvious that for the computer these two data sets have absolutely nothing in common. Only for humans these are quite similar pictures showing two related animals. Therefore it would make no difference whether we take *any* other enough textured images.

5.2.2 Registration Results

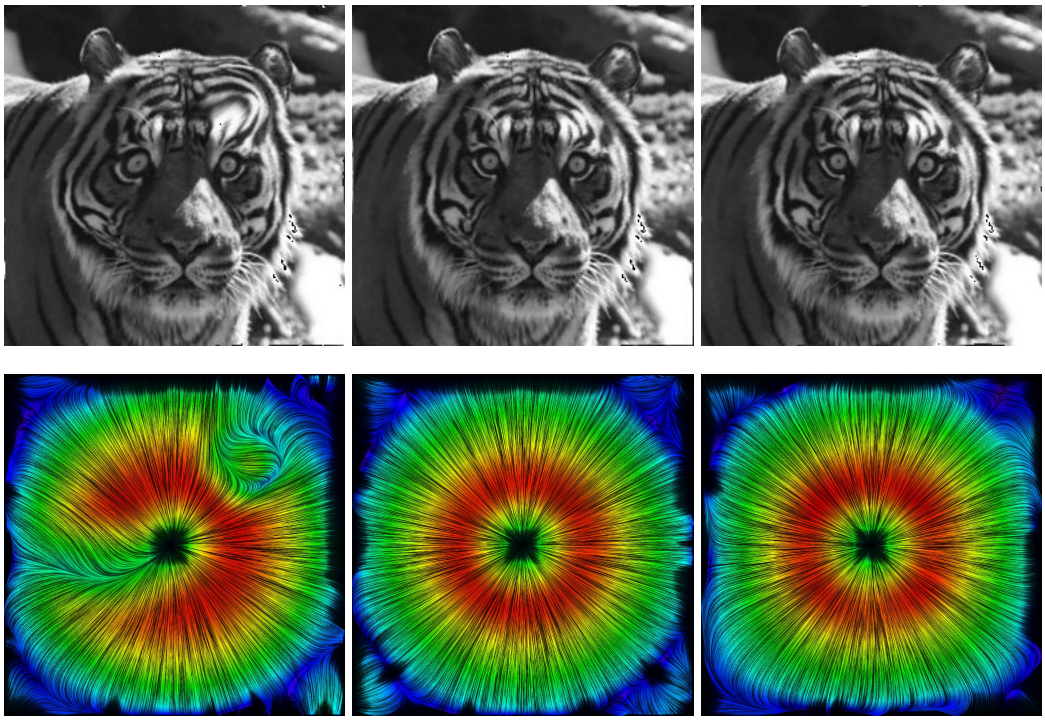


Figure 5.5: The upper row shows the results of the registration methods. Below every tiger one can see the LIC of the belonging displacement field. From left to right following methods are shown: Curvature-based, curvature-based with vortex suppression and curvature-based with vortex extraction.

In this example there are two significant regions where the curvature-based method runs into errors and even there the vortex suppression and the vortex extraction registration improves the solution.

Again it seems to be the texture of the tiger that lets the method run into errors. The curvature-based registration is not able to handle the stripes of the coat.

In the upper row of figure 5.5 one can make sure that the difference between the two vortex handle methods cannot be determined by the images. Both tigers seem to be registered perfectly. Even the LICs show no significant difference. Only in the border regions they differ in their structure.

The error plots in figure 5.6 disclose two things. Firstly one can see that the buckles in the tiger of the curvature-based method are the points with the greatest registration error. Secondly they show that contrary to the first impression that the two vortex handling methods do not work perfectly.

The registrations with vortex suppression has notable errors in the middle and directly

onto the borders of the image. Whereas our vortex extraction method has, beside the same error in the middle, significant problems with the *boundary* of the frog-eye.

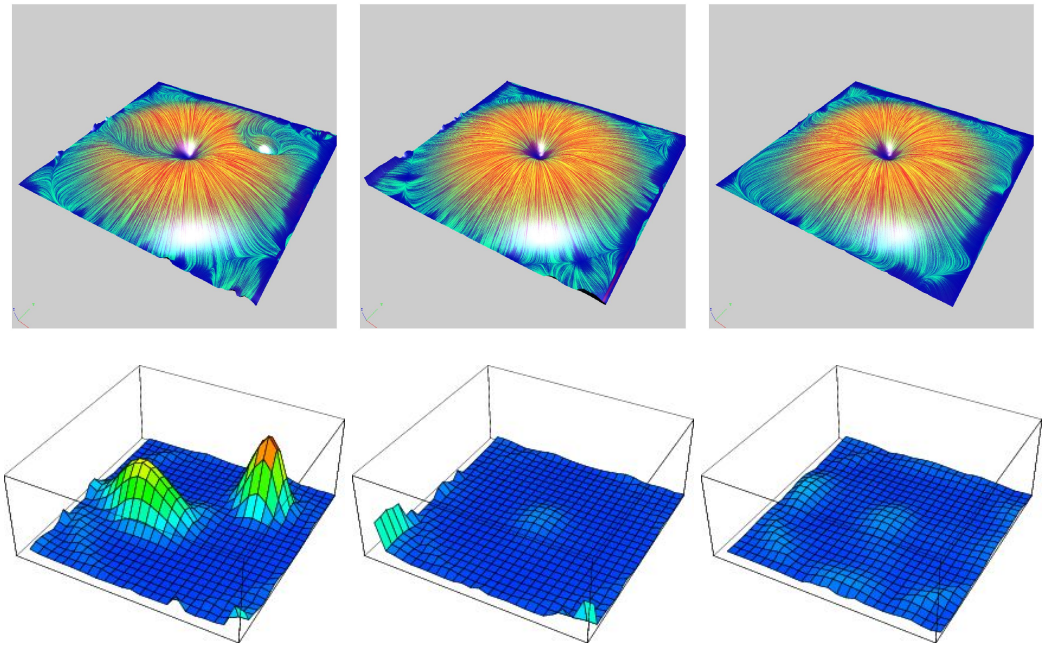


Figure 5.6: First row: LIC on a bump map representing the strength of the displacement. Second row: Error plots of the obtained displacement vector fields.

Finally, the worst registration is again the curvature-based without any processing of vortices. The other two methods work reasonably well. Apparently the vortex suppression has an error maximum at the border whereas the errors of the vortex extraction are distributed on the boundary of the frog-eye.

5.3 Discussion

Before we present a realistic example we want to discuss how the results can be interpreted. Firstly we have shown that the behavior of the underlying method to produce vortices and to run into wrong solutions can be easily reproduced. Furthermore we pointed out that it is not important what is actually shown on the images. As long as the data is highly stuctured and we have an appropriate deformation of the template, the method will run into more or less heavy errors.

Our new method improves this behavior significantly. During the process of finding the solution the algorithm moves along one of many possible paths. In each step it goes into a direction that improves the actual result. Important is that the algorithm does not

see the *end*. It can only decide its next move by looking exactly one time step forward in each direction.

If you have to find a valley in the mountains you would probably do the same. Going downwards with every footstep until you reach a point where it is not possible anymore to walk deeper. The question is, how can you be sure you found the deepest valley and not only a little one between some mountains. This is problematic because you cannot be sure.

The old algorithm has an equivalent problem. It runs into a local minima without the knowledge that there is a much better transformation for the registration problem.

Our new approach uses an advantage we have: We know that vortices are very unlikely. Projecting them out of the intermediate steps ensures that the part of the movement that goes into the wrong direction is sorted out. Therefore the algorithm cannot deviate from the right path and walks into the global solution.

The results show that our method has the highest amount of errors at the boundary of the frog-eye. There are two possible reasons for this. Firstly one should notice that the boundary[1] of the frog-eye is not continuous. For a comparison the reader can look back on page 41 at the visualizations of the frog-eye transformation. With a smooth displacement field it is not possible to represent this boundary correctly.

The second reason is that those regions are very near to the border of the image. We already pointed out that the vortex projection at the boundary is problematic. It is possible that errors in the projection on a certain level have notable influences on the result of the image registration.

We have shown that the registration extended by a vortex extraction improves the old algorithm significantly. We gave an explanation why our new approach is working and where its drawbacks are. Since many readers may be distrustful whether we can simply apply the conclusions on a real application, we will give a third example which is more realistic than the first two.

[1] This is where the bump changes into the untransformed grid.

5.4 Registration of Uterine Cervix Slices

5.4.1 Input images

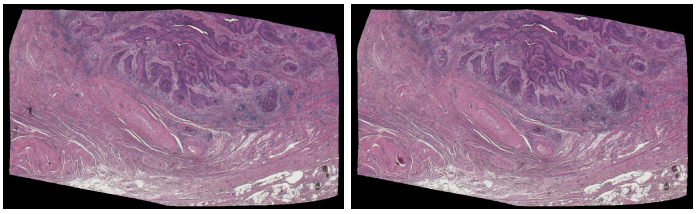


Figure 5.7: These images are two HE-stained histological slices consecutively cut from a specimen of a squamous cell carcinoma of the uterine cervix which are to be registered onto another.

For the last example we chose a realistic registration problem given in two HE-stained histological slices which were consecutively cut from a specimen of uterine cervix. As described in the introduction our method was designed to improve the last registration step in the process of a 3D reconstruction of a tumor invasion front. Therefore the two given slices were already processed through a rigid and a polynomial registration.

At this point it is impossible for a viewer to find differences in the reference and the template image which are shown in figure 5.7. Nevertheless, the two images are not equal but the only differences are small and difficult to recognise.

At least now it should become clear why we did not use those kind of sample data up to now. What we used in the previous examples is the human ability to observe faces. Everyone is extremely trained in recognizing even the smallest deformations of those well known objects. Interpreting faces is a great accomplishment of the human eye-brain system.

In contrast to that only few people are trained to look on medical tissue images. Much experience is required to recognize differences and registration errors. This is the reason for using animals for demonstration purposes.

Nevertheless we want to provide this example. Since the difference between the reference and the registered template is unrecognizable we cannot use them to evaluate the quality of the registration. Instead of these images we will use the LICs to determine the work of the methods.

5.4.2 REGISTRATION RESULTS

In figure 5.8 the LICs of the uterine cervix registrations are given. Subfigure (a) belongs to the original curvature-based non-linear registration. Since this method was used in [BKE+05a] the same line integral convolution is given there. This first displacement field consists of many particular features. Since we do not know the real solution of the registration we are not able to state which of these features are errors and which not.

This is a problem and we have to discuss what results are expected by applying our new method. The previous test situations showed that highly textured regions of the image are now handled very well. Local significant errors of the old method vanish completely.

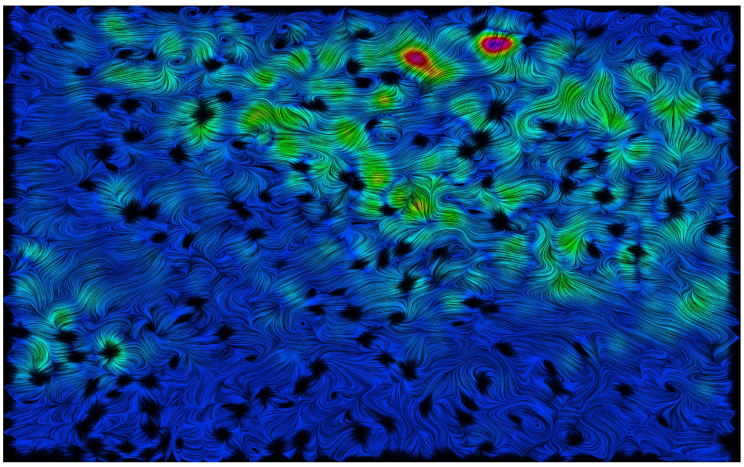
Therefore we expect the method to reduce the number of features in the result. As we can see in the LIC (a) of figure 5.8 those features are uniformly distributed to a certain degree over the displacement field. We say "features" because we cannot exactly determine whether they are sources, sinks, vortices or a combination of them.

What we know is that the input data was highly textured and it is very likely this causes an amount of wrong displacements. If our method smooths the displacement field to some extent, we can assume that those regions contained wrong displacements.

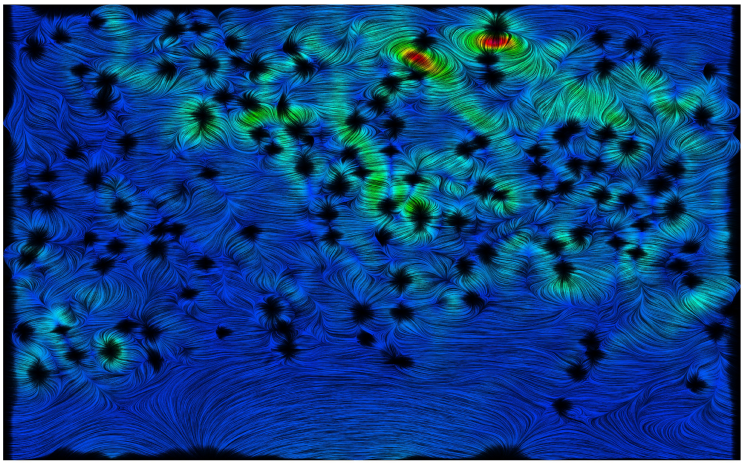
Observing the pictures (b) of figure 5.8, which represents the result of the vortex suppression method, one can see that the displacement field appears to be much smoother. Especially at the bottom a big difference is notable. We can assume that the input pictures did not differ that much in the lower part. The promiscuous displacements seem to be caused by registration itself.

The result of our method is given in the subfigure (c). Surprisingly our method seems to smooth the displacement field more than the vortex suppression registration.

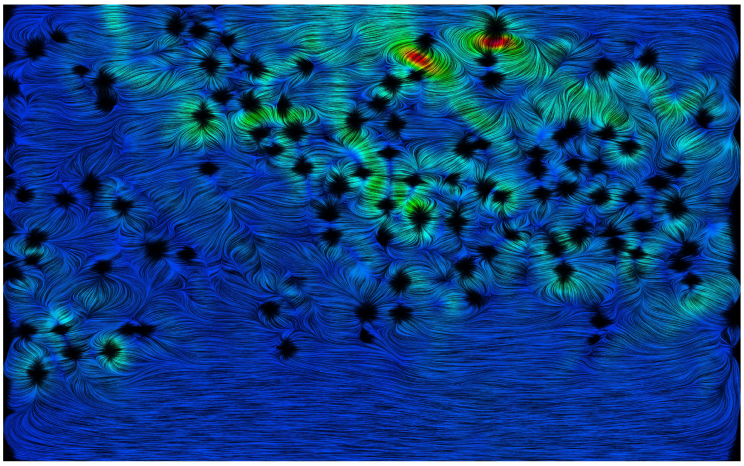
Finally we can only say that our method works as expected. Since the tissue is of course highly textured and since we know that our algorithm works equivalent on all input data it is very likely that our method extracted unwanted vortices from the solution.



(a) The LIC that belongs to the displacement field of the registration method without vortex handling.



(b) The LIC that belongs to the displacement field of the registration method with vortex suppression.



(c) The LIC that belongs to the displacement field of our new registration method with vortex extraction.

Figure 5.8: LICs of the uterine cervix example.

CHAPTER 6

CONCLUSION

IMPORTANT NOTICE TO PURCHASERS:
The Entire Physical Universe, Including This Product, May One Day Collapse Back into an Infinitesimally Small Space. Should Another Universe Subsequently Reemerge, the Existence of This Product in That Universe Cannot Be Guaranteed. [HS91]

The target of our work was to show whether we can influence successfully an image registration during the solution procedure. It is a great advance that we are able to lead the method to solutions that we decide to be more likely than others.

In the case of this thesis we developed a projection technique that allows us to suppress unwanted vortices. Although this projection has some known problems on discrete data, the integrated process of curvature-based registration and vortex projection works better than expected.

The problematic, high-textured regions which were a pitfall for the curvature-based registration are now handled excellently. Even in comparison with the existing vortex suppression method our new registration yields nearly the same results in the analytic examples.

Since there is significantly more smoothness compared to the vortex suppression method in the real-world example, we are confident that this work will be of great use in the registration of tissue images.

Beside this, the reader should notice that the results of this work prepare the ground for broadly based future plans. The image registration in this special case required solutions which are vortex-free. It is very likely that there are other applications which have different assumptions on the resulting displacement field.

For instance a divergence-free approach would be of great interest. This would represent a situation where the registration saves the mass of the template because without sources and sinks a change of this property is not possible.

APPENDIX A

LIST OF ABBREVIATIONS.

Some of the abbreviations are described in more detail in the text and some are only used and details are given here. The page number after the description refers to the first occurence of the acronym.

CCD Charge coupled device, page 4

FFTW Fastest Fourier Transform in the West, page 50

LIC Line integral convolution, page 9

LSE Linear system of equations, page 47

MRI Magnetic resonance imaging. Also known as magnetic resonance tomography (MRT) or nuclear magnetic resonance (NMR), page 4

PAT Principal axes transformation, page 25

PDE Partial differential equation, page 32

pixel Picture element, page 5

w.r.t. With respect to, page 16

x-ray X-rays are a form of radiation discovered by Wilhelm Röntgen. They are often called Röntgen radiation and in medicine they are used for imaging bone structures, page 4

APPENDIX B

SUMMERY OF USED SYMBOLS AND THEIR MEANING

B.1 OPERATORS, NORMS AND ATTRIBUTES

$\|\cdot\|$**:** The norm of a vector. The dimension of the space depends on the context.

∇**:** The nabla operator is a symbolic vector and often used in vector analysis. It is defined by

$$\nabla := \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{pmatrix}$$

$\operatorname{grad} f(\boldsymbol{x})$**:** The gradient of a scalar function $f : \mathbb{R}^3 \to \mathbb{R}$ is a vector field that represents the partial derivatives of $f(\boldsymbol{x})$ in every direction. It can be expressed with the Nabla operator (∇).

$$\operatorname{grad} f(\boldsymbol{x}) = (\nabla f)(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_2} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_3} f(\boldsymbol{x}) \end{pmatrix}$$

$\operatorname{div} \boldsymbol{f}(\boldsymbol{x})$**:** The divergence of a vector-valued function $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^3$ is a scalar field that represents the flow.

$$\operatorname{div} \boldsymbol{f}(\boldsymbol{x}) = \nabla \cdot \boldsymbol{f}(\boldsymbol{x}) = \frac{\partial}{\partial x_1} f_1(\boldsymbol{x}) + \frac{\partial}{\partial x_2} f_2(\boldsymbol{x}) + \frac{\partial}{\partial x_3} f_3(\boldsymbol{x})$$

$\operatorname{curl} \boldsymbol{f}(\boldsymbol{x})$**:** The curl of a vector-valued function $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^3$ is a vector field that

represents the rotation of $\boldsymbol{f}(\boldsymbol{x})$ in every direction.

$$\operatorname{curl} \boldsymbol{f}(\boldsymbol{x}) = \nabla \times \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x_2} f_3(\boldsymbol{x}) & - & \frac{\partial}{\partial x_3} f_2(\boldsymbol{x}) \\ \frac{\partial}{\partial x_3} f_1(\boldsymbol{x}) & - & \frac{\partial}{\partial x_1} f_3(\boldsymbol{x}) \\ \frac{\partial}{\partial x_1} f_2(\boldsymbol{x}) & - & \frac{\partial}{\partial x_2} f_1(\boldsymbol{x}) \end{pmatrix}$$

Δ**:** The Laplace operator. It is defined by

$$\Delta f = \operatorname{div} \operatorname{grad} f = \nabla \cdot \nabla f$$

$\mathbf{X}^\top$**:** The transpose of a matrix which makes the columns to rows and vice versa.

$\mathbf{A} \times \mathbf{B}$**:** The cross-product of the matrices $\mathbf{A}$ and $\mathbf{B}$.

$\delta F(\boldsymbol{u} + \epsilon \boldsymbol{g})$**:** Gâteaux derivative of $\boldsymbol{u}$ in the direction of $\boldsymbol{g}$.

B.2 General notations

$\mathbb{N}, \mathbb{R}, \mathbb{C}$**:** Symbols for natural, real and complex numbers.

$\boldsymbol{p}, \boldsymbol{v}(x, y), \boldsymbol{x}$**:** Symbols printed in bold italic font indicate that it is a vector or a vector-valued function. Sometimes it depends on the context whether a single symbol like $\boldsymbol{p}$ refers to a vector or a vector-valued function.

$\mathbf{A}, \mathbf{B}, \mathbb{1}$**:** In formulas and mathematical expressions an upper-case symbol typed in bold font denotes a matrix. The symbol $\mathbb{1}$ denotes the unity matrix which has a 1 on every position on the main diagonal and 0 otherwise. Its dimension depends on the context.

$\mathfrak{T}, \mathfrak{T}(R(\boldsymbol{x}))$**:** When the context does not guess another meaning, then it is commonly used for transformations in registration methods. $\mathfrak{T}(R)$ denotes a transformation $\mathfrak{T}$ of an image R.

$\mathcal{D}, \mathcal{S}$**:** $\mathcal{D}(\boldsymbol{u})$ (difference) and $\mathcal{S}(\boldsymbol{u})$ (smoothness) are functionals which are required for defining non-parametric image registration problems.

$\boldsymbol{u}$**:** In almost every case denotes $\boldsymbol{u}$ or $\boldsymbol{u}(\boldsymbol{x})$ a displacement field for a given registration problem.

Ω**:** General notation for the compact area $\Omega =]0, 1[^d$ where a d-dimensional image is defined.

BIBLIOGRAPHY

[Ami94] Yali Amit. A nonlinear variational problem for image matching. *SIAM Journal on Scientific Computing*, 15(1):207–224, January 1994.

[AS84] Milton Abramowitz and Irene A. Stegun. *Pocketbook of Mathematical Functions*. Verlag Harri Deutsch, Frankfurt/Main, 1984.

[BD69] W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations*. John Wiley, New York, second edition, 1969.

[BEH+06] Ulf-Dietrich Braumann, Jens Einenkel, Lars-Christian Horn, Jens-Peer Kuska, Markus Löffler, Nico Scherf, and Nicolas Wentzensen. Registration of histologic colour images of different staining. In Heinz Handels, Jan Ehrhardt, Alexander Horsch, Hans-Peter Meinzer, and Thomas Tolxdorff, editors, *Bildverarbeitung für die Medizin 2006 – Algorithmen, Systeme, Anwendungen*, Informatik aktuell, pages 231–235. Springer-Verlag, March 2006.

[BFH+06] Ulf-Dietrich Braumann, Heike Franke, Jan Hengstler, Jens-Peer Kuska, and Marco Weber. Graph-based quantification of astrocytes. In Heinz Handels, Jan Ehrhardt, Alexander Horsch, Hans-Peter Meinzer, and Thomas Tolxdorff, editors, *Bildverarbeitung für die Medizin 2006 – Algorithmen, Systeme, Anwendungen*, Informatik aktuell, pages 379–383. Springer-Verlag, March 2006.

[BK05] Ulf-Dietrich Braumann and Jens-Peer Kuska. Influence of the boundary conditions on the result of non-linear image registration. In *Proceedings of the IEEE International Conference on Image Processing*, pages I-1129–I-1132. IEEE Signal Processing Society, September 2005.

[BK06] Ulf-Dietrich Braumann and Jens-Peer Kuska. A new equation for nonlinear image registration with control over the vortex structure in the displacement

field. In *Proceedings of the IEEE International Conference on Image Processing*, pages 329–332. IEEE Signal Processing Society, October 2006.

[BKE03] Ulf-Dietrich Braumann, Jens-Peer Kuska, and Jens Einenkel. Dreidimensionale rekonstruktion der invasionsfront von gebärmutterhalskarzinomen. In *Bildverarbeitung für die Medizin*, pages 230–234, 2003.

[BKE+04] Ulf-Dietrich Braumann, Jens-Peer Kuska, Jens Einenkel, Lars-Christian Horn, and Michael Höckel. Quantification of tumour invasion fronts using 3D reconstructed histological serial section. In *Bildverarbeitung für die Medizin*, pages 70–74, 2004.

[BKE+05a] Ulf-Dietrich Braumann, Jens-Peer Kuska, Jens Einenkel, Lars-Christian Horn, Markus Löffler, and Michael Höckel. Three-dimensional reconstruction and quantification of cervical carcinoma invasion fronts from histological serial sections. *IEEE Transactions on Medical Imaging*, 24(10):1286–1307, October 2005.

[BKE+05b] Ulf-Dietrich Braumann, Jens-Peer Kuska, Jens Einenkel, Lars-Christian Horn, Markus Löffler, and Michael Höckel. Three-dimensional reconstruction and quantification of cervical carcinoma invasion fronts from histological serial sections. *IEEE Transactions on Medical Imaging*, 24(10):1286–1307, October 2005.

[BKE+06] Ulf-Dietrich Braumann, Jens-Peer Kuska, Jens Einenkel, Lars-Christian Horn, Markus Löffler, and Michael Höckel. 3-d tumour invasion analysis based on large histological serial sections. In *Proceedings of the 8th European Congress on Telepathology and 2nd International Congress on Virtual Microscopy*, page 27, October 2006.

[BN96] Morten Bro-Nielsen. *Medical Image Registration and Surgery Simulation*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, August 1996. IMM-PHD-1996-25.

[BNG96] M. Bro-Nielsen and C. Gramkow. Fast fluid registration of medical images. *Lecture Notes in Computer Science*, 1131:267, 1996.

[Bro81] C. Broit. *Optimal registration of deformed images*. Ph.d. thesis, University of Pennsylvania, 1981.

[BSM05] Ilja N. Bronstein, Konstantin A. Semendjajew, and Gerhard Musiol. *Taschenbuch der Mathematik*. Deutsch (Harri), 2005.

[CL93] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings SIGGRAPH '93*, pages 263–272, 1993.

[Eco05] Umberto Eco. *Wie man eine wissenschaftliche Abschlußarbeit schreibt. Doktor-, Diplom- und Magisterarbeit in den Geistes- und Sozialwissenschaften*. UTB Uni-Taschenbücher Verlag, 2005.

[EKH+06] Jens Einenkel, Jens-Peer Kuska, Lars-Christian Horn, Nicolas Wentzensen, Michael Höckel, and Ulf-Dietrich Braumann. Combined three-dimensional microscopic visualisation of tumour-invasion front of cervical carcinoma. *The Lancet Oncology*, 7(8):698, August 2006.

[FJ98] Matteo Frigo and Steven G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume 3, pages 1381–1384, Seattle, WA, May 1998.

[FM02a] B. Fischer and J. Modersitzki. Curvature based registration with applications to MR-mammography. In PMA Sloot, CJK Tan, JJ Dongarra, and AG Hoekstra, editors, *Computational Science – ICCS 2002*, pages 203–206. Springer, LNCS 2331, 2002.

[FM02b] B. Fischer and J. Modersitzki. Fast curvature based registration of MR-mammography images. In M Meiler et al., editor, *Bildverarbeitung für die Medizin*, pages 139–143. Springer, 2002.

[FM03a] B. Fischer and J. Modersitzki. Curvature based image registration. *J. of Mathematical Imaging and Vision*, 18(1):81–85, 2003.

[FM03b] B. Fischer and J. Modersitzki. Curvature based image registration. *J. of Mathematical Imaging and Vision*, 18(1):81–85, 2003.

[HS81] B. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 20, 1981.

[HS91] Susan Hewitt and Ed Subitzky. A call for more scientific truth in product warning labels. *The Journal of Irreproducible Results*, 36(1), 1991.

[KBS+06a] Jens-Peer Kuska, Ulf-Dietrich Braumann, Nico Scherf, Jens Einenkel, Lars-Christian Horn, Nico Wentzensen, Magnus von Knebel Doeberitz, and

Markus Löffler. Large histological serial sections for computational tissue volume reconstruction. In Markus Löffler and Alfred Winter, editors, *Tagungsband der 51. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds) e.V.*, pages 64–65, September 2006.

[KBS+06b] Jens-Peer Kuska, Ulf-Dietrich Braumann, Nico Scherf, Markus Löffler, Jens Einenkel, Michael Höckel, Lars-Christian Horn, Nicolas Wentzensen, and Magnus von Knebel Doeberitz. Image registration of differently stained histological sections. In *Proceedings of the IEEE International Conference on Image Processing*, pages 333–336. IEEE Signal Processing Society, October 2006.

[Kus97] Jens-Peere Kuska. *Mathematica und C in der modernen Theoretischen Physik, m. CD-ROM*. Springer, Berlin, 1997.

[LP05] Christian Lang and Norbert Pucker. *Mathematische Methoden in der Physik*, volume 2. Spektrum Akademischer Verlag, 2005.

[MF93] C. Maurer and J. Fitzpatrick. A review of medical image registration, 1993.

[Mod04] Jan Modersitzki. *Numerical Methods for Image Registration (Numerical Mathematics and Scientific Computation)*. Oxford University Press, USA, 2004.

[MV90] Meyberg and Vachenauer. *Höhere Mathematik*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1990.

[MV98] J. B. A. Maintz and M. A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–16, 1998.

[MV01] Meyberg and Vachenauer. *Höhere Mathematik 2*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2001.

[Str95] Walter A. Strauss. *Partielle Differentialgleichungen, Eine Einführung*. Verlag Vieweg, 1995.

[Str03] Gilbert Strang. *Introduction to linear algebra*. pub-WCP, pub-WCP:adr, third edition, 2003.

[Wol03] Stephen Wolfram. *The Mathematica Book, Fifth Edition*. Wolfram Media, 2003.

EIGENSTÄNDIGKEITSERKLÄRUNG

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Räpitz, Februar 2007 Patrick Scheibe